The recursive path and polynomial ordering*

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.

We present a simple ordering that combines both RPO and POLO and defines a family of orderings that includes both, and extends 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 an automatable use of polynomial interpretations in combination with beta-reduction.

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 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 in a sequential way, first trying one method (maybe under some time limit) and, in case of failure, trying the other one afterwards.

As an alternative to this sequential application 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 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.

Matrix interpretations [8, 7], have recently been adopted as the third alternative to define term orderings. As future work, it would be interesting to study if our results can be generalized to matrix interpretations and to more general interpretations fulfilling some required properties[1].

* This work has been partially supported by the Spanish MEC/MICINN under grants TIN2008-04547 and TIN 2010-68093-C02-01

[1] We would like to thank the anonymous referee that pointed us this possible extension

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.

Due to this, we show its practical usefulness by extending it, using the same techniques as for the higher-order recursive path ordering [10] (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 [9] (AFS), and provides an automatable termination method that allows the user to have polynomial interpretations on some symbols in a higher-order setting. Note that, although some polynomial interpretations for higher-order rewrite systems à la Nipkow where extensively studied in [11], it is unclear how to implement those techniques in an automatic tool.

Due to the space limitations we have not included here the definitions of the higher-order version of the ordering, but it is the natural extension following the same ideas applied to extend RPO to HORPO (for a full version of this work see [3]).

2 The recursive path and polynomial ordering (RPOLO)

Here we present the ordering for first-order terms. Let $\mathcal{F}$ be a signature split into two sets $\mathcal{F}_{POLO}$ and $\mathcal{F}_{RPO}$. We have 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 $\mathcal{T}(\mathcal{F}, \mathcal{X})$. 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 $\mathbb{Z}^+$. 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.

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 compatible quasi-ordering $\sqsupseteq_{RPOLO}$.

▶ **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 $top(v) \in \mathcal{F}_{RPO}$.

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.

▶ **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 $s \sqsupseteq_{RPOLO} t$ and $\mathcal{A}cc(s)$ is defined as $\{u \mid x_u \in \mathcal{V}ar(I(s))\}$ and, to ease the reading, $C(s)$ denotes $C(\mathcal{A}cc(s))$.

Note that ordering keeps the flavor of the RPO definition, but adding some cases to handle the terms headed by polynomially interpreted symbols.

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 5.** Consider the following constraint consisting of three literals:

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

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 constraints cannot be proved terminating by an integer polynomial interpretation either.

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, applying case 4.2(c)ii, we need to prove $\{f(g(g(x)), y), x\}(\succ_{RPOLO})_{mul}\{f(g(y), x), f(g(y), x)\}$, 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 6.** Consider the following constraint coming from a termination proof:

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

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

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

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 we have $f(s(x),y) \succ_{RPOLO} s(f(x,y))$ and $I(s(f(x,y))) = x_{f(x,y)} + 1$. ◀

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

HORPOLO has been implemented as base ordering in THOR-1.0 [2], a higher-order termination prover based on the monotonic higher-order semantic path ordering [6].

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 is handled by the Barcelogic [2, 5] SMT-solver.

Just to hint on the power of the extension of the ordering to the higher-order case we provide an example that cannot be proved with other existing methods.

▶ **Example 7.** 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:

$$\begin{array}{lcl}
dec(0,x) & \to & 0 \\
dec(x,0) & \to & x \\
dec(s(x),s(y)) & \to & dec(x,y) \\
grec(0,d,u,F) & \to & u \\
grec(s(x),s(y),u,F) & \to & grec(dec(x,y),s(y),@(@(F,u),x),F)
\end{array}$$

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

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

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, the rules compute the square root 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 $p \in \mathcal{F}_{POLO}$.

Some more examples as well as all details and proofs can be found in [3].

References

1 T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs. *Theoretical Computer Science*, 236(1-2):133-178, 2000.

2 M. Bofill, R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell and A. Rubio. The Barcelogic SMT Solver. In *Proc. 20th International Conference on Computer Aided Verification (CAV)*. Springer LNCS 5123, pp. 294–298, 2008.

3 M. Bofill, C. Borralleras, E. Rodríguez-Carbonell, and A. Rubio. The recursive path and polynomial ordering for first-order and higher-order terms. Journal submission, 2011.

4 C. Borralleras, M. Ferreira and A. Rubio. Complete monotonic semantic path orderings. In *Proc. 17th International Conference on Automated Deduction (CADE)*. Springer, LNAI 1831, pp. 346–364, 2000.

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*. Springer, 2012.

6 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.

7 J. Endrullis, J. Waldmann and H. Zantema. Matrix interpretations for proving termination of term rewriting. *Journal of Automated Reasoning*, 40(2–3):195–220, 2008.

8 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.

9 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)*, pp. 350–361. IEEE Computer Society Press, 1991.

10 J.-P. Jouannaud and A. Rubio. Polymorphic higher-order recursive path orderings. *Journal of the ACM*, 54(1):1-48, 2007.

11 J. van de Pol. Termination of Higher-order Rewrite Systems. PhD. Thesis, Utrecht University, 1996.