The recursive path and polynomial ordering for first-order and higher-order terms*

Miquel Bofill[1], Cristina Borralleras[2], Enric Rodríguez-Carbonell[3], and Albert Rubio[3]

[1] Universitat de Girona, Spain
[2] Universitat de Vic, Spain
[3] Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract. In most termination tools two ingredients, namely recursive path orderings (RPO) and polynomial interpretation orderings (POLO), are used in a consecutive disjoint way to solve the final constraints generated from the termination problem.
In this paper we present a simple ordering that combines both RPO and POLO and defines a family of orderings that includes both, and extend them with the possibility of having, at the same time, an RPO-like treatment for some symbols and a POLO-like treatment for the others.
The ordering is extended to higher-order terms, providing a new fully automatable use of polynomial interpretations in combination with beta-reduction.
Key words: Program correctness, Term rewriting, Typed lambda-calculus, Termination, Term orderings.

1 Introduction

Term orderings have been extensively used in termination proofs of rewriting. They are used both in direct proofs of termination showing decreasingness of every rule or as ingredients for solving the constraints generated by other methods like the Dependency Pair approach [1] or the Monotonic Semantic Path Ordering [4].

The most widely used term orderings in automatic termination tools (e.g., AProVE [13], C*i*ME [8], MU-TERM [21], TTT [15], ...) are the recursive path ordering (RPO) and the polynomial ordering (POLO). Almost all known tools implement these orderings. RPO and POLO are incomparable, so that they are used independently in a sequential way, trying first one method (maybe under some time limit) and, in case of failure, trying the other one afterwards. Some recent implementations use independent parallel calls to both methods.

As an alternative to this independent application of both methods we propose a new ordering that combines both RPO and POLO. The new family of orderings, called RPOLO, includes strictly both RPO and POLO as well as the

* This work has been partially supported by the Spanish MEC/MICINN under grants TIN2008-04547 and TIN 2010-21062-C02-01

sequential combination of both. Our approach is based on splitting the set of symbols into those handled in an RPO-like way (called RPO-symbols) and those that are interpreted using a polynomial interpretation (called POLO-symbols). In this paper, only linear polynomial interpretations are considered. These interpretations are never applied to terms headed by an RPO-symbol. Instead, the term is interpreted as a new variable (labeled by the term). This is crucial to be able to extend the ordering to the higher-order case, since applying polynomial interpretations to beta-reduction is not easy. However, the introduction of different unrelated variables for every term makes us lose stability under substitutions and (weak) monotonicity. To avoid that, a context relating the variables is introduced, but then a new original proof of well-foundedness is needed.

Although the new ordering is strictly more powerful than its predecessors and thus examples that can be handled by RPOLO and neither by RPO nor by POLO can be cooked, in practice, there is no real gain when using RPOLO on the first-order examples coming from the Termination Problem Data Base (TPDB)[4].

Due to this, we show its practical usefulness by extending it, using the same techniques as for the higher-order recursive path ordering [18] (HORPO), to rewriting on simply typed higher-order terms union beta-reduction. The resulting ordering, called HORPOLO, can hence be used to prove termination of the so called Algebraic Functional Systems [17] (AFS), and provides a new automatable termination method that allows the user to have polynomial interpretations on some symbols in a higher-order setting. Polynomial interpretations for higher-order rewrite systems à la Nipkow, where studied in [22,23]. In contrast to our approach, there, under some conditions, even the application operator can be interpreted. Automation of these kind of interpretations has recently been studied in [10].

In order to ease the reading, the ordering is first presented in Section 3 for first-order terms and later on for higher-order terms. All the novelties of the ordering are already present in the first-order version, so that the method can be fully understood within this part. Then the ordering is extended to the higher-order case by adapting all proofs for HORPO in [18] to HORPOLO.

The paper is structured as follows. In Section 2 some preliminaries on first-order terms and orderings are given. Section 3 is devoted to presenting the ordering for first-order terms. In Section 4, basic definitions and notions on higher-order terms and orderings are provided. In Section 5, the ordering is extended to higher-order terms. A necessary improvement of the ordering is given in Section 6 and the resulting ordering is applied to several examples in Section 7. Finally, experiments and some details about the implementation of the technique are given in Section 8 before the conclusions.

[4] See `http://termination-portal.org/wiki/TPDB`

2 First-order term rewriting and termination

A *signature* $\mathcal{F}$ is a set of function symbols. Given a signature $\mathcal{F}$ and a set of variable symbols $\mathcal{X}$ with $\mathcal{F} \cap \mathcal{X} = \varnothing$, $\mathcal{T}(\mathcal{F}, \mathcal{X})$ denotes the terms build from $\mathcal{F}$ and $\mathcal{X}$. Given $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $|t|$ denotes the number of symbols in $\mathcal{F} \cup \mathcal{X}$ occurring in t; $|t|_{\mathcal{F}}$ denotes the number of symbols in $\mathcal{F}$ occurring in t; $\mathcal{V}ar(t)$ is the set of variables occurring in t; and for $t \notin \mathcal{X}$, $top(t)$ denotes the topmost symbol of t, i.e., $top(f(t_1, \ldots, t_n)) = f$. The notation $\bar{t}$ shall be ambiguously used for a list, a multiset, or a set of terms $t_1, \ldots, t_n$.

The subterm of t at position p is denoted by $t|_p$, where p is a sequence of positive integers describing the path from the root of t to the subterm, where λ denotes the empty sequence and hence the root position. Let $\trianglerighteq$ denote the subterm relation, defined as $t \trianglerighteq t|_p$ for all position p of t, and let $\triangleright$ be its strict part, which fulfils $t \triangleright t|_p$ for all position $p \neq \lambda$ of t.

A *context* $u[\,]$ is a term with a hole, and $u[t]$ denotes the term resulting from placing t in the hole of u. A *substitution* γ is a mapping from variables to terms. We denote the domain and the range of γ by $\mathcal{D}om(\gamma)$ and $\mathcal{R}an(\gamma)$ respectively. The application of a substitution γ to a term s is denoted by $s\gamma$.

2.1 Term orderings

The reflexive and transitive closure of a binary relation $\succ$ is denoted by $\succ^*$, its reflexive closure by $\succ^=$ and its transitive closure by $\succ^+$.

An *equivalence relation* is a reflexive, symmetric and transitive relation. A transitive and irreflexive binary relation is a (strict partial) *ordering* and a transitive and reflexive binary relation is a *quasi-ordering*. The equivalence relation associated to a quasi ordering $\succeq$, denoted by $=$, is the intersection of $\succeq$ with its inverse. The associated strict partial ordering, denoted by $\succ$, is their difference. A relation $\succ$ is said to be *compatible* with another relation $\succeq$ if $\succ \cdot \succeq \subseteq \succ$ or $\succeq \cdot \succ \subseteq \succ$. If $\succ$ is the strict part of a quasi-ordering $\succeq$, then $\succ$ is compatible with $\succeq$.

Given a binary relation $\succ$, a term s is *strongly normalizing* with respect to $\succ$ if there is no infinite sequence with $\succ$ issuing from s. The relation $\succ$ itself is *strongly normalizing*, *well-founded* or *terminating*, if all terms are strongly normalizing with respect to $\succ$. A substitution is said to be strongly normalizing if all terms in its range are strongly normalizing. If $\succ$ is a well-founded relation then $\succ^+$ is a well-founded ordering.

A *precedence* $\succeq_{\mathcal{F}}$ is a quasi-ordering on a signature $\mathcal{F}$ such that $\succ_{\mathcal{F}}$ is well-founded. Note that, in particular, if $\mathcal{F}$ is finite then $\succ_{\mathcal{F}}$ is always well-founded.

Let s and t be arbitrary terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A relation $\succ$ is *monotonic* if $s \succ t$ implies $u[s] \succ u[t]$ for all contexts $u[\,]$, and *stable under substitutions* if $s \succ t$ implies $s\gamma \succ t\gamma$ for all substitutions γ.

Monotonic orderings that are stable under substitutions are called *rewrite orderings*. A *reduction ordering* $\succ$ is a rewrite ordering that is well-founded. An ordering $\succ$ included in the strict part of a quasi-ordering $\succeq$ is *weakly-monotonic* if $u[s] \succeq u[t]$ for all contexts $u[\,]$ whenever $s \succ t$.

Given a relation $\succeq$, the *monotonic extension* of $\succeq$ on tuples of elements, denoted by $(\succeq)_{mon}$, is defined as $\langle s_1, \dots, s_n \rangle \ (\succeq)_{mon} \ \langle t_1, \dots, t_n \rangle$ if $s_i \succeq t_i$ $\forall i \in \{1, \dots, n\}$ with $n \geq 0$. Similarly, the *monotonic extension* of $\succeq$ on finite multisets is defined as $\{s_1, \dots, s_n\} \ (\succeq)_{mon} \ \{t_1, \dots, t_n\}$ if for some permutation π of $\{1 \dots n\}$ we have $s_{\pi(i)} \succeq t_i$ for all $i \in \{1 \dots n\}$.

Note that, by a straightforward induction, $M(\succeq)^*_{mon}N$ implies $M(\succeq^*)_{mon}N$ for both tuples and multisets. Additionally, if $\succeq$ is stable under substitutions then the extension $(\succeq)_{mon}$ on both tuples and finite multisets is stable under substitutions.

Given a relation $\succ$, the *multiset extension* of $\succ$ w.r.t. a relation $\succeq$ on finite multisets, denoted by $(\succ)_{mul}$, is defined as $M \cup S \ (\succ)_{mul} \ N \cup T$ if $M \ (\succeq)_{mon} \ N$ and either $S \neq T = \varnothing$ or $T \neq \varnothing$ and $\forall t \in T \ \exists s \in S$ such that $s \succ t$.

If $\succ$ is a well-founded ordering on terms which is compatible with $\succeq$, then $(\succ)_{mul}$ is a well-founded ordering on finite multisets of terms. If $\succ$ and $\succeq$ are stable under substitutions then $(\succ)_{mul}$ is stable under substitutions. Moreover, if $\succ$ is compatible with $\succeq$ then $(\succ)_{mul}$ is compatible with $(\succeq)_{mon}$.

Given a relation $\succ$, the *lexicographic extension* of $\succ$ w.r.t. a relation $\succeq$ on bounded tuples, denoted by $(\succ)_{lex}$, is defined as $\langle s_1, \dots, s_n \rangle \ (\succ)_{lex} \ \langle t_1, \dots, t_m \rangle$ if $\langle s_1, \dots, s_k \rangle (\succeq)_{mon} \langle t_1, \dots, t_k \rangle$ for some k and $k = m < n$ or $s_{k+1} \succ t_{k+1}$.

If $\succ$ is a well-founded ordering on terms which is compatible with $\succeq$ then $(\succ)_{lex}$ is a well-founded ordering on bounded tuples of terms. If $\succ$ and $\succeq$ are stable under substitutions then $(\succ)_{lex}$ is stable under substitutions. Moreover, if $\succ$ is compatible with $\succeq$ then $(\succ)_{lex}$ is compatible with $(\succeq)_{mon}$.

Let $\succeq_1, \dots, \succeq_n$ be quasi-orderings and $\succ_1, \dots, \succ_n$ be orderings such that $\succ_i$ is compatible with $\succeq_i$ for all $i \in \{1 \dots n\}$. The *lexicographic combination* of $\succ_1, \dots, \succ_n$, denoted by $(\succ_1, \dots, \succ_n)_{lex}$, is defined as $s \langle \succ_1, \dots, \succ_n \rangle_{lex} t$ if either $s \succ_i t$ for some i and $s \succeq_j t$ for all $j < i$.

If all $\succ_i$ are well-founded (stable under substitutions) then so is their lexicographic combination.

2.2 Term rewriting and termination

A *term rewrite system* (TRS) over a signature $\mathcal{F}$ is a set of rules $l \to r$ where $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$.

Given a TRS R, s rewrites to t with R, denoted by $s \to_R t$, if there is some rule $l \to r$ in R, such that $s = u[l\gamma]$ and $t = u[r\gamma]$ for some context u and substitution γ. We denote by $\mathrel{+\!\!\!\!\!\!\to}_R$ a parallel rewriting step using rules in R, i.e. the application of several rewriting steps at the same time in disjoint positions (i.e. non-overlapped).

A TRS R is terminating if $\to_R$ is terminating. Thus, the transitive closure $\xrightarrow{+}_R$ of $\to_R$ for any terminating TRS R is a reduction ordering. Furthermore, reduction orderings characterize termination of TRSs: a rewrite system R is terminating iff all rules are contained in a reduction ordering $\succ$, i.e., $l \succ r$ for every $l \to r \in R$.

Instead of finding reduction orderings, most automatic tools for proving termination are based on constraint frameworks like [12] or [6]. There, a termination problem is transformed into an ordering constraint problem using, for instance, the Dependency Pair approach, which is successively simplified by applying different sound (and sometimes complete) rules. Some of these transformation rules are based on finding a so-called reduction pair $(\succeq, \succ)$ satisfying a set of literals of the form $s \succeq t$ or $s \succ t$.

A quasi-ordering $\succeq$ and an ordering $\succ$ form a *reduction pair* $(\succeq, \succ)$ if

$\succeq$ is monotonic and stable under substitutions,
$\succ$ is well-founded and stable under substitutions, and
$\succ$ is *compatible* with $\succeq$.

3 The Recursive Path and Polynomial Ordering

We have a signature $\mathcal{F}$ split into two sets $\mathcal{F}_{POLO}$ and $\mathcal{F}_{RPO}$ where the arity of the symbols is bounded, a precedence $\succeq_{\mathcal{F}}$ on $\mathcal{F}_{RPO}$ and a polynomial interpretation I over the non-negative integers $\mathbb{Z}^+$ for the terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. All symbols f in $\mathcal{F}_{RPO}$ have a status, denoted by $stat(f)$, which indicates if the arguments of f are compared recursively using a multiset extension or a lexicographic extension. We assume that if $f =_{\mathcal{F}} g$ then $stat(f) = stat(g)$. Moreover, the interpretation I is defined by a linear interpretation f_I with coefficients in $\mathbb{Z}^+$ for every symbol f in $\mathcal{F}_{POLO}$ and a variable x_s for every term s with top symbol in $\mathcal{F}_{RPO}$:

$$I(s) = \begin{cases} f_I(I(s_1), \ldots, I(s_n)) & \text{if } s = f(s_1, \ldots, s_n) \text{ and } f \in \mathcal{F}_{POLO} \\ x_s & \text{otherwise} \end{cases}$$

In order to handle these introduced variables x_s, we define a context information to be used when comparing the interpretations. In what follows a *(polynomial) context* is a set of constraints of the form $x \geq E$ where x is a variable and E is a linear polynomial expression over $\mathbb{Z}^+$.

The following arithmetic properties on polynomials will be extensively used.

Lemma 1. *Let* $P_1 = a_0 + a_1 \cdot x_1 + \ldots + a_n \cdot x_n$ *and* $P_2 = b_0 + b_1 \cdot x_1 + \ldots + b_n \cdot x_n$ *be two linear polynomials over* $\mathbb{Z}^+$*. Then*

- $\forall x_1, \ldots, x_n \in \mathbb{Z}^+[P_1 \geq P_2]$ *if and only if* $a_i \geq b_i$ *for all* $i \in \{0 \ldots n\}$*.*
- $\forall x_1, \ldots, x_n \in \mathbb{Z}^+[P_1 > P_2]$ *if and only if* $a_i \geq b_i$ *for all* $i \in \{1 \ldots n\}$ *and* $a_0 > b_0$*.*

Proof. In both cases the right-to-left implication is trivial. For the left-to-right, replacing all variables by 0 we conclude that $a_0 \geq b_0$ and if $P_1 > P_2$ then $a_0 > b_0$. For the remaining coefficients, for every i we replace all variables x_j with $j \neq i$ by 0 and x_i by $a_0 + 1$. Then we have $a_0 + a_i \cdot (a_0 + 1) \geq b_0 + b_i \cdot (a_0 + 1)$. Assume $b_i > a_i$, then $b_i \geq a_i + 1$, which implies the following sequence $b_0 + b_i \cdot (a_0 + 1) \geq b_0 + (a_i + 1) \cdot (a_0 + 1) = b_0 + a_i \cdot a_0 + a_i + a_0 + 1 > a_i \cdot a_0 + a_i + a_0 = a_0 + a_i \cdot (a_0 + 1)$, which is a contradiction. □

Corollary 1. *Let* P_1 *and* P_2 *be two linear polynomials over* $\mathbb{Z}^+$*.*

– *If $P_1 \geq P_2$ then $P_1 = P_2 + P$ for some linear polynomial P over $\mathbb{Z}^+$.*
– *If $P_1 > P_2$ then $P_1 = P_2 + P + 1$ for some linear polynomial P over $\mathbb{Z}^+$.*

Let us now show the way contexts are used when comparing polynomials.

Definition 1. *Let C be a context. The relation $\rightarrow_C$ on linear polynomial expressions over $\mathbb{Z}^+$ is defined by the rules $P + x \rightarrow_C P + E$ for every $x \geq E \in C$.*

Let p and q be linear polynomial expressions over $\mathbb{Z}^+$. Then $p >_C q$ (resp. $p \geq_C q$) if there is some u such that $p \twoheadrightarrow\!\!\!\!\!\!\!+\;\;^{=}_{C} u > q$ (resp. $p \twoheadrightarrow\!\!\!\!\!\!\!+\;\;^{=}_{C} u \geq q$).

We use here (the reflexive closure of) a parallel rewriting step $\twoheadrightarrow\!\!\!\!\!\!\!+\;\;^{=}_{C}$ instead of the transitive closure of $\rightarrow_C$ because it simplifies the proofs without losing any power. Additionally, we can keep the same definition of $>_C$ in the higher-order case, where, as we will see, the relations are no longer transitive.

The following three mutually recursive definitions introduce respectively the context $C(\mathcal{S})$ of a set of terms $\mathcal{S}$, the ordering $\succ_{RPOLO}$ and the two quasi-orderings $\sqsupseteq_{RPOLO}$ and $\succeq_{RPOLO}$.

Before giving the definition of the context we introduce some notation. Note that for all variables $x_u \in \mathcal{V}ar(I(s))$ we have either $u = f(u_1, \ldots, u_n)$ and $f \in \mathcal{F}_{RPO}$ or u is a variable. Let us define the set of *accessible terms* $\mathcal{A}cc(s)$ of s as $\{u \mid x_u \in \mathcal{V}ar(I(s))\}$. By $\mathcal{X}_{\mathcal{S}}$ we denote the set of labeled variables x_u with $u \in \mathcal{S}$. Note that then we have $\mathcal{V}ar(I(s)) = \mathcal{X}_{\mathcal{A}cc(s)}$.

Definition 2. *Let $\mathcal{S}$ be a set of terms u such that $top(u) \notin \mathcal{F}_{POLO}$. The context $C(\mathcal{S})$ is defined as the union of*

1. *$x_u \geq E + 1$ for all $u \in \mathcal{S}$ and for all linear polynomial expressions E over $\mathbb{Z}^+$ and variables $\{x_{v_1}, \ldots, x_{v_n}\}$ such that $u \succ_{RPOLO} v_i$ for all $i \in \{1, \ldots, n\}$.*
2. *$x_u \geq x_v$ for all $u \in \mathcal{S}$ and all v such that $u \sqsupseteq_{RPOLO} v$ and $top(v) \in \mathcal{F}_{RPO}$.*

To ease the reading of the paper we define $C(s)$ for some term s with $top(s) \in \mathcal{F}_{POLO}$ as $C(\mathcal{A}cc(s))$.

Note that $C(s)$ can be infinite. For this reason, in practice, when comparing a pair of terms s and t we only generate the part of $C(s)$ that is needed. This part is chosen by inspecting t.

Now we can define $\sqsupseteq_{RPOLO}$, $\succ_{RPOLO}$ and $\succeq_{RPOLO}$.

Definition 3. *$s \sqsupseteq_{RPOLO} t$ iff*

1. *$s = t \in \mathcal{X}$, or*
2. *$s = f(s_1, \ldots, s_n)$ and*
   (a) *$f \in \mathcal{F}_{POLO}$, $I(s) \geq_{C(s)} I(t)$ or*
   (b) *$t = g(t_1, \ldots, t_n)$, $f, g \in \mathcal{F}_{RPO}$, $f =_{\mathcal{F}} g$ and*
      i. *$stat(f) = mul$ and $\{s_1, \ldots, s_n\}(\sqsupseteq_{RPOLO})_{mon}\{t_1, \ldots, t_n\}$, or*
      ii. *$stat(f) = lex$ and $\langle s_1, \ldots, s_n\rangle(\sqsupseteq_{RPOLO})_{mon}\langle t_1, \ldots, t_n\rangle$.*

Definition 4. *$s = f(s_1, \ldots, s_n) \succ_{RPOLO} t$ iff*

1. *$f \in \mathcal{F}_{POLO}$ and $I(s) >_{C(s)} I(t)$, or*

2. *$f \in \mathcal{F}_{RPO}$, and*
   (a) *$s_i \succeq_{RPOLO} t$ for some $i \in \{1, \ldots, n\}$, or*
   (b) *$t = g(t_1, \ldots, t_m)$, $g \in \mathcal{F}_{POLO}$ and $s \succ_{RPOLO} u$ for all $u \in \mathcal{A}cc(t)$, or*
   (c) *$t = g(t_1, \ldots, t_m)$, $g \in \mathcal{F}_{RPO}$ and*
      i. *$f \succ_{\mathcal{F}} g$ and $s \succ_{RPOLO} t_i$ for all $i \in \{1, \ldots, m\}$, or*
      ii. *$f =_{\mathcal{F}} g$, $stat(f) = mul$ and $\{s_1, \ldots, s_n\}(\succ_{RPOLO})_{mul}\{t_1, \ldots, t_m\}$, or*
      iii. *$f =_{\mathcal{F}} g$, $stat(f) = lex$, $\langle s_1, \ldots, s_n\rangle(\succ_{RPOLO})_{lex}\langle t_1, \ldots, t_m\rangle$ and $s \succ_{RPOLO} t_i$ for all $i \in \{1, \ldots, m\}$,*

where $s \succeq_{RPOLO} t$ iff $s \succ_{RPOLO} t$ or $s \sqsupseteq_{RPOLO} t$.

We show that $\succ_{RPOLO}$ and $\sqsupseteq_{RPOLO}$ are well-defined by induction on the pair $\langle s, t\rangle$ comparing lexicographically the sizes of the terms. It is easy to see that all recursive calls in the definition of $\succ_{RPOLO}$ and $\sqsupseteq_{RPOLO}$ decrease in this ordering. Moreover, in the definition of the context $C(\mathcal{A}cc(s))$ all calls to $\succ_{RPOLO}$ and $\sqsupseteq_{RPOLO}$ are of the form $u \succ_{RPOLO} v$ or $u \sqsupseteq_{RPOLO} v$ where $|s| > |u|$, because $top(s) \in \mathcal{F}_{POLO}$ and hence all $u \in \mathcal{A}cc(s)$ are proper subterms of s.

Now, we provide some examples of comparisons between terms that are included in our ordering and are neither included in RPO nor in POLO, i.e., using (linear) integer polynomial interpretations. In fact, since we consider constraints including both strict and non-strict literals, what we show is that they are included in the pair $(\succ_{RPOLO}, \succeq_{RPOLO})$.

Example 1. Consider the following constraint consisting of three literals:

$$\begin{array}{ll} H(f(g(g(x)),y),x) & > H(f(g(y),x),f(g(y),x)) \\ H(x,g(y)) & \geq H(y,x) \\ f(g(x),y) & \geq f(y,x) \end{array}$$

The first literal cannot be proved by RPO since $f(g(g(x)),y)$ cannot be proved larger than $f(g(y),x)$ as no argument of the former is greater than $g(y)$. The constraint cannot be proved by an integer polynomial interpretation either. In order to explain this claim, let us consider $I(H(x,y)) = a_{H_0} + a_{H_1} \cdot x + a_{H_2} \cdot y$, $I(f(x,y)) = a_{f_0} + a_{f_1} \cdot x + a_{f_2} \cdot y$ and $I(g(x)) = a_{g_0} + a_{g_1} \cdot x$. First note that the combination of the three literals requires that all non-constant coefficients must be strictly greater than 0, and then, due to the occurrences of the variable y in the first literal we have that $a_{H_1} \cdot a_{f_2} \geq a_{H_1} \cdot a_{f_1} \cdot a_{g_1} + a_{H_2} \cdot a_{f_1} \cdot a_{g_1}$ and due to the occurrences of the variable x in the third literal $a_{f_1} \cdot a_{g_1} \geq a_{f_2}$ is required which altogether is a contradiction.

Let us prove it using RPOLO. We take $H \in \mathcal{F}_{RPO}$ with $stat(H) = mul$ and $f, g \in \mathcal{F}_{POLO}$ with $f_I(x,y) = x + y$ and $g_I(x) = x + 1$.

For the first literal we apply case 4.2(c)ii, and then $\{f(g(g(x)),y), x\}(\succ_{RPOLO})_{mul}\{f(g(y),x), f(g(y),x)\}$ is needed, which holds since $f(g(g(x)),y) \succ_{RPOLO} f(g(y),x)$ by case 4.1 as $I(f(g(g(x)),y)) = x_x + x_y + 2 > x_x + x_y + 1 = I(f(g(y),x))$.
The proof of the other two literals reuses part of the previous argument. □

Let us now show an example where we need symbols in $\mathcal{F}_{RPO}$ occurring below symbols that need to be in $\mathcal{F}_{POLO}$. Moreover, in this example a non-trivial use of the context is also necessary.

Example 2. Consider the following constraint coming from a termination proof:

$$\begin{array}{ll} f(0,x) & \geq x \\ f(s(x),y) & \geq s(f(x,f(x,y))) \\ H(s(f(s(x),y)),z) & > H(s(z),s(f(x,y))) \end{array}$$

The third literal needs H and s to be in $\mathcal{F}_{POLO}$. To hint this fact, note that we cannot remove s and, in that case, no argument in $H(s(f(s(x),y)),z)$ can be greater than or equal to $s(z)$. On the other hand, since due to the third literal, s cannot be removed and needs a non-zero coefficient for its argument, there is no polynomial interpretation for f fulfilling the first two literals, i.e., f must be in $\mathcal{F}_{RPO}$.

Therefore, we take $H, s \in \mathcal{F}_{POLO}$ with $H_I(x,y) = x + y$ and $s_I(x) = x + 1$, and $f \in \mathcal{F}_{RPO}$ with $stat(f) = lex$.

The first literal holds by case 4.2a. For the second one, $f(s(x),y) \succ_{RPOLO} s(f(x,f(x,y)))$ is proved by applying case 4.2b which requires $f(s(x),y) \succ_{RPOLO} f(x,f(x,y))$. We apply then case 4.2(c)iii, showing $s(x) \succ_{RPOLO} x$, by case 4.1, since $I(s(x)) = x_x+1 > x_x = I(x)$, and $f(s(x),y) \succ_{RPOLO} x$ and $f(s(x),y) \succ_{RPOLO} f(x,y)$ for the arguments. The first one holds by applying cases 4.2a and 4.1 consecutively, and the second one by case 4.2(c)iii as before.

Finally, for the third literal we apply case 4.1, since

$$x_{f(s(x),y)} + x_z + 1 \rightarrow_{\{x_{f(s(x),y)} \geq x_{f(x,y)}+2\}} x_{f(x,y)} + 2 + x_z + 1 > x_z + x_{f(x,y)} + 2$$

Note that $x_{f(s(x),y)} \geq x_{f(x,y)} + 2$ belongs to the context of $H(s(f(s(x),y)),z)$ since $f(s(x),y) \succ_{RPOLO} f(x,y)$. $\square$

Let us mention that, although in the previous example we have used the context, in all non crafted examples we have tried the context is not used (see Section 8). However, the context is still necessary, since otherwise we can prove neither stability under substitutions nor (weak) monotonicity.

The rest of this section is devoted to proving that $(\succeq_{RPOLO}, \succ_{RPOLO})$ is a reduction pair.

Lemma 2. *The relations $\sqsupseteq_{RPOLO}$ and $\succeq_{RPOLO}$ are reflexive.*

Proof. $s \sqsupseteq_{RPOLO} s$ easily follows by induction on $|s|$ and case analysis on the definition of $\sqsupseteq_{RPOLO}$. Then reflexivity of $\succeq_{RPOLO}$ directly follows by definition. $\square$

The following four lemmas provide some necessary results on subterms, being the first three on accessible terms.

Lemma 3. *Let s and w be terms in $\mathcal{T}(\mathcal{F},\mathcal{X})$. Then*

(i) $s \succeq_{RPOLO} w$ implies $s \succeq_{RPOLO} u$ for all $u \in \mathcal{A}cc(w)$.

(ii) $s \succ_{\scriptscriptstyle RPOLO} w$ *implies* $s \succ_{\scriptscriptstyle RPOLO} u$ *for all* $u \in \mathcal{A}cc(w)$.

See the proof of Lemma 19 which extends this lemma to the higher-order case.

Lemma 4. *Let* s, u *and* v *be terms. If* $u \in \mathcal{A}cc(s)$ *and* $u \succ_{\scriptscriptstyle RPOLO} v$ *then* $x_u \geq I(v) + 1$ *is in* $C(s)$.

Proof. By Lemma 3, $u \succ_{\scriptscriptstyle RPOLO} v$ implies $u \succ_{\scriptscriptstyle RPOLO} w$ for all $w \in \mathcal{A}cc(v)$. Therefore, by applying case 1 of the definition of $C(s)$ we have that $x_u \geq E + 1$ for all linear polynomial E over variables x_w such that $w \in \mathcal{A}cc(v)$. Therefore, as $I(v)$ is one of such polynomials, we have that $x_u \geq I(v) + 1$ is in $C(s)$. $\square$

The proof of the following lemma is very similar to the one of Lemma 21, which is an adaptation to what is needed in the higher-order case.

Lemma 5. *Let* s *and* t *be terms. Then* $s \succeq_{\scriptscriptstyle RPOLO} t$ *implies* $\forall u \in \mathcal{A}cc(t)$, $\exists v \in \mathcal{A}cc(s)$ *such that* $v \succeq_{\scriptscriptstyle RPOLO} u$.

As in the RPO we have the following property for terms headed by a symbol in $\mathcal{F}_{RPO}$.

Lemma 6. *Let* $t = g(t_1, \ldots, t_n)$ *with* $g \in \mathcal{F}_{RPO}$.
$s \succeq_{\scriptscriptstyle RPOLO} t$ *implies* $s \succ_{\scriptscriptstyle RPOLO} t_i$ *for all* $i \in \{1, \ldots, n\}$.

Proof. We proceed by induction on $|s| + |t|$. The cases with $top(s) \in \mathcal{F}_{RPO}$ are standard. Notice that having $s \succ_{\scriptscriptstyle RPOLO} t$ by case 4.2b is not possible since $top(t) \in \mathcal{F}_{RPO}$. Consider now $top(s) \in \mathcal{F}_{POLO}$. Then, as $top(t) \in \mathcal{F}_{RPO}$, we have that $\mathcal{A}cc(t) = \{t\}$ and hence, by Lemma 5, there is some $u \in \mathcal{A}cc(s)$ s.t. $u \succeq_{\scriptscriptstyle RPOLO} t$. By induction hypothesis, $u \succeq_{\scriptscriptstyle RPOLO} t$ implies $u \succ_{\scriptscriptstyle RPOLO} t_i$, and hence, by Lemma 4, $x_u \geq I(t_i) + 1 \in C(s)$. This implies $I(s) = p + x_u \rightarrow_{C(s)} p + I(t_i) + 1 > I(t_i)$ for some polynomial p, and hence we have $I(s) >_{C(s)} I(t_i)$ which let us conclude that $s \succ_{\scriptscriptstyle RPOLO} t_i$ holds by case 4.1. $\square$

The following lemma proves some compatibility and transitivity properties for $\succ_{\scriptscriptstyle RPOLO}$ and $\sqsupseteq_{\scriptscriptstyle RPOLO}$, which are trivially extended to $\succeq_{\scriptscriptstyle RPOLO}$.

Lemma 7. *Let* s, t *and* u *be terms.*

1. *Let* $top(s), top(t) \in \mathcal{F}_{POLO}$. *If* $s \succeq_{\scriptscriptstyle RPOLO} t$ *and* $I(s) = p \twoheadrightarrow^{=}_{C(s)} q \twoheadrightarrow^{=}_{C(t)} r$ *then* $p \twoheadrightarrow^{=}_{C(s)} r$.
2. *If* $s \sqsupseteq_{\scriptscriptstyle RPOLO} t \succ_{\scriptscriptstyle RPOLO} u$ *then* $s \succ_{\scriptscriptstyle RPOLO} u$.
3. *If* $s \succ_{\scriptscriptstyle RPOLO} t \sqsupseteq_{\scriptscriptstyle RPOLO} u$ *then* $s \succ_{\scriptscriptstyle RPOLO} u$.
4. *If* $s \sqsupseteq_{\scriptscriptstyle RPOLO} t \sqsupseteq_{\scriptscriptstyle RPOLO} u$ *then* $s \sqsupseteq_{\scriptscriptstyle RPOLO} u$.
5. *If* $s \succ_{\scriptscriptstyle RPOLO} t \succ_{\scriptscriptstyle RPOLO} u$ *then* $s \succ_{\scriptscriptstyle RPOLO} u$.

Proof. We prove all four properties by induction on $\langle |s|, |t| \rangle$ compared lexicographically. For cases from 2 to 5, we use a second induction on $|u|$.

1. We proceed by a second induction on the number of steps with $\rightarrow_{C(t)}$ in $q \twoheadrightarrow^{=}_{C(t)} r$. If there are no steps then it trivially holds. Otherwise we have $q \rightarrow_{\{x_u \geq E\}} q' \twoheadrightarrow^{=}_{C(t)} r$ for some $x_u \geq E$ in $C(t)$. If the occurrence of x_u has not been introduced in the parallel rewriting step $p \twoheadrightarrow^{=}_{C(s)} q$ then we have that $u \in \mathcal{A}cc(s)$, $x_u \geq E$ also occurs in $C(s)$ and the step with $\rightarrow_{\{x_u \geq E\}}$ can also be performed in parallel with $p \twoheadrightarrow^{=}_{C(s)} q$, giving $p \twoheadrightarrow^{=}_{C(s)} q'$, and hence we conclude by the inner induction. Otherwise, x_u has been introduced in $p \twoheadrightarrow^{=}_{C(s)} q$. Let x_v be the variable that has been rewritten in the step introducing x_u. Therefore, we have some $x_v \geq E'$ in $C(s)$ with x_u occurring in E'. We distinguish four cases:
   - If $E' = x_u$ and $E = x_w$ for some term w then we have $v \sqsupseteq_{RPOLO} u \sqsupseteq_{RPOLO} w$ and, by the outer induction hypothesis 4, $v \sqsupseteq_{RPOLO} w$. Therefore $x_v \geq x_w$ is in $C(s)$ and thus we can replace the step using $x_v \geq x_u$ by a step using $x_v \geq x_w$ and we have $p \twoheadrightarrow^{=}_{C(s)} q'$, and conclude by the inner induction.
   - If $E' = x_u$ and $u \succ_{RPOLO} w$ for all x_w occurring in E then we have $v \sqsupseteq_{RPOLO} u \succ_{RPOLO} w$ and, by the outer induction hypothesis 2, $v \succ_{RPOLO} w$. Therefore, $x_v \geq E$ is in $C(s)$ and thus, as before, we can replace the step using $x_v \geq x_u$ by a step using $x_v \geq E$ obtaining $p \twoheadrightarrow^{=}_{C(s)} q'$ which allows us to conclude by the inner induction.
   - If $E' = E'' + x_u$ with $E'' \neq 0$ and $E = x_w$ for some term w then we have $v \succ_{RPOLO} u \sqsupseteq_{RPOLO} w$ and, by the outer induction hypothesis 3, $v \succ_{RPOLO} w$. Moreover, since $x_v \geq E'' + x_u \in C(s)$, by definition, for all x_{z_i} occurring in E'' we have $v \succ_{RPOLO} z_i$. Therefore, we have that $x_v \geq E'' + x_w$ is in $C(s)$, since $E'' + x_w$ is a linear polynomial over $\mathbb{Z}^+$ and variables $\{x_{z_1}, \ldots, x_{z_n}, x_w\}$, and thus, we conclude as in the previous cases, replacing the step using $x_v \geq E'' + x_u$ by a step using $x_v \geq E'' + x_w$.
   - If $E' = E'' + x_u$ with $E'' \neq 0$ and $u \succ_{RPOLO} w$ for all x_w occurring in E then we have $v \succ_{RPOLO} u \succ_{RPOLO} w$ and, by the outer induction hypothesis 5, $v \succ_{RPOLO} w$ for all x_w occurring in E. Therefore, since, as in the previous case, we also have $v \succ_{RPOLO} z$ for all x_z occurring in E'', we have again that $x_v \geq E'' + E$ is in $C(s)$ and thus, we conclude replacing the step as in the previous cases.
2. We distinguish cases according to $s \sqsupseteq_{RPOLO} t$. Since $t \succ_{RPOLO} u$ we have that neither s nor t are variables.
   (a) If $s \sqsupseteq_{RPOLO} t$ by case 3.2a and $top(t) \in \mathcal{F}_{POLO}$ then we have that $t \succ_{RPOLO} u$ holds necessarily by case 4.1. Thus, we have $I(s) \geq_{C(s)} I(t) >_{C(t)} I(u)$, which implies, $I(s) \twoheadrightarrow^{=}_{C(s)} p_s \geq I(t) \twoheadrightarrow^{=}_{C(t)} p_t > I(u)$ for some polynomials p_s and p_t, and, by Corollary 1, $I(s) \twoheadrightarrow^{=}_{C(s)} I(t)+p' \twoheadrightarrow^{=}_{C(t)} I(u) + q' + 1 + p'$ for some polynomials p' and q'. Now, by Case 7.1, we have $I(s) \twoheadrightarrow^{=}_{C(s)} I(u) + q' + 1 + p'$, and hence $I(s) >_{C(s)} I(u)$, which implies $s \succ_{RPOLO} u$ by case 4.1.
   (b) If $s \sqsupseteq_{RPOLO} t$ by case 3.2a and $top(t) \in \mathcal{F}_{RPO}$ then we have $\mathcal{A}cc(t) = \{t\}$ and thus $I(t) = x_t$. By Lemma 5, there is some $v \in \mathcal{A}cc(s)$ such that

$v \succeq_{RPOLO} t$. Now, by induction hypothesis 2 and 5, we have $v \succ_{RPOLO} u$ and hence, by Lemma 4, $x_v \geq I(u)+1$ belongs to $C(s)$. Then $I(s) = x_v + p_s \twoheadrightarrow^{=}_{C(s)} I(u)+1+p_s > I(u)$ holds, and we conclude by case 4.1.

(c) If $s \sqsupseteq_{RPOLO} t$ by case 3.2b, then we have that $t = f(t_1, \ldots, t_n)$ with $f \in \mathcal{F}_{RPO}$. We distinguish the cases according to $t \succ_{RPOLO} u$.

i. If case 4.2a applies then $t_i \succeq_{RPOLO} u$ for some i and, by Lemma 6, we have $s \succ_{RPOLO} t_i$. Thus, we conclude $s \succ_{RPOLO} u$ by the induction hypothesis 3 and 5.

ii. If case 4.2b applies then $t \succ_{RPOLO} w$ for all $w \in \mathcal{A}cc(u)$. By the induction hypothesis 2, we have $s \succ_{RPOLO} w$ and hence $s \succ_{RPOLO} u$ by case 4.2b.

iii. If one of the cases 4.2c applies then we have $s = f(\overline{s})$, $t = g(\overline{t})$ and $u = h(\overline{u})$, with $f, g, h \in \mathcal{F}_{RPO}$ and $f =_{\mathcal{F}} g \succeq_{\mathcal{F}} h$. First, if we are in case 4.2(c)i or 4.2(c)iii since $t \succ_{RPOLO} u'$ for all $u' \in \overline{u}$ by induction hypothesis 2 we have $s \succ_{RPOLO} u'$. Now, if $g \succ_{\mathcal{F}} h$, we conclude by case 4.2(c)i. Otherwise, $f =_{\mathcal{F}} g =_{\mathcal{F}} h$ and either $stat(f) = stat(g) = mul$ or $stat(f) = stat(g) = lex$. In the first case $\{\overline{s}\}(\sqsupseteq_{RPOLO})_{mon}\{\overline{t}\}(\succ_{RPOLO})_{mul}\{\overline{u}\}$, and by induction hypothesis 2 and 4, we have $\{\overline{s}\}(\succ_{RPOLO})_{mul}\{\overline{u}\}$, and hence $s \succ_{RPOLO} u$ by case 4.2(c)ii. In the second case $\langle\overline{s}\rangle(\sqsupseteq_{RPOLO})_{mon}\langle\overline{t}\rangle(\succ_{RPOLO})_{lex}\langle\overline{u}\rangle$, and, by induction hypothesis 2 and 4, we have $\langle\overline{s}\rangle(\succ_{RPOLO})_{lex}\langle\overline{u}\rangle$, and hence $s \succ_{RPOLO} u$ by case 4.2(c)iii.

3. We distinguish the cases according to $s \succ_{RPOLO} t$.

(a) If $s \succ_{RPOLO} t$ by case 4.1 and $top(t) \in \mathcal{F}_{POLO}$ then we have that $t \sqsupseteq_{RPOLO} u$ holds necessarily by case 3.2a. Thus, we have $I(s) >_{C(s)} I(t) \geq_{C(t)} I(u)$, which implies $I(s) \twoheadrightarrow^{=}_{C(s)} p_s > I(t) \twoheadrightarrow^{=}_{C(t)} p_t \geq I(u)$ for some polynomials p_s and p_t, and, by Corollary 1, $I(s) \twoheadrightarrow^{=}_{C(s)} I(t) + p' + 1 \twoheadrightarrow^{=}_{C(t)} I(u) + q' + p' + 1$ for some p' and q'. Now, by Case 7.1, we have $I(s) \twoheadrightarrow^{=}_{C(s)} I(u) + q' + p' + 1$, and hence $I(s) >_{C(s)} I(u)$, which implies $s \succ_{RPOLO} u$ by case 4.1.

(b) If $s \succ_{RPOLO} t$ by case 4.1 and $top(t) \in \mathcal{F}_{RPO}$ or t is a variable then $top(u) \in \mathcal{F}_{RPO}$ or u is a variable as well. Therefore, we have $\mathcal{A}cc(t) = \{t\}$ and $\mathcal{A}cc(u) = \{u\}$, and thus $I(t) = x_t$ and $I(u) = x_u$. By Lemma 5, there is some $v \in \mathcal{A}cc(s)$ such that $v \succeq_{RPOLO} t$. Now, if $v \succ_{RPOLO} t$ then, by induction hypothesis 3, we have $v \succ_{RPOLO} u$ and hence, by Lemma 4, $x_v \geq I(u)+1$ belongs to $C(s)$. Then $I(s) = x_v + p_s \twoheadrightarrow^{=}_{C(s)} I(u)+1+p_s > I(u)$ holds. Otherwise, if $v \sqsupseteq_{RPOLO} t$ then, by induction hypothesis 4, we have $v \sqsupseteq_{RPOLO} u$, and thus both $x_v \geq x_t$ and $x_v \geq x_u$ belong to $C(s)$. Therefore, since $I(s) >_{C(s)} I(t) = x_t$, by Lemma 1, we have that $I(s) \twoheadrightarrow^{=}_{C(s)} x_t + 1 + p_s > x_t = I(t)$. There are two cases.

If x_t has not been introduced by a rewriting step then we have that $t \in \mathcal{A}cc(s)$ and $x_t \geq x_u$ belongs to $C(s)$, and hence we can also rewrite x_t in the parallel step obtaining $I(s) \twoheadrightarrow^{=}_{C(s)} x_u + 1 + p_s > x_u = I(u)$, and we conclude by case 4.1. Otherwise, there must be a step replacing x_v by x_t in $I(s) = x_v + p_s \twoheadrightarrow^{=}_{C(s)} x_t + p'_s > x_t = I(t)$. Then, since

$x_v \geq x_u$ belongs to $C(s)$, we can replace x_v by x_u instead of x_t, and we obtain $I(s) = x_v + p_s \twoheadrightarrow^{=}_{C(s)} x_u + p'_s > x_u = I(u)$, and again $s \succ_{RPOLO} u$ by case 4.1.

(c) If $s \succ_{RPOLO} t$ by case 4.2a then $s_i \succeq_{RPOLO} t$ for some argument s_i of s. By induction hypothesis 3 and 4, we have $s_i \succeq_{RPOLO} u$ and hence we conclude $s \succ_{RPOLO} u$ by case 4.2a.

(d) If $s \succ_{RPOLO} t$ by case 4.2b then we have $s \succ_{RPOLO} w$ for all $w \in \mathcal{A}cc(t)$. By Lemma 5, for all $v \in \mathcal{A}cc(u)$ there is some $w \in \mathcal{A}cc(t)$ such that $w \succeq_{RPOLO} v$, and, therefore, by induction hypothesis 3 and 5, $s \succ_{RPOLO} v$ for all $v \in \mathcal{A}cc(u)$. Now, if $top(u) \notin \mathcal{F}_{POLO}$, we already have $s \succ_{RPOLO} u$, since $\mathcal{A}cc(u) = \{u\}$. Otherwise, we have $s \succ_{RPOLO} u$ by case 4.2b.

(e) If $s \succ_{RPOLO} t$ by one of the cases 4.2c then we have $s = f(\overline{s})$, $t = g(\overline{t})$ and $u = h(\overline{u})$, with $f, g, h \in \mathcal{F}_{RPO}$ and $f \succeq_{\mathcal{F}} g =_{\mathcal{F}} h$. First, since $t \succ_{RPOLO} u'$ for all $u' \in \overline{u}$ by Lemma 6, we have $s \succ_{RPOLO} u'$ by induction hypothesis 5. Now, if $f \succ_{\mathcal{F}} g$, we conclude by case 4.2(c)i. Otherwise, $f =_{\mathcal{F}} g =_{\mathcal{F}} h$ and either $stat(f) = stat(g) = mul$ or $stat(f) = stat(g) = lex$. In the first case $\{\overline{s}\}(\succ_{RPOLO})_{mul}\{\overline{t}\}(\sqsupseteq_{RPOLO})_{mon}\{\overline{u}\}$, and by induction hypothesis 3 and 4 $\{\overline{s}\}(\succ_{RPOLO})_{mul}\{\overline{u}\}$, and hence $s \succ_{RPOLO} u$ by case 4.2(c)ii. In the second case $\langle\overline{s}\rangle(\succ_{RPOLO})_{lex}\langle\overline{t}\rangle(\sqsupseteq_{RPOLO})_{mon}\langle\overline{u}\rangle$, and, by induction hypothesis 3 and 4, $\langle\overline{s}\rangle(\succ_{RPOLO})_{lex}\langle\overline{u}\rangle$, and hence $s \succ_{RPOLO} u$ by case 4.2(c)iii.

4. We distinguish the cases according to $s \sqsupseteq_{RPOLO} t$.

(a) If $s \in \mathcal{X}$ then by definition of $\sqsupseteq_{RPOLO}$ we have $s = t = u \in \mathcal{X}$ and hence, $s \sqsupseteq_{RPOLO} u$ by case 3.1.

(b) If $s \sqsupseteq_{RPOLO} t$ and $t \sqsupseteq_{RPOLO} u$ by case 3.2a then we have $I(s) \geq_{C(s)} I(t) \geq_{C(t)} I(u)$, which implies, $I(s) \twoheadrightarrow^{=}_{C(s)} p_s \geq I(t) \twoheadrightarrow^{=}_{C(t)} p_t \geq I(u)$ for some polynomials p_s and p_t, and, by Corollary 1, $I(s) \twoheadrightarrow^{=}_{C(s)} I(t) + p' \twoheadrightarrow^{=}_{C(t)} I(u) + q' + p'$. Now, by Case 7.1, we have $I(s) \twoheadrightarrow^{=}_{C(s)} I(u) + q' + p'$, and hence $I(s) \geq_{C(s)} I(u)$, which implies $s \sqsupseteq_{RPOLO} u$ by case 3.2a.

(c) If $s \sqsupseteq_{RPOLO} t$ by case 3.2a and $top(t) \in \mathcal{F}_{RPO}$ or t is a variable, then, $top(u) \in \mathcal{F}_{RPO}$ or u is a variable. By Lemma 5, we have that there is some $v \in \mathcal{A}cc(s)$ such that $v \succeq_{RPOLO} t$, since $\mathcal{A}cc(t) = \{t\}$, and, by the induction hypothesis 3 and 4, $v \succeq_{RPOLO} u$. Therefore, since $\mathcal{A}cc(u) = \{u\}$, by definition $x_v \geq x_u$ belongs to $C(s)$, and hence $I(s) = x_v + p_s \twoheadrightarrow^{=}_{C(s)} x_u + p_s \geq x_u = I(u)$ and, hence $s \sqsupseteq_{RPOLO} u$ by case 3.2a.

(d) If $s \sqsupseteq_{RPOLO} t$ and $t \sqsupseteq_{RPOLO} u$ by case 3.2b then $s = f(\overline{s})$, $t = g(\overline{t})$ and $u = h(\overline{u})$, with $f =_{\mathcal{F}} g =_{\mathcal{F}} h$ and either $stat(f) = stat(g) = mul$ or $stat(f) = stat(g) = lex$. In the first case $\{\overline{s}\}(\sqsupseteq_{RPOLO})_{mon}\{\overline{t}\}(\sqsupseteq_{RPOLO})_{mon}\{\overline{u}\}$, and by induction hypothesis 4 $\{\overline{s}\}(\sqsupseteq_{RPOLO})_{mon}\{\overline{u}\}$, and hence $s \sqsupseteq_{RPOLO} u$ by case 3.2(b)i. In the second case $\langle\overline{s}\rangle(\sqsupseteq_{RPOLO})_{mon}\langle\overline{t}\rangle(\sqsupseteq_{RPOLO})_{mon}\langle\overline{u}\rangle$, and, by induction hypothesis 4, $\langle\overline{s}\rangle(\sqsupseteq_{RPOLO})_{mon}\langle\overline{u}\rangle$, and hence $s \sqsupseteq_{RPOLO} u$ by case 3.2(b)ii.

5. We distinguish cases according to $s \succ_{RPOLO} t$ and $t \succ_{RPOLO} u$. Notice that neither s nor t are variables.

(a) If $s \succ_{RPOLO} t$ by case 4.1 and $top(t) \in \mathcal{F}_{POLO}$ then we have that $t \succ_{RPOLO} u$ holds necessarily by case 4.1. Thus, we have $I(s) >_{C(s)} I(t) >_{C(t)}$

$I(u)$, which implies, $I(s) \twoheadrightarrow^{=}_{C(s)} p_s > I(t) \twoheadrightarrow^{=}_{C(t)} p_t > I(u)$ for some polynomials p_s and p_t, and, by Corollary 1, $I(s) \twoheadrightarrow^{=}_{C(s)} I(t) + p' + 1 \twoheadrightarrow^{=}_{C(t)} I(u) + q' + 1 + p' + 1$. Now, by Case 7.1, we have $I(s) \twoheadrightarrow^{=}_{C(s)} I(u) + q' + p' + 2$, and hence $I(s) >_{C(s)} I(u)$, which implies $s \succ_{RPOLO} u$ by case 4.1.

(b) If $s \succ_{RPOLO} t$ by case 4.1 and $top(t) \in \mathcal{F}_{RPO}$ then it holds like in case 2b.

(c) If $s \succ_{RPOLO} t$ by case 4.2a then it holds like case 3c.

(d) If $t \succ_{RPOLO} u$ by case 4.2a then it holds like case 2(c)i.

(e) If $s \succ_{RPOLO} t$ by case 4.2b then it holds like case 3d.

(f) If $t \succ_{RPOLO} u$ by case 4.2b and $top(s) \in \mathcal{F}_{RPO}$ then it holds like 2(c)ii.

(g) If we apply one of the cases 4.2c then we have $s = f(\overline{s})$, $t = g(\overline{t})$ and $u = h(\overline{u})$, with $f, g, h \in \mathcal{F}_{RPO}$ and $f \succeq_{\mathcal{F}} g \succeq_{\mathcal{F}} h$. First, since $t \succ_{RPOLO} u'$ for all $u' \in \overline{u}$ by Lemma 6, we have $s \succ_{RPOLO} u'$ by induction hypothesis 5. Now, if $f \succ_{\mathcal{F}} g$ or $g \succ_{\mathcal{F}} h$, we conclude by case 4.2(c)i. Otherwise, $f =_{\mathcal{F}} g =_{\mathcal{F}} h$ and either $stat(f) = stat(g) = mul$ or $stat(f) = stat(g) = lex$. In the first case $\{\overline{s}\}(\succ_{RPOLO})_{mul}\{\overline{t}\}(\succ_{RPOLO})_{mul}\{\overline{u}\}$, and by induction hypothesis 2,3,4 and 5, $\{\overline{s}\}(\succ_{RPOLO})_{mul}\{\overline{u}\}$, and hence $s \succ_{RPOLO} u$ by case 4.2(c)ii. In the second case $\langle\overline{s}\rangle(\succ_{RPOLO})_{lex}\langle\overline{t}\rangle(\succ_{RPOLO})_{lex}\langle\overline{u}\rangle$, and, by induction hypothesis 2,3,4 and 5, $\langle\overline{s}\rangle(\succ_{RPOLO})_{lex}\langle\overline{u}\rangle$, and hence $s \succ_{RPOLO} u$ by case 4.2(c)iii. □

Corollary 2. *$\succ_{RPOLO}$ and $\succeq_{RPOLO}$ are transitive and $\succ_{RPOLO}$ is compatible with $\succeq_{RPOLO}$.*

Now we provide the lemmas needed for proving the monotonicity and stability under substitutions.

Lemma 8. *If $s \succeq_{RPOLO} t$ then $I(f(\dots, s, \dots)) \geq_{C(f(\dots,s,\dots))} I(f(\dots, t, \dots))$, for all function symbols $f \in \mathcal{F}_{POLO}$.*

Lemma 9. *$\sqsupseteq_{RPOLO}$ and $\succeq_{RPOLO}$ are monotonic.*

See respectively the proofs of lemmas 22 and 23 which extend these lemmas to the higher-order case.

In order to prove stability under substitutions, we extend the definition of polynomial interpretation to substitutions in the natural way.

Definition 5. *Let γ be a substitution. Then we define*

$$\gamma_I = \{x_t \mapsto I(t\gamma) \mid t \in \mathcal{T}(\mathcal{F}, \mathcal{X})\}.$$

The following lemma easily holds by induction on the size of the term (see Lemma 24 for the details).

Lemma 10. *Let γ be a substitution such that $x_t \notin \mathcal{D}om(\gamma)$ for any term t. Then $I(s\gamma) = I(s)\gamma_I$ for every term s.*

Using the previous lemma we can show stability under substitutions of our relations (see the proof of Lemma 25 for details).

Lemma 11. *$\sqsupseteq_{RPOLO}$ and $\succ_{RPOLO}$ are stable under substitutions.*

Corollary 3. *$\succ_{RPOLO}$ and $\succeq_{RPOLO}$ are stable under substitutions.*

The following lemmas are devoted to proving well-foundedness of the ordering. The first one roughly shows that $>_C$ is well-founded on linear polynomials if the context C is built using a set of strongly normalizing terms (the proof is exactly as the one for Lemma 28).

Lemma 12. *Let $\mathcal{S}$ be a set of terms closed w.r.t. $\succeq_{RPOLO}$.*

1. *If p and q are linear polynomials, $\mathcal{V}ar(p) \subseteq \mathcal{X}_{\mathcal{S}}$ and $p >_{C(\mathcal{S})} q$ then $\mathcal{V}ar(q) \subseteq \mathcal{X}_{\mathcal{S}}$.*
2. *If all terms in $\mathcal{S}$ are strongly normalizing w.r.t. $\succ_{RPOLO}$ then $>_{C(\mathcal{S})}$ is well-founded on linear polynomials over $\mathbb{Z}^+$ and $\mathcal{X}_{\mathcal{S}}$.*

The following two lemmas show roughly that POLO symbols and RPO symbols preserve strong normalization. In the first case we assume that the accessible terms are strongly normalizing, while in the second case we assume that the arguments are strongly normalizing.

Lemma 13. *Let t be a term with $top(t) \in \mathcal{F}_{POLO}$ s.t. $\mathcal{A}cc(t) = \{u_1, \ldots, u_k\}$.*

If $u_1, \ldots, u_k$ are strongly normalizing w.r.t. $\succ_{RPOLO}$ then t is strongly normalizing w.r.t. $\succ_{RPOLO}$.

Lemma 14. *Let $t_1, \ldots t_n$ be terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$.*

If $t_1, \ldots, t_n$ are strongly normalizing w.r.t. $\succ_{RPOLO}$ and $f \in \mathcal{F}_{RPO}$ then $f(t_1, \ldots, t_n)$ is strongly normalizing w.r.t. $\succ_{RPOLO}$.

See the proofs of lemmas 29 and 30, which extend these lemmas to the higher-order case.

Using the previous lemmas we can prove well-foundedness of the ordering (see the proof of Lemma 31 which is the extension of this lemma to the higher-order case).

Lemma 15. *$\succ_{RPOLO}$ is well-founded.*

We conclude the Section with the main Theorem of the first-order part, which follows directly from corollaries 2 and 3 and lemmas 9 and 15.

Theorem 1. *$(\succeq_{RPOLO}, \succ_{RPOLO})$ is a reduction pair.*

As we have seen, in order to form a reduction pair we just need the quasi-ordering to be monotonic. However, if we want to use the ordering for directly proving termination then we need a reduction ordering and, hence, monotonicity of the strict ordering is required. In this case, to make $\succ_{RPOLO}$ monotonic there are two possibilities.

The first and simplest one relies on considering only monotonic polynomial interpretations, i.e. interpretations $f_I(x_1, \ldots, x_n) = c_0 + c_1 \cdot x_1 + \ldots + c_n \cdot x_n$

where $c_j > 0$ for all $j \in \{1 \ldots n\}$ and for all $f \in \mathcal{F}_{POLO}$. Then, if $s \succ_{RPOLO} t$ by case 4.1 then since $top(s) \in \mathcal{F}_{POLO}$ we have that $C(s) \subseteq C(f(\ldots s \ldots))$, and hence $I(s) >_{C(s)} I(t)$ implies $I(f(\ldots s \ldots)) >_{C(f(\ldots s \ldots))} I(f(\ldots t \ldots))$, which implies $f(\ldots s \ldots) \succ_{RPOLO} f(\ldots t \ldots)$ by case 4.1. Otherwise, if $s \succ_{RPOLO} t$ by case 4.2b then, since $top(s) \in \mathcal{F}_{RPO}$ and $top(t) \in \mathcal{F}_{POLO}$, we have, by Lemma 4, $x_s \geq I(t) + 1 \in C(f(\ldots, s, \ldots))$, and since $I(f(\ldots s \ldots)) = \ldots + x_s + \ldots$ and $I(f(\ldots t \ldots)) = \ldots + I(t) + \ldots$, we can conclude by case 4.1, again. Finally, the rest of the cases are proved as in the recursive path ordering.

The second solution is based on splitting the case 4.1 of RPOLO (where $top(s) \in \mathcal{F}_{POLO}$) in two cases. The first one is as before, i.e. requires $I(s) >_{C(s)} I(t)$, and the second one requires $I(s) \geq_{C(s)} I(t)$ and $\mathcal{A}cc(s)(\succ_{RPOLO})_{mul}\mathcal{A}cc(t)$. Now, it is easy to prove monotonicity, but well-foundedness needs to be revised. To this end we just need to consider a second component (combined lexicographically) in the induction argument of the proof of Lemma 13. Being precise, we have to proceed by induction on the pair $\langle I(t), \mathcal{A}cc(t)\rangle$ w.r.t. the ordering $(>_C, (\succ_{RPOLO})_{mul})_{lex}$, which is well-founded since, by assumption, all terms in $\mathcal{A}cc(t)$ are strongly normalizing w.r.t. $\succ_{RPOLO}$. This is very similar to what is necessary for proving Lemma 14. Since these proofs are omitted, see the proofs of Lemma 29 and Lemma 30 which respectively extend both lemmas to the higher-order case.

4 Higher-order terms and orderings

Given a set $\mathcal{S}$ of *sort symbols*, the set of *types* is the usual set of simple types generated by the constructor $\rightarrow$ for *functional types*:

$$\mathcal{T}_{\mathcal{S}} := s \in \mathcal{S} \mid \mathcal{T}_{\mathcal{S}} \rightarrow \mathcal{T}_{\mathcal{S}}$$

Types are *functional* when headed by the $\rightarrow$ symbol, and *data types* otherwise. The operator $\rightarrow$ associates to the right. We use $\sigma, \tau, \rho, \theta$ for arbitrary types.

Function symbols are meant to be algebraic operators equipped with a fixed number n of arguments (called the *arity*) of respective types $\sigma_1, \ldots, \sigma_n$, and an *output type* σ. Let $\mathcal{F} = \biguplus_{\sigma_1,\ldots,\sigma_n,\sigma} \mathcal{F}_{\sigma_1 \times \ldots \times \sigma_n \rightarrow \sigma}$. The membership of a given function symbol f to $\mathcal{F}_{\sigma_1 \times \ldots \times \sigma_n \rightarrow \sigma}$ is called a *type declaration* and written $f : [\sigma_1 \times \ldots \times \sigma_n] \rightarrow \sigma$.

The set $\mathcal{T}(\mathcal{F}, \mathcal{X})$ of *raw algebraic λ-terms* is generated from the signature $\Sigma = (\mathcal{S}, \mathcal{F})$ and a denumerable set $\mathcal{X}$ of variables according to the grammar:

$$\mathcal{T} := \mathcal{X} \mid (\lambda\mathcal{X} : \mathcal{T}_{\mathcal{S}}.\mathcal{T}) \mid @(\mathcal{T}, \mathcal{T}) \mid \mathcal{F}(\mathcal{T}, \ldots, \mathcal{T}).$$

Raw terms of the form $\lambda x : \sigma.u$ are called *abstraction*, while the other raw terms are said to be *neutral*. $@(u, v)$ denotes the *application* of u to v. We may sometimes omit the type σ in $\lambda x : \sigma.u$. As a matter of convenience, we may write $@(u, v_1, \ldots, v_n)$ for $@(\ldots @(u, v_1), \ldots, v_n)$, assuming $n > 0$. The raw term $@(u, v_1, \ldots, v_n)$ is called a (partial) left-flattening of $s = @(\ldots @(u, v_1), \ldots, v_n)$,

u being possibly an application itself (hence the word 'partial'). We use $\mathcal{V}ar(t)$ for the set of free variables of t.

An *environment* Γ is a finite set of pairs written as $\{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$, where x_i is a variable, σ_i is a type, and $x_i \neq x_j$ for $i \neq j$. $\mathcal{V}ar(\Gamma) = \{x_1, \ldots, x_n\}$ is the set of variables of Γ. Given two environments Γ and Γ', their composition, denoted by $\Gamma \cdot \Gamma'$, is the environment $\Gamma' \cup \{x : \sigma \in \Gamma \mid x \notin \mathcal{V}ar(\Gamma')\}$. Two environments Γ and Γ' are compatible if $\Gamma \cdot \Gamma' = \Gamma \cup \Gamma'$.

Given a signature Σ, our typing judgements are written as $\Gamma \ \vdash_\Sigma s : \sigma$. A raw term s has type σ in the environment Γ if the judgement $\Gamma \ \vdash_\Sigma s : \sigma$ is provable in the inference system given at Figure 1. An important property of our type system is that a raw term typable in a given environment has a unique type. Typable raw terms are called *terms*.

Variables:

$$\frac{x : \sigma \in \Gamma}{\Gamma \ \vdash_\Sigma x : \sigma}$$

Functions:

$$\frac{f : [\sigma_1 \times \ldots \times \sigma_n] \to \sigma \in \mathcal{F} \quad \Gamma \ \vdash_\Sigma t_1 : \sigma_1 \ \ldots \ \Gamma \ \vdash_\Sigma t_n : \sigma_n}{\Gamma \ \vdash_\Sigma f(t_1, \ldots, t_n) : \sigma}$$

Abstraction:

$$\frac{\Gamma \cdot \{x : \sigma\} \ \vdash_\Sigma t : \tau}{\Gamma \ \vdash_\Sigma (\lambda x : \sigma.t) : \sigma \to \tau}$$

Application:

$$\frac{\Gamma \ \vdash_\Sigma s : \sigma \to \tau \qquad \Gamma \ \vdash_\Sigma t : \sigma}{\Gamma \ \vdash_\Sigma @(s, t) : \tau}$$

Fig. 1. The type system for monomorphic higher-order algebras

A *substitution* γ of domain $\mathcal{D}om(\gamma) = \{x_1, \ldots, x_n\}$ is a set of triples $\gamma = \{\Gamma_1 \ \vdash_\Sigma x_1 \mapsto t_1, \ldots, \Gamma_n \ \vdash_\Sigma x_n \mapsto t_n\}$, such that x_i and t_i have the same type in the environment Γ_i. Substitutions are extended to terms by morphism, variable capture being avoided by renaming bound variables when necessary. We use post-fixed notation for substitution application.

Given a signature Σ, a *higher-order rewrite rule* is a triple written $\Gamma \ \vdash_\Sigma l \to r$ where Γ is an environment and l, r are higher-order terms such that $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l) \subseteq \mathcal{V}ar(\Gamma)$ and for all substitutions γ such that $\Gamma \ \vdash_\Sigma l\gamma : \sigma$, then $\Gamma \ \vdash_\Sigma r\gamma : \sigma$. A *higher-order rewrite system* is a set of higher-order rewrite rules.

The rewrite relation which is considered in the following sections is the union of the one induced by a set of higher-order rewrite rules and the β- and η-reduction relations all working modulo α-conversion, i.e.,

$$\to_{\mathcal{R}\beta\eta} = \to_\mathcal{R} \cup \to_\beta \cup \to_\eta$$

where

$$\begin{array}{l} \{u : \alpha, v : \beta\} \ \vdash_\Sigma @(\lambda x : \alpha.v, u) \to_\beta \ v\{x \mapsto u\} \\ \{x : \alpha, u : \alpha \to \beta\} \ \vdash_\Sigma \lambda x.@(u, x) \to_\eta \ u \quad \text{if } x \notin \mathcal{V}ar(u) \end{array}$$

For simplicity reasons, typing environments are omitted in the rest of the paper, except when presenting type inference rules.

For some proofs we will use an extension of the subterm relation $\rhd$ based on α-conversion, which is denoted by $\rhd_\alpha$ and defined as $u \rhd_\alpha v$ if $u \rhd v$ or $u = \lambda x.u'$ and $v = u'\{x \mapsto y\}$ for some fresh variable y. We have that $\rhd_\alpha$ is well-founded since $u \rhd_\alpha v$ implies $|u| > |v|$. Moreover, since $\rhd_\alpha$ is compatible with β-reduction, we have $\rightarrow_\beta \cup \rhd_\alpha$ is well-founded on typed terms.

Definition 6. *A* higher-order reduction ordering $\succ$ *is a well-founded ordering which is*

(i) monotonic: $s : \sigma \succ t : \sigma$ *implies* $u[s] : \tau \succ u[t] : \tau$ *for all* $u[x : \sigma] : \tau$*;*
(ii) stable: $s : \sigma \succ t : \sigma$ *implies* $s\gamma : \sigma \succ t\gamma : \sigma$ *for every substitution* γ*;*
(iii) functional: $s : \sigma \rightarrow_\beta \cup \rightarrow_\eta t : \sigma$ *implies* $s : \sigma \succ t : \sigma$*.*

Higher-order reduction orderings allow us to show that the relation $\rightarrow_{\mathcal{R}\beta\eta}$ is terminating by simply comparing the left-hand and right-hand sides of each rule in $\mathcal{R}$.

Theorem 2. *Let* $\mathcal{R} = \{l_i \rightarrow r_i\}_{i \in I}$ *be a higher-order rewrite system and* $\succ$ *be a higher-order reduction ordering such that* $l_i \succ r_i$ *for every* $i \in I$*. Then the relation* $\rightarrow_{\mathcal{R}\beta\eta}$ *is strongly normalizing.*

As in the first-order case, methods like the ones in [7,24,19] require higher-order reduction pairs to solve the constraints generated by the method.

Definition 7. *Given* $\succeq$ *and* $\succ$*,* $(\succeq, \succ)$ *is a* higher-order reduction pair *if* $\succeq$ *is a quasi-ordering which is monotonic, stable under substitutions and functional and* $\succ$ *is a well-founded ordering which is stable under substitutions, functional and compatible with* $\succeq$*.*

In the following section we will provide a higher-order reduction pair by extending the results of Section 3.

5 The Higher-Order Recursive Path and Polynomial Ordering

We have, as for the first-order case, $\mathcal{F}$ split into two sets $\mathcal{F}_{POLO}$ and $\mathcal{F}_{RPO}$, a precedence $\succeq_\mathcal{F}$ on $\mathcal{F}_{RPO}$ and a polynomial interpretation over the natural numbers I on terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. For any symbol f whose output type is functional, we have $f \in \mathcal{F}_{RPO}$ (i.e., for any term $f(s_1 : \sigma_1, \ldots, s_n : \sigma_n) : \rho \rightarrow \tau$, $f \in \mathcal{F}_{RPO}$).

The interpretation I is enlarged as follows. If t has a functional type then $I(t) = 0$ and otherwise if the top symbol of t is @ then $I(t) = x_t$. In any other case we proceed as before. However, to be able to handle the introduced variables x_t, while keeping monotonicity of beta-reduction, we need to add context information recording the relation between x_t and t.

$$I(s) = \begin{cases} 0 & \text{if } s:\sigma \text{ and } \sigma \text{ is functional} \\ x_s & \text{if } s = @(s_1,\ldots,s_n):\sigma \text{ and } \sigma \text{ is a data type} \\ f_I(I(s_1),\ldots,I(s_n)) & \text{if } s = f(s_1,\ldots,s_n):\sigma,\ \sigma \text{ is a data type, } f \in \mathcal{F}_{POLO} \\ x_s & \text{if } s = f(s_1,\ldots,s_n):\sigma,\ \sigma \text{ is a data type, } f \in \mathcal{F}_{RPO} \\ x_s & \text{if } s = x:\sigma \in \mathcal{X}, \text{ and } \sigma \text{ is a data type} \end{cases}$$

Given a term s, its set of accessible terms $\mathcal{A}cc(s)$ is defined identically as in the first-order case. Remark that for all terms $u:\tau \in \mathcal{A}cc(s)$, τ is a data type and either $top(u) \in \mathcal{F}_{RPO} \cup \{@\}$ or u is a variable.

Now we define a quasi-ordering on types exactly as done in [2], which is also an ingredient of the ordering:

Definition 8. *Let $\geq_T$ be a quasi-ordering on types satisfying the following properties:*

1. *Well-foundedness: $>_T^{\rightarrow} = >_T \cup \rhd_{\rightarrow}$ is well-founded, where $\sigma \rightarrow \tau \rhd_{\rightarrow} \sigma$;*
2. *Right arrow subterm: $\sigma \rightarrow \tau >_T \tau$;*
3. *Arrow preservation: $\tau \rightarrow \sigma =_T \alpha$ iff $\alpha = \tau' \rightarrow \sigma'$, $\tau =_T \tau'$ and $\sigma =_T \sigma'$;*
4. *Arrow decreasingness: $\tau \rightarrow \sigma >_T \alpha$ implies either $\sigma \geq_T \alpha$ or $\alpha = \tau' \rightarrow \sigma'$, $\tau =_T \tau'$ and $\sigma >_T \sigma'$;*

where $>_T$ is its strict part and $=_T$ its equivalence.

Remark 1. The type ordering $\geq_T$ and the polynomial interpretation I have to satisfy the following condition. For all $f : [\sigma_1 \times \ldots \times \sigma_n] \rightarrow \sigma \in \mathcal{F}_{POLO}$, and for all $i \in \{1,\ldots,n\}$, if $\forall x_1 \ldots x_n : I(f(x_1,\ldots,x_n)) \geq x_i$ then $\sigma \geq_T \sigma_i$.

Using this condition we can infer the following crucial property.

Property 1. Let $t:\sigma$ and $w:\tau$ be typed terms. If $w \in \mathcal{A}cc(t)$ then $\sigma \geq_T \tau$.

Proof. By induction on $|t|$. Since $w \in \mathcal{A}cc(t)$, we have that τ is a data type. If $top(t) \in \mathcal{F}_{RPO} \cup \{@\}$ or t is a variable then it trivially holds since $\mathcal{A}cc(t) = \{t\}$. Otherwise, $t = f(t_1,\ldots,t_n)$ and $f : [\sigma_1 \times \ldots \times \sigma_n] \rightarrow \sigma \in \mathcal{F}_{POLO}$, and moreover $w \in \mathcal{A}cc(t_i)$ for some $i \in \{1,\ldots,n\}$ such that $I(f(x_1,\ldots,x_n)) \geq x_i$. Therefore, by Remark 1, we have $\sigma \geq_T \sigma_i$ and since, by induction hypothesis, $\sigma_i \geq_T \tau$ we conclude $\sigma \geq_T \tau$. □

The following three mutually recursive definitions introduce respectively the context $C(\mathcal{S})$ of a set of terms $\mathcal{S}$, and the relations $\sqsupseteq_{HORPOLO}$, $\succ_{HORPOLO}$ and $\succeq_{HORPOLO}$.

We give first the definition of context as in the first-order case, but using $\succ_{HORPOLO}$ instead of $\succ_{RPOLO}$. Note that beta-reduction is included in the definition of $\succ_{HORPOLO}$ and hence it is also considered in the definition of context.

Definition 9. *Let $\mathcal{S}$ be a set of terms $u:\sigma$ such that σ is a data type and $top(u) \notin \mathcal{F}_{POLO}$. The context $C(\mathcal{S})$ is defined as the union of*

1. $x_u \geq E+1$ *for all* $u \in \mathcal{S}$ *and for all linear polynomial expressions* E *over* $\mathbb{Z}^+$ *and variables* $\{x_{v_1}, \ldots, x_{v_n}\}$ *such that* $u \succ_{\scriptscriptstyle HORPOLO} v_i$ *for all* $i \in \{1, \ldots, n\}$.
2. $x_u \geq x_v$ *for all* $u \in \mathcal{S}$ *and for all* v *such that* $u \sqsupseteq_{\scriptscriptstyle HORPOLO} v$ *and* $top(v) \in \mathcal{F}_{RPO} \cup \{@\}$.

As before, we define $C(t)$ for some term t with $top(t) \in \mathcal{F}_{POLO}$ as $C(\mathcal{A}cc(t))$. Now we can define the relations $\sqsupseteq_{\scriptscriptstyle HORPOLO}$, $\succ_{\scriptscriptstyle HORPOLO}$ and $\succeq_{\scriptscriptstyle HORPOLO}$.

Definition 10. $s : \sigma \sqsupseteq_{\scriptscriptstyle HORPOLO} t : \tau$ *iff*

1. $s = t \in \mathcal{X}$, *or*
2. $s = f(s_1, \ldots, s_n)$ *and*
   (a) $f \in \mathcal{F}_{POLO}$, $\sigma \geq_T \tau$ *with* τ *a data type and* $I(s) \geq_{C(s)} I(t)$, *or*
   (b) $t = g(t_1, \ldots, t_n)$, $f, g \in \mathcal{F}_{RPO}$, $f =_{\mathcal{F}} g$, $\sigma =_T \tau$ *and*
      i. $stat(f) = mul$ *and* $\{s_1, \ldots, s_n\}(\sqsupseteq_{\scriptscriptstyle HORPOLO})_{mon}\{t_1, \ldots, t_n\}$, *or*
      ii. $stat(f) = lex$ *and* $\langle s_1, \ldots, s_n\rangle(\sqsupseteq_{\scriptscriptstyle HORPOLO})_{mon}\langle t_1, \ldots, t_n\rangle$
3. $s = @(u, v)$, $t = @(u', v')$, $\{u, v\}(\sqsupseteq_{\scriptscriptstyle HORPOLO})_{mon}\{u', v'\}$ *and* $\sigma =_T \tau$
4. $s = \lambda x : \alpha.u$ *and* $t = \lambda y : \beta.v$ *with* $\alpha =_T \beta$, $\sigma =_T \tau$ *and* $u\{x \mapsto z\} \sqsupseteq_{\scriptscriptstyle HORPOLO} v\{y \mapsto z\}$ *for some fresh variable* $z : \alpha$.

Definition 11. $s : \sigma \succ_{\scriptscriptstyle HORPOLO} t : \tau$ *iff* $\sigma \geq_T \tau$ *and*

1. $s = f(s_1, \ldots, s_n)$ *and*
   (a) $f \in \mathcal{F}_{POLO}$ *and*
      i. τ *is a data type and* $I(s) >_{C(s)} I(t)$, *or*
      ii. τ *is functional and* $u \succ_{\scriptscriptstyle HORPOLO} t$ *for some* $u \in \mathcal{A}cc(s)$, *or*
   (b) $f \in \mathcal{F}_{RPO}$ *and*
      i. $s_i \succeq_{\scriptscriptstyle HORPOLO} t$ *for some* $i \in \{1, \ldots, n\}$, *or*
      ii. $t = g(t_1, \ldots, t_m)$, $g \in \mathcal{F}_{POLO}$ *and* $s \succ_{\scriptscriptstyle HORPOLO} u$ *for all* $u \in \mathcal{A}cc(t)$, *or*
      iii. $t = g(t_1, \ldots, t_m)$, $g \in \mathcal{F}_{RPO}$,
         A. $f \succ_{\mathcal{F}} g$ *and for all* $i \in \{1, \ldots, m\}$ *we have* $s \succ_{\scriptscriptstyle HORPOLO} t_i$ *or* $s_j \succeq_{\scriptscriptstyle HORPOLO} t_i$ *for some* $j \in \{1, \ldots, n\}$, *or*
         B. $f =_{\mathcal{F}} g$, $stat(f) = mul$ *and* $\{s_1, \ldots, s_n\}(\succ_{\scriptscriptstyle HORPOLO})_{mul}\{t_1, \ldots, t_m\}$, *or*
         C. $f =_{\mathcal{F}} g$, $stat(f) = lex$, $\langle s_1, \ldots, s_n\rangle(\succ_{\scriptscriptstyle HORPOLO})_{lex}\langle t_1, \ldots, t_m\rangle$ *and for all* $i \in \{1, \ldots, m\}$ *we have* $s \succ_{\scriptscriptstyle HORPOLO} t_i$ *or* $s_j \succeq_{\scriptscriptstyle HORPOLO} t_i$ *for some* $j \in \{1, \ldots, n\}$, *or*
      iv. $t = @(t_1 \ldots, t_m)$ *for some partial left-flattening of* t *and for all* $i \in \{1, \ldots, m\}$ *we have* $s \succ_{\scriptscriptstyle HORPOLO} t_i$ *or* $s_j \succeq_{\scriptscriptstyle HORPOLO} t_i$ *for some* $j \in \{1, \ldots, n\}$, *or*
2. $s = @(s_1, s_2)$ *and either*
   (a) $s_1 \succeq_{\scriptscriptstyle HORPOLO} t$ *or* $s_2 \succeq_{\scriptscriptstyle HORPOLO} t$, *or*
   (b) $t = @(t_1, \ldots, t_m)$ *for some partial left-flattening of* t *and* $\{s_1, s_2\}(\succ_{\scriptscriptstyle HORPOLO})_{mul}\{t_1, \ldots, t_m\}$, *or*
3. $s = @(\lambda x.w, v)$ *and* $w\{x \mapsto v\} \succeq_{\scriptscriptstyle HORPOLO} t$, *or*

4. $s = \lambda x : \alpha.u$ and
   (a) $u\{x \mapsto z\} \succeq_{HORPOLO} t$ for some fresh variable $z : \alpha$, or
   (b) $t = \lambda y : \beta.v$, $\alpha =_T \beta$ and $u\{x \mapsto z\} \succ_{HORPOLO} v\{y \mapsto z\}$ for some fresh variable $z : \alpha$, or
   (c) $u = @(w, x)$, $x \notin \mathcal{V}ar(w)$ and $w \succeq_{HORPOLO} t$,

where $s : \sigma \succeq_{HORPOLO} t : \tau$ iff $s : \sigma \succ_{HORPOLO} t : \tau$ or $s : \sigma \sqsupseteq_{HORPOLO} t : \tau$.

In order to show that $\succ_{HORPOLO}$ and $\sqsupseteq_{HORPOLO}$ are well-defined we proceed like in the first order case but by induction on the pair $\langle s, t\rangle$ compared lexicographically with $(\rightarrow_\beta \cup \rhd_\alpha, \rhd)_{lex}$.

Let us show the use of the defined relations in a simple example. Some larger examples will be given in Section 7.

Example 3. Let o be a data type, $\mathcal{F} = \{s : [o] \to o, p : [o \times o] \to o, r : [o \times (o \to o \to o)] \to o\}$ and $\mathcal{X} = \{x : o, y : o, F : o \to o \to o\}$.

Consider the following constraint literal:

$$r(p(s(s(x)), y), F) > @(F, y, r(p(s(y), x), F))$$

First of all, let us mention that HORPO fails to prove this literal since we need $p(s(s(x)), y)$ to be greater than $p(s(y), x)$, which cannot happen in any RPO-like ordering. However, it is proved with HORPOLO taking $r \in \mathcal{F}_{RPO}$ with $stat(r) = mul$ and $p, s \in \mathcal{F}_{POLO}$ with $p_I(x, y) = x + y$ and $s_I(x) = x + 1$.
By case 11.1(b)iv, since F is already an argument of $r(p(s(s(x)), y), F)$, we just need to show that $r(p(s(s(x)), y), F) \succ_{HORPOLO} y$ and $r(p(s(s(x)), y), F) \succ_{HORPOLO} r(p(s(y), x), F)$. For the former we first apply case 11.1(b)i and then case 11.1(a)i, since $I(p(s(s(x)), y)) = x_x + x_y + 2 > x_y = I(y)$. For the latter, we apply first case 11.1(b)iiiB, which requires $p(s(s(x)), y) \succ_{HORPOLO} p(s(y), x)$, that holds by case 11.1(a)i, since $I(p(s(s(x)), y)) = x_x + x_y + 2 > x_x + x_y + 1 = I(p(s(y), x))$. □

As done in the previous example, in practice we would like to use the defined relations $\succeq_{HORPOLO}$ and $\succ_{HORPOLO}$ to prove termination. However, the relation $\succ_{HORPOLO}$ cannot be proved transitive and moreover $\succeq_{HORPOLO}$ and $\succ_{HORPOLO}$ are not compatible, and hence $(\succeq_{HORPOLO}, \succ_{HORPOLO})$ is not a higher-order reduction pair. Fortunately, we can define two more relations $\geq_{HORPOLO}$ and $>_{HORPOLO}$, which extend respectively $\succeq_{HORPOLO}$ and $\succ_{HORPOLO}$, and such that $(\geq^*_{HORPOLO}, >^+_{HORPOLO})$ is a higher-order reduction pair. Note that, since $\succ_{HORPOLO}$ is included in $>_{HORPOLO}$ and thus in $>^+_{HORPOLO}$, we can use $\succ_{HORPOLO}$ in the termination proofs (and analogously for $\succeq_{HORPOLO}$ and $\geq^*_{HORPOLO}$).

Definition 12. *The relations $>_{HORPOLO}$ and $\geq_{HORPOLO}$ are defined as*

$$>_{HORPOLO} = \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO}$$
$$\geq_{HORPOLO} = \sqsupseteq^*_{HORPOLO} \cdot \succeq_{HORPOLO}$$

Notice that $\geq_{HORPOLO} = \sqsupseteq^*_{HORPOLO} \cup >_{HORPOLO}$, *since* $\succeq_{HORPOLO} = \sqsupseteq_{HORPOLO} \cup \succ_{HORPOLO}$ *implies* $\geq_{HORPOLO} = \sqsupseteq^*_{HORPOLO} \cdot (\sqsupseteq_{HORPOLO} \cup \succ_{HORPOLO}) = \sqsupseteq^+_{HORPOLO} \cup (\sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO}) = \sqsupseteq^+_{HORPOLO} \cup >_{HORPOLO} = \sqsupseteq^*_{HORPOLO} \cup >_{HORPOLO}$.

Moreover, $\geq^*_{HORPOLO}$ *is a quasi-ordering and* $>^+_{HORPOLO}$ *is an ordering.*

The rest of this section is devoted to proving that $(\geq^*_{HORPOLO}, >^+_{HORPOLO})$ is a higher-order reduction pair. To this end we will show the following properties.

- $>^+_{HORPOLO}$ is compatible with $\geq^*_{HORPOLO}$.
- $>_{HORPOLO}$ is strongly-normalizing.
- $\geq_{HORPOLO}$ is monotonic.
- both $>_{HORPOLO}$ and $\geq_{HORPOLO}$ are stable under substitutions.
- both $>_{HORPOLO}$ and $\geq_{HORPOLO}$ are functional.

The strong normalization proof proceeds like in [18] but uses ideas already introduced for the first-order case when handling polynomial interpretations.

5.1 Candidate terms

Our strong normalization proof is based on Tait and Girard's reducibility technique. In order to apply this method we have to define for each type σ, a set of terms called the computability predicate $[\![\sigma]\!]$. Terms in $[\![\sigma]\!]$ are called *computable*. In practice, $[\![\sigma]\!]$ can be defined by the properties it should satisfy. The most important one is that computable terms must be strongly normalizable. The rest of the proof is based on lemmas similar to the ones introduced in the first order case but working on computable terms instead of strongly normalizing terms. See [14] for a detailed exposition of the method in case of system F, and [11] for a discussion about the different possibilities for defining computability predicates in practice.

Since we work with an equivalence relation $=_T$ on types, the set $[\![\sigma]\!]$ is actually associated to the equivalence class of σ modulo $=_T$. Moreover, we need, for instance, that if $s \in [\![\sigma \to \tau]\!]$ and $t \in [\![\sigma]\!]$, then the raw term $@(s,t)$ must belong to the set $[\![\tau]\!]$ even if it is not typable, which may arise in case t does not have type σ but $\sigma' =_T \sigma$. Now we give a type system in which all raw terms needed in the strong normalization proof become typable candidate terms.

Definition 13. *A raw term* s *is a* candidate term *if the judgement* $\Gamma \vdash_\Sigma s :_\mathcal{C} \sigma$ *is provable in the type system of Figure 2.*

The set of types of a typable candidate term of type σ is a union of type equivalence classes modulo $=_T$:

Lemma 16. *For all* σ *and* τ *such that* $\sigma =_T \tau$, *we have* $\Gamma \vdash_\Sigma s :_\mathcal{C} \sigma$ *iff* $\Gamma \vdash_\Sigma s :_\mathcal{C} \tau$.

This allows us to talk about the types of a candidate term up to type equivalence.

Equivalence:
$$\frac{\Gamma \vdash_\Sigma s :_{\mathcal{C}} \sigma \quad \sigma =_T \tau}{\Gamma \vdash_\Sigma s :_{\mathcal{C}} \tau}$$

Variables:
$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash_\Sigma x :_{\mathcal{C}} \sigma}$$

Functions:
$$\frac{f : [\sigma_1 \times \ldots \times \sigma_n] \rightarrow \sigma \in \mathcal{F} \\ \Gamma \vdash_\Sigma t_1 :_{\mathcal{C}} \sigma_1 \ \ldots \ \Gamma \vdash_\Sigma t_n :_{\mathcal{C}} \sigma_n}{\Gamma \vdash_\Sigma f(t_1, \ldots, t_n) :_{\mathcal{C}} \sigma}$$

Abstraction:
$$\frac{\Gamma \cdot \{x : \sigma\} \vdash_\Sigma t :_{\mathcal{C}} \tau}{\Gamma \vdash_\Sigma (\lambda x : \sigma.t) :_{\mathcal{C}} \sigma \rightarrow \tau}$$

Application:
$$\frac{\Gamma \vdash_\Sigma s :_{\mathcal{C}} \sigma \rightarrow \tau \quad \Gamma \vdash_\Sigma t :_{\mathcal{C}} \sigma}{\Gamma \vdash_\Sigma @(s, t) :_{\mathcal{C}} \tau}$$

Fig. 2. The type system for Candidate Terms

5.2 Properties of the order

First of all, let us mention that, in this Section Definition 9, 10 and 11 are applied to candidate terms, and moreover, since the proof of strong normalization works on candidate terms and relies on all the following properties, we will prove them for candidate terms as well.

Lemma 17. *The ordering $>^+_{HORPOLO}$ is compatible with the quasi-ordering $\geq^*_{HORPOLO}$.*

Proof. We show that if $s \geq^*_{HORPOLO} t >^+_{HORPOLO} u$ then $s >^+_{HORPOLO} u$. We proceed by induction on the length of the sequence from s to t. If $s = t$ we are done. Otherwise, we have a sequence $s = s_0 \geq_{HORPOLO} s_1 \ldots \geq_{HORPOLO} s_n = t >^+_{HORPOLO} u$ with $n > 0$. Now, on the one hand, by induction hypothesis we have $s_1 >^+_{HORPOLO} u$. On the other hand, $\geq_{HORPOLO} = \sqsupseteq^*_{HORPOLO} \cup >_{HORPOLO}$. Thus, if $s_0 \geq_{HORPOLO} s_1$, then either $s_0 \sqsupseteq^*_{HORPOLO} s_1$ or $s_0 >_{HORPOLO} s_1$. If $s_0 \sqsupseteq^*_{HORPOLO} s_1$ then $s_0 \sqsupseteq^*_{HORPOLO} \cdot >^+_{HORPOLO} u$ and, since $>_{HORPOLO} = \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO}$, we have $s_0 >^+_{HORPOLO} u$. If, otherwise, $s_0 >_{HORPOLO} s_1$, then we trivially have $s_0 >^+_{HORPOLO} u$. Thus, we have proved that $s >^+_{HORPOLO} u$. □

In order to be able to prove most of the properties, some lemmas holding for first-order terms need to be generalized for higher-order terms.

Lemma 18 (Generalization of Lemma 2). *The relations $\sqsupseteq_{HORPOLO}$ and $\succeq_{HORPOLO}$ are reflexive.*

The proof holds trivially like in the first-order case.

Lemma 19 (Generalization of Lemma 3). *Let $s :_{\mathcal{C}} \sigma$ and $w :_{\mathcal{C}} \tau$ be candidate terms. Then*

(i) $s \succeq_{HORPOLO} w$ *implies* $s \succeq_{HORPOLO} u$ *for all* $u \in \mathcal{A}cc(w)$.
(ii) $s \succ_{HORPOLO} w$ *implies* $s \succ_{HORPOLO} u$ *for all* $u \in \mathcal{A}cc(w)$.

Proof. We proceed by induction on s w.r.t. $\rightarrow_\beta \cup \rhd_\alpha$. If τ is functional then $\mathcal{A}cc(w) = \varnothing$, and hence the lemma trivially holds. Otherwise, τ is a data type. If $top(w) \in \mathcal{F}_{RPO} \cup \{@\}$ or w is a variable, then $\mathcal{A}cc(w) = \{w\}$ and the lemma trivially holds as well. Otherwise, $top(w) \in \mathcal{F}_{POLO}$. Then, by definition, we have $\sigma \geq_T \tau$ and, by Property 1, for all $u :_\mathcal{C} \tau'$ in $\mathcal{A}cc(w)$ we have $\tau \geq_T \tau'$ and hence $\sigma \geq_T \tau'$.

If $s \sqsupseteq_{\scriptscriptstyle HORPOLO} w$ then, since $top(w) \in \mathcal{F}_{POLO}$, it can only be by case 10.2a. Then we have $I(s) \geq_{C(s)} I(w)$. By definition of I, for all $u :_\mathcal{C} \tau' \in \mathcal{A}cc(w)$, we have $I(w) = P + x_u$ for some polynomial P, which implies $I(s) \geq_{C(s)} I(u) = x_u$. Hence, since $\sigma \geq_T \tau'$ and all terms in $\mathcal{A}cc(w)$ have a data type, $s \sqsupseteq_{\scriptscriptstyle RPOLO} u$ holds by case 10.2a.

Otherwise, we have $s \succ_{\scriptscriptstyle HORPOLO} w$. Since $top(w) \in \mathcal{F}_{POLO}$, these are the only applicable cases.

- $s \succ_{\scriptscriptstyle HORPOLO} w$ holds by case 11.1(a)i. Then we have both $\sigma \geq_T \tau$ and $I(s) >_{C(s)} I(w)$. By definition of I, for all $u :_\mathcal{C} \tau' \in \mathcal{A}cc(w)$, we have $I(w) = P + x_u$ for some polynomial P, which implies $I(s) >_{C(s)} I(u) = x_u$ and, since all terms in $\mathcal{A}cc(w)$ have a data type, we conclude $s \succ_{\scriptscriptstyle RPOLO} u$ by case 11.1(a)i.
- $s \succ_{\scriptscriptstyle HORPOLO} w$ holds by case 11.1(b)i. Then $s_i \succeq_{\scriptscriptstyle HORPOLO} w$ for some argument s_i of s. By induction hypothesis, we have $s_i \succeq_{\scriptscriptstyle HORPOLO} u :_\mathcal{C} \tau'$ for all $u \in \mathcal{A}cc(w)$, and hence, since $\sigma \geq_T \tau'$, we conclude $s \succ_{\scriptscriptstyle HORPOLO} u :_\mathcal{C} \tau'$ by case 11.1(b)i for all $u \in \mathcal{A}cc(w)$.
- $s \succ_{\scriptscriptstyle HORPOLO} w$ holds by case 11.2a. The proof is analogous to the previous, using case 11.2a instead of case 11.1(b)i.
- $s \succ_{\scriptscriptstyle HORPOLO} w$ holds by case 11.1(b)ii. Then $s \succ_{\scriptscriptstyle HORPOLO} u$ for all $u \in \mathcal{A}cc(w)$ holds by definition.
- $s \succ_{\scriptscriptstyle HORPOLO} w$ holds by case 11.3. Then $s \rightarrow_\beta s' \succeq_{\scriptscriptstyle HORPOLO} w$ for some $s' :_\mathcal{C} \sigma$. Then, by induction hypothesis, $s' :_\mathcal{C} \sigma \succeq_{\scriptscriptstyle HORPOLO} u :_\mathcal{C} \tau'$ for all $u \in \mathcal{A}cc(w)$, and hence $s :_\mathcal{C} \sigma \succ_{\scriptscriptstyle HORPOLO} u :_\mathcal{C} \tau'$ by case 11.3 for all $u \in \mathcal{A}cc(w)$.
- $s \succ_{\scriptscriptstyle HORPOLO} w$ holds by case 11.4a. Then $s = \lambda x : \alpha.v$ and $v\{x \mapsto z\} \succeq_{\scriptscriptstyle HORPOLO} w$ for some fresh variable $z : \alpha$. Since $\lambda x : \alpha.v \rhd_\alpha v\{x \mapsto z\}$, by induction hypothesis $v\{x \mapsto z\} \succeq_{\scriptscriptstyle HORPOLO} u$ for all $u \in \mathcal{A}cc(w)$ and hence $s \succ_{\scriptscriptstyle HORPOLO} u$ by case 11.4a for all $u \in \mathcal{A}cc(w)$.
- $s \succ_{\scriptscriptstyle HORPOLO} w$ holds by case 11.4c. Then $s = \lambda x : \alpha.@(v, x)$, $x \notin \mathcal{V}ar(v)$ and $v \succeq_{\scriptscriptstyle HORPOLO} w$. By induction hypothesis $v \succeq_{\scriptscriptstyle HORPOLO} u$ for all $u \in \mathcal{A}cc(w)$ and hence $s \succ_{\scriptscriptstyle HORPOLO} u$ by case 11.4c for all $u \in \mathcal{A}cc(w)$. $\square$

The following lemma is adapted from Lemma 4, but using $\succ_{\scriptscriptstyle HORPOLO}$. The proof is analogous.

Lemma 20. *Let s, u and v be terms. If $u \in \mathcal{A}cc(s)$ and $u \succ_{\scriptscriptstyle HORPOLO} v$ then $x_u \geq I(v) + 1$ is in $C(s)$.*

The next lemma extends Lemma 5 to both $\succeq_{\scriptscriptstyle HORPOLO}$ and $\geq_{\scriptscriptstyle HORPOLO}$. The first one is used in Lemma 25 and the second one in Lemma 29.

Lemma 21 (Generalization of Lemma 5). *Let $s :_\mathcal{C} \sigma$ with σ a data type and $t :_\mathcal{C} \tau$ be candidate terms.*

- *If $s \succeq_{\scriptscriptstyle HORPOLO} t$ then $\forall u \in \mathcal{A}cc(t)$, $\exists v \in \mathcal{A}cc(s)$ such that $v \succeq_{\scriptscriptstyle HORPOLO} u$.*

– *If $s \geq_{\scriptscriptstyle HORPOLO} t$ then $\forall u \in \mathcal{A}cc(t)$, $\exists v \in \mathcal{A}cc(s)$ such that $v \geq^*_{\scriptscriptstyle HORPOLO} u$.*

Proof. For the first property we distinguish two cases according to the type of t.

If τ is functional then it trivially holds since $\mathcal{A}cc(t) = \varnothing$. Otherwise τ is a data type. Then if s is a variable we necessarily have $s = t$ and since $\mathcal{A}cc(s) = \{s\} = \mathcal{A}cc(t)$ it trivially holds. If $top(s) \in \mathcal{F}_{RPO} \cup \{@\}$ then $\mathcal{A}cc(s) = \{s\}$ and hence we conclude by Lemma 19.

Otherwise $top(s) \in \mathcal{F}_{POLO}$. Then whether $s \sqsupseteq_{\scriptscriptstyle HORPOLO} t$ or $s \succ_{\scriptscriptstyle HORPOLO} t$, we have that $I(s) \geq_{C(s)} I(t)$, and hence $I(s) \twoheadrightarrow^{=}_{C(s)} p \geq I(t)$ for some polynomial p. Note that, by definition, for all $x_u \geq E$ in $C(s)$ we have that $u \in \mathcal{A}cc(s)$. Moreover, for all terms $u \in \mathcal{A}cc(t)$ there is a variable x_u occurring in $I(t)$. The property trivially holds for those variables x_u occurring in $I(t)$ such that x_u also occurs in $I(s)$. For those variables x_u occurring in $I(t)$ such that x_u does not occur in $I(s)$ we will have a step with $\rightarrow_{C(s)}$ in $I(s) \twoheadrightarrow^{=}_{C(s)} p \geq I(t)$ introducing x_u. Therefore, there exists some variable x_v in $I(s)$ such that either $x_v \geq x_u$ is in $C(s)$ and x_v is replaced by x_u or $x_v \geq P + x_u + 1$ is in $C(s)$ for some polynomial P and x_v is replaced by $P + x_u + 1$. Now, we have that $v \in \mathcal{A}cc(s)$ and $v \sqsupseteq_{\scriptscriptstyle HORPOLO} u$ in the first case and $v \succ_{\scriptscriptstyle HORPOLO} u$ in the second one.

For the second property we proceed by induction on the number of steps with $\sqsupseteq_{\scriptscriptstyle HORPOLO}$ that we have in $s \geq_{\scriptscriptstyle HORPOLO} t$.

If there are no steps, it holds by the first property. Otherwise, we have $s \sqsupseteq_{\scriptscriptstyle HORPOLO} s' \geq_{\scriptscriptstyle HORPOLO} t$. Then, by induction we have that for all $u \in \mathcal{A}cc(t)$ there is some $v' \in \mathcal{A}cc(s')$ such that $v' \geq^*_{\scriptscriptstyle HORPOLO} u$, and by the first property we have that for all $v' \in \mathcal{A}cc(s')$ there is some $v \in \mathcal{A}cc(s)$ such that $v \succeq_{\scriptscriptstyle HORPOLO} v'$. Therefore for all $u \in \mathcal{A}cc(t)$ there is some $v \in \mathcal{A}cc(s)$ such that $v \geq^*_{\scriptscriptstyle HORPOLO} u$. □

The next lemma is used in the following one, which states the monotonicity of our non-strict relations on candidate terms.

Lemma 22 (Generalization of Lemma 8). *If $s :_{\mathcal{C}} \sigma \succeq_{\scriptscriptstyle HORPOLO} t :_{\mathcal{C}} \sigma$ then $I(f(\ldots,s,\ldots)) \geq_{C(f(\ldots,s,\ldots))} I(f(\ldots,t,\ldots))$, for all terms $f(\ldots,x:\sigma,\ldots)$ such that $f \in \mathcal{F}_{POLO}$*

Proof. If $I(f(\ldots,s,\ldots)) = I(f(\ldots,t,\ldots))$ then it trivially holds. Otherwise σ is not functional and we have $I(f(\ldots,s,\ldots)) = p + I(s)$ and $I(f(\ldots,t,\ldots)) = p + I(t)$ for some polynomial p, and $\mathcal{A}cc(s) \subseteq \mathcal{A}cc(f(\ldots,s,\ldots))$ which implies $C(s) \subseteq C(f(\ldots,s,\ldots))$.

Now, if $top(s) \in \mathcal{F}_{RPO} \cup \{@\}$ then $I(s) = x_s$, i.e., $\mathcal{A}cc(s) = \{s\}$. In this case $s \succeq_{\scriptscriptstyle HORPOLO} t$ gives us either $x_s \geq I(t) \in C(f(\ldots,s,\ldots))$ or, by Lemma 20, $x_s \geq I(t)+1 \in C(f(\ldots,s,\ldots))$, and hence, since $I(f(\ldots,s,\ldots)) = p + x_s \rightarrow_{\{x_s \geq I(t)\}} p + I(t) = I(f(\ldots,t,\ldots))$ and $I(f(\ldots,s,\ldots)) = p + x_s \rightarrow_{\{x_s \geq I(t)+1\}} p + I(t) + 1 > p + I(t) = I(f(\ldots,t,\ldots))$, we conclude $I(f(\ldots,s,\ldots)) \geq_{C(f(\ldots,s,\ldots))} I(f(\ldots,t,\ldots))$. If $top(s) \in \mathcal{F}_{POLO}$ then $s \succeq_{\scriptscriptstyle HORPOLO} t$ gives us $I(s) \geq_{C(s)} I(t)$ and since $C(s) \subseteq C(f(\ldots,s,\ldots))$ we have $I(f(\ldots,s,\ldots)) = p + I(s) \geq_{C(f(\ldots,s,\ldots))} p + I(t) = I(f(\ldots,t,\ldots))$. Therefore, we conclude $I(f(\ldots,s,\ldots)) \geq_{C(f(\ldots,s,\ldots))} I(f(\ldots,t,\ldots))$.

Finally, if s is a variable then necessarily $s \sqsupseteq_{\scriptscriptstyle HORPOLO} t$ by case 10.1, which implies $s = t$ and hence $I(f(\ldots,s,\ldots)) = I(f(\ldots,t,\ldots))$. □

Lemma 23 (Generalization of Lemma 9). *$\sqsupseteq_{HORPOLO}$ and $\succeq_{HORPOLO}$ are monotonic for candidate terms.*

Proof. We have to prove that $s :_{\mathcal{C}} \sigma \sqsupseteq_{HORPOLO} t :_{\mathcal{C}} \sigma$ implies $u[s] :_{\mathcal{C}} \tau \sqsupseteq_{HORPOLO} u[t] :_{\mathcal{C}} \tau$ and $s :_{\mathcal{C}} \sigma \succ_{HORPOLO} t :_{\mathcal{C}} \sigma$ implies $u[s] :_{\mathcal{C}} \tau \succeq_{HORPOLO} u[t] :_{\mathcal{C}} \tau$ for all $u[x :_{\mathcal{C}} \sigma] :_{\mathcal{C}} \tau$.

We proceed by induction on the size of u. If u is empty it trivially holds. For the induction step we have to prove the following properties.

- $s :_{\mathcal{C}} \sigma \sqsupseteq_{HORPOLO} t :_{\mathcal{C}} \sigma$ implies $f(\dots, s, \dots) :_{\mathcal{C}} \theta \sqsupseteq_{RPOLO} f(\dots, t, \dots) :_{\mathcal{C}} \theta$ and $s :_{\mathcal{C}} \sigma \succeq_{HORPOLO} t :_{\mathcal{C}} \sigma$ implies $f(\dots, s, \dots) :_{\mathcal{C}} \theta \succeq_{RPOLO} f(\dots, t, \dots) :_{\mathcal{C}} \theta$ for all function symbol f. If $f \in \mathcal{F}_{POLO}$, we apply Lemma 22 and case 10.2a. Otherwise, it holds as in RPO applying the case depending on the status of f.
- If $s :_{\mathcal{C}} \sigma \sqsupseteq_{HORPOLO} t :_{\mathcal{C}} \sigma$ then $@(s, v) :_{\mathcal{C}} \theta \sqsupseteq_{HORPOLO} @(t, v) :_{\mathcal{C}} \theta$ (respectively $@(v, s) :_{\mathcal{C}} \theta \sqsupseteq_{HORPOLO} @(v, t) :_{\mathcal{C}} \theta$), which holds by case 10.3.
- If $s :_{\mathcal{C}} \sigma \succ_{HORPOLO} t :_{\mathcal{C}} \sigma$ then $@(s, v) :_{\mathcal{C}} \theta \succ_{HORPOLO} @(t, v) :_{\mathcal{C}} \theta$ (respectively $@(v, s) :_{\mathcal{C}} \theta \succ_{HORPOLO} @(v, t) :_{\mathcal{C}} \theta$), which holds by case 11.2b.
- If $s :_{\mathcal{C}} \sigma \sqsupseteq_{HORPOLO} t :_{\mathcal{C}} \sigma$ then $\lambda y.s :_{\mathcal{C}} \theta \sqsupseteq_{HORPOLO} \lambda y.t :_{\mathcal{C}} \theta$, which holds by case 10.4.
- If $s :_{\mathcal{C}} \sigma \succ_{HORPOLO} t :_{\mathcal{C}} \sigma$ then $\lambda y.s :_{\mathcal{C}} \theta \succ_{HORPOLO} \lambda y.t :_{\mathcal{C}} \theta$, which holds by case 11.4b. □

Corollary 4. *$\geq_{HORPOLO}$ is monotonic and $>_{HORPOLO}$ is weakly monotonic for candidate terms.*

Now we proceed to prove stability under substitutions of both $\sqsupseteq_{HORPOLO}$ and $\succ_{HORPOLO}$, which also implies that $\succeq_{HORPOLO}$ is stable under substitutions.

Lemma 24 (Generalization of Lemma 10). *Let γ be a substitution such that $x_t \notin \mathcal{D}om(\gamma)$ for any term t. Then $I(s\gamma) = I(s)\gamma_I$ for every candidate term $s :_{\mathcal{C}} \tau$.*

Proof. By induction on $|s|$. If τ is functional then $I(s) = 0$ and, since substitutions preserve types, $I(s\gamma) = 0$ as well. Hence, in this case the lemma trivially holds. Otherwise, τ is a data type. If s is a variable or a term with $top(s) \in \mathcal{F}_{RPO} \cup \{@\}$ then $I(s) = x_s$ and hence $I(s)\gamma_I = I(s\gamma)$. Finally, if $s = f(s_1, \dots, s_n)$ with $f \in \mathcal{F}_{POLO}$ then $I(s) = f_I(I(s_1), \dots, I(s_n))$ and since, by induction hypothesis, $I(s_i\gamma) = I(s_i)\gamma_I$ for all $i \in \{1, \dots, n\}$, we have that $I(s\gamma) = f_I(I(s_1\gamma), \dots, I(s_n\gamma)) = f_I(I(s_1)\gamma_I, \dots, I(s_n)\gamma_I) = I(s)\gamma_I$. □

Lemma 25 (Generalization of Lemma 11). *The relations $\sqsupseteq_{HORPOLO}$ and $\succ_{HORPOLO}$ are stable under substitutions for candidate terms.*

Proof. Given $s :_{\mathcal{C}} \sigma$ and $t :_{\mathcal{C}} \tau$, we have to prove that $s \sqsupseteq_{HORPOLO} t$ implies $s\gamma \sqsupseteq_{HORPOLO} t\gamma$ and $s \succ_{HORPOLO} t$ implies $s\gamma \succ_{HORPOLO} t\gamma$, for every substitution γ s.t. $x_u \notin \mathcal{D}om(\gamma)$ for any term u, by induction on the pair $\langle s, t\rangle$ w.r.t. $(\rightarrow_\beta \cup \rhd_\alpha, \rhd)_{lex}$. Since substitutions preserve types, we will only consider the term comparisons of the ordering and omit all references to types. Notice that we can restrict the domain of the substitution in the indicated way, as we can assume that no variable of the form x_u occur in the terms. There are the following cases according to the definitions:

1. If $s \sqsupseteq_{HORPOLO} t$ by case 10.1 then $s = t \in \mathcal{X}$ and hence $s\gamma \sqsupseteq_{HORPOLO} t\gamma$ by reflexivity of $\sqsupseteq_{HORPOLO}$.
2. If $s \sqsupseteq_{HORPOLO} t$ by case 10.2a we must show that $I(s) \geq_{C(s)} I(t)$ implies $I(s\gamma) \geq_{C(s\gamma)} I(t\gamma)$. First we will show that if $x_u \geq E$ is in $C(s)$ then $x_{u\gamma} \geq E\gamma_I$ is in $C(s\gamma)$.
   - If $x_u \geq x_v$ is in $C(s)$ then we have $u \in \mathcal{A}cc(s)$, $u \sqsupseteq_{HORPOLO} v$ and $top(v) \in \mathcal{F}_{RPO} \cup \{@\}$. Hence we have that $top(u\gamma), top(v\gamma) \in \mathcal{F}_{RPO} \cup \{@\}$, $u\gamma \in \mathcal{A}cc(s\gamma)$ and, by induction hypothesis, $u\gamma \sqsupseteq_{HORPOLO} v\gamma$. Therefore we have that $x_{u\gamma} \geq x_{v\gamma}$ is in $C(s\gamma)$. Now, since $I(v\gamma) = x_{v\gamma}$ and, by Lemma 24, $I(v\gamma) = I(v)\gamma_I$, and $I(v) = x_v$, we conclude $x_{u\gamma} \geq x_v\gamma_I$ is in $C(s\gamma)$.
   - If $x_u \geq P + 1$ is in $C(s)$ then we have $u \in \mathcal{A}cc(s)$ and $u \succ_{HORPOLO} v$ for all labeled variables x_v in P. Hence, as $top(u) \in \mathcal{F}_{RPO} \cup \{@\}$ we have that $top(u\gamma) \in \mathcal{F}_{RPO} \cup \{@\}$ and $u\gamma \in \mathcal{A}cc(s\gamma)$. Moreover, by induction hypothesis, we have $u\gamma \succ_{HORPOLO} v\gamma$ for all labeled variables x_v in P, and, by Lemma 21, $u\gamma \succ_{HORPOLO} v\gamma$ implies $u\gamma \succ_{HORPOLO} w$ for all $w \in \mathcal{A}cc(v\gamma)$. Now, as all labeled variables in $P\gamma_I$ are either $x_{v\gamma}$ for some labeled variable x_v in P, or x_w for some $w \in \mathcal{A}cc(v\gamma)$ and labeled variable x_v in P, we conclude $x_{u\gamma} \geq P\gamma_I + 1$ is in $C(s\gamma)$.

   We have proved that if $x_u \geq E$ is in $C(s)$ then $x_{u\gamma} \geq E\gamma_I$ is in $C(s\gamma)$. Now we will prove that if $p + x_u \rightarrow_{\{x_u \geq E\}} p + E$ then $p\gamma_I + x_u\gamma_I \rightarrow_{\{x_{u\gamma} \geq E\gamma_I\}} p\gamma_I + E\gamma_I$. By Lemma 24, we have that $I(u\gamma) = I(u)\gamma_I$, and since $top(u) \in \mathcal{F}_{RPO} \cup \{@\}$, we have $I(u) = x_u$ and $I(u\gamma) = x_{u\gamma}$. Thus, we have that $x_{u\gamma} = x_u\gamma_I$ and hence, we conclude $p\gamma_I + x_u\gamma_I \rightarrow_{\{x_{u\gamma} \geq E\gamma_I\}} p\gamma_I + E\gamma_I$.

   Finally, we have that $I(s) \geq_{C(s)} I(t)$ implies $I(s) \twoheadrightarrow^{=}_{C(s)} p \geq I(t)$ for some polynomial p, and by Corollary 1, $p = q + I(t)$. Moreover, we have seen that $I(s) \twoheadrightarrow^{=}_{C(s)} q + I(t)$ implies that $I(s)\gamma_I \twoheadrightarrow^{=}_{C(s\gamma)} q\gamma_I + I(t)\gamma_I$ and hence, by lemma 24, we have $I(s\gamma) \twoheadrightarrow^{=}_{C(s\gamma)} q\gamma_I + I(t\gamma)$. Therefore, we conclude $I(s\gamma) \geq_{C(s\gamma)} I(t\gamma)$ and hence $s\gamma \sqsupseteq_{HORPOLO} t\gamma$ by case 10.2a.
3. If $s \sqsupseteq_{HORPOLO} t$ by case 10.2b, by case 10.3 or by case 10.4, we conclude by induction hypothesis and by each of the cases, respectively.
4. If $s \succ_{HORPOLO} t$ by case 11.1(a)i, the proof is analogous to that of $s \sqsupseteq_{HORPOLO} t$ by case 10.2a.
5. If $s \succ_{HORPOLO} t$ by case 11.1(a)ii, then there is some $u \in \mathcal{A}cc(s)$ such that $u \succ_{HORPOLO} t$. Then u cannot be a variable and hence $u\gamma \in \mathcal{A}cc(s\gamma)$. Therefore, since by induction hypothesis, $u\gamma \succ_{HORPOLO} t\gamma$ we conclude by case 11.1(a)ii.
6. If $s \succ_{HORPOLO} t$ by case 11.1(b)i, by case 11.1(b)iii, by case 11.1(b)iv, or by case 11.2, we conclude by induction hypothesis and by each of the subcases.
7. If $s \succ_{HORPOLO} t$ by case 11.1(b)ii, then we have $top(s) \in \mathcal{F}_{RPO}$, $top(t) \in \mathcal{F}_{POLO}$ and $s \succ_{HORPOLO} u$ for all $u \in \mathcal{A}cc(t)$. Therefore, we have that

$top(s\gamma) \in \mathcal{F}_{RPO}, top(t\gamma) \in \mathcal{F}_{POLO}$ and, by induction hypothesis, $s\gamma \succ_{HORPOLO} u\gamma$ for all $u \in \mathcal{A}cc(t)$. Now we will prove that $s\gamma \succ_{HORPOLO} v\gamma$ for all $v \in \mathcal{A}cc(t\gamma)$ which will allow us to conclude that $s\gamma \succ_{HORPOLO} t\gamma$ by case 11.1(b)ii. If $v \in \mathcal{A}cc(t\gamma)$ with $v = u\gamma$ for some $u \in \mathcal{A}cc(t)$ then since $s\gamma \succ_{HORPOLO} u\gamma$ for all $u \in \mathcal{A}cc(t)$ we have $s\gamma \succ_{HORPOLO} v$. Otherwise $v \in \mathcal{A}cc(t\gamma)$ with $v \in \mathcal{A}cc(x\gamma)$ for some variable $x \in \mathcal{A}cc(t)$ and hence we are done since $s\gamma \succ_{HORPOLO} x\gamma$, by Lemma 19, implies $s\gamma \succ_{HORPOLO} v$.

8. If $s \succ_{HORPOLO} t$ by case 11.3, then the property holds by induction hypothesis, stability of β-reduction, and case 11.3.
9. If $s \succ_{HORPOLO} t$ by case 11.4a, then $s = \lambda x.u$ and $u\{x \mapsto z\} \succeq_{HORPOLO} t$ for some fresh variable z. Let γ be a substitution of domain $\mathcal{V}ar(s) \cup \mathcal{V}ar(t)$. By induction hypothesis $u\{x \mapsto z\}\gamma \succ_{HORPOLO} t\gamma$. Then, since $\mathcal{D}om(\gamma)$ only contains free variables we have $x \notin \mathcal{D}om(\gamma)$, and since z is fresh, we have that $z \notin \mathcal{D}om(\gamma)$. Thus, we have $u\{x \mapsto z\}\gamma = u\gamma\{x \mapsto z\}$ which implies $u\gamma\{x \mapsto z\} \succ_{HORPOLO} t\gamma$ and hence, we conclude $s\gamma = \lambda x.u\gamma \succ_{HORPOLO} t\gamma$ by case 11.4a.
10. If $s \succ_{HORPOLO} t$ by case 11.4b, then $s = \lambda x.u$, $t = \lambda y.v$ and $u\{x \mapsto z\} \succ_{HORPOLO} v\{y \mapsto z\}$ for some fresh variable z. By induction hypothesis we have that $u\{x \mapsto z\}\gamma \succ_{HORPOLO} v\{y \mapsto z\}\gamma$, and assuming that $x, y, z \notin \mathcal{D}om(\gamma)$ (x and y are not free variables and z is fresh) then we have $u\{x \mapsto z\}\gamma = u\gamma\{x \mapsto z\}$ and $v\{y \mapsto z\}\gamma = v\gamma\{y \mapsto z\}$ which implies $u\gamma\{x \mapsto z\} \succ_{HORPOLO} v\gamma\{y \mapsto z\}$. Therefore, we conclude that $s\gamma = \lambda x.u\gamma \succ_{HORPOLO} \lambda y.v\gamma = t\gamma$ by case 11.4b.
11. If $s \succ_{HORPOLO} t$ by case 11.4c, the property holds by induction hypothesis, stability of η-reduction and case 11.4c. $\square$

Corollary 5. *$>_{HORPOLO}$ and $\geq_{HORPOLO}$ are stable under substitutions for candidate terms.*

5.3 Candidate interpretations

To prove well-foundedness of the ordering we follow Tait and Girard's computability predicate proof method. We denote by $[\![\sigma]\!]$ the computability predicate for candidate terms of type σ. Our definition of computability for candidate terms is standard and it is like the one in [18], but without considering polymorphism.

Definition 14. *The family of candidate interpretations $\{[\![\sigma]\!]\}_{\sigma \in \mathcal{T}_S}$ is the family of subsets of the set of typed terms whose elements are the least sets satisfying the following properties:*

1. *If σ is a data type, then $s :_{\mathcal{C}} \sigma \in [\![\sigma]\!]$ iff $\forall t :_{\mathcal{C}} \tau$ such that $s >_{HORPOLO} t$, $t \in [\![\tau]\!]$.*
2. *If $s :_{\mathcal{C}} \sigma = \tau \rightarrow \rho$ then $s \in [\![\sigma]\!]$ iff $@(s, t) \in [\![\rho]\!]$ for every $t \in [\![\tau]\!]$.*

A typed term s of type σ is said to be computable if $s \in [\![\sigma]\!]$. A vector $\overline{s}$ of terms is computable if and only if so are all its components.

As in [18], computability is shown to be well-defined by a lexicographic combination of an induction on the well-founded type ordering $>_{\mathcal{T}}^{\rightarrow}$ (which includes $>_{\mathcal{T}}$ and $\rhd_{\rightarrow}$), and a fixpoint computation for equal data types. Since case 1 does not involve any negation, it is monotonic with respect to set inclusion, which ensures the existence of a least fix point.

Then, if we apply case 2 we decrease in $>_{\mathcal{T}}^{\rightarrow}$ as it includes $\rhd_{\rightarrow}$ and if we apply case 1 either we decrease in $>_{\mathcal{T}}^{\rightarrow}$, as it includes $>_{\mathcal{T}}$, or $\sigma =_{\mathcal{T}} \tau$ and both are data types, and then we conclude by the fixpoint computation. Note that for data types this definition can be seen as a closure w.r.t. case 1, taking as initial set for each data type the set of minimal, w.r.t. $>_{\textsc{horpolo}}$, terms (which includes the variables).

Preservation of data types follows easily from arrow preservation:

Lemma 26. *Assume that $\sigma =_{\mathcal{T}} \tau$ and σ is a data type, then τ is a data type as well.*

Lemma 27. *If $\sigma =_{\mathcal{T}} \tau$ then $[\![\sigma]\!] = [\![\tau]\!]$.*

Proof. We proceed by induction on $\rhd_{\rightarrow}$. Consider first σ is a data type. By Lemma 16, $s :_{\mathcal{C}} \sigma$ and $\sigma =_{\mathcal{T}} \tau$ implies $s :_{\mathcal{C}} \tau$. On the other hand, $\sigma =_{\mathcal{T}} \tau$ gives us $s :_{\mathcal{C}} \sigma \succ_{\textsc{horpolo}} t :_{\mathcal{C}} \rho$ iff $s :_{\mathcal{C}} \tau \succ_{\textsc{horpolo}} t :_{\mathcal{C}} \rho$. Consequently, since by Lemma 26 we have that τ is also a data type, by applying Definition 14.1 we conclude $s \in [\![\tau]\!]$.

If $\sigma =_{\mathcal{T}} \tau$ and σ is a functional type of the form $\alpha \rightarrow \rho$ then, by arrow preservation, τ is also a functional type of the form $\alpha' \rightarrow \rho'$, where $\alpha =_{\mathcal{T}} \alpha'$ and $\rho =_{\mathcal{T}} \rho'$. Moreover, by Definition 14.2, $s \in [\![\sigma]\!]$ iff $@(s,t) \in [\![\rho]\!]$ for every $t \in [\![\alpha]\!]$. By induction hypothesis, $[\![\alpha]\!] = [\![\alpha']\!]$ and $[\![\rho]\!] = [\![\rho']\!]$, and hence $s \in [\![\tau]\!]$. □

In order to prove the well-foundedness of $>_{\textsc{horpolo}}$, we first prove five properties on computability of candidate terms. Recall that a term is neutral if it is not an abstraction.

Property 2. Computability properties.

1. Every computable term is strongly normalizing w.r.t. $>_{\textsc{horpolo}}$.
2. If s is computable and $s \geq_{\textsc{horpolo}}^{*} t$ then t is computable.
3. A neutral term s is computable if t is computable for every t s.t. $s >_{\textsc{horpolo}} t$.
4. If $\bar{t}$ is a vector of at least two computable terms s.t. $@(\bar{t})$ is a candidate term, then $@(\bar{t})$ is computable.
5. $\lambda x : \sigma.u$ is computable iff $u\{x \mapsto w\}$ is computable for every computable term $w :_{\mathcal{C}} \sigma$.

All proofs are adapted from [18], with some additional difficulties.

Proof.

– Property 4. By induction on the length of $\bar{t}$ and applying case 2 of the definition of candidate interpretations.

– Properties 1, 2, 3 are proved together.
  Given a type σ, we prove by induction on the definition of $[\![\sigma]\!]$ that
  1. Given $s :_{\mathcal{C}} \sigma \in [\![\sigma]\!]$, then s is strongly normalizing w.r.t. $>_{\mathit{HORPOLO}}$.
  2. Given $s :_{\mathcal{C}} \sigma \in [\![\sigma]\!]$ such that $s \geq_{\mathit{HORPOLO}} t$ for some $t :_{\mathcal{C}} \tau$, then $t \in [\![\tau]\!]$. By repeated applications of such property we have that given $s :_{\mathcal{C}} \sigma \in [\![\sigma]\!]$ such that $s \geq^*_{\mathit{HORPOLO}} t$ for $t :_{\mathcal{C}} \tau$, then $t \in [\![\tau]\!]$.
  3. A neutral candidate term $u :_{\mathcal{C}} \sigma \in [\![\sigma]\!]$ if $w :_{\mathcal{C}} \theta \in [\![\theta]\!]$ for all w such that $u >_{\mathit{HORPOLO}} w$. In particular, variables are computable.

  We prove each property distinguishing in each case whether σ is a data type or functional.
  1. Given $s :_{\mathcal{C}} \sigma \in [\![\sigma]\!]$, then s is strongly normalizing.
     If σ is a data type then, by definition 14.1, s computable implies t computable for every t such that $s >_{\mathit{HORPOLO}} t$. By induction hypothesis t is strongly normalizing, and hence s is strongly normalizing.
     Otherwise $\sigma = \theta \to \tau$. We will prove that $t :_{\mathcal{C}} \rho$ is strongly normalizing for every t such that $s >_{\mathit{HORPOLO}} t$. Then, let $s' :_{\mathcal{C}} \sigma'$ be a term such that $s :_{\mathcal{C}} \sigma \sqsupseteq^*_{\mathit{HORPOLO}} s' :_{\mathcal{C}} \sigma' \succ_{\mathit{HORPOLO}} t :_{\mathcal{C}} \rho$. By definition of $\sqsupseteq_{\mathit{HORPOLO}}$, we have that $\sigma' = \theta' \to \tau'$, with $\theta =_T \theta'$ and $\tau =_T \tau'$. By definition of $\succ_{\mathit{HORPOLO}}$, we have $\sigma' \geq_T \rho$ and hence, by arrow preservation and arrow decreasingness, there are only two cases to be considered:
     - Case $\tau =_T \tau' \geq_T \rho$. Then we have $@(s, y) \sqsupseteq^*_{\mathit{HORPOLO}} @(s', y)$ by case 10.3 and $@(s', y) \succ_{\mathit{HORPOLO}} t$ by case 11.2a for some variable $y : \theta$. Since $y :_{\mathcal{C}} \theta$ is computable by induction hypothesis 3, $@(s, y) :_{\mathcal{C}} \tau$ is computable by assumption and definition of $[\![\tau]\!]$. Then, by induction hypothesis on $@(s, y) :_{\mathcal{C}} \tau$, we have that t is strongly normalizing.
     - Case $\rho = \theta'' \to \tau''$ with $\theta =_T \theta' =_T \theta''$ and $\tau =_T \tau' \geq_T \tau''$. Then we have $@(s, y) \sqsupseteq^*_{\mathit{HORPOLO}} @(s', y)$ by case 10.3 and $@(s', y) \succ_{\mathit{HORPOLO}} @(t, y)$ by case 11.2b for some variable $y : \theta$. Since $y :_{\mathcal{C}} \theta$ is computable by induction hypothesis 3, $@(s, y) :_{\mathcal{C}} \tau$ is computable by assumption and definition of $[\![\tau]\!]$. Then, by induction hypothesis on $@(s, y) :_{\mathcal{C}} \tau$, we have that $@(t, y)$ is strongly normalizing, which again by case 11.2b, implies strong normalization of t.
  2. Given $s :_{\mathcal{C}} \sigma \in [\![\sigma]\!]$ such that $s \geq_{\mathit{HORPOLO}} t$ for some $t :_{\mathcal{C}} \tau$, then $t \in [\![\tau]\!]$.
     Consider first that σ is a data type. Then if $s >_{\mathit{HORPOLO}} t$ the property follows from case 1 of the definition. Otherwise, $s \sqsupseteq^*_{\mathit{HORPOLO}} t$. Since $s \in [\![\sigma]\!]$, we have $\forall w :_{\mathcal{C}} \tau$ such that $s >_{\mathit{HORPOLO}} w$, $w \in [\![\tau]\!]$. Now as $\forall w :_{\mathcal{C}} \tau$ such that $t >_{\mathit{HORPOLO}} w$ we have $s \sqsupseteq^*_{\mathit{HORPOLO}} t >_{\mathit{HORPOLO}} w$ and hence $s >_{\mathit{HORPOLO}} w$, we conclude $t \in [\![\tau]\!]$ by case 14.1.
     Consider now $\sigma = \theta \to \rho$. First, note that, by definition of $\sqsupseteq_{\mathit{HORPOLO}}$, for every $s' :_{\mathcal{C}} \sigma'$, such that $s :_{\mathcal{C}} \sigma \sqsupseteq_{\mathit{HORPOLO}} s' :_{\mathcal{C}} \sigma'$, we have $\sigma' =_T \sigma$ and, thus, σ' is functional. Now, by arrow decreasingness and by arrow preservation there are two cases:
     (a) $\rho \geq_T \tau$. Since $s :_{\mathcal{C}} \sigma$ is computable, by case 14.2, $@(s, u)$ is computable $\forall u \in [\![\theta]\!]$. Let $y :_{\mathcal{C}} \theta$. By induction hypothesis 3, $y \in [\![\theta]\!]$, hence $@(s, y)$ is computable. Then, as $@(s, y) :_{\mathcal{C}} \rho \sqsupseteq^*_{\mathit{HORPOLO}} \cdot \succ_{\mathit{HORPOLO}} t :_{\mathcal{C}} \tau$ by cases 10.3 and 11.2a and hence $@(s, y) :_{\mathcal{C}} \rho >_{\mathit{HORPOLO}} t :_{\mathcal{C}} \tau$, we conclude t is computable by induction hypothesis 2.

(b) $\tau = \theta' \to \rho'$ with $\theta =_T \theta'$ and $\rho \geq_T \rho'$. Now, by case 14.2, for all $u \in [\![\theta]\!]$ we have $@(s,u) \in [\![\rho]\!]$. Then, since $@(s,u) \sqsupseteq^*_{\textsc{horpolo}} \cdot \succeq_{\textsc{horpolo}} @(t,u)$ by cases 10.3 and 11.2b and hence, $@(s,u) \geq_{\textsc{horpolo}} @(t,u)$, by induction hypothesis 2, $@(t,u) \in [\![\rho']\!]$. Since by Lemma 27 $[\![\theta]\!] = [\![\theta']\!]$, then $t \in [\![\tau]\!]$ by case 14.2.

3. A neutral candidate term $u :_{\mathcal{C}} \sigma \in [\![\sigma]\!]$ if $w :_{\mathcal{C}} \theta \in [\![\theta]\!]$ for all w such that $u >_{\textsc{horpolo}} w$.
If σ is a data type then, the property follows from the definition of candidate interpretations.
Assume now that $\sigma = \sigma_1 \to \ldots \to \sigma_n \to \tau$ where $n > 0$ and τ is a data type. By case 14.2, $u \in [\![\sigma]\!]$ if $@(u, u_1, \ldots, u_n) \in [\![\tau]\!]$ for arbitrary terms $u_1 \in [\![\sigma_1]\!], \ldots, u_n \in [\![\sigma_n]\!]$ which are strongly normalizing by induction hypothesis 1. Since τ is a data type then, by definition, $@(u, u_1, \ldots, u_n)$ is computable iff so are all its reducts.
We prove that for all $w :_{\mathcal{C}} \rho$ such that $@(u, u_1, \ldots, u_k) >_{\textsc{horpolo}} w$, $w \in [\![\rho]\!]$, for all $k \in \{0 \ldots n\}$. It is proved by induction on the multiset $\{u_1, \ldots, u_k\}$ ordered by $(>_{\textsc{horpolo}})_{mul}$. Taking $k = n$ yields the desired property, implying that u is computable.
If $k = 0$ then we have to prove $w \in [\![\rho]\!]$ for all $w :_{\mathcal{C}} \rho$ such that $u >_{\textsc{horpolo}} w$ which holds by assumption. For the general case, let $k = (j+1) \leq n$. We need to consider all terms w s.t. $@(@(u, u_1, \ldots, u_j), u_{j+1}) >_{\textsc{horpolo}} w$ i.e. $@(@(u, u_1, \ldots, u_j), u_{j+1}) \sqsupseteq^*_{\textsc{horpolo}} \cdot \succ_{\textsc{horpolo}} w$. We consider several cases according to the definition of $\succ_{\textsc{horpolo}}$. Note that $@(u, u_1, \ldots, u_j)$ is neutral even if $j = 0$. Then, for every u' such that $@(u, u_1, \ldots, u_j) \sqsupseteq^*_{\textsc{horpolo}} u'$, we have that u' is neutral since otherwise, by definition of $\sqsupseteq_{\textsc{horpolo}}$, $@(u, u_1, \ldots, u_j)$ must be an abstraction. Therefore case 11.3 of $\succ_{\textsc{horpolo}}$ does not apply and hence we only need to consider case 11.2.

- If $@(@(u, u_1, \ldots, u_j), u_{j+1}) \sqsupseteq^*_{\textsc{horpolo}} \cdot \succ_{\textsc{horpolo}} w$ using case 11.2a then we have that either $@(u, \ldots, u_j) \sqsupseteq^*_{\textsc{horpolo}} \cdot \succeq_{\textsc{horpolo}} w$ or $u_{j+1} \sqsupseteq^*_{\textsc{horpolo}} \cdot \succeq_{\textsc{horpolo}} w$.
  - ∗ If $@(u, \ldots, u_j) \sqsupseteq^*_{\textsc{horpolo}} \cdot \succeq_{\textsc{horpolo}} w$ then, as w is also a reduct of $@(@(u, \ldots, u_j), u_{j+1})$, for typing reasons, we actually have $@(u, \ldots, u_j) \sqsupseteq^*_{\textsc{horpolo}} \cdot \succ_{\textsc{horpolo}} w$ that is $@(u, \ldots, u_j) >_{\textsc{horpolo}} w$. We conclude w is computable by inner induction hypothesis.
  - ∗ If $u_k \sqsupseteq^*_{\textsc{horpolo}} \cdot \succeq_{\textsc{horpolo}} w$ and hence $u_k \geq_{\textsc{horpolo}} w$, we conclude by assumption and by induction hypothesis 2.
- If $@(@(u, u_1, \ldots, u_j), u_{j+1}) \sqsupseteq^*_{\textsc{horpolo}} \cdot \succ_{\textsc{horpolo}} w = @(\overline{w})$ using case 11.2b then, for some partial left-flattening $@(w_1, \ldots, w_s)$ of w, we have $\{@(u, u_1, \ldots, u_j), u_{j+1}\} (\sqsupseteq_{\textsc{horpolo}})^*_{mon} \cdot (\succ_{\textsc{horpolo}})_{mul} \{w_1, \ldots, w_s\}$ and hence we have $\{@(u, u_1, \ldots, u_j), u_{j+1}\} (\sqsupseteq^*_{\textsc{horpolo}})_{mon} \cdot (\succ_{\textsc{horpolo}})_{mul} \{w_1, \ldots, w_s\}$. By definition of the monotonic and multiset extensions and for type reasons, there are two possibilities:
  - ∗ Consider first the case $@(u, u_1, \ldots, u_j) \sqsupseteq^*_{\textsc{horpolo}} w_1$ and $u_{j+1} \sqsupseteq^*_{\textsc{horpolo}} \cdot \succ_{\textsc{horpolo}} w_i$ $\forall i \in \{2, \ldots, s\}$, implying that w_i is computable by assumption and induction hypothesis 2 (since $>_{\textsc{horpolo}} = \sqsupseteq^*_{\textsc{horpolo}} \cdot \succ_{\textsc{horpolo}}$). By induction hypothesis all

reducts of $@(u, u_1, \ldots, u_j)$ are computable. Therefore, all reducts of w_1 are computable and, as w_1 is neutral (it is an application), we have that w_1 is computable by induction hypothesis 3. It follows that w is computable by Property 2.4.

    * Otherwise, $\forall w_i \in \{w_1, \ldots, w_s\}$ we have either $@(u, u_1, \ldots, u_j) \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} w_i$ or $u_{j+1} \sqsupseteq^*_{HORPOLO} \cdot \succeq_{HORPOLO} w_i$. In the first case we have $@(u, u_1, \ldots, u_j) >_{HORPOLO} w_i$ and we conclude w_i is computable by induction hypothesis. For the second case we have $u_{j+1} \geq_{HORPOLO} w_i$ and as u_{j+1} is computable by assumption, we conclude w_i is computable by induction hypothesis 2. Then, by Property 2.4, we conclude w is computable.

  As a consequence, all reducts of $@(u, u_1, \ldots, u_n)$ are computable and we are done.

- Property 5: $\lambda x : \sigma.u$ is computable iff $u\{x \mapsto w\}$ is computable for every computable term $w :_{\mathcal{C}} \sigma$.
  First we prove the only if part. By definition 14.2, $\lambda x : \sigma.u \in [\![\sigma \to \tau]\!]$ implies $@(\lambda x.u, w) \in [\![\tau]\!]$ for all $w \in [\![\sigma]\!]$. Then, by Property 2.2, $@(\lambda x.u, w) >_{HORPOLO} v$ implies v is computable. Therefore, we have that $u\{x \mapsto w\}$ is computable since $@(\lambda x.u, w) \succ_{HORPOLO} u\{x \mapsto w\}$ by case 11.3 and hence, by definition of $>_{HORPOLO}$, $@(\lambda x.u, w) >_{HORPOLO} u\{x \mapsto w\}$.
  For the if part we will prove that $u\{x \mapsto w\} \in [\![\tau]\!]$ for every $w \in [\![\sigma]\!]$ implies $@(\lambda x.u, w) \in [\![\tau]\!]$ since, by definition 14.2, this implies $\lambda x : \sigma.u$ is computable.
  Since variables are computable by Property 2.3, $u\{x \mapsto z\}$ is computable by assumption for some fresh variable z. Therefore, by Property 2.1 we have that $u\{x \mapsto z\}$ and w are strongly normalizing and hence we can use induction on the pair $\langle u\{x \mapsto z\}, w\rangle$ ordered by $(>_{HORPOLO})_{lex}$ to prove $@(\lambda x.u, w)$ is computable.
  Since $@(\lambda x.u, w)$ is neutral, by Property 2.3, it is computable if v is computable for all v such that $@(\lambda x.u, w) >_{HORPOLO} v$, that is, $@(\lambda x.u, w) \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} v$. There are three cases:
  1. $@(\lambda x.u, w) \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} v$ using case 10.3 for $\sqsupseteq_{HORPOLO}$ and case 11.2a for $\succ_{HORPOLO}$. There are two cases:
     (a) if $w \sqsupseteq^*_{HORPOLO} \cdot \succeq_{HORPOLO} v$ we conclude that v is computable by Property 2.2 as $\geq_{HORPOLO} = \sqsupseteq^*_{HORPOLO} \cdot \succeq_{HORPOLO}$.
     (b) if $\lambda x.u \sqsupseteq^*_{HORPOLO} \cdot \succeq_{HORPOLO} v$ then, for typing reasons, we have $\lambda x.u \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} v$. Steps of $\sqsupseteq_{HORPOLO}$ hold by case 10.4 and for $\succ_{HORPOLO}$ there are three cases:
        i. If $\lambda x.u \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} v$ by case 11.4a then we have that $u\{x \mapsto z\} \sqsupseteq^*_{HORPOLO} \cdot \succeq_{HORPOLO} v$ and hence $u\{x \mapsto z\} \geq_{HORPOLO} v$. We conclude again that v is computable by Property 2.2.
        ii. If $\lambda x.u \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} v = \lambda y : \beta.u'$ by case 11.4b then, since z is fresh, we have $u\{x \mapsto z\} \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} u'\{y \mapsto z\}$. Now, for every computable term $w' :_{\mathcal{C}} \beta$, we have that $u\{x \mapsto w'\}$ is computable by assumption, and $u\{x \mapsto w'\} >_{HORPOLO}$

$u'\{y \mapsto w'\}$, by stability under substitutions, which, by Property 2.2, implies that $u'\{y \mapsto w'\}$ is computable. Now, by induction hypothesis applied to the pair $\langle u'\{y \mapsto z\}, w'\rangle$, we have that $@(\lambda y.u', w')$ is computable, and we conclude that v is computable by definition 14.2.

iii. If $\lambda x.u \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} v$ by case 11.4c then we have $u = @(u', x)$, $x \notin \mathcal{V}ar(u')$ and $u' \sqsupseteq^*_{HORPOLO} \cdot \succeq_{HORPOLO} v$, and hence $u' \geq_{HORPOLO} v$. By the main assumption we have that $@(u', x)\{x \mapsto t\} = @(u', t)$ is computable for all computable term t and hence, by definition 14.2, we have u' is computable. Therefore, since $u' \geq_{HORPOLO} v$, we conclude v is computable by Property 2.2.

2. $@(\lambda x.u, w) \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} @(\overline{v})$ using case 10.3 for $\sqsupseteq_{HORPOLO}$ and case 11.2b for $\succ_{HORPOLO}$. Then, $\{\lambda x.u, w\}((\sqsupseteq_{HORPOLO})_{mon})^* \cdot (\succ_{HORPOLO})_{mul}\{\overline{v}\}$ and hence $\{\lambda x.u, w\}(\sqsupseteq^*_{HORPOLO})_{mon} \cdot (\succ_{HORPOLO})_{mul}\{\overline{v}\}$. For typing reasons, there are two cases:
   - $\lambda x.u \sqsupseteq^*_{HORPOLO} v_1 = \lambda y.u'$ and $w \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} v_j$ for all $j > 1$. By Property 2.2 we have that v_j is computable for all v_j s.t. $w \geq_{HORPOLO} v_j$. As $\lambda x.u \sqsupseteq^*_{HORPOLO} v_1 = \lambda y.u'$ implies $u\{x \mapsto z\} \sqsupseteq^*_{HORPOLO} u'\{y \mapsto z\}$, then by stability it also holds $u\{x \mapsto v_2\} \sqsupseteq^*_{HORPOLO} u'\{y \mapsto v_2\}$. Now, as $u\{x \mapsto v_2\}$ is computable by assumption, and we have $u\{x \mapsto v_2\} \geq_{HORPOLO} u'\{y \mapsto v_2\}$, we conclude that $u'\{y \mapsto v_2\}$ is computable by Property 2.2. Hence, by induction hypothesis we have $@(v_1, v_2)$ is computable. Now if $v = @(v_1, v_2)$ we are done and we conclude by Property 2.4 otherwise.
   - $\lambda x.u \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} v_j$ or $w \sqsupseteq^*_{HORPOLO} \cdot \succeq_{HORPOLO} v_j$ for all $j \geq 1$. By Property 2.2 we have that v_j is computable for all v_j s.t. $w \geq_{HORPOLO} v_j$ or $u\{x \mapsto z\} \geq_{HORPOLO} v_j$. If $\lambda x.u \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} v_j$ holds by case 11.4c we can prove that v_j is computable as done in the previous case 1(b)iii. Otherwise it holds by case 11.4b and hence $\lambda x.u \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} \lambda y.u' = v_j$ with $u\{x \mapsto z\} \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} u'\{y \mapsto z\}$, and hence $u\{x \mapsto z\} >_{HORPOLO} u'\{y \mapsto z\}$. Then, by assumption $u\{x \mapsto v''\}$ is computable for an arbitrary computable v'' and, by stability under substitutions, it also holds $u\{x \mapsto v''\} >_{HORPOLO} u'\{y \mapsto v''\}$. Now by Property 2.2 we conclude $u'\{y \mapsto v''\}$ is computable. Hence, by induction hypothesis $\lambda y.u' = v_j$ is computable. Therefore $@(\overline{v})$ is computable by Property 2.4.

3. $@(\lambda x.u, w) \sqsupseteq^*_{HORPOLO} \cdot \succ_{HORPOLO} v$ using case 10.3 for $\sqsupseteq_{HORPOLO}$ and case 11.3 for $\succ_{HORPOLO}$, that is $@(\lambda x.u, w) \sqsupseteq^*_{HORPOLO} @(\lambda y.u', w')$ and $u'\{y \mapsto w'\} \succeq_{HORPOLO} v$. Then, by definition of $\sqsupseteq_{HORPOLO}$ we have $u\{x \mapsto z\} \sqsupseteq^*_{HORPOLO} u'\{y \mapsto z\}$ and $w \sqsupseteq^*_{HORPOLO} w'$, and hence, by stability and monotonicity $u\{x \mapsto w\} \sqsupseteq^*_{HORPOLO} u'\{y \mapsto w'\}$. Therefore we have $u\{x \mapsto w\} >_{HORPOLO} v$ and since $u\{x \mapsto w\}$ is computable by the main assumption, then v is computable by Property 2.2. □

Lemma 28 (Generalization of Lemma 12). *Let $\mathcal{S}$ be a set of terms closed w.r.t. $\geq_{HORPOLO}$.*

1. *If p and q are linear polynomials, $\mathcal{V}ar(p) \subseteq \mathcal{X}_{\mathcal{S}}$ and $p >_{C(\mathcal{S})} q$ then $\mathcal{V}ar(q) \subseteq \mathcal{X}_{\mathcal{S}}$.*
2. *If all terms in $\mathcal{S}$ are computable then $>_{C(\mathcal{S})}$ is well-founded on linear polynomials over $\mathbb{Z}^+$ and $\mathcal{X}_{\mathcal{S}}$.*

Proof. Let C be $C(\mathcal{S})$. Given a polynomial $p = a_0 + a_1 \cdot x_{u_1} + \ldots + a_n \cdot x_{u_n}$ over labeled variables in $\mathcal{X}_{\mathcal{S}}$, we define $M(p)$ as the multiset containing a_i occurrences of u_i for all $i \in \{1 \ldots n\}$, and $K(p)$ as the non-negative integer a_0.

Then, if we have $p + x_u \rightarrow_C p + E$ then there are two cases:

1. $E = x_v$ with $u \sqsupseteq_{HORPOLO} v$. Then $v \in \mathcal{S}$ which implies $x_v \in \mathcal{X}_{\mathcal{S}}$, $M(p + x_u)(\sqsupseteq_{HORPOLO})_{mon} M(p + E)$ and $K(p + x_u) = K(p + E)$.
2. $E = q + 1$ with $u \succ_{HORPOLO} v \in \mathcal{S}$ for all x_v occurring in E, which implies $x_v \in \mathcal{X}_{\mathcal{S}}$ and $M(p + x_u)\,(\succ_{HORPOLO})_{mul}\, M(p + E)$.

Repeatedly applying both cases above, if $p \twoheadrightarrow\!\!\!\!\!\rightarrow^{=}_{C} p'$ then we have that (i) $\mathcal{V}ar(p) \subseteq \mathcal{X}_{\mathcal{S}}$ implies $\mathcal{V}ar(p') \subseteq \mathcal{X}_{\mathcal{S}}$, and (ii) either $M(p)\,(\succ_{HORPOLO})_{mul}\, M(p')$ or both $M(p)\,(\sqsupseteq_{HORPOLO})_{mon}\, M(p')$ and $K(p) = K(p')$. Moreover, by Corollary 1, if $p' > q$ then $p' = q + q' + 1$ for some q', which implies that (iii) $\mathcal{V}ar(q) \subseteq \mathcal{V}ar(p')$ and (iv) either $M(q + q' + 1)\,(\succ_{HORPOLO})_{mul}\, M(q)$ or $M(q') = \varnothing$.

Therefore, by (i) and (iii) we have that if $\mathcal{V}ar(p) \subseteq \mathcal{X}_{\mathcal{S}}$ then $\mathcal{V}ar(q) \subseteq \mathcal{V}ar(p') \subseteq \mathcal{X}_{\mathcal{S}}$, which ends the proof of (1). Now, we conclude with the proof of (2). Assume we have an infinite sequence $p_0 >_C p_1 >_C p_2 >_C \ldots$ with $\mathcal{V}ar(p_0) \subseteq \mathcal{X}_{\mathcal{S}}$. By definition, we have $p_0 \twoheadrightarrow\!\!\!\!\!\rightarrow^{=}_{C} p'_0 > p_1 \twoheadrightarrow\!\!\!\!\!\rightarrow^{=}_{C} p'_1 > p_2 \twoheadrightarrow\!\!\!\!\!\rightarrow^{=}_{C} p'_2 > \ldots$ Then, for all $i \geq 0$, it holds that

- $\mathcal{V}ar(p_i) \cup \mathcal{V}ar(p'_i) \subseteq \mathcal{X}_{\mathcal{S}}$, by induction on i using (1) to prove $\mathcal{V}ar(p_i) \subseteq \mathcal{X}_{\mathcal{S}}$ and then (i), as above, to conclude that $\mathcal{V}ar(p'_i) \subseteq \mathcal{X}_{\mathcal{S}}$;
- $M(p_i)\,(\succ_{HORPOLO})_{mul}\, M(p'_i)$ or both $M(p_i)\,(\sqsupseteq_{HORPOLO})_{mon}\, M(p'_i)$ and $K(p_i) = K(p'_i)$, by (ii);
- $M(p'_i)\,(\succ_{HORPOLO})_{mul}\, M(p_{i+1})$ or $M(p'_i)\,(\sqsupseteq_{HORPOLO})_{mon}\, M(p_{i+1})$, by (iv).

As, $\mathcal{V}ar(p_i) \cup \mathcal{V}ar(p'_i) \subseteq \mathcal{X}_{\mathcal{S}}$, we have $M(p_i) \cup M(p'_i) \subseteq \mathcal{S}$ for all $i \geq 0$, and since all terms in $\mathcal{S}$ are strongly normalizing w.r.t. $>_{HORPOLO}$, we have that $(>^{+}_{HORPOLO})_{mul}$ is well-founded. Thus, from some point k on we have that $M(p_i)(\sqsupseteq_{HORPOLO})_{mon} M(p'_i)(\sqsupseteq_{HORPOLO})_{mon} M(p_{i+1})$ and $K(p_i) = K(p'_i)$ for all $i \geq k$, since otherwise, as $\succ_{HORPOLO} \subseteq >_{HORPOLO}$ and $\sqsupseteq_{HORPOLO} \subseteq \geq_{HORPOLO}$ and $>^{+}_{HORPOLO}$ is compatible with $\geq^{*}_{HORPOLO}$, we would have an infinite sequence with $(>^{+}_{HORPOLO})_{mul}$. Therefore, we have $K(p_i) = K(p'_i) > K(p_{i+1})$ for all $i \geq k$, which is a contradiction. □

Lemma 29 (Generalization of Lemma 13). *Let $t :_{\mathcal{C}} \tau$ be a candidate term with $top(t) \in \mathcal{F}_{POLO}$ s.t. $\mathcal{A}cc(t) = \{u_1, \ldots, u_k\}$. If $u_1, \ldots, u_k$ are computable then t is computable.*

Proof. Let $\mathcal{S}$ be the closure of $\{u_1, \ldots, u_k\}$ w.r.t. $\geq_{HORPOLO}$. Note that, since computability implies strong normalization by Property 2.1, we have that $>_{C(\mathcal{S})}$ is well-founded by Lemma 28.2.

Now, since τ is a data type, by Property 2.3, we have to prove that w is computable for all candidate terms $w :_{\mathcal{C}} \rho$ such that $t >_{HORPOLO} w$. We proceed by induction on $I(t)$ w.r.t. $>_C$, taking $C = C(\mathcal{S})$.

If ρ is a functional type then there is some $u \in \mathcal{A}cc(t)$ such that $u \succ_{HORPOLO} w$, and hence $u >_{HORPOLO} w$. Since u is computable by assumption, we have w is computable by Property 2.2.

Otherwise, ρ is a data type. By Lemma 21, $t >_{RPOLO} w$ implies that for all $v \in \mathcal{A}cc(w)$ there is some $u \in \mathcal{A}cc(t)$ such that $u \geq^*_{HORPOLO} v$. Therefore, since $\mathcal{S}$ is the closure of $\mathcal{A}cc(t)$ w.r.t. $\geq_{HORPOLO}$, we have that all terms in $\mathcal{A}cc(w)$ are in $\mathcal{S}$, and thus computable by Property 2.2.

There are two cases to be considered. If $top(w) \in \mathcal{F}_{RPO} \cup \{@\}$ or w is a variable then, since $\mathcal{A}cc(w) = \{w\}$, we have that w is computable. Otherwise, $top(w) \in \mathcal{F}_{POLO}$ and hence, by definition of the ordering, $I(t) >_{C(t)} I(w)$. Since $C(t) = C(\mathcal{A}cc(t))$ and $\mathcal{A}cc(t) \subseteq \mathcal{S}$, we have that $C(t) \subseteq C(\mathcal{S}) = C$, which implies $I(t) >_C I(w)$. Finally, by Lemma 28.1, $\mathcal{V}ar(I(w)) \subseteq \mathcal{X}_{\mathcal{S}}$, which allows us to conclude, by induction hypothesis, that w is computable. □

Lemma 30 (Generalization of Lemma 14). *Let $f :_{\mathcal{C}} \overline{\sigma} \to \tau \in \mathcal{F}_{RPO}$ and $t_1 :_{\mathcal{C}} \sigma_1, \ldots, t_n :_{\mathcal{C}} \sigma_n$ be a set of candidate terms. If $t_1, \ldots, t_n$ are computable then $f(t_1, \ldots, t_n) :_{\mathcal{C}} \tau$ is computable.*

Proof. The proof is done by induction on $\langle f, \{t_1, \ldots, t_n\}\rangle$ ordered lexicographically by $(\succ_{\mathcal{F}}, (>^+_{HORPOLO})_{stat(f)})_{lex}$ where *stat* is either *mul* or *lex* depending on the symbol f and $(>^+_{HORPOLO})_{mul}$ is the multiset extension of $>^+_{HORPOLO}$ w.r.t. $\geq^*_{HORPOLO}$ and $(>^+_{HORPOLO})_{lex}$ is the lexicographic extension of $>^+_{HORPOLO}$ w.r.t. $\geq^*_{HORPOLO}$. This relation is well-founded since we are assuming that $t_1, \ldots, t_n$ are computable and hence strongly normalizing w.r.t. $>_{HORPOLO}$ by Property 2.1.

Since $f(t_1, \ldots, t_n)$ is neutral, by Property 2.3, it is computable if every w such that $f(t_1, \ldots, t_n) >_{HORPOLO} w :_{\mathcal{C}} \rho$ is computable, which we prove by an inner induction on $|w|$.

By definition of $>_{HORPOLO}$, $t = f(t_1, \ldots, t_n) >_{HORPOLO} w$ implies that $t = f(t_1, \ldots, t_n) \sqsupseteq^*_{HORPOLO} t' \succ_{HORPOLO} w$ for some $t' :_{\mathcal{C}} \tau'$. Now, by definition of $\sqsupseteq_{HORPOLO}$ we have that $\tau =_T \tau'$, $top(t) =_{\mathcal{F}} top(t') \in \mathcal{F}_{RPO}$ and we also have that for all t'_j argument of t' there is some t_i argument of t such that $t_i \sqsupseteq^*_{HORPOLO} t'_j$, and hence, by definition of $\geq_{HORPOLO}$, $t_i \geq_{HORPOLO} t'_j$. Therefore, as by assumption $t_1, \ldots, t_n$ are computable then, by Property 2.2, all arguments of t' are computable.

We distinguish several cases according to the definition of $\succ_{HORPOLO}$.

- If $t' \succ_{HORPOLO} w$ holds by case 11.1(b)i then we have $t'_j \succeq_{HORPOLO} w$ for some t'_j argument of t'. Thus, by Property 2.2, w is computable since $\succeq_{HORPOLO} \subseteq \geq_{HORPOLO}$ and as we have seen, t'_j is computable.
- If $t' \succ_{HORPOLO} w$ holds by case 11.1(b)ii then we have $top(w) \in \mathcal{F}_{POLO}$ and $t' \succ_{HORPOLO} w'$ $\forall w' \in \mathcal{A}cc(w)$. By the inner induction hypothesis, w' is computable and hence, by Lemma 29, w is computable.
- If $t' \succ_{HORPOLO} w$ holds by case 11.1(b)iiiA then $top(w) \in \mathcal{F}_{RPO}$, $top(t') \succ_{\mathcal{F}} top(w)$ and for every w_i argument of w either $t' \succ_{HORPOLO} w_i$, in which case w_i

is computable by the inner induction hypothesis (as $t \sqsupseteq^*_{HORPOLO} t' \succ_{HORPOLO} w_i$ implies $t >_{HORPOLO} w_i$), or $t'_j \succeq_{HORPOLO} w_i$ for some t'_j argument of t', and since $\succeq_{HORPOLO} \subseteq \geq_{HORPOLO}$, w_i is computable by Property 2.2. Therefore, w_i is computable for all w_i argument of w, and since $top(t) =_{\mathcal{F}} top(t') \succ_{\mathcal{F}} top(w)$, we conclude that w is computable by the outer induction hypothesis.

- If $t' \succ_{HORPOLO} w$ holds by case 11.1(b)iiiB then we have $top(t') =_{\mathcal{F}} top(w) \in \mathcal{F}_{RPO}$ and $\{t'_1, \dots, t'_n\}(\succ_{HORPOLO})_{mul}\{w_1, \dots, w_m\}$. Thus, we have $top(t) =_{\mathcal{F}} top(w)$ and $\{t_1, \dots, t_n\}(\sqsupseteq_{HORPOLO})^*_{mon}\{t'_1, \dots, t'_n\}(\succ_{HORPOLO})_{mul}\{w_1, \dots, w_m\}$, and hence, $\{t_1, \dots, t_n\}(\sqsupseteq^*_{HORPOLO})_{mon}\{t'_1, \dots, t'_n\}(\succ_{HORPOLO})_{mul}\{w_1, \dots, w_m\}$. Now, we prove that $\{t_1, \dots, t_n\}(>_{HORPOLO})_{mul}\{w_1, \dots, w_m\}$, which implies that all terms $w_1, \dots, w_m$ are computable and allows us to conclude that w is computable by the outer induction hypothesis using the second component of $(\succ_{\mathcal{F}}, (>_{HORPOLO})^+_{stat})_{lex}$.
  By definition, $\{t_1, \dots, t_n\}(\sqsupseteq^*_{HORPOLO})_{mon}\{t'_1, \dots, t'_n\}$ implies that there is some permutation π such that $t_{\pi(i)} \sqsupseteq^*_{HORPOLO} t'_i$ for all i, and $\{t'_1, \dots, t'_n\}(\succ_{HORPOLO})_{mul}\{w_1, \dots, w_m\}$ implies $\{t'_1, \dots, t'_n\} = M' \cup S'$ and $\{w_1, \dots, w_m\} = N \cup T$ and $M'(\sqsupseteq_{HORPOLO})_{mon}N$ and for all $w_j \in T$ there is a $t'_i \in S'$ such that $t'_i \succ_{HORPOLO} w_j$. Take $M = \{t_{\pi(i)} \mid t'_i \in M'\}$ and $S = \{t_{\pi(i)} \mid t'_i \in S'\}$. Then $M(\sqsupseteq^*_{HORPOLO})_{mon}M'(\sqsupseteq_{HORPOLO})_{mon}N$, which implies $M(\sqsupseteq^*_{HORPOLO})_{mon}N$, and for all $w_j \in T$ we have that $t_{\pi(i)} \in S$ fulfils $t_{\pi(i)} \sqsupseteq^*_{HORPOLO} t'_i \succ_{HORPOLO} w_j$. Finally, by definition of $\geq_{HORPOLO}$ and $>_{HORPOLO}$, we have $M(\geq_{HORPOLO})_{mon}N$ and $\forall w_j \in T\ \exists t_{\pi(i)} \in S$ such that $t_{\pi(i)} >_{HORPOLO} w_j$, and hence $\{t_1, \dots, t_n\}(>_{HORPOLO})_{mul}\{w_1, \dots, w_m\}$.
- If $t' \succ_{HORPOLO} w$ holds by case 11.1(b)iiiC then we have $top(w) \in \mathcal{F}_{RPO}$, $top(t') =_{\mathcal{F}} top(w)$, $\langle t'_1, \dots, t'_n\rangle(\succ_{HORPOLO})_{lex}\langle w_1, \dots, w_m\rangle$ and for all w_j argument of w, either $t' \succ_{HORPOLO} w_j$ or $t'_i \succeq_{HORPOLO} w_j$ for some t'_i argument of t'. As in case 11.1(b)iiiA, we conclude w_j is computable for all w_j argument of w. Moreover, we have $\langle t_1, \dots, t_n\rangle(\sqsupseteq_{HORPOLO})^*_{mon}\langle t'_1, \dots, t'_n\rangle(\succ_{HORPOLO})_{lex}\langle w_1, \dots, w_m\rangle$ and hence, by definition of the lexicographic extension we have that $\langle t'_1, \dots, t'_n\rangle(\succ_{HORPOLO})_{lex}\langle w_1, \dots, w_m\rangle$ implies $\langle t'_1, \dots, t'_k\rangle(\sqsupseteq_{HORPOLO})_{mon}\langle w_1, \dots, w_k\rangle$ for some k and either $k = m < n$ or $t'_{k+1} \succ_{HORPOLO} w_{k+1}$. Thus, by definition of the monotonic extension on tuples, for all $i \in \{1, \dots, k\}$, we have $t_i \sqsupseteq^*_{HORPOLO} t'_i \sqsupseteq_{HORPOLO} w_i$, and either $k = m < n$ or $t_{k+1} \sqsupseteq^*_{HORPOLO} t'_{k+1} \succ_{HORPOLO} w_{k+1}$. Hence, by definition of $>_{HORPOLO}$, we have $t_i \geq_{HORPOLO} w_i$ for all $i \in \{1, \dots, k\}$, and either $k = m < n$ or $t_{k+1} >_{HORPOLO} w_{k+1}$. Therefore we conclude $\langle t_1, \dots, t_n\rangle(>_{HORPOLO})_{lex}\langle w_1, \dots, w_m\rangle$. Hence, as $top(t) = top(w)$, we conclude that w is computable by the outer induction hypothesis using the second component of $(\succ_{\mathcal{F}}, (>_{HORPOLO})^+_{stat})_{lex}$.
- Consider now $t' \succ_{HORPOLO} w$ by case 11.1(b)iv for some $@(w_1, \dots, w_m)$ partial left-flattening of w. Hence, we have for all $i \in \{1, \dots, m\}$, either $t' \succ_{HORPOLO} w_i$ or $t'_j \succeq_{HORPOLO} w_i$ for some $j \in \{1, \dots, n\}$. By the inner induction in the first case and by Property 2.2 in the second case, we conclude w_i is computable for all $i \in \{1, \dots, m\}$, and hence w is computable by Property 2.4.

□

Lemma 31 (Generalization of Lemma 15). *$>_{HORPOLO}$ is well-founded.*

Proof. We will prove that $t\gamma$ is computable for every typed term t and computable substitution γ. Then, taking the empty substitution we have that all terms are computable and hence, by Property 2.1, strongly normalizing w.r.t. $>_{HORPOLO}$.

The proof is by induction on $|t|$.

- If t is a variable x then either $x \in \mathcal{D}om(\gamma)$ and $x\gamma$ is computable by assumption, or $x\gamma = x$, which is computable by Property 2.3.
- If $t = \lambda x.u$ then, by Property 2.5, $t\gamma$ is computable if $u\gamma\{x \mapsto w\}$ is computable for every well-typed computable term w. Let $\delta = \gamma \cup \{x \mapsto w\}$. Then we have $u\gamma\{x \mapsto w\} = u(\gamma \cup \{x \mapsto w\}) = u\delta$ since x may not occur in γ. Since δ is computable, and $|t| > |u|$, by induction hypothesis, $u\delta$ is computable and hence $t\gamma$ is computable.
- $t = f(t_1, \ldots, t_n)$ with $f \in \mathcal{F}_{RPO}$. By induction hypothesis $t_i\gamma$ is computable for all i, and hence, $t\gamma$ is computable by Lemma 30.
- $t = @(t_1, t_2)$. By induction hypothesis $t_1\gamma$ and $t_2\gamma$ are computable, and hence, $t\gamma$ is computable by Property 2.4.
- $t = f(t_1, \ldots, t_n)$, $f \in \mathcal{F}_{POLO}$ and $\mathcal{A}cc(t) = \{u_1, \ldots, u_k\}$. By induction hypothesis $u_i\gamma$ is computable for all $i \in \{1, \ldots, k\}$. Now, for all $w \in \mathcal{A}cc(t\gamma)$ we have that $w \in \mathcal{A}cc(u_i\gamma)$ and, by Lemma 19, we have that $u_i\gamma \succeq_{HORPOLO} w$. Therefore, w is computable by Property 2.2, and hence $t\gamma$ is computable by Lemma 29. $\square$

As in the first-order case, we conclude with our main theorem stating that we have a reduction pair.

Theorem 3. $(\geq^*_{HORPOLO}, >^+_{HORPOLO})$ *is a higher-order reduction pair.*

Proof. We show that all required properties hold.

- By Corollary 4 we have that $\geq_{HORPOLO}$ is monotonic and hence, as the reflexive and transitive closure of a monotonic relation is also monotonic, we have that $\geq^*_{HORPOLO}$ is monotonic.
- By Corollary 5 we have that both $\geq_{HORPOLO}$ and $>_{HORPOLO}$ are stable under substitutions and hence, as the reflexive and transitive closure of a stable under substitutions relation is also stable under substitutions, we have that both $\geq^*_{HORPOLO}$ and $>^+_{HORPOLO}$ are stable under substitutions.
- By Lemma 31 we have that $>_{HORPOLO}$ is well-founded, and hence, $>^+_{HORPOLO}$ also is.
- By Lemma 17, we have that $>^+_{HORPOLO}$ is compatible with $\geq^*_{HORPOLO}$.
- Finally, we have $\succ_{HORPOLO} \subseteq >_{HORPOLO} \subseteq \geq_{HORPOLO}$, and $\rightarrow_\beta \cup \rightarrow_\eta \subset \succ_{HORPOLO}$ by cases 3 and 4c of Definition 11. Therefore both $>^+_{HORPOLO}$ and $\geq^*_{HORPOLO}$ are functional. $\square$

6 The computability closure

Like for the HORPO in [18], in order to have a useful ordering the HORPOLO has to be extended with the so called *computability closure*.

In order to ease the reading we will only consider a simplified definition of the computability closure that includes the cases that are used most often.

Definition 15. *Given a term $s = f(\overline{s})$ with $f \in \mathcal{F}_{RPO}$, we define its computability closure $\mathcal{CC}(s)$ as $\mathcal{CC}(s, \varnothing)$, where $\mathcal{CC}(s, \mathcal{V})$ for a set of variables $\mathcal{V}$ with $\mathcal{V} \cap \mathcal{V}ar(s) = \varnothing$, is the smallest set of typable terms containing all variables in $\mathcal{V}$, all terms in $\overline{s}$ and closed under the following operations:*

1. *precedence: let $g \in \mathcal{F}_{RPO}$ such that $f \succ_{\mathcal{F}} g$ and $\overline{t} \subseteq \mathcal{CC}(s, \mathcal{V})$, then $g(\overline{t}) \in \mathcal{CC}(s, \mathcal{V})$;*
2. *status: let $g \in \mathcal{F}_{RPO}$ such that $f =_{\mathcal{F}} g$, if $\overline{s}(\succ_{HORPOLO})_{stat(f)}\overline{t}$ and $\overline{t} \subseteq \mathcal{CC}(s, \mathcal{V})$ then $g(\overline{t}) \in \mathcal{CC}(s, \mathcal{V})$;*
3. *polo: let $t = g(\overline{t})$ with $g \in \mathcal{F}_{POLO}$ and $\mathcal{A}cc(t) \subseteq \mathcal{CC}(s, \mathcal{V})$, then $t \in \mathcal{CC}(s, \mathcal{V})$;*
4. *application: let $@(\overline{t})$ be a partial left-flattening of t and $\overline{t} \subseteq \mathcal{CC}(s, \mathcal{V})$, then $t \in \mathcal{CC}(s, \mathcal{V})$;*
5. *abstraction: let $x \notin \mathcal{V}ar(s) \cup \mathcal{V}$ and $t \in \mathcal{CC}(s, \mathcal{V} \cup \{x\})$, then $\lambda x.t \in \mathcal{CC}(s, \mathcal{V})$.*
6. *reduction: let $u \in \mathcal{CC}(s, \mathcal{V})$ and $u \succeq_{HORPOLO} t$, then $t \in \mathcal{CC}(s, \mathcal{V})$.*

Note that in case 2, if the status is *mul* then the condition $\overline{t} \subseteq \mathcal{CC}(s, \mathcal{V})$ already holds applying the *reduction* case and the fact that $\overline{s}\,(\succ_{HORPOLO})_{mul}\,\overline{t}$.

Now we give a modified version of $\succ_{HORPOLO}$ which includes the use of the computability closure in some of the cases with $top(s) \in \mathcal{F}_{RPO}$.

Definition 16. *$s : \sigma \succ_{HORPOLO} t : \tau$ iff $\sigma \geq_T \tau$ and*

1. *$s = f(s_1, \ldots, s_n)$ and*
   (a) *$f \in \mathcal{F}_{POLO}$ and*
      i. *τ is a data type and $I(s) >_{C(s)} I(t)$, or*
      ii. *τ is functional and $u \succ_{HORPOLO} t$ for some $u \in \mathcal{A}cc(s)$, or*
   (b) *$f \in \mathcal{F}_{RPO}$ and*
      i. *$s_i \succeq_{HORPOLO} t$ for some $i \in \{1, \ldots, n\}$, or $t \in \mathcal{CC}(s)$, or*
      ii. *$t = g(t_1, \ldots, t_m)$, $g \in \mathcal{F}_{POLO}$ and $s \succ_{HORPOLO} u$ or $u \in \mathcal{CC}(s)$ for all $u \in \mathcal{A}cc(t)$, or*
      iii. *$t = g(t_1, \ldots, t_m)$, $g \in \mathcal{F}_{RPO}$,*
         A. *$f \succ_{\mathcal{F}} g$ and for all $i \in \{1, \ldots, m\}$ we have $s \succ_{HORPOLO} t_i$ or $t_i \in \mathcal{CC}(s)$, or*
         B. *$f =_{\mathcal{F}} g$, $stat(f) = mul$ and*
         $\{s_1, \ldots, s_n\}(\succ_{HORPOLO})_{mul}\{t_1, \ldots, t_m\}$, or
         C. *$f =_{\mathcal{F}} g$, $stat(f) = lex$,*
         $\langle s_1, \ldots, s_n\rangle(\succ_{HORPOLO})_{lex}\langle t_1, \ldots, t_m\rangle$ and
         for all $i \in \{1, \ldots, m\}$ we have $s \succ_{HORPOLO} t_i$ or $t_i \in \mathcal{CC}(s)$, or
      iv. *$t = @(t_1 \ldots, t_m)$ for some partial left-flattening of t and for all $i \in \{1, \ldots, m\}$ we have $s \succ_{HORPOLO} t_i$ or $t_i \in \mathcal{CC}(s)$, or*
2. *$s = @(s_1, s_2)$ and either*
   (a) *$s_1 \succeq_{HORPOLO} t$ or $s_2 \succeq_{HORPOLO} t$, or*
   (b) *$t = @(t_1, \ldots, t_m)$ for some partial left-flattening of t and*
   $\{s_1, s_2\}(\succ_{HORPOLO})_{mul}\{t_1, \ldots, t_m\}$, or

3. $s = @(\lambda x.w, v)$ *and* $w\{x \mapsto v\} \succeq_{HORPOLO} t$*, or*
4. $s = \lambda x : \alpha.u$ *and*
   (a) $u\{x \mapsto z\} \succeq_{HORPOLO} t$ *for some fresh variable* $z : \alpha$*, or*
   (b) $t = \lambda y : \beta.v$*,* $\alpha =_T \beta$ *and* $u\{x \mapsto z\} \succ_{HORPOLO} v\{y \mapsto z\}$ *for some fresh variable* $z : \alpha$*, or*
   (c) $u = @(w, x)$*,* $x \notin \mathcal{V}ar(w)$ *and* $w \succeq_{HORPOLO} t$*,*

where $s : \sigma \succeq_{HORPOLO} t : \tau$ *iff* $s : \sigma \succ_{HORPOLO} t : \tau$ *or* $s : \sigma \sqsupseteq_{HORPOLO} t : \tau$*.*

First of all note that now the definition of $\succ_{HORPOLO}$ is also mutually recursive with the definition of $\mathcal{CC}$, and due to the Case 1(b)i of $\succ_{HORPOLO}$ and the Case 6 of $\mathcal{CC}$ we cannot use a measure on (s, t) to prove well-definedness. Therefore, in this case, we prove that the ordering is well-defined by a fixpoint computation. Note that, since no case involves negation, the definitions are monotonic with respect to set inclusion, and hence the existence of the least fixpoint is guaranteed.

A more compact and elegant way to combine the ordering with the computability closure is given in [2]. There, a single ordering, called the *computability path ordering* (CPO) is defined. Combining CPO with polynomial interpretations can be done in the same way as in this paper for HORPO.

The proofs are extended following the same approach as for HORPO in [18]. In what follows, only the proofs that differ from the previous section are provided.

We first need to prove again Lemma 19 stated for the ordering with the computability closures, whose proof has to consider two additional cases.

Lemma 32 (Revision of Lemma 19). *Let* $s :_{\mathcal{C}} \sigma$ *and* $w :_{\mathcal{C}} \tau$ *be candidate terms. Then*

(i) $s \succeq_{HORPOLO} w$ *implies* $s \succeq_{HORPOLO} u$ *for all* $u \in \mathcal{A}cc(w)$*.*
(ii) $s \succ_{HORPOLO} w$ *implies* $s \succ_{HORPOLO} u$ *for all* $u \in \mathcal{A}cc(w)$*.*

Proof. The proof proceeds as before, but due to the use of the Computability Closure in the definition of $\succ_{HORPOLO}$, two new cases must be considered when $top(w) \in \mathcal{F}_{POLO}$. Note that, by definition, if u has type τ' then we have that $\sigma \geq_T \tau \geq_T \tau'$.

- $s \succ_{HORPOLO} w$ because $w \in \mathcal{CC}(s)$ by case 16.1(b)i. We have already proved (without any use of the induction hypothesis) that $w \succeq_{HORPOLO} u$ for all $u \in \mathcal{A}cc(w)$. Therefore, by the case of *reduction* we also have that $u \in \mathcal{CC}(s)$, and since $\sigma \geq \tau'$ we conclude that $s \succ_{HORPOLO} u$ by case 16.1(b)i.
- $s \succ_{HORPOLO} w$ by case 16.1(b)ii, that is, $s \succ_{HORPOLO} u$ or $u \in \mathcal{CC}(s)$ for all $u \in \mathcal{A}cc(t)$. If $u \in \mathcal{CC}(s)$ then since $\sigma \geq \tau'$ we have that $s \succ_{HORPOLO} u$ by case 16.1(b)i and hence, we can conclude in both cases.

□

Due to the mutually recursive definition of HORPOLO and the computability closure, the stability under substitutions of $\sqsupseteq_{HORPOLO}$ and $\succ_{HORPOLO}$ has to be proved along with the stability under substitutions of the membership in $\mathcal{CC}$.

Lemma 33 (Extension of Lemma 25). *The relations $\sqsupset_{HORPOLO}$ and $\succ_{HORPOLO}$ are stable under substitutions for candidate terms, and if $u \in \mathcal{CC}(t)$ then $u\gamma \in \mathcal{CC}(t\gamma)$ for all type-preserving substitutions γ.*

Proof. We prove all three properties by induction on the four mutually recursive definitions (the context, $\sqsupset_{HORPOLO}$, $\succ_{HORPOLO}$ and $\mathcal{CC}$). The proofs of stability under substitutions of $\sqsupset_{HORPOLO}$ and $\succ_{HORPOLO}$ are as before but now use the new induction argument and the property on $\mathcal{CC}$ when needed.

Now, we prove that if $u \in \mathcal{CC}(t, \mathcal{V})$ with $\mathcal{V} \subseteq \mathcal{X} \setminus (\mathcal{V}ar(t) \cup \mathcal{V}ar(t\gamma) \cup \mathcal{D}om(\gamma))$, then $u\gamma \in \mathcal{CC}(t\gamma, \mathcal{V})$. Note that the property on $\mathcal{V}$ depends on t and γ, but not on u. It will therefore be trivially satisfied in all cases but *polo* and *abstraction*. And indeed, these are the only cases in the proof which are not routine, hence we do them in detail.

- Case 3: $u = g(\overline{u})$ with $g \in \mathcal{F}_{POLO}$ where $\mathcal{A}cc(u) \subseteq \mathcal{CC}(t, \mathcal{V})$. By induction hypothesis $\mathcal{A}cc(u)\gamma \subseteq \mathcal{CC}(t\gamma, \mathcal{V})$. We will prove that $\mathcal{A}cc(u\gamma) \subseteq \mathcal{CC}(t\gamma, \mathcal{V})$ and hence $u\gamma \in \mathcal{CC}(t\gamma, \mathcal{V})$ by Case 3. Let $v \in \mathcal{A}cc(u\gamma)$. If $v \in \mathcal{A}cc(u)\gamma$ then we are done. Otherwise $v \in \mathcal{A}cc(x\gamma)$ for some variable $x \in \mathcal{A}cc(u)$ and $top(x\gamma) \in \mathcal{F}_{POLO}$. Since $x\gamma \in \mathcal{A}cc(u)\gamma$ and $\mathcal{A}cc(u)\gamma \subseteq \mathcal{CC}(t\gamma, \mathcal{V})$ then $x\gamma \in \mathcal{CC}(t\gamma, \mathcal{V})$. Now, since by Lemma 32 $x\gamma \succeq_{HORPOLO} v$, we conclude that $v \in \mathcal{CC}(t\gamma, \mathcal{V})$ by Case 6.
- Case 5: let $u = \lambda x.s$ with $x \in \mathcal{X} \setminus (\mathcal{V} \cup \mathcal{V}ar(t))$ and $s \in \mathcal{CC}(t, \mathcal{V} \cup \{x\})$. At the price of renaming the variable x in s if necessary, we can assume in addition that $x \notin \mathcal{V}ar(t\gamma) \cup \mathcal{D}om(\gamma)$, and therefore $\mathcal{V} \cup \{x\} \subseteq \mathcal{X} \setminus (\mathcal{V}ar(t) \cup \mathcal{V}ar(t\gamma) \cup \mathcal{D}om(\gamma))$. By induction hypothesis, $s\gamma \in \mathcal{CC}(t\gamma, \mathcal{V} \cup \{x\})$. Since $x \notin \mathcal{V}ar(t\gamma) \cup \mathcal{V}$, by Case 5 of the definition $\lambda x.s\gamma \in \mathcal{CC}(t\gamma, \mathcal{V})$ and, since $x \notin \mathcal{D}om(\gamma)$, we have $u\gamma = \lambda x.s\gamma$.
- Case 6: $v \succeq_{HORPOLO} u$ for some $v \in \mathcal{CC}(t, \mathcal{V})$. Then by induction hypothesis we have $v\gamma \succeq_{HORPOLO} u\gamma$ and $v\gamma \in \mathcal{CC}(t\gamma, \mathcal{V})$, which implies $u\gamma \in \mathcal{CC}(t\gamma, \mathcal{V})$, by Case 6. □

The precise formulation of this statement arises from its forthcoming use in the proof of Lemma 35.

Lemma 34. *Assume $\overline{t} : \overline{\tau}$ is computable, as well as every term $g(\overline{s})$ with $g \in \mathcal{F}_{RPO}$, $\overline{s}$ computable and smaller than $t = f(\overline{t})$ in the ordering $(\succ_{\mathcal{F}}, (>^{+}_{HORPOLO})_{stat(f)})_{lex}$ operating on pairs $\langle f, \overline{t} \rangle$. Then every term in $\mathcal{CC}(t)$ is computable.*

Proof. We prove that $u\gamma : \sigma$ is computable for every computable substitution γ of domain $\mathcal{V}$ and every $u \in \mathcal{CC}(t, \mathcal{V})$ such that $\mathcal{V} \cap \mathcal{V}ar(t) = \varnothing$. We obtain the result by taking $\mathcal{V} = \varnothing$. We proceed by induction on the definition of $\mathcal{CC}(t, \mathcal{V})$.

For the basic case: if $u \in \mathcal{V}$, then $u\gamma$ is computable by the assumption on γ; and if $u \in \overline{t}$, we conclude by the assumption that $\overline{t}$ is computable, since $u\gamma = u$ by the assumption that $\mathcal{V} \cap \mathcal{V}ar(t) = \varnothing$. For the induction step, we discuss the successive operations to form the closure:

- Case 1: $u = g(\overline{u})$ where $\overline{u} \subseteq \mathcal{CC}(t, \mathcal{V})$. By induction hypothesis, $\overline{u}\gamma$ is computable. Since $f \succ_{\mathcal{F}} g$, $u\gamma$ is computable by our assumption that terms smaller than $f(\overline{t})$ are computable.

- Case 2: $u = g(\overline{u})$ where $f =_{\mathcal{F}} g$, $\overline{t}(\succ_{\scriptscriptstyle HORPOLO})_{stat(f)}\overline{u}$, $\overline{u} \subseteq \mathcal{CC}(t, \mathcal{V})$ and $\mathcal{V}ar(u) \subseteq \mathcal{V}ar(t)$. By induction hypothesis, $\overline{u}\gamma$ is computable. By assumption, and by stability of the ordering under substitutions, $\overline{t}\gamma(\succ_{\scriptscriptstyle HORPOLO})_{stat(f)}\overline{u}\gamma$ and hence $\overline{t}\gamma(>_{\scriptscriptstyle HORPOLO})_{stat(f)}\overline{u}\gamma$. Note that $\overline{t}\gamma = \overline{t}$ by our assumption that $\mathcal{V} \cap \mathcal{V}ar(t) = \varnothing$. Therefore $u\gamma = g(\overline{u}\gamma)$ is computable by our assumption that terms smaller than $f(\overline{t})$ are computable.
- Case 3: $u = g(\overline{u})$ with $g \in \mathcal{F}_{POLO}$ where $\mathcal{A}cc(u) \subseteq \mathcal{CC}(t, \mathcal{V})$. By induction hypothesis, for all $u' \in \mathcal{A}cc(u)$ we have $u'\gamma$ is computable. Then for all $v \in \mathcal{A}cc(u\gamma)$, if $v \in \mathcal{A}cc(u)\gamma$ then v is computable; otherwise $v \in \mathcal{A}cc(x\gamma)$ for some variable $x \in \mathcal{A}cc(u)$ and $top(x\gamma) \in \mathcal{F}_{POLO}$. By Lemma 32, $v \in \mathcal{A}cc(x\gamma)$ implies that $x\gamma \succeq_{\scriptscriptstyle HORPOLO} v$ and hence $x\gamma \geq_{\scriptscriptstyle HORPOLO} v$ which implies that v is computable by Property 2.2. Therefore, we have that $\mathcal{A}cc(u\gamma)$ is computable and hence $u\gamma$ is computable by Lemma 29.
- Case 4: by induction hypothesis and Property 2.4.
- Case 5: let $u = \lambda x.s$ with $x \notin \mathcal{V} \cup \mathcal{V}ar(t) \cup \mathcal{D}om(\gamma)$ and $s \in \mathcal{CC}(t, \mathcal{V} \cup \{x\})$. Then we have $(\mathcal{V} \cup \{x\}) \cap \mathcal{V}ar(t) = \varnothing$ and, given an arbitrary computable term w, we have that $\gamma' = \gamma \cup \{x \mapsto w\}$ is a computable substitution of domain $\mathcal{V} \cup \{x\}$. Therefore, by induction hypothesis, $s\gamma'$ is computable and, by Property 2.5, $(\lambda x.s)\gamma$ is computable as well.
- Case 6: by induction hypothesis, stability and Property 2.2. $\square$

Lemma 35. *Let $f :_{\mathcal{C}} \overline{\sigma} \to \tau \in \mathcal{F}_{RPO}$ and $t_1 :_{\mathcal{C}} \sigma_1, \ldots, t_n :_{\mathcal{C}} \sigma_n$ be a set of candidate terms. If $t_1, \ldots, t_n$ are computable then $f(t_1, \ldots, t_n) :_{\mathcal{C}} \tau$ is computable.*

Proof. The proof is done by induction on $\langle f, \{t_1, \ldots, t_n\}\rangle$ ordered lexicographically by $(\succ_{\mathcal{F}}, (>^{+}_{\scriptscriptstyle HORPOLO})_{stat(f)})_{lex}$ and is exactly like the one for Lemma 30, but applying Lemma 34, when the computability closure of $f(t_1, \ldots, t_n)$ is used. $\square$

7 Examples

In the following examples, the type ordering $\geq_T$ is defined by equating all sort symbols and considering that $\sigma \to \tau \geq \rho$ if $\tau \geq \rho$. This ordering can be easily extended to one fulfilling all conditions of Definition 8.

The constraints we show in all the examples have been automatically generated by our tool THOR (see Section 8 for more details).

Example 4. Let nat be a data type, $\mathcal{F} = \{s : [nat] \to nat, 0 : [] \to nat, dec : [nat \times nat] \to nat, grec : [nat \times nat \times nat \times (nat \to nat \to nat)] \to nat, + : [nat \times nat] \to nat, log2 : [nat \times nat] \to nat, sumlog : [nat] \to nat\}$ and $\mathcal{X} = \{x : nat, y : nat, u : nat, F : nat \to nat \to nat\}$.

Consider the following set of rules:

$$\begin{array}{ll}
dec(0,x) & \to 0 \\
dec(x,0) & \to x \\
dec(s(x),s(y)) & \to dec(x,y) \\
grec(0,y,u,F) & \to u \\
grec(s(x),s(y),u,F) & \to grec(dec(x,y),s(y),@(@(F,u),x),F) \\
0+x & \to x \\
s(x)+y & \to s(x+y) \\
 & \\
log2(s(0),0) & \to 0 \\
log2(0,s(y)) & \to s(log2(s(y),0)) \\
log2(s(0),s(y)) & \to s(log2(s(y),0)) \\
log2(s(s(x)),y) & \to log2(x,s(y)) \\
sumlog(x) & \to grec(x,s(s(0)),0,\lambda z_1:nat.\lambda z_2:nat.log2(s(z_2),0)+z_1)
\end{array}$$

The first rules define a tail recursive generalized form of the Gödel recursor where we can decrease in any given fixed amount at every recursive call. Using it we define a function that given a natural number n computes the sum of the series of the logarithm to base two of n, $n-2$, etc. Note that $log2(n,0)$ computes the the logarithm to base two of n.

In order to prove this example we need to solve, among others, the constraint below

$$GREC(s(x),s(y),u,F) > GREC(dec(x,y),s(y),@(@(F,u),x),F)$$

$$\begin{array}{lll}
dec(0,x)\geq 0 & dec(x,0)\geq x & dec(s(x),s(y))\geq dec(x,y) \\
0+x\geq x & s(x)+y\geq s(x+y) & grec(0,y,u,F)\geq u
\end{array}$$
$$grec(s(x),s(y),u,F) \ \geq \ grec(dec(x,y),s(y),@(@(F,u),x),F)$$
$$\begin{array}{ll}
log2(s(0),0)\geq 0 & log2(0,s(y))\geq s(log2(s(y),0)) \\
log2(s(0),s(y))\geq s(log2(s(y),0)) & log2(s(s(x)),y)\geq log2(x,s(y))
\end{array}$$
$$sumlog(x)\geq grec(x,s(s(0)),0,\lambda z_1:nat.\lambda z_2:nat.log2(s(z_2),0)+z_1)$$

Note that HORPO fails to prove this constraint due to the combination of the first literal (the one on terms headed by $GREC$) and the last three literals on terms headed by $log2$.

The constraint can be proved by HORPOLO taking $\mathcal{F}_{RPO}=\{GREC, grec, +, sumlog\}$ with $sumlog \succ_{\mathcal{F}} grec$, $sumlog \succ_{\mathcal{F}} +$ and all having status lex, and $\mathcal{F}_{POLO}=\{dec, log2, s, 0\}$ with $dec_I(x,y)=x$, $log2_I(x,y)=x+2{\cdot}y$, $s_I(x)=x+1$ and $0_I=0$.

In the remainder of the example we show how three of these literals are included in the reduction pair $(\succ^+_{HORPOLO}, \succeq^*_{HORPOLO})$. Note that, in fact, we just use $\succ_{HORPOLO}$ and $\succeq_{HORPOLO}$.

– $GREC(s(x),s(d),u,F) \succ_{HORPOLO} GREC(dec(x,d),s(d),@(@(F,u),x),F)$.
  We start applying case 16.1(b)iiiC. To this end we first need $\langle s(x),s(d),u,F\rangle$ $(\succ_{HORPOLO})_{lex}\langle dec(x,d),s(d),@(@(F,u),x),F\rangle$, which holds since $s(x)\succ_{HORPOLO}$

$dec(x,d)$ by case 16.1(a)i as $I(s(x)) = x+1 >_{C(s(x))} x = I(dec(x,d))$. We conclude showing that

- $s(d), F \in \mathcal{CC}(GREC(s(x), s(d), u, F))$, by the base case.
- $GREC(s(x), s(d), u, F) \succ_{HORPOLO} dec(x,d)$, which holds by case 16.1(b)ii, showing $x, d \in \mathcal{CC}(GREC(s(x), s(d), u, F))$ by the base case and case 6, since both $s(x) \succ_{HORPOLO} x$ and $s(d) \succ_{HORPOLO} d$ hold by case 16.1(b)i.
- $GREC(s(x), s(d), u, F) \succ_{HORPOLO} @(F, u, x)$, which holds by case 16.1(b)iv, since as seen $F, x \in \mathcal{CC}(GREC(s(x), s(d), u, F))$ and $u \in \mathcal{CC}(GREC(s(x), s(d), u, F))$ by the base case.

– $log2(s(s(x)), y) \succeq_{HORPOLO} log2(x, s(y))$.
We show that $t = log2(s(s(x)), y) \sqsupseteq_{HORPOLO} log2(x, s(y))$, by case 10.2a, since $I(log2(s(s(x)), y)) = x + 2y + 2 \geq_{C(t)} x + 2y + 2 = I(log2(x, s(y)))$.

– $sumlog(x) \succeq_{HORPOLO} grec(x, s(s(0)), 0, \lambda z_1 : nat.\lambda z_2 : nat.log2(s(z_2), 0) + z_1)$. We show that $sumlog(x) \succ_{HORPOLO} grec(x, s(s(0)), 0, \lambda z_1 : nat.\lambda z_2 : nat.log2(s(z_2), 0) + z_1)$ by case 16.1(b)iiiA, showing that

- $x \in \mathcal{CC}(sumlog(x))$ by the base case.
- $sumlog(x) \succ_{HORPOLO} s(s(0))$ and $sumlog(x) \succ_{HORPOLO} 0$ by case 16.1(b)ii. Note that $\mathcal{A}cc(s(s(0))) = \mathcal{A}cc(0) = \varnothing$.
- $\lambda z_1 : nat.\lambda z_2 : nat.log2(s(z_2), 0) + z_1 \in \mathcal{CC}(sumlog(x))$ by case 5 twice and showing that $log2(s(z_2), 0) + z_1 \in \mathcal{CC}(sumlog(x), \{z_1, z_2\})$, applying case 1 first and then proving $log2(s(z_2), 0) \in \mathcal{CC}(sumlog(x), \{z_1, z_2\})$ by case 3 and $z_2, z_1 \in \mathcal{CC}(sumlog(x), \{z_1, z_2\})$ by the base case. □

Example 5. The example is based on the previous one, replacing the last five rules by the following ones.

$$\begin{array}{ll}
quad(0) & \to 0 \\
quad(s(x)) & \to s(s(s(s(quad(x))))) \\
sqr(x) & \to sqrp(p(x, 0)) \\
sqrp(p(0, 0)) & \to 0 \\
sqrp(p(s(s(x)), y)) & \to sqrp(p(x, s(y))) \\
sqrp(p(0, s(y))) & \to quad(sqrp(p(s(y), 0))) \\
sqrp(p(s(0), y)) & \to quad(sqrp(p(y, 0))) + s(quad(y)) \\
sumsqr(x) & \to grec(x, s(s(0)), 0, \lambda z_1 : nat.\lambda z_2 : nat.sqr(s(z_2)) + z_1)
\end{array}$$

and $\mathcal{F}$ extended with $\{quad : [nat] \to nat,\ sqr : [nat] \to nat,\ p : [nat \times nat] \to pair,\ sqrp : [pair] \to nat,\ sumsqr : [nat] \to nat\}$.

This example computes the square of x using the recurrence $x^2 = 4(x\, div\, 2)^2$ when x is even and $x^2 = 4(x\, div\, 2)^2 + 4(x\, div\, 2) + 1$ when x is odd. Note that in the square definitions the even/odd checking is done along with the computation. To be able to handle this example we need to introduce the symbol p, which allows us to have $sqrp \in \mathcal{F}_{RPO}$ and $p \in \mathcal{F}_{POLO}$. Checking termination of these rules requires solving the constraint obtained from replacing the last five literals of the previous example by:

$$\begin{array}{ll}
quad(0) \geq 0 & quad(s(x)) \geq s(s(s(s(quad(x))))) \\
sqr(x) \geq sqrp(p(x,0)) & sqrp(p(0,0)) \geq 0 \\
\multicolumn{2}{l}{sqrp(p(s(s(x)),y)) \geq sqrp(p(x,s(y)))} \\
\multicolumn{2}{l}{sqrp(p(0,s(y))) \geq quad(sqrp(p(s(y),0)))} \\
\multicolumn{2}{l}{sqrp(p(s(0),y)) \geq quad(sqrp(p(y,0))) + s(quad(y))} \\
\multicolumn{2}{l}{sumsqr(x) \geq grec(x, s(s(0)), 0, \lambda z_1 : nat.\lambda z_2 : nat.sqr(s(z_2)) + z_1)}
\end{array}$$

Using HORPOLO, the constraint holds using the following settings: $\mathcal{F}_{RPO} = \{GREC, grec, +, sumsqr, sqr, sqrp\}$ with $sumsqr \succ_{\mathcal{F}} grec$, $sumsqr \succ_{\mathcal{F}} +$, $sumsqr \succ_{\mathcal{F}} sqr \succ_{\mathcal{F}} sqrp$ and all status *lex* and $\mathcal{F}_{POLO} = \{0, s, +, p, quad\}$ with $0_I = 0$, $s_I(x) = x + 1$, $+_I(x,y) = x + y$, $p_I(x,y) = x + 2 \cdot y$ and $quad_I(x) = 4 \cdot x$. □

The following two examples introduce several standard operations on lists. The first one computes some permutation of the tail of a given list by shuffling the elements a logarithmic number of times with respect to the first element of the list.

Example 6. Let *nat* and *list* be data types, $\mathcal{F} = \{s : [nat] \to nat,\ nil : [\,] \to list,\ app : [list \times list] \to list,\ cons : [nat \times list] \to list,\ shuffle : [list] \to list,\ rshuffle : [list] \to list,\ reverse : [list] \to list,\ hrepeat : [nat \times (list \to list) \times list] \to list,\ 0 : [\,] \to nat,\ ceilhalf : [nat] \to nat,\ tail : [list] \to list,\ head : [list] \to nat\}$ and $\mathcal{X} = \{n : nat, x : list, y : list, l : list, F : list \to list\}$.

$$\begin{array}{ll}
app(nil,l) & \to l \\
app(cons(n,l),y) & \to cons(n, app(l,y)) \\
reverse(nil) & \to nil \\
reverse(cons(n,l)) & \to app(reverse(l), cons(n,nil)) \\
shuffle(nil) & \to nil \\
shuffle(cons(n,l)) & \to cons(n, shuffle(reverse(l))) \\
ceilhalf(0) & \to 0 \\
ceilhalf(s(0)) & \to s(0) \\
ceilhalf(s(s(n))) & \to s(ceilhalf(n)) \\
hrepeat(0,F,l) & \to l \\
hrepeat(s(n),F,l) & \to hrepeat(ceilhalf(n), F, @(F,l)) \\
tail(cons(n,l)) & \to l \\
head(cons(n,l)) & \to n \\
rshuffle(l) & \to hrepeat(head(l), \lambda z.shuffle(z), tail(l))
\end{array}$$

In order to prove this example we need to solve, among others, the constraint below

$$HREPEAT(s(n),F,l) \ > \ HREPEAT(ceilhalf(n), F, @(F,l))$$

$$\begin{array}{ll}
app(nil,l) \geq l & app(cons(n,l),y) \geq cons(n, app(l,y)) \\
reverse(nil) \geq nil & reverse(cons(n,l)) \geq app(reverse(l), cons(n,nil)) \\
shuffle(nil) \geq nil & shuffle(cons(n,l)) \geq cons(n, shuffle(reverse(l)))
\end{array}$$

$\mathit{ceilhalf}(0) \geq 0 \qquad \mathit{ceilhalf}(s(0)) \geq s(0) \qquad \mathit{ceilhalf}(s(s(n))) \geq s(\mathit{ceilhalf}(n))$
$\mathit{hrepeat}(0, F, l) \geq l \qquad \mathit{hrepeat}(s(n), F, l) \geq \mathit{hrepeat}(\mathit{ceilhalf}(n), F, @(F, l))$
$\mathit{tail}(\mathit{cons}(n, l)) \geq l \qquad \mathit{head}(\mathit{cons}(n, l)) \geq n$
$\mathit{rshuffle}(l) \geq \mathit{hrepeat}(\mathit{head}(l), \lambda z.\mathit{shuffle}(z), \mathit{tail}(l))$

The system can be proved using HORPOLO where $\mathcal{F}_{RPO} = \{\mathit{rshuffle}, \mathit{HREPEAT}, \mathit{hrepeat}, \mathit{head}, \mathit{tail}\}$ with $\mathit{rshuffle} \succ_{\mathcal{F}} \mathit{hrepeat}$, $\mathit{rshuffle} \succ_{\mathcal{F}} \mathit{head}$ and $\mathit{rshuffle} \succ_{\mathcal{F}} \mathit{tail}$ and all having status lex, and $\mathcal{F}_{POLO} = \{\mathit{app}, \mathit{cons}, \mathit{nil}, \mathit{reverse}, \mathit{shuffle}, s, \mathit{ceilhalf}\}$ with $\mathit{app}_I(x, y) = x + y$, $\mathit{cons}_I(x, y) = x + y + 1$, $\mathit{nil}_I = 0$, $\mathit{reverse}_I(x) = x$, $\mathit{shuffle}_I(x) = x$, $s_I(x) = x + 1$ and $\mathit{ceilhalf}_I(x) = x$. □

The last example is somehow different, as the only strict literal of the generated constraint does not contain any application symbol.

Example 7. Let nat, $\mathit{natlist}$, plist and pair be data types, $\mathcal{F} = \{0 : [\,] \to \mathit{nat},\ s : [\mathit{nat}] \to \mathit{nat},\ \mathit{nil} : [\,] \to \mathit{natlist},\ \mathit{pnil} : [\,] \to \mathit{plist},\ \mathit{app} : [\mathit{natlist} \times \mathit{natlist}] \to \mathit{natlist},\ \mathit{cons} : [\mathit{nat} \times \mathit{natlist}] \to \mathit{natlist},\ p : [\mathit{natlist} \times \mathit{natlist}] \to \mathit{pair},\ \mathit{pcons} : [\mathit{pair} \times \mathit{plist}] \to \mathit{plist},\ \mathit{fst} : [\mathit{pair}] \to \mathit{natlist},\ \mathit{shuffle} : [\mathit{natlist}] \to \mathit{natlist},\ \mathit{reverse} : [\mathit{natlist}] \to \mathit{natlist},\ \mathit{pshuffle} : [\mathit{natlist}] \to \mathit{pair},\ \mathit{prefixshuffle} : [\mathit{pair} \times \mathit{natlist}] \to \mathit{plist},\ \mathit{pps} : [\mathit{natlist}] \to \mathit{plist}\ \mathit{apply2} : [(\mathit{pair} \to \mathit{nat} \to \mathit{pair}) \times \mathit{pair} \times \mathit{nat}] \to \mathit{pair}\}$ and $\mathcal{X} = \{n : \mathit{nat}, x : \mathit{natlist}, y : \mathit{natlist}, l : \mathit{natlist}, z : \mathit{pair}, F : \mathit{pair} \to \mathit{nat} \to \mathit{pair}\ \}$. Consider the higher-order rewrite system consisting of the rules in Example 6 for app, $\mathit{reverse}$ and $\mathit{shuffle}$ plus these ones

$$
\begin{array}{ll}
\mathit{fst}(p(x, y)) & \to x \\
\mathit{pshuffle}(l) & \to p(l, \mathit{shuffle}(l)) \\
\mathit{prefixshuffle}(z, \mathit{nil}) & \to \mathit{pcons}(z, \mathit{pnil}) \\
\mathit{prefixshuffle}(z, \mathit{cons}(n, l)) & \to \mathit{pcons}(z, \mathit{prefixshuffle}(\mathit{apply2}(\lambda x.\lambda y. \\
& \quad \mathit{pshuffle}(\mathit{app}(\mathit{fst}(x), \mathit{cons}(y, \mathit{nil}))), z, n), \mathit{reverse}(l))) \\
\mathit{apply2}(F, z, 0) & \to z \\
\mathit{apply2}(F, z, s(n)) & \to @(@(F, z), s(n)) \\
\mathit{pps}(l) & \to \mathit{prefixshuffle}(p(\mathit{nil}, \mathit{nil}), l)
\end{array}
$$

The constraint generated by the termination prover contains the following literals:

$$
\begin{array}{ll}
\mathit{Prefixshuffle}(z, \mathit{cons}(n, l)) > \mathit{Prefixshuffle}(@(\lambda x.\lambda y. & \\
\quad \mathit{pshuffle}(\mathit{app}(\mathit{fst}(x), \mathit{cons}(y, \mathit{nil}))), z, n), \mathit{reverse}(l)) & \\
\mathit{fst}(p(x, y)) & \geq x \\
\mathit{pshuffle}(l) & \geq p(l, \mathit{shuffle}(l)) \\
\mathit{prefixshuffle}(z, \mathit{nil}) & \geq \mathit{pcons}(z, \mathit{pnil}) \\
\mathit{prefixshuffle}(z, \mathit{cons}(n, l)) & \geq \mathit{pcons}(z, \mathit{prefixshuffle}(\mathit{apply2}(\lambda x.\lambda y. \\
& \quad \mathit{pshuffle}(\mathit{app}(\mathit{fst}(x), \mathit{cons}(y, \mathit{nil}))), z, n), \mathit{reverse}(l))) \\
\mathit{apply2}(F, z, 0) & \geq z \\
\mathit{apply2}(F, z, s(n)) & \geq @(@(F, z), s(n)) \\
\mathit{pps}(l) & \geq \mathit{prefixshuffle}(p(\mathit{nil}, \mathit{nil}), l)
\end{array}
$$

and the ones for *app*, $reverse$ and *shuffle* given in the Example 6.

The system can be proved using HORPOLO where $\mathcal{F}_{POLO} = \{shuffle, cons, pcons, app, reverse, nil, pnil, fst\}$ with $shuffle_I(x, y) = x + y$, $cons_I(x, y) = x + y + 1$, $pcons_I(x, y) = y$, $app_I(x, y) = x + y$, $reverse_I(x) = x$, $nil_I = 0$, $pnil_I = 0$ and $fst_I(x) = x$, and where $\mathcal{F}_{RPO} = \{pshuffle, p, prefixshuffle, Prefixshuffle, pps, apply2\}$, with $pps \succ_{\mathcal{F}} prefixshuffle \succ_{\mathcal{F}} pshuffle \succ_{\mathcal{F}} p$, $Prefixshuffle \succ_{\mathcal{F}} pshuffle \succ_{\mathcal{F}} apply2$ and all with status lex (right-to-left). □

8 Implementation and experiments

HORPOLO has been implemented as base ordering in THOR-1.0 [5], a higher-order termination prover based on the monotonic higher-order semantic path ordering [7].

The implementation of HORPOLO is done by translating the ordering constraints $s > t$ and $s \geq t$ into problems in SAT modulo non-linear integer arithmetic (NIA) which are handled by the Barcelogic [3,5] SMT-solver.

In order to perform an evaluation of the experiments we have used three new different versions of our tool. The first one, called "DISJOINT" in Figure 3, can use both HORPO and polynomial interpretations (POLO), but in a disjoint way. Note that this version already properly extends the old version of THOR that only uses HORPO. This way we have been able to check that the examples shown in Section 7 (4, 5, 6 and 7) can only be proved using HORPOLO. The other two are versions using HORPOLO. The first one, called "HORPOLO", implements the ordering using the context $C(s)$ when comparing polynomial interpretations while the second one, called "HORPOLO-NC", implements the ordering without the context. The first one is, in principle, more powerful. However, all examples can also be proved with the second one, which is more efficient, as can be seen in Figure 3. Note that, although the context is necessary to ensure monotonicity and stability under substitutions of the ordering, the relation obtained from removing the context is included in the original relation, which makes it suitable for proving termination.

For our experiments, we are using the problem database TPDB8.1, which is the one used in the recent 2011 Termination Competition. There are three families in the Higher-Order category, which altogether include 156 problems. We have also considered a fourth category called "HORPOLO-need" which has got our four new examples.

We have performed the experiments in Figure 3 on a 2.4 GHz 2.9 GB Intel Core Duo with a 32-bit architecture. Runtimes are given in seconds. The table contains a row for each problem set, which displays for every tool version the number of instances that can be proved terminating (YES) and the ones that cannot (MAYBE) together with their respective solving times.

The analysis of the results on the considered benchmarks shows that the time consumption using HORPOLO is comparable to all other variants. Although most of the examples do not involve very complex constraints, there are

[5] See `http://www.lsi.upc.edu/~albert/term.html`

	DISJOINT				HORPOLO				HORPOLO-NC			
	YES		MAYBE		YES		MAYBE		YES		MAYBE	
	#	Time	#	Time	#	Time	#	Time	#	Time	#	Time
Kop_11	2	0.325	4	0.110	2	0.326	4	0.123	2	0.314	4	0.112
Mixed_HO_10	26	2.993	14	0.445	26	3.207	14	0.495	26	3.045	14	0.479
U_A_11	67	26.246	43	5.801	67	28.610	43	6.289	67	26.253	43	5.885
HORPOLO-need	0	0	4	2.166	4	4.104	0	0	4	3.652	0	0

Fig. 3. Experimental results with THOR-1.0

enough instances to conclude that the implementation of HORPOLO is feasible in practice.

Moreover, we have to mention that the reason why HORPOLO does not improve the results on the already existing examples in the TPDB is due to the fact that most of them contain many first-order rules combined with a small higher-order part, which is mainly handled in the constraint generation and simplification process. Therefore the constraints sent to HORPOLO do not require the combined ordering. However, our four new examples show that it is not that hard to find situations where the HORPOLO is needed, and hence probably in the near future if the database is extended with more and larger examples some more constraints needing HORPOLO will appear.

Finally, we have tried to prove termination of our four new examples using WANDA [20]. In fact, we have tried with two different versions of it. The first one is a release of 2010 and the second one is a release of 2011, both available at the author's web page. Both versions use different techniques described, for instance, in [19], but finally rely on HORPO for solving the generated constraints on higher-order terms. Both versions of WANDA fail to prove termination of all four examples. However, we have been able to check that the constraints that cause the failure of WANDA 2011 could be proved using HORPOLO, which shows that the new ordering is not only useful in our tool.

9 Conclusions

In this paper we have shown a new way of combining polynomial interpretation based orderings with RPO-like orderings. Terms that have to be compared with RPO are abstracted and replaced by variables when the polynomial interpretation is applied. The relations between the introduced variables are kept in a context that is used when comparing the interpretations of the terms. This way we can have monotonicity properties without needing any kind of interpretation on symbols that are handled with RPO.

In the second part of the paper the ordering is extended to the higher-order case. Polynomial interpretations in the higher-order case have only been successfully used in [23] and implemented in [10]. In our approach we combine polynomial interpretations with HORPO, obtaining an automatable technique, which has been successfully implemented within our tool THOR. Finally, in order to avoid some weaknesses of the ordering in presence of functional type terms, we have introduced an additional technique, called computability closure, which is directly inherited from HORPO. Several examples illustrating the power of the presented techniques have been given.

As future work, we plan to combine the polynomial interpretations considered in [23] and [10] with our combined method, since this would allow one to decide whether the application symbol should be handled by the polynomial interpretation or by the RPO part of the ordering, getting the best of both methods.

Finally, it would be interesting to study whether our results can be extended to handle *matrix interpretations* [16,9], which have recently been adopted in automated tools as an alternative to RPO and polynomial interpretations for solving ordering constraints.

Acknowledgments:
We would like to sincerely thank the referees for the very useful feedback we have obtained in the revision process and especially to Cynthia Kop for her outstanding revision effort which has definitely helped us to improve the quality of the paper and to detect some problems in earlier versions.

References

1. T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs. *Theoretical Computer Science*, 236(1-2):133-178, 2000.
2. F. Blanqui, J.-P. Jouannaud and A. Rubio. The Computability Path Ordering: The End of a Quest. In *Proceedings of the 22nd International Workshop Computer Science Logic (CSL) and 17th Annual Conference of the EACSL*, volume 5213 of *LNCS*, pages 1–14, Bertinoro (Italy), September 2008. Springer.
3. M. Bofill, R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell and A. Rubio. The Barcelogic SMT Solver. In *Proceedings of the 20th International Conference on Computer Aided Verification (CAV)*, volume 5123 of *LNCS*, pages 294–298, Princeton (USA), July 2008. Springer.
4. C. Borralleras, M. Ferreira and A. Rubio. Complete monotonic semantic path orderings. In *Proceedings of the 17th International Conference on Automated Deduction (CADE)*, volume 1831 of *LNAI*, pages 346–364, Pittsburgh (USA), June 2000. Springer.
5. C. Borralleras, S. Lucas, E. Rodríguez-Carbonell, A. Oliveras and A. Rubio. SAT Modulo Linear Arithmetic for Solving Polynomial Constraints. *Journal of Automated Reasoning*. To appear in print. Springer. Published on-line in September 2010.
6. C. Borralleras and A. Rubio. Orderings and Constraints: Theory and Practice of Proving Termination. In *Rewriting, Computation and Proof, Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, volume 4600 of *LNCS*, pages 28–43, 2007. Springer.
7. C. Borralleras and A. Rubio. A Monotonic Higher-Order Semantic Path Ordering. In *Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence (LPAR)*, volume 2250 of *LNAI*, pages 531–547, La Havana (Cuba), December 2001. Springer.
8. E. Contejean, C. Marché, B. Monate and X. Urbain. Proving termination of rewriting with CiME. In *Proceedings of the 6th International Workshop on Termination (WST)*, Technical Report DSIC II/15/03, pages 71–73, Valencia (Spain), June 2003.

9. J. Endrullis, J. Waldmann and H. Zantema. Matrix interpretations for proving termination of term rewriting. *Journal of Automated Reasoning*, 40(23):195220, 2008.
10. C. Fuhs and C. Kop. Polynomial Interpretations for Higher-Order Rewriting. In *Proceedings of the 23rd International Conference on Rewriting Techniques and Applications (RTA)*, volume 15 of LIPIcs, Nagoya (Japan), May 2012. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
11. J. Gallier. On Girard's "candidats de reductibilité". In *Logic and Computer Science*, pages 123–203. Academic Press, 1990.
12. J. Giesl, R. Thiemann and P. Schneider-Kamp. The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs. In *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence (LPAR)*, volume 3452 of *LNAI*, pages 301–331, Montevideo (Uruguay), March 2005. Springer.
13. J. Giesl, P. Schneider-Kamp and R. Thiemann. AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR)*, volume 4130 of *LNAI*, pages 281–286, Seattle (USA), August 2006. Springer.
14. J.-Y. Girard, Y. Lafont and P. Taylor. Proofs and Types. *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.
15. N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205(4):474-511, 2007.
16. D. Hofbauer and J. Waldmann. Termination of string rewriting with matrix interpretations. In *Proc. 17th International Conference on Rewriting Techniques and Applications (RTA 2006)*, LNCS 4098, pp. 328–342, 2006.
17. J.-P. Jouannaud and M. Okada. A computation model for executable higher-order algebraic specification languages. In *Proceedings of the 6th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 350–361, Amsterdam (Netherlands), July 1991. IEEE Computer Society Press.
18. J.-P. Jouannaud and A. Rubio. Polymorphic higher-order recursive path orderings. *Journal of the ACM*, 54(1):1-48, 2007.
19. C. Kop and F. van Raamsdonk. Higher Order Dynamic Dependency Pairs for Algebraic Functional Systems. In *Proceedings of the 22nd International Conference on Rewriting Techniques and Applications (RTA)*, volume 10 of LIPIcs, pages 203–218, Novi Sad (Serbia), May 2011. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
20. C. Kop. WANDA, a higher order termination tool. `http://www.few.vu.nl/~kop/code.html`.
21. S. Lucas. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In *Proceedings of 15h International Conference on Rewriting Techniques and Applications (RTA)*, volume 3091 of *LNCS*, pages 200-209, Aachen (Germany), June 2004. Springer. `http://zenon.dsic.upv.es/muterm`.
22. J. van de Pol. Termination proofs for higher-order rewrite systems. In *Proceedings of First Int. Workshop Higher-Order Algebra, Logic, and Term Rewriting (HOA'93)*, volume 816 of *LNCS*, pages 305-325, 1994. Springer.
23. J. van de Pol. Termination of Higher-order Rewrite Systems. PhD. Thesis, Utrecht University, 1996.
24. S. Suzuki, K. Kusakari and F. Blanqui. Argument filterings and usable rules in higher-order rewrite systems. *IPSJ Transactions on Programming*, 4(2):1-12, 2011.