# Dependency Pairs are a Simple Semantic Path Ordering

Nachum Dershowitz

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

**Abstract**

We explicate the relation between the older semantic path ordering of Kamin and Lévy and the newer dependency-pair method of Arts and Giesl.

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

**Keywords and phrases** Termination, semantic path orderings, dependency pairs

## 1 Introduction

As pointed out by Cristina Borralleras in her dissertation [5, Thm. 7.3.2] (see also [4, Section 5.4]), the dependency-pair method of Thomas Arts and Jürgen Giesl [1] is actually a special case of the semantic path ordering of Sam Kamin and Jean-Jacques Lévy [14]. We expand and elaborate on that relationship in what follows.

## 2 The Semantic Path Ordering

Let irreflexive $\gtrdot$ and reflexive $\geq$ be two binary relations on terms such that their combined "modulo" relation $\gtrdot/\geq$ ($= \geq^* \circ \gtrdot \circ \geq^*$) is terminating (in the sense of [2]). This means that there is no sequence of terms

$$s_0 \geq \cdots \geq s_0' \gtrdot s_1 \geq \cdots \geq s_1' \gtrdot s_2 \geq \cdots \geq s_2' \gtrdot \cdots$$

containing *infinitely* many $\gtrdot$ steps and is equivalent to saying that the modulo relation's transitive closure $(\gtrdot/\geq)^+$ is well-founded. Let's refer to this relation $\gtrdot/\geq$ as the *semantic* ordering. Typically, but not exclusively, it is defined via a homomorphism from terms to some well-founded domain.

Despite the deliberately misleading choice of symbols, $\geq$ need not be the reflexive-closure of $\gtrdot$. But if it is, then $\gtrdot/\geq$ is terminating if (and only if) $\gtrdot$ is.

If both relations are transitive and also are compatible with each other (meaning that either $\geq \circ \gtrdot \subseteq \gtrdot$ or $\gtrdot \circ \geq \subseteq \gtrdot$; see [14, p. 14] and [1, n. 5]), and provided $\gtrdot$ is itself well-founded, then the above termination condition holds. We prefer, however, not to bother requiring transitivity.

Let $\gtrdot$ and $\geq$ be as above and let $\blacktriangleright$ denote the *immediate* subterm relation. We can define a very simple semantic path ordering $\succ/\succsim$, with (base) semantic ordering $\gtrdot/\geq$, as follows:

$$\frac{\exists s_i.\ s \blacktriangleright s_i \succsim t}{s \succ t,\ s \succsim t} \qquad \frac{s \gtrdot t,\ \forall t_j \blacktriangleleft t.\ s \succ t_j}{s \succ t,\ s \succsim t} \qquad \frac{s \geq t,\ \forall t_j \blacktriangleleft t.\ s \succ t_j}{s \succsim t}$$

If $\doteq$ is the intersection $\geq \cap \leqslant$ and $\approx$ is $\succsim \cap \precsim$, then it follows that

$$\frac{s \doteq t,\ \forall t_j \blacktriangleleft t.\ s \succ t_j, \forall s_i \blacktriangleleft s.\ t \succ s_i}{s \approx t}$$

This is in essence the semantic path ordering of [14] (cf. [11, Def. 4]) over the semantic relation $\gtrsim$ with a trivial (empty) functional (lifting the ordering from subterms to terms), so that there is no recursion on subterms. We have added $\gtrsim$ to the definition (in particular, in the third case) in the obvious way. (The use of a quasi-order was also suggested in [14, p. 10].) Technically, the relation satisfies the *weak* (non-strict) monotonicity condition on the functional (cf. [14, p. 12]). It also satisfies the requirements of the general path ordering [11]. The conditions in the definition could be relaxed somewhat to take into account the possible non-transitivity of the relations.

▸ **Theorem 1.** *The simple semantic path ordering $\succ/\gtrsim$ is terminating.*

The proof is essentially as in [14] (and [11, Thm. 2]), but takes the quasi-ordered case into account.

**Proof.** Suppose the path relation is not terminating and look at a minimal counterexample $u_0 \gtrsim^* \succ \gtrsim^* u_1 \gtrsim^* \succ \gtrsim^* u_2 \gtrsim^* \succ \gtrsim^* \cdots$, minimal with respect to subterm. Let's number the cases as follows:

$$\frac{s_i \gtrsim t}{s \succ_1 t} \qquad \frac{s \gtrsim t,\ s \succ t_j}{s \succ_2 t} \qquad \frac{s \geq t,\ s \succ t_j}{s \gtrsim_3 t}$$

The minimal counterexample never employs the first case: Clearly, it is not the case that $u_0 \gtrsim_1 u_1$, since then the sequence beginning with the subterm of $u_0$ that justifies the inequality would be smaller. Suppose $u_i \succ_1 u_{i+1}$ is the first occurrence of case 1 in the counterexample ($i \geq 1$), and that it is justified by $t_i \gtrsim u_{i+1}$ for subterm $t_i$ of $u_i$. Whether $u_{i-1} \succ_2 u_i$ or $u_{i-1} \gtrsim_3 u_i$, we would have $u_{i-1} \succ t_i \gtrsim u_{i+1}$ by the side requirement $s \succ t_j$ of the other two cases.

Whenever $u_i \succ_2 u_{i+1}$, we have $u_i \succ u_{i+1}$; when $u_i \gtrsim_3 u_{i+1}$, we have $u_i \geq u_{i+1}$. This contradicts termination of $\succ/\geq$. ◂

For reduction in a semantic path ordering of a term-rewriting system to provide termination, one shows first of all that $\ell \succ r$ for every rule $\ell \to r$ (meaning that $\ell\sigma \succ r\sigma$ for every substitution $\sigma$). As explained in [14, pp. 14–15], the semantic path ordering is not necessarily weakly monotonic. That is, it need *not* be the case that

$$s \succ t \Rightarrow f(\ldots, s, \ldots) \gtrsim f(\ldots, t, \ldots)$$

Therefore, one also needs to demonstrate ([14, p. 14, *post correctionem*])

$$s \to t \Rightarrow f(\ldots, s, \ldots) \geq f(\ldots, t, \ldots) \qquad (*)$$

so that $s \to t$ implies either $s \succ t$ (if it is a top-rewrite) or $s \gtrsim t$ (if not), which is enough to ensure that $s \gtrsim t$ whenever $s \to t$ and give termination [7, Second Termination Theorem]. With condition (*), the intersection of $\gtrsim$ and $\to$ is monotonic, since $s \gtrsim t$ and $f(\ldots, s, \ldots) \geq f(\ldots, t, \ldots)$ yield $f(\ldots, s, \ldots) \gtrsim_3 f(\ldots, t, \ldots)$.

## 3 The Dependency-Pair Method

Consider now the (basic) dependency-pair framework [1, Thm. 7]. For every rule $\ell \to r$ and nonvariable (not necessarily proper) subterm $u$ of $r$ that is not headed by a constructor (a symbol that never appears at the head of a left-hand side of any rule), we have a dependency

pair $\ell \to u$. Suppose we are given a pair of (partial and quasi-) orderings $\gtrdot, \gtrsim$ that are compatible (as above), and such that $\gtrdot$ is well-founded and $\gtrsim$ is weakly monotonic, meaning:

$$s \gtrsim t \Rightarrow f(\ldots, s, \ldots) \gtrsim f(\ldots, t, \ldots)$$

Then, a rewrite system terminates if $\ell \gtrsim r$ for every rule and $\ell \gtrdot u$ for every dependency pair (again, for all substitutions).

▸ **Theorem 2.** *If a rewrite system can be shown terminating by the basic dependency-pair method using the pair $\gtrsim$ and $\gtrdot$, then it is terminating by the semantic path ordering method using the same pair.*

**Proof.** Modify the ordering $\gtrdot$ so that all terms headed by constructors are smaller than all those that are not (see [5, Sect. 7.3], [9, n. 9], [4, Section 5.4]). Clearly, the ordering remains terminating and this change has no effect on dependency pairs, because constructor-headed terms are never compared. To maintain compatibility, also remove from $\gtrdot$ any pair whose left-side is a constructor term (they are never needed), and remove from $\gtrsim$ any pair with left-side a constructor and right-side not (also unnecessary).

Consider a rule $\ell \to r$. We show that $\ell \succ r$. If $r$ is a proper subterm of $\ell$, and in particular if $r$ is a variable, then $\ell \succ_1 r$. If not, then $\ell \gtrdot r$, since it is one of the dependency pairs or else $r$ is headed by a constructor. Furthermore, $\ell \succ r_j$, for every subterm $r_j$ of $j$, either because $r_j$ is a subterm of $\ell$, or because of a dependency pair $\ell \to r_j$, or because $r_j$ is headed by a constructor, so $\ell \gtrdot r_j$, and $r_j$'s subterms are smaller (by induction).

For reduction in the semantic path ordering to provide termination, we said that one also needs condition (*) to hold. But the dependency pair conditions tell us that $\ell \gtrsim r$ for every rule, and weak monotonicity tells us that $c[\ell] \gtrsim c[r]$ for any context $c$. Therefore, $s \to t \Rightarrow f(\ldots, s, \ldots) \gtrsim f(\ldots, t, \ldots)$, as required. ◂

It is clear why there is no need to consider dependency pairs $\ell \to u$ when $u$ is a proper subterm of $\ell$, as suggested in [9, n. 8], since then $\ell \succ_1 u$. In fact, the pair can be ignored if $\ell$ has any proper subterm $t$, such that $t \gtrsim u$, as suggested in [10, Sect. 6.3].

## 4 The Monotonic Semantic Path Ordering

The dependency-pair conditions for termination also fulfill the requirements for the monotonic semantic path ordering of [3] (preceded by [12]). This method combines a (multiset) semantic path ordering $\succ$ over a well-founded quasi-order $\gtrsim$ with a (weakly-) monotonic quasi-ordering $\geq$. It demands that $\ell \gtrsim r$ and $\ell \succ r$ for each rule, and, furthermore, that the two base orderings satisfy

$$s \geq t \Rightarrow f(\ldots, s, \ldots) \geq f(\ldots, t, \ldots) \qquad (**)$$

See [9, Sect. 4]. (The latter condition is called "quasi-monotonicity" of $\gtrsim$ with respect to $\geq$ in [3] and "harmony" of $\gtrsim$ with $\geq$ in [9, Sect. 3].)

Suppose now that $\geq$ and $\gtrsim$ are one and the same well-founded monotonic quasi-ordering. The above condition (**) translates into weak-monotonicity of $\gtrsim$. Then to satisfy the requirements of the corresponding monotonic semantic path ordering, we have $\ell \gtrsim r$ by the demands of the dependency method and $\ell \succ r$ for the same reasons as in the above proof. See [5, Thm. 7.3.2] and [4, Section 5.4].

## 5 Conclusion

The ordinary semantic path ordering [14], general path ordering [11, 13], and monotonic semantic path ordering [3] all include recursive cases, where subterms are examined recursively (in some order or other) if two terms are semantically equivalent (vis-à-vis $\doteq$). As noted in [4, Section 5.4], dependency pairs do not make use of the recursive case of the path ordering. Including recursion on subterms refines the simple-minded ordering and can only be of service in termination proofs. (Some might view the absence of recursive comparisons an "advantage", in that the search space is reduced.)

On the other hand, the dependency-pair formulation of this termination method has the practical advantage of rephrasing the task as the termination problem of an enlarged rewrite system (one that includes rewrite rules that force $s \succ t_j$ to hold) for which it may be relatively easy to adapt ordinary termination-proof systems. In one standard version of the method, additional rules—with altered root symbols—are used to disentangle the strong-monotonicity and weak-monotonicity requirements.

It is commonplace with the dependency-pair method for $\gtrdot/\geq$ to be some version of the recursive path ordering [7, 8]. The same is true for the semantic path ordering, which often uses a simpler recursive path ordering for its semantic ordering. This kind of semantic ordering is something that David Plaisted and I used from the earliest days of path orderings.[1]

On account of the *weak* monotonicity requirement for the component ordering $\geq$ used for the semantic path ordering, general path ordering, or dependency-pair method, the ordering $\geq$ in all three cases can ignore selected subterms, which is very often useful.

Nothing we have said relates to the powerful data-flow techniques of [1], which take the narrowing ideas of [6] and others to a high degree of utility. Were one to want to, the analysis of dependency-pair chains could be captured by pattern-based semantic, perhaps akin to that in [15].

**Acknowledgement.** I thank Jügen Giesl, Jean-Pierre Jouannaud, and Ori Lahav for their advice and comments.

## References

1 Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000. Preliminary version available at `http://verify.rwth-aachen.de/giesl/papers/ibn-97-46.ps`.

2 Leo Bachmair and Nachum Dershowitz. Commutation, transformation, and termination. In J. H. Siekmann, editor, *Proceedings of the Eighth International Conference on Automated Deduction (Oxford, England)*, volume 230 of *Lecture Notes in Computer Science*, pages 5–20, Berlin, July 1986. Springer-Verlag. Available at `http://nachum.org/papers/CommutationTermination.pdf`.

3 Cristina Borralleras, Maria Ferreira, and Albert Rubio. Complete monotonic semantic path orderings. In *Proceedings of the 17th International Conference on Automated Deduction (Pittsburgh, PA)*, volume 1831 of *Lecture Notes in Artificial Intelligence*, pages 346–364, Berlin, June 2000. Springer-Verlag. Available at `http://www.lsi.upc.edu/~albert/papers/mspo.ps.gz`.

[1] This is what was being referred to in the citation of the personal communication "[Plaisted, 1979]" in [8]).

**4** Cristina Borralleras and Albert Rubio. Orderings and constraints: Theory and practice of proving termination. In H. Comon-Lundh, C. Kirchner, and H. Kirchner, editors, *Rewriting, Computation and Proof: Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, volume 4600 of *Lecture Notes in Computer Science*, pages 28–43. Springer-Verlag, Berlin, June 2007. Available at `http://www.lsi.upc.edu/~albert/papers/jean-pierre-60.pdf`.

**5** Cristina Borralleras Andreu. *Ordering-Based Methods for Proving Termination Automatically*. PhD thesis, Departament de Llenguatges i Sistemes Informàtics de la Universitat Politècnica de Catalunya, Barcelona, Spain, April 2003. Available at `http://www.lsi.upc.edu/~albert/cristinaphd.ps.gz`.

**6** Jacques Chabin and Pierre Réty. Narrowing directed by a graph of terms. In Ronald V. Book, editor, *Rewriting Techniques and Applications*, volume 488 of *Lecture Notes in Computer Science*, pages 112–123. Springer-Verlag, Berlin, 1991. Available at `http://www.univ-orleans.fr/lifo/Members/chabin/articles/rta1991.pdf`.

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

**8** 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`.

**9** Nachum Dershowitz. Termination by abstraction. In *Proceedings of the Twentieth International Conference on Logic Programming (St. Malo, France)*, volume 3132 of *Lecture Notes in Computer Science*, pages 1–18, Berlin, September 2004. Springer-Verlag. Available at `http://nachum.org/papers/TerminationByAbstraction.pdf`.

**10** 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`.

**11** Nachum Dershowitz and Charles Hoot. Natural termination. *Theoretical Computer Science*, 142(2):179–207, 1995. Available at `http://nachum.org/papers/natural-sterm94.pdf`.

**12** Alfons Geser. On a monotonic semantic path ordering. Technical Report 92-13, Ulmer Informatik-Berichte, Universität Ulm, Germany, 1992.

**13** Alfons Geser. An improved general path order. *Applicable Algebra in Engineering, Communication and Computing*, 7(6):469–511, 1996. Available at `http://webdoc.sub.gwdg.de/ebook/e/2001/mip/gpo.ps.Z`.

**14** Sam Kamin and Jean-Jacques Lévy. Two generalizations of the recursive path ordering. Unpublished letter to Nachum Dershowitz, Department of Computer Science, University of Illinois, Urbana, IL, February 1980. Available at `http://nachum.org/term/kamin-levy80spo.pdf`.

**15** Laurence Puel. Embedding with patterns and associated recursive path ordering. In N. Dershowitz, editor, *Proceedings of the Third International Conference on Rewriting Techniques and Applications (Chapel Hill, NC)*, volume 387 of *Lecture Notes in Computer Science*, pages 371–387, Berlin, 1989. Springer-Verlag.