Another Proof for the Recursive Path Ordering

Nachum Dershowitz

School of Computer Science, Tel Aviv University
Ramat Aviv, Israel
nachum.dershowitz@cs.tau.ac.il

Abstract

Yet another proof of well-foundedness of the (multiset) recursive path ordering is provided.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases Path orderings, recursive path ordering, well-foundedness, termination, order types

1 Introduction

The *recursive path ordering* [3] is a popular family of well-founded orderings on terms (trees), used for proving termination of functional programs (e.g. [8]) and rewrite systems (e.g. [11, 9, 2]) and for guiding completion procedures and theorem provers (e.g. [13]). See [4].

We give a new proof of its well-foundedness in what follows. Some previous proof approaches may be found in [3, 15, 10, 12, 1, 5]. Though some of the orderings in these references differ, when function symbols are totally ordered, they all coincide [16]. So a proof of one is a proof of all.

2 Basics

Let F be a set of symbols, let $A, G \subseteq F$, and let $T_G(A)$ denote the (finite ordered) trees constructed with leaves (atoms) taken from A and (internal) nodes from G. We are also given a well-founded partial ordering $>$ on F (a *precedence*); *we will assume throughout that each leaf in A is greater in this ordering than every node in G.*

Let $T^n_G(A)$ denote those trees in $T_G(A)$ in which the (maximum) nesting of *maximal* nodes in G is at most n. So

$$T_F(A) \;=\; \bigcup_{n=0}^{\infty} T^n_F(A)$$

We refer to n as the *altitude* of those trees that are in $T^n_G(A) \setminus T^{n-1}_G(A)$.

It is convenient to let

$$\max B \;=\; \{f \in F : \nexists\, g \in B.\, g > f\}$$

be all the *maximal* elements of partially ordered set B and

$$^{<}B \;=\; B \setminus \max B \;=\; \{f \in F : \exists g \in B.\, g > f\}$$

be all the rest (the non-maximal elements in B). It can be that $B = {}^{<}B$ when B has no maximal elements ($\max B = \varnothing$), as for the natural numbers, for example.

Bags (finite multisets)

$$\mathcal{M}(A) \;=\; \{\wr a_1, \ldots, a_\ell \int : a_1, \ldots, a_\ell \in A,\; \ell \in \mathbb{N}\}$$

of elements of well-founded A are known to be well-founded under the *bag* (multiset) ordering [7]. The bag ordering $\gg$ on $A \cup \mathcal{M}(A)$ may be defined as the transitive closure of the following rules:

$$\frac{}{\{\!\{a\}\!\} \gg a} \qquad \frac{a > b_1, \ldots, b_\ell}{\{\!\{a\}\!\} \gg \{\!\{b_1, \ldots, b_\ell\}\!\}} \qquad \frac{\{\!\{a_1, \ldots, a_k\}\!\} \gg \{\!\{b_1, \ldots, b_\ell\}\!\}}{\{\!\{c, a_1, \ldots, a_k\}\!\} \gg \{\!\{c, b_1, \ldots, b_\ell\}\!\}}$$

$k, \ell \geq 0$, $a, a_i, b_j, c \in A$. This makes a bag bigger than each of its elements.

We also allow colored bags, like red bags $\{\!\{1, 1, 3\}\!\} \in \mathcal{M}_r(\mathbb{N})$ and blue bags $\{\!\{a, c, c\}\!\} \in \mathcal{M}_b(a..z)$, which are incomparable. The ordering rules are color specific.

Given a partial ordering $\gtrdot$ on $F = A \cup G$, the original (multiset) path ordering $>$ on $T_G(A)$ is the transitive closure of the following recursive rules:

$$\frac{}{f(\ldots, a_i, \ldots) > a_i} \qquad \frac{f \gtrdot g,\ \ f(a_1, \ldots, a_k) > b_1, \ldots, b_\ell}{f(a_1, \ldots, a_k) > g(b_1, \ldots, b_\ell)} \qquad \frac{\{\!\{a_1, \ldots, a_k\}\!\} \gg \{\!\{b_1, \ldots, b_\ell\}\!\}}{f(a_1, \ldots, a_k) > f(b_1, \ldots, b_\ell)}$$

$k, \ell \in \mathbb{N}$, $f, g \in F$, $a_i, b_j \in T_G(A)$. The bags in the last rule are compared recursively in the presently defined ordering.

In the next section, we propose an alternative definition for this path ordering. The idea is to transform trees before comparing by turning each subtree rooted in a node labeled by maximal symbol into a leaf containing the bag of that node's children.

It pays to recall some properties of the original definition: The following facts hold for the path ordering $>$:

- If a tree s contains a symbol (node or leaf) that is larger than every symbol in another tree t, then $s > t$.
- A leaf s is bigger than a non-leaf tree t iff s is bigger than all leaves (and nodes) of t. (As stated earlier, we are presuming that all leaves are bigger than all nodes.)
- A leaf s is smaller than a non-leaf tree t iff s is smaller than or the same as some leaf (or node) of t.

3 Repackaging

We deal here with the case where F is totally (well-) ordered. To compare trees over F, take each maximal subtree rooted in the maximal element in F and turn it into a bag of lower trees. Let this operation on a tree t be denoted $\widehat{t}$. If g is the largest node in $t = u[v_1, \ldots, v_k]$, which can be decomposed into a "cap" u not containing g and subtrees $v_1, \ldots, v_k$ each headed by g, then $\widehat{t} = u[a_1, \ldots, a_k]$ where each a_i is a leaf labeled by the bag of immediate subtrees of v_i. The nodes of $\widehat{t}$ are all smaller than g. The new leaves of $\widehat{t}$ contain only strictly lower trees than the original t.

To compare trees s and t, one first sees which has the largest leaf, then which has the largest node. Those being equal, this is followed by comparing the decomposed trees $\widehat{s}$ and $\widehat{t}$, with the new leaves made larger than the remaining node labels.

Let B_n be short for $T^n_G(A)$ and let $\mathcal{M}^+(B_n) = \mathcal{M}(B_n) \cup A$ be bags of these trees plus leaves A, with leaves ordered above than these bags. Trees $T_G(A) = \bigcup_{n=0}^{\infty} B_n$ are viewed and compared inductively as follows:

$$\begin{aligned} B_0 &= T_{\lessdot G}(A) && \text{— provided } {}^{\lessdot}G \subsetneq G && (1) \\ B_{n+1} &= \mathcal{M}^+_g(B_n) \times G \times T_{\lessdot G}\big(\mathcal{M}^+_g(B_n)\big) && \text{— where } g = \max G && (2) \\ B_0 &= \bigcup_{H \lessdot G} T_H(A) && \text{— if } \max G = \varnothing && (3) \end{aligned}$$

where the $H \lessdot G$ are proper initial segments of ordered G. (The initial segments of $\mathbb{N}$ are $[0..i]$ for all $i \in \mathbb{N}$, for example.) The new leaves $\mathcal{M}_g(B_n)$ are placed below A and above ${}^{<}G$ in the leaf ordering.

1. The first case is just two ways of saying that all nodes are non-maximal in G.
2. The second means that a tree of altitude $n+1$ can be viewed instead as a tree built from smaller nodes and from leaves that are bags of lower trees B_n, first comparing the trees' maximal leaves and nodes and then the trees themselves.
3. What if G has no maximal element and so $G = {}^{<}G$, in which case the first case doesn't apply? Still each tree has a maximal element, and so any two trees may be compared according to the ordering of trees with nodes up to the largest node in either. (Technically, the first case is subsumed by this one.)

▸ **Example 1.** Consider binary trees built from leaves $A = a > b > c$ and internal nodes $G : \uparrow\, > \times > +$. The trees $T^1_G(A)$ of altitude one include $(b \uparrow b) + (a \times (b+c))$ and $(b \uparrow b) + ((a \times b) + (a \times c))$—think distributivity. They both have the same maximal node $\uparrow$ and the same maximal leaf a. Applying the above decomposition yields

$$\boxed{\Lbag b, b \Rbag} + (a \times (b+c))$$

and

$$\boxed{\Lbag b, b \Rbag} + ((a \times b) + (a \times c))$$

respectively, where each box is a leaf. Now they both have the same maximal node $\times$ and the same maximal leaf a as before. The next decompositions are

$$\boxed{\Lbag b, b \Rbag} + \boxed{\Lbag a, b+c \Rbag}$$

and

$$\boxed{\Lbag b, b \Rbag} + \left(\boxed{\Lbag a, b \Rbag} + \boxed{\Lbag a, c \Rbag} \right)$$

with maximal leaf $\Lbag b, b \Rbag$. One more step gives

$$\boxed{\left\Lbag \boxed{\Lbag b, b \Rbag}, \boxed{\Lbag a, b+c \Rbag} \right\Rbag}$$

and

$$\boxed{\left\Lbag \boxed{\Lbag b, b \Rbag}, \boxed{\Lbag a, b \Rbag} + \boxed{\Lbag a, c \Rbag} \right\Rbag}$$

To compare these two leaves, we compare the bags

$$\left\Lbag \boxed{\Lbag b, b \Rbag}, \boxed{\Lbag a, b+c \Rbag} \right\Rbag$$

and

$$\left\Lbag \boxed{\Lbag b, b \Rbag}, \boxed{\Lbag a, b \Rbag} + \boxed{\Lbag a, c \Rbag} \right\Rbag$$

The first is larger since its leaf element $\Lbag a, b+c \Rbag$ is larger than the leaves $\Lbag a, b \Rbag$ and $\Lbag a, c \Rbag$ of the tree $\Lbag a, b \Rbag + \Lbag a, c \Rbag$. That is because $b+c$ is bigger than both b and c.

To see that the new version is well-founded, consider an infinite descending sequence and reason by induction on maximal node and altitude. Since the leaves and nodes are well-ordered, the maximal leaf and maximal node stabilize from some point on. Look at the decompositions of those trees, all of whose nodes are strictly smaller than the maximal node in the original sequence. So, by induction, the old and new leaves are well-ordered, and hence that sequence of decompositions must in fact be finite.

The new tree ordering on $T_F(\{\top\})$ (starting with maximal leaves) is identical to the original path ordering. Any tree can be put in this form by sprouting a $\top$-leaf from each original leaf.

4 Discussion

We are hopeful that our redefinition of the recursive path ordering may facilitate extensions beyond Γ_0, which are of value for demonstrating termination of "non-simplifying" rewriting systems. The reason is that the definition given here bears a measure of similarity to the ordering of ordinal diagrams; see [14, 6].

When the ordering on nodes is partial, there may be more than one maximal node. Each should get its own incomparable bag of shallower trees. The details remain to be worked out.

References

1 Frédéric Blanqui, Jean-Pierre Jouannaud, and Albert Rubio. The computability path ordering: The end of a quest. In Michael Kaminski and Simone Martini, editors, *Proceedings of the 17th Annual Conference on Computer Science Logic (CSL, Bertinoro, Italy)*, volume 5213 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2008. Available at `http://www.lsi.upc.edu/~albert/papers/csl08.pdf` (viewed May 20, 2014).

2 Evelyne Contejean, Pierre Courtieu, Julien Forest, Olivier Pons, and Xavier Urbain. Automated certified proofs with CiME 3. In Manfred Schmidt-Schauß, editor, *Proceedings of the 22nd International Conference on Rewriting Techniques and Applications (RTA, Novi Sad, Serbia)*, volume 10 of *LIPIcs*, pages 21–30. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011. Available at `http://cedric.cnam.fr/fichiers/art_2044.pdf` (viewed May 20, 2014).

3 Nachum Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982. Available at `http://nachum.org/papers/Orderings4TRS.pdf` (viewed May 20, 2014).

4 Nachum Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1&2):69–115, February/April 1987. Available at `http://nachum.org/papers/termination.pdf` (viewed May 20, 2014).

5 Nachum Dershowitz. Jumping and escaping: Modular termination and the abstract path ordering. *Theoretical Computer Science*, 464:35–47, 2012. Available at `http://nachum.org/papers/Toyama.pdf` (viewed May 20, 2014).

6 Nachum Dershowitz. Ordinal path orderings. In *Informal Proceedings of the 13th International Workshop on Termination (WST 2013, Bertinoro, Italy)*, pages 31–35, August 2013. Available at `http://nachum.org/papers/OPO.pdf` (viewed May 20, 2014).

7 Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Communications of the ACM (CACM)*, 22(8):465–476, August 1979.

8 Jürgen Giesl. Termination analysis for functional programs using term orderings. In Alan Mycroft, editor, *Proceedings Second International Symposium on Static Analysis (SAS '95, Glasgow, UK)*, volume 983 of *Lecture Notes in Computer Science*, pages

154–171. Springer, 1995. Available at `http://verify.rwth-aachen.de/giesl/papers/SAS-report.ps` (viewed May 20, 2014).

9 Nao Hirokawa and Aart Middeldorp. Tsukuba Termination Tool. In *Proceedings of the International Conference on Rewriting Techniques and Applications (RTA)*, 2003. Available at `http://colo6-c703.uibk.ac.at/ttt/rta03.pdf` (viewed May 20, 2014).

10 Jean-Pierre Jouannaud, Pierre Lescanne, and Fernand Reinig. Recursive decomposition ordering. In Dines Bjørner, editor, *Proceedings of the Second IFIP Workshop on Formal Description of Programming Concepts (Garmisch-Partenkirchen, West Germany)*, pages 331–348. North-Holland, Amsterdam, June 1982.

11 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, and Stephan Falke. Automated termination proofs with AProVE. In *Proceedings of the International Conference on Rewriting Techniques and Applications (RTA)*, 2004. Available at `http://verify.rwth-aachen.de/giesl/papers/RTA04-distribute.pdf` (viewed May 20, 2014).

12 Jan Willem Klop, Vincent van Oostrom, and Roel C. de Vrijer. Iterative lexicographic path orders. In Kokichi Futatsugi, Jean-Pierre Jouannaud, and José Meseguer, editors, *Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, volume 4060 of *Lecture Notes in Computer Science*, pages 541–554. Springer, 2006. Available at `http://www.cs.vu.nl/~tcs/trs/ilpogoguen.pdf` (viewed May 19, 2014).

13 William McCune. Prover9 manual. `http://www.cs.unm.edu/~mccune/prover9/manual/2009-11A` (viewed May 19, 2014).

14 Mitsuhiro Okada. A simple relationship between Buchholz's new system of ordinal notations and Takeuti's system of ordinal diagrams. *The Journal of Symbolic Logic*, 52(3):577–581, 1987.

15 David A. Plaisted. A recursively defined ordering for proving termination of term rewriting systems. Report R-78-943, Department of Computer Science, University of Illinois, Urbana, IL, September 1978. Available at `https://ia601701.us.archive.org/9/items/recursivelydefin943plai/recursivelydefin943plai.pdf` (viewed May 19, 2014).

16 Michael Rusinowitch. Path of subterms ordering and recursive decomposition ordering revisited. In Jean-Pierre Jouannaud, editor, *Rewriting Techniques and Applications*, volume 202 of *Lecture Notes in Computer Science*, pages 225–240. Springer, Berlin, 1985.