Hierarchical Termination*

Nachum Dershowitz

Department of Computer Science
University of Illinois at Urbana-Champaign
1304 West Springfield Avenue
Urbana, IL 61801, U.S.A.
nachum@cs.uiuc.edu

Abstract. From a practical perspective, it is important for programs to have modular correctness properties. Some (largely syntactic) sufficient conditions are given here for the union of terminating rewrite systems to be terminating, particularly in the hierarchical case, when one of the systems makes no reference to functions defined by the other.

1 Introduction

A *rewrite rule* is an equation between first-order terms used to replace equals-by-equals in one direction only. A rewrite system, that is a set of rewrite rules, is a form of applicative program that computes by reducing (that is, repeatedly rewriting) a variable-free term to its normal form (an unrewritable term), where the order in which rules are applied and the choice of subterm to which to apply a rule is arbitrary. Rewrite systems have other important applications in programming language semantics and automated deduction. For recent surveys of rewriting, see [Dershowitz and Jouannaud, 1990; Avenhaus and Madlener, 1990; Klop, 1992; Plaisted, 1993].

When no infinite sequences of rewrites are possible, a rewrite system is said to have the (*strong*, or *uniform*) *termination* property. In practice, one usually guarantees termination by devising a well-founded partial ordering for which a rewritten term is always smaller than the original. For a survey of methods of proving termination, see [Dershowitz, 1987]; for examples of these methods, see [Dershowitz, 1995].

Rewrite systems provide a simple, intuitive, nondeterministic functional language. As such, it would be quite valuable to be able to combine systems possessing desirable properties. In particular, we look for sufficient conditions under which the union of two terminating systems would be terminating. The conditions given here are based on syntactic restrictions of the systems in question. The first to consider modularity issues in rewriting was Bidoit [1981] with his "gracious" conditions.

A rule $l \to r$ is used to rewrite a term s containing an instance $l\sigma$ of its left-hand side l at some position p to $s[r\sigma]_p$, the same term, except that the subterm at p has

* This research was supported in part by the U. S. National Science Foundation under Grants CCR-90-07195 and CCR-90-24271, by a Lady Davis fellowship at the Hebrew University of Jerusalem, Israel, and by a Meyerhoff Visiting Professorship at the Weizmann Institute of Science, Rehovot, Israel.

been replaced by the corresponding instance $r\sigma$ of the rule's right-hand side r. We will have recourse to the notation $s|_p$ for the subterm at position p in term s.

Various modularity properties (such as termination and uniqueness of normal forms) have been intensely studied since the appearance of [Toyama, 1987b], particularly for combinations of systems that have no function symbols (or constants) in common.[1] Toyama [1987a] gave the following example, showing that even in this simplest case the combination of two terminating systems is not necessarily terminating:

$$\boxed{f(0,1,x) \to f(x,x,x) \;\Big|\; \begin{array}{c} g(x,y) \to x \\ g(x,y) \to y \end{array}} \qquad \text{(A)}$$

In the next section, we review what is known about termination in this disjoint case, and present the major syntactic restrictions of interest in this regard. (For other modular properties, see [Middeldorp, 1990].) Then, in Section 3, the case where "constructor" symbols are shared by the two systems is considered.

Section 4 considers the "hierarchical" case where one system is allowed to refer to symbols appearing in the other, but not vice-versa. For example, we want to be able to use terminating systems for addition and multiplication in conjunction with a terminating program for factorial:

$$\begin{array}{c} fact(x) \to f(x, s(0)) \\ f(s(x), y) \to f(x, s(x) \cdot y) \\ f(0, x) \to x \ . \end{array} \;\Big|\; \begin{array}{c} x \cdot 0 \to 0 \\ x \cdot s(y) \to (x \cdot y) + x \\ \hline x + 0 \to x \\ x + s(y) \to s(x + y) \end{array} \qquad (1)$$

The individual systems can be shown to terminate by showing that the terms in any derivation decrease in some well-founded measure (such as the natural numbers). For the right systems, one can, for example, let $[\![0]\!] = 2$, $[\![s(x)]\!] = [\![x]\!] + 1$, $[\![x \cdot y]\!] = [\![x]\!]^{2^{[\![y]\!]}}$, and $[\![x + y]\!] = [\![x]\!][\![y]\!]$. For the "higher" system, let $[\![0]\!] = 1$, $[\![s(x)]\!] = [\![x]\!] + 1$, $[\![x \cdot y]\!] = [\![x]\!][\![y]\!]$, $[\![f(x,y)]\!] = [\![y]\!]([\![x]\!] + 1)!$, and $[\![fact(x)]\!] = ([\![x]\!] + 2)!$. With either system, whenever $s \to t$, we have $[\![s]\!] > [\![t]\!]$. The question is how to ensure that the union of the two systems terminates, without having to find an independent proof for the combination. The measures used above for the individual systems cannot be combined. (Termination of the combined system could be proved instead using the methods in [Dershowitz, 1982].) This eminently practical case has received virtually no attention.[2]

Section 5 mentions some results for the fully general case, when both systems can refer to all symbols, and is followed by a brief discussion of some remaining questions.

---

[1] Some authors reserve the designation "modular" for this disjoint case; we prefer, however, to use the term generically, specifying "disjoint," "shared constructors," or "hierarchical," as the case may be.

[2] A draft of this paper [Dershowitz, 1992] was distributed in December 1992.

2 Disjoint termination

Let A and B be disjoint sets of function symbols (including constants) and X be a set of variables. Let a *red* rewrite system contain terms built from A and X only (*red* terms), while a blue system has terms from B and X only (*blue* terms).[3] Supposing the red and blue systems are both terminating for all terms, that is, there are no infinite sequences of red rewrites, nor of blue rewrites, for terms constructed from A and B, then termination is said to be *modular* when the union is also terminating. The notion of modularity of properties for the disjoint vocabulary case was first studied by Toyama [1987b].

It is worth repeating the following bit of folk wisdom:

Proposition1. *If a system is terminating for all terms constructed from symbols appearing in it (plus one new constant if the rules display none), then it also terminates for terms constructed from any richer set of symbols.*

Thus, to show that a red (blue) system terminates for "all" terms, it suffices to show termination for red (blue) terms.

Proof. Suppose a red system terminates for all red terms. One way to prove that it also terminates for mixed terms, containing "foreign" (blue) symbols, is to decompose mixed terms into pure red subterms, with some red constant replacing subterms of the components headed by blue symbols. The nesting depth of these pure-red components in a term cannot increase by rewriting. Terms are compared by looking lexicographically at a tuple of multisets, the most significant element of the tuple containing the uppermost red components, and so on. Multisets are compared in the multiset ordering [Dershowitz and Manna, 1979] and components in the red rewrite relation. For example, if f, a, b, c, and d are red, then $g(f(g(f(a,g(b))),f(a,f(g(d),g(d)))))$ would have components $\langle\{f(c,f(a,f(c,c)))\},\{f(a,c),d,d\},\{b\}\rangle$. □

Various sufficient conditions for modularity of termination (and related properties in subsequent sections) make use of the following decidable notions, all but the last of which are syntactic:[4]

- A *non-erasing* system has no rule with a variable on the left not also appearing on its right.
- A *non-collapsing* system has no rule with a variable as its right-hand side.
- A *right-linear* system has no rule with more than one occurrence of a variable on its right-hand side.
- A *non-duplicating* system[5] has no rule with more occurrences of a variable on its right-hand side than on its left. Of course, right-linear systems are non-duplicating, since we normally disallow rules that introduce a variable on the right not already on the left.

[3] With apologies to some previous authors, the color scheme has been changed here for added mnemonic value.

[4] These restrictions are ordered from the more severe to the less, taking a programmer's point of view.

[5] Called "conservative" in [Fernandez and Jouannaud, 1995].

- A *left-linear* system has no rule with more than one occurrence of a variable on its left-hand side.
- A *non-overlapping* system has no left-hand side that unifies with a non-variable subterm of another left-hand side or with a proper non-variable subterm of itself, after renaming variables in the terms so that they are disjoint. (This means there are no non-trivial "critical pairs" in the terminology of Knuth and Bendix [1970].) Clearly, rules with different colors at the top of their left sides cannot overlap.
- A *constructor-based* system is one in which no left-hand side has a symbol below the top that appears at the top of any left-hand side.
- An *overlaying* system is one in which no left-hand side unifies with a non-variable proper subterm of any left-hand side (including itself), after "standardizing apart" (renaming variables in the terms so that they are disjoint). Of course, non-overlapping and constructor-based systems are also overlaying.
- A *locally confluent* system is one for which any two terms that can be obtained each by one step of rewriting from the same term can both be rewritten in zero or more steps to the identical term. (Local confluence is decidable for finite terminating systems [Knuth and Bendix, 1970].) In particular, non-overlapping systems are locally confluent [Huet, 1980].

The following results are known:

Theorem 2 [Rusinowitch, 1987]. *The union of non-collapsing red and blue terminating systems is terminating.*

Theorem 3 [Rusinowitch, 1987]. *The union of non-duplicating red and blue terminating systems is terminating.*

Theorem 4 [Middeldorp, 1989]. *The union of a non-collapsing non-duplicating red terminating system with a blue terminating system is terminating.*

Theorem 5 [Toyama *et al.*, 1989]. *The union of left-linear locally-confluent red and blue terminating systems is terminating.*

Theorem 6 [Middeldorp and Toyama, 1991]. *The union of constructor-based locally-confluent red and blue terminating systems is terminating.*

Let us call overlaying locally-confluent systems *overlay-confluent*. The conditions of the previous theorem have been weakened to include this class of systems:

Theorem 7 [Gramlich, 1995]. *The union of overlay-confluent red and blue terminating systems is terminating.*

All the above results apply to the union of the following non-erasing, non-collapsing, non-duplicating, left-linear, non-overlapping systems:

$$x + s(y) \to s(x + y) \quad \Big| \quad p(x) - p(y) \to x - y \ , \tag{2}$$

where $+$ is red and $-$ is blue.

Generalizations of Theorems 2, 3, 4, and 7 will be proved in the sequel.

A more semantic approach was developed in [Gramlich, 1994; Ohlebusch, 1993] (based on the syntactic ideas in [Kurihara and Ohuchi, 1990]):

Theorem 8 [Ohlebusch, 1993]. *The union of red and blue systems that are each terminating when joined with the system* $\{h(x,y) \to x, h(x,y) \to y\}$*, for new function symbol* h *not appearing in either system, is terminating. (This is an undecidable property.)*

Theorem 9 [Ohlebusch, 1993]. *The union of a non-duplicating red system that is terminating when joined with* $\{h(x,y) \to x, h(x,y) \to y\}$*, for new function symbol* h *not appearing in either system, with a terminating blue system is terminating.*

3 Shared termination

Requiring that two systems have no symbols in common is much too restrictive in practice. In this section, we investigate the case where constructors are shared by the two systems (as also considered in [Middeldorp and Toyama, 1991; Gramlich, 1994; Ohlebusch, 1993]).

For our purposes, a *constructor* is any function symbol (in the given vocabulary) that never appears as the outermost symbol of a left-hand side (of either system), while a *defined symbol* is one that does. Let C be a set of *yellow* constructors, disjoint from A and B. Terms built from $A \cup C \cup X$ are *orange*; those over $B \cup C \cup X$ are *green*. An orange (green) term is deemed *bright* when its top symbol is red (blue).

An *orange* system has only orange terms; a *green* system, only green. Note that an orange (green) system must have a red (blue) symbol on the top of the left-hand side (since constructors never appear on the top left), and that red and blue symbols may be nested on either side of a rule. We will call an orange (green) system *bright* if the top symbol on the right is always red (blue). A bright system cannot have just a variable for right-hand side (that is, it is non-collapsing), nor can its right-hand side be headed by a constructor.[6] Defined (red or blue) symbols may be nested on either side of a rule, unless otherwise stated.

We count the number of alternations of red and blue symbols (ignoring yellow ones) along the path from the root leading to each symbol f in a term (in its tree representation) and assign a *level* to f accordingly. Yellow symbols in a term are assigned to the level of the nearest blue or red symbol preceding it along the path from the root. Orange and green rewrites can never increase the number of homogeneously colored layers in the terms of a derivation.

Were a shared system non-terminating, there would be a minimum number of layers for non-termination. An infinite derivation with that number of layers would have to have an infinite number of rewrites in the top layer (or else fewer layers would suffice for non-termination), as well as an infinite number below (or else the top system alone would be non-terminating).

The next theorem extends Theorem 2 (due to Rusinowitch [1987]) to systems with shared constructors.

Theorem 10 [Gramlich, 1994]. *The union of bright-orange and bright-green terminating systems is terminating.*

[6] In the terminology of [Gramlich, 1994], it is not "constructor-lifting."

Thus,

$$x + s(y) \to s(x) + y \quad \Big| \quad s(x) - s(y) \to x - y \tag{3}$$

is terminating (red $+$, blue $-$, yellow s), since each rule by itself is.

Proof. We use a well-founded ordering for which a rewrite within the top layer decreases the term in the ordering and a rewrite below the top layer does not increase it, precluding more than finitely many top rewrites. Since the systems are "brightly colored," layers never collapse, that is, the top layer never grows on account of a step below the top. Thus, to compare two terms, we simply compare their top layers—with one arbitrary term replacing all lower-level subterms—in the terminating rewrite relation of the top system. □

Similarly, the following theorem extends Theorem 3 [Rusinowitch, 1987] for shared constructors:[7]

Theorem 11. *The union of orange and green non-duplicating terminating systems is terminating.*

Proof. Rewriting does not increase the number of levels (except to add constructors at the top, which we can safely ignore). Consider the multiset of subterms below the top layer (the "aliens") in an infinite derivation in the combined system having no more levels than necessary for non-termination. If the top level is red (say), these subterms are headed by the highest blue symbol. Terms are compared in the union of the combined rewrite relation (which may be presumed terminating for terms of fewer layers) and the proper subterm relation. (This combined rewrite and subterm relation is terminating since the two commute, that is, any rewritten subterm is the subterm of the whole term rewritten.) Since the systems are non-duplicating, a top rewrite can only remove elements from the multiset (or leave them all intact). Furthermore, each of the ostensibly infinitely many rewrites below the top decreases the multiset in the rewrite relation. If a rewrite in the second, blue layer creates more than one (disjoint) blue subterm (connected by constructors), they are each smaller in the composition of the rewrite and subterm relation. Similarly, when a segment of the second layer collapses (which it can do in the non-bright case), some elements of the multiset may be replaced by some of their subterms. □

The following extends Theorem 4:

Theorem 12. *The union of a non-duplicating bright-orange terminating system with a green terminating system is terminating.*

Proof. Consider any derivation with a fixed number of layers. If the top layer is red, then the argument is just as in the previous proof. If the second layer is red, then we can use the same ordering as for Theorem 10. □

[7] A similar proof was given independently in [Ohlebusch, 1993]. In [Fernandez and Jouannaud, 1995], the result is extended to allow sharing of symbols other than constructors, provided the same proof method still applies.

The following non-terminating example [Dershowitz, 1981], with red f, blue 2, and yellow 0 and 1, shows the necessity of brightness (as in Theorem 10) or non-duplication (as in Theorem 11):

$$\left| \begin{array}{l|l} f(0,1,x) \to f(x,x,x) & \begin{array}{l} 2 \to 0 \\ 2 \to 1 \end{array} \end{array} \right| \tag{B}$$

Proposition 13 [Gramlich, 1995]. *An overlay-confluent system is terminating for a given term if, and only if, it is by innermost rewriting.*

This is analogous to the well-known fact that termination of call-by-value implies termination of call-by-name [Cadiou, 1972]. It includes, as a common special case, non-overlapping systems, proved in [Geupel, 1989].

Since, as it is easy to ascertain, innermost termination is preserved by unions of orange and green systems (cf. [Kurihara and Kaji, 1990; Gramlich, 1995]), it follows that[8]

Theorem 14. *The union of overlay-confluent orange and green terminating systems is terminating.*

In particular, non-overlapping systems can be combined. This extends Theorem 7 to systems with shared constructors, like:

$$\left| \begin{array}{l|l} \begin{array}{r} x + 0 \to x \\ 0 + x \to x \\ s(x) + y \to s(x + y) \\ x + s(y) \to s(x + y) \end{array} & \begin{array}{l} s(x) \uparrow 0 \to s(0) \\ x \uparrow s(y) \to (x \uparrow y) \cdot x \ . \end{array} \end{array} \right| \tag{4}$$

It also extends the result in [Middeldorp and Toyama, 1991] for constructor-sharing constructor-based systems.[9]

The overlaying requirement is necessary, as seen in this locally confluent example [Drosten, 1989]:

$$\left| \begin{array}{l|l} \begin{array}{r} f(0,1,x) \to f(x,x,x) \\ f(x,y,z) \to 2 \\ 0 \to 2 \\ 1 \to 2 \end{array} & \begin{array}{l} g(x,y,y) \to x \\ g(x,x,y) \to y \end{array} \end{array} \right| \tag{C}$$

Local-confluence is likewise essential (cf. Example (B)).

[8] This result also appears in [Gramlich, 1995].

[9] Middeldorp and Toyama [1991] also consider the case where certain rules are shared by both systems, also easily handled by our method.

4 Hierarchical termination

Suppose one has defined some blue functions, recursively, using green rules. Typically, these rules would reduce any green term to a yellow (constructor) normal form. Then, one goes ahead and defines red functions, also recursively, but using blue functions in an auxiliary manner. We are thinking of a system like the right half of System (1), where $\cdot$ is red, $+$ is blue, 0 and s are yellow. We'll call such systems, with bright-orange left-hand sides and arbitrary right-hand sides, *purple*. This common situation also arises in applicative programs; semantics (that is, knowing the values computed by the functions) are usually needed for termination proofs.

At least two approaches are possible. We can endeavor to show that any infinite derivation in the combined system could be rearranged to provide an infinite monochrome derivation, which is impossible. Or we can try to extend the results of the previous section which require that the number of layers not increase in a derivation (which is not in general true for the hierarchical case).

An easy result using the first approach is:

Theorem 15. *The union of a left-linear purple terminating system with a right-linear bright-green terminating system is terminating.*

This theorem applies, for example, to:

$$\begin{array}{r} x \cdot 0 \to 0 \\ x \cdot s(y) \to (x \cdot y) + x \end{array} \quad \Big| \quad x + s(y) \to s(x) + y \ . \tag{5}$$

It does not apply to (1) with its "dull" rule $x + 0 \to x$.

The necessity of brightness and right-linearity can both be seen from Example (B); left-linearity is needed to exclude:

$$\boxed{f(x,x) \to f(g(a),g(b)) \quad \Big| \quad g(a) \to g(b)} \tag{D}$$

Here f is red, g is blue, and a and b are constructors. With blue symbols on both sides of purple rules, we invite non-termination, as in:

$$\boxed{f(b) \to f(g(a)) \quad \Big| \quad \begin{array}{r} g(a) \to b \\ b \to g(c) \end{array}} \tag{E}$$

(g and b are blue).

A more general version of this theorem will be proved at the end of the next section.

Right-linearity is not a very natural requirement. To get a better handle on the hierarchical non-right-linear case, we further restrict the form of purple rules.

Theorem 16. *The union of a left-linear overlay-confluent purple terminating system with an overlay-confluent bright-green terminating system is terminating.*

This theorem applies to System (5) and corrects the result in [Bidoit, 1981] for "gracious" systems by requiring that the green system be bright. Without brightness, we could be fooled by

$$\boxed{f(a) \to f(b) \quad | \quad b \to a} \tag{F}$$

where a is the only constructor.[10] We've already seen the need for left-linearity in Example (D). Example (B) shows the need for confluence (of the purple system at least).

Proof. The union is overlaying (since the purple left sides cannot unify with non-variable green subterms, nor green left sides with the orange subterms of purple left sides) and locally-confluent (by the Critical Pair Lemma [Knuth and Bendix, 1970; Huet, 1980], since the union cannot introduce any new overlapping left-hand sides). Hence, by Proposition 13, we need only show innermost termination. In any innermost derivation, there cannot be a purple step taking place in the variable part of a preceding green step, since that would mean that the purple step could have been applied to a proper subterm of the green redex. So, if a derivation has a green step immediately preceding a purple step, the two either occur at disjoint positions (neither at a subterm of the other redex), in which case they can be interchanged, or else the green step occurs in the variable part of the purple step (on account of brightness), in which case the left-linear purple step can be applied first, followed by some number of green steps (one for each occurrence of that variable on the right-hand side of the purple rule). Thus, from any innermost derivation with infinitely many purple steps, an infinite purple derivation could be constructed. □

A purple system is *red-increasing* if the maximum number of red symbols along a path from the root of a term to a leaf can increase in a derivation. For example,

$$\begin{aligned} f(x, g(y)) &\to f(y, g(x)) \\ g(x) &\to x \ , \end{aligned} \tag{6}$$

where f and g are red, is red-increasing. If a purple system is not red-increasing, then neither is its union with a green system.

Theorem 10 has the following analogue in the hierarchical case:

Theorem 17. *The union of a left-linear non-red-increasing bright-purple terminating system with a bright-green terminating system is terminating.*

System (1), sans its three dull rules, is an example. Example (B) shows the need for brightness of both systems; (D) demonstrates the need for left-linearity.

Proof. Since the nesting of reds does not increase, were the union non-terminating, there would be a non-terminating derivation of minimal depth, with infinitely many rewrites at topmost red symbols. (Were the top red symbols to all become inactive, fewer levels would suffice for non-termination, since the green steps above the top red layer could not go on in perpetuity.) Transform this derivation by replacing

[10] It wouldn't help—nor would it make sense—to insist that all terms reduce to constructor terms via the purple system, since we could just add $f(x) \to a$ to this counter-example.

all subterms headed by a non-topmost red with some green constant, since bright purple steps at the second level cannot impinge on rewritability of higher purple or green redexes, and lower steps can certainly have no effect. That leaves an infinite derivation of terms having green above and below a single layer of red. If there are green symbols above the red, they cannot sustain more than finitely many green steps, since bright purple steps do not contribute to that green layer. Thus, any green step preceding a purple step must either be disjoint from the latter, or else it is in the variable part of the purple rule, since the green right-hand side is headed by a blue symbol, which can appear nowhere in the purple left side. As in Theorem 16, since the purple system is left-linear, that green step can be delayed until after the purple step, and then performed once for each occurrence of the variable in question on the right side. Hence, a derivation containing infinitely many purple steps in succession can be constructed, contradicting the presumed termination of the purple system on its own. □

A *flat* system is one in which red symbols are not nested on the left or right. That is, no path from the root symbol has more than one red symbol along it. Flat systems, in addition to being constructor-based, cannot invoke nested recursion.

Lemma 18. *Flat purple systems are not red-increasing.*

Even for flat systems, hierarchical termination is by no means ensured (Example (F)). In the remainder of this section, we develop sufficient conditions for termination in the flat case.

Violet (a bluish purple) systems have no blue symbols below a red on the right side, as in the right half of System (1). When innermost termination suffices, we can show:[11]

Theorem 19. *The union of a locally-confluent flat violet (only yellow below red) terminating system with an overlay-confluent green terminating system is terminating.*

The right half of System (1) is an example. More generally, this theorem applies to hierarchies of primitive-recursive definitions. It is a corollary of the one that follows.

If we desire to allow blue symbols to also appear below the red ones, we need to be able to ignore the effects of green rewriting. We will say that a terminating purple system is *oblivious (of green)* if it remains terminating even when the rules are replicated so that each bright-green subterm on a right-hand side (headed by a blue symbol) is replaced by all possible green (and yellow) variable-free terms. (We assume that there is at least one blue or yellow constant—or else we must add one so that the set of green terms is not void.) That is, a purple system R is oblivious of green terms if $R_g = \{l \to r[g]_p : l \to r \in R;\ r|_p \text{ is bright green};\ g \text{ is green}\}$ is also terminating. (Actually, we need only replace maximal green subterms.)

[11] This result also appeared in [Krishna Rao, 1992]. Instead of requiring flatness, Krishna Rao [1993] forbids those nestings that *seem* able to lead eventually to a blue symbol that can cause an increase in the depth of red. See also [Gramlich, 1995].

For System (1), the recursive rule on the right adds $f(s(x), y) \to f(x, ?)$, where ? is an arbitrary green term (containing any combination of $\cdot$, $+$, s, and 0). The extended system is still terminating.

Flat violet systems are oblivious by definition. Systems for which there is one green argument position that decreases (taking subterms, say) with each "recursive call" are also oblivious.

Obliviousness compensates for the appearance of blue symbols below reds on the right, and allows us to generalize the previous theorem:

Theorem 20. *The union of an oblivious locally-confluent flat purple terminating system with an overlay-confluent green terminating system is terminating.*

Note that green levels can grow deeper; hence, more than two levels of hierarchically-defined functions are possible. This result applies, for example, to the three parts of System (1), as well as to the following "tail recursive" program:

$$\begin{array}{c} sum(x) \to f(0, x) \\ f(x, \epsilon) \to x \\ f(x, y \cdot z) \to f(x + y, z) \end{array} \quad \Big| \quad \begin{array}{c} x + 0 \to x \\ x + s(y) \to s(x + y) \ , \end{array} \tag{7}$$

Flatness of purple right-hand sides is necessary as can be seen from the following non-terminating union:

$$\boxed{\begin{array}{c} f(x, x) \to f(a, g(x)) \\ a \to f(c, d) \end{array} \quad \Big| \quad g(x) \to x} \tag{G}$$

(f and a are red; g is blue). That non-increasing red depth is insufficient can be seen from the following variant:

$$\boxed{\begin{array}{c} f(x, x, a) \to f(a, g(x), a) \\ a \to f(c, d, d) \end{array} \quad \Big| \quad g(x) \to x} \tag{H}$$

These two systems are oblivious, since no *green* replacement for $g(x)$ can match the red a needed for the first purple rule to reapply. One part of (B) is not locally confluent, which explains its non-termination; (D) is not oblivious; the green left half of (C) is not overlaying.

Proof. By flatness, there is a bound on the depth of red symbols in any derivation. As was the case for Theorem 16, the union is locally confluent and overlaying, so we need only consider innermost rewriting (Proposition 13). Purple steps at the lowest red level may be followed by some green steps lower down. In an innermost derivation those green steps cannot be in the variable part of the purple right-hand side, since those are already in normal form. Thus, the purple system is oblivious of those green steps, and the net effect (reordering the green steps, as necessary, to follow immediately upon the red step that created them) is just a sequence of "oblivious purple steps," guaranteed to terminate with the lowest red level and everything below in normal form. Those subterms of red normal forms that have a red symbol at the top can play no further role in the derivation, since green rules

cannot "see" them at all, nor can the applicability of constructor-based purple rules depend on red symbols below the redex. The only impact they can have is in allowing or disallowing a non-left-linear rule to fire. They can all, therefore, be replaced with one non-red constant, giving a term with fewer levels of red. Thus, any derivation with the original red normal forms can be mimicked by shallower terms. □

This proof only requires that the purple system be oblivious of green subterms that are below a red symbol.

By combining the commutation-based approach with the layer-based approach, we get the following modification of Theorem 11:

Theorem 21. *The union of an oblivious right-linear flat purple terminating system with a non-duplicating green terminating system is terminating.*

System (7) falls in this category. Without flatness we have non-termination, as before (G); non-duplication by green rewrites rules out systems like (B); right-linearity of purple cannot be weakened to non-duplication, witness:

$$\left| \begin{array}{l} f(0,1,x,x) \to f(x,x,g(0,1),g(0,1)) \end{array} \right| \left| \begin{array}{l} g(x,y) \to x \\ g(x,y) \to y \end{array} \right| \qquad \text{(I)}$$

Proof. By flatness, the nesting of red does not increase, so we may consider an infinite derivation of minimal red depth. The multiset of subterms headed by second-layer red symbols (call them the "aliens") decreases with each rewrite at or below the second red layer and does not increase with a (purple or green) rewrite above—in the union of the subterm relation and the combined rewrite relation for terms with shallower nesting of red. Hence, from some point on, the minimal infinite derivation only has top red steps and green steps above the second layer. We can, therefore, replace all the aliens by a green constant without affecting any of those steps. This yields an infinite derivation of terms with green symbols above and below a single layer of red. (Flatness comes into play here; without it, new aliens would be produced by purple steps.) On account of right-linearity, any purple rewrite at a topmost red symbol followed immediately by a green rewrite within the position of a particular variable of the purple right-hand side can be rearranged to first apply the green rule as many times as necessary to rewrite the (one or more) occurrences of that variable on the left-hand side of the purple rule, followed by the same purple step. That can only transpire finitely many times (since green terminates), leaving an infinite sequence of oblivious purple steps, plus green steps in the top, non-duplicating, green layer. But there can be only finitely many of either, since green is non-duplicating. □

5 Combined termination

We mention here a few results for the non-hierarchical case in which either terminating system can refer to symbols appearing also in the other. We will refer to the systems as black and white. In particular, we will generalize the first two theorems of the previous section.

Proposition 22 [Dershowitz and Hoot, 1995; Gramlich, 1995].
A non-erasing, non-overlapping system terminates if it is normalizing (that is, if there is always some derivation leading to a normal form).

This improves the result in [O'Donnell, 1977] which requires that the system be left-linear, and which, consequently, has the same behavior as Church's [1941] λ-I calculus.

We say that a white system *preserves normal forms* of a black system if the former always rewrites black normal forms to black normal forms.

Theorem 23. *The union of black and white non-erasing terminating systems, the union of which is non-overlapping, and such that the white system preserves normal forms of the black, is terminating.*

For example,

$$\begin{array}{c|c} x + s(y) \to s(x) + y & s(x) - s(y) \to x - y \\ x + 0 \to x & s(x) \uparrow 0 \to 1\ . \end{array} \tag{8}$$

Proof. Use the preceding proposition and the fact that the union is normalizing under the stated conditions, taking white normal forms of black normal forms. □

The necessity of preservation is demonstrated by Example (F); the need for non-overlapping, by (B).[12]

We say that terms s and t are *separate* if s does not unify with a renamed non-variable subterm of t, nor vice-versa.

Proposition 24. *A white system preserves normal forms of a left-linear black system whenever white right-hand sides and black left-hand sides are separate. (In particular, the white system must be non-collapsing.)*

Proof. White cannot create an occurrence of a black left-hand side. Since black is left linear, white cannot create a new black redex by making making the latter's variable parts equal. □

We have:

Theorem 25. *The union of a left-linear overlay-confluent black terminating system with an overlay-confluent white terminating system, such that white right-hand sides are separate from black left-hand sides, is terminating.*

Theorem 16 is a corollary.[13] The proof is unchanged. An example is

$$x \cdot (y + z) \to (x \cdot y) + (x \cdot z) \quad | \quad x \cdot x \to 0\ . \tag{9}$$

We need the following:

[12] Though the non-erasing requirement is needed for the above proposition, an example of non-termination for non-overlapping preserving systems is lacking.

[13] This idea of decomposing proofs of termination by looking at overlappings between rules, but ignoring the difficulties engendered by non-left-linear rules, appeared in [Pettorossi, 1981].

Lemma 26 [Raoult and Vuillemin, 1980]. *If u rewrites to v using a right-linear rule $l \to r$, and then to w using a left-linear rule $s \to t$, and r and s are separate, then w can also be derived from u using at least one application of $s \to t$ followed by some number of applications of $l \to r$.*

Theorem 27 [Bachmair and Dershowitz, 1986]. *The union of a left-linear black terminating system with a right-linear white terminating system, such that white right-hand sides are separate from black left-hand sides, is terminating.*

Theorem 15 is a corollary.[14] System (9) is again an example.

Proof. Since the white right-hand sides are separate from black left sides and both the white right sides and black left sides do not have repeated variables, by the preceding lemma, any white step followed by a black step can always be replaced by at least one black step followed by some number of white steps. By induction, any number of white steps followed by one black can be replaced by at least one black step followed by some number of white steps. Hence, from any derivation with infinitely many black steps, an infinite purely black derivation could be constructed, contradicting the assumption of black termination. □

6 Discussion

It appears that Theorem 20 is the most useful result we have obtained for hierarchical systems, since it does not require brightness, right-linearity, or non-duplication. Both systems must be overlaying and locally-confluent (which implies that innermost rewriting will lead to non-termination if any strategy can), but that is normal in a functional programming style. In the absence of prescience as to the semantics (normal-form computations) of the green system, the purple system must be oblivious of green rewrites taking place in arguments of red functions (or have no blue symbols below red recursive calls, as in Theorem 19), but that is similar to the situation with ordinary functional languages. The purple system must be flat, but—in future work—we hope to use more general notions of obliviousness (such as obliviousness of terms built from green symbols *and* subterms of the purple left side) to perhaps weaken some of our restrictions. In any case, we need to develop additional sufficient conditions for obliviousness.

There still seems to be room for improving the various results we have given here, though we have provided counter-examples to most (but not quite all) ways of relaxing the conditions for termination. Some extensions are obvious: Bright-green rules, as in Theorem 16 for example, were only needed to preclude a green rule "creating" a purple redex; a constructor on the top right of a green rule that does not appear below the defined function of a purple left-hand side poses no problem (they are "separate" in the terminology of Section 5). Transformation methods of [Bachmair and Dershowitz, 1986; Bellegarde and Lescanne, 1990] can perhaps be used to handle certain collapsing cases.

[14] This theorem was claimed in [Dershowitz, 1981], but an overly weak condition of separateness was implied. (The examples in [Bachmair and Dershowitz, 1986] were also wrong on this account.) This direction was pursued further in [Geser, 1989].

This paper has only considered modularity of termination. The preservation of other properties, such as existence and uniqueness of normal forms, is also worth exploring for hierarchical systems.

To conclude with one more example, consider the fact that none of the theorems we have given apply to the union of

$$\begin{aligned} mapf(\epsilon) &\to \epsilon \\ mapf(x \cdot y) &\to f(x) \cdot mapf(y) \end{aligned} \tag{10}$$

with an *arbitrary* (green) system for computing f, not containing $mapf$. If the latter is non-duplicating or overlay-confluent, then it's okay, but it is highly unlikely (in this case, at least) that *any* terminating green system could cause problems.

Acknowledgements

I thank Bernhard Gramlich, M. R. K. Krishna-Rao and Aart Middeldorp for their helpful comments.

References

[Avenhaus and Madlener, 1990] Jürgen Avenhaus and Klaus Madlener. Term rewriting and equational reasoning. In R. B. Banerji, editor, *Formal Techniques in Artificial Intelligence: A Sourcebook*, pages 1–41. Elsevier, Amsterdam, 1990.

[Bachmair and Dershowitz, 1986] 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.

[Bellegarde and Lescanne, 1990] François Bellegarde and Pierre Lescanne. Termination by completion. *Applicable Algebra in Engineering, Communication and Computing*, 1990.

[Bidoit, 1981] Michel Bidoit. *Une méthode de présentation de types abstraits: Applications.* PhD thesis, Université de Paris-Sud, Orsay, France, June 1981. Rapport 3045.

[Cadiou, 1972] J. M. Cadiou. *Recursive Definitions of Partial Functions and their Computations.* PhD thesis, Stanford University, Stanford, CA, March 1972.

[Church, 1941] Alonzo Church. *The Calculi of Lambda Conversion*, volume 6 of *Ann. Mathematics Studies.* Princeton University Press, Princeton, NJ, 1941.

[Dershowitz, 1981] Nachum Dershowitz. Termination of linear rewriting systems. In *Proceedings of the Eighth International Colloquium on Automata, Languages and Programming (Acre, Israel)*, volume 115 of *Lecture Notes in Computer Science*, pages 448–458, Berlin, July 1981. European Association of Theoretical Computer Science, Springer-Verlag.

[Dershowitz, 1982] Nachum Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17(3):279–301, March 1982.

[Dershowitz, 1987] Nachum Dershowitz. Termination of rewriting. *J. Symbolic Computation*, 3(1&2):69–115, February/April 1987. Corrigendum: *4*, 3 (December 1987), 409–410; reprinted in *Rewriting Techniques and Applications*, J.-P. Jouannaud, ed., pp. 69—115, Academic Press, 1987.

[Dershowitz, 1992] Nachum Dershowitz. Hierarchical termination. Unpublished report, Leibnitz Center for Research in Computer Science, Hebrew University, Jerusalem, Israel, December 1992.

[Dershowitz, 1995] Nachum Dershowitz. 33 examples of termination. In H. Comon and J.-P. Jouannaud, editors, *French Spring School of Theoretical Computer Science Advanced Course on Term Rewriting (Font Romeux, France, May 1993)*, volume 909 of *Lecture Notes in Computer Science*, pages 16–26, Berlin, 1995. Springer-Verlag.

[Dershowitz and Hoot, 1995] Nachum Dershowitz and Charles Hoot. Natural termination. *Theoretical Computer Science*, 142(2):179–207, May 1995.

[Dershowitz and Jouannaud, 1990] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Methods and Semantics, chapter 6, pages 243–320. North-Holland, Amsterdam, 1990.

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

[Drosten, 1989] K. Drosten. *Termersetzungssysteme*. PhD thesis, Universitat Passau, Berlin, Germany, 1989. Informatik Fachberichte 210, Springer-Verlag.

[Fernandez and Jouannaud, 1995] Maribel Fernandez and Jean-Pierre Jouannaud. Modular termination of term rewriting systems revisited. In *Proceedings of the Eleventh Workshop on Specification of Abstract Data Types (Santa Margherita de Ligura, Italy)*, Lecture Notes in Computer Science, Berlin, 1995. Springer-Verlag.

[Geser, 1989] Alfons Geser. *Termination Relative*. PhD thesis, Universität Passau, Passau, West Germany, 1989.

[Geupel, 1989] Oliver Geupel. Overlap closures and termination of term rewriting systems. Report MIP-8922, Universität Passau, Passau, West Germany, July 1989.

[Gramlich, 1994] Bernhard Gramlich. Generalized sufficient conditions for modular termination of rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 5:131–158, 1994. A preliminary version appeared in the Proceedings of the Third Internationnal Conference on Algebraic and Logic Programming, Lecture Notes in Computer Science 632, Springer-Verlag, Berlin, pp. 53–68, 1992.

[Gramlich, 1995] Bernhard Gramlich. Abstract relations between restricted termination and confluence properties of rewrite systems. *Fundamenta Informaticae*, September 1995. Preliminary versions appeared as "Relating Innermost, Weak, Uniform and Modular Termination of Term Rewriting Systems" in Proceedings of the Conference on Logic Programming and Automated Reasoning (St. Petersburg, Russia), A. Voronkov, ed., Lecture Notes in Artificial Intelligence 624, Springer-Verlag, Berlin, pp. 285–296 and as SEKI-Report SR-93-09, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, Germany, 1993.

[Huet, 1980] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *J. of the Association for Computing Machinery*, 27(4):797–821, October 1980.

[Klop, 1992] Jan Willem Klop. Term rewriting systems. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 1, pages 1–117. Oxford University Press, Oxford, 1992.

[Knuth and Bendix, 1970] Donald E. Knuth and P. B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, U. K., 1970. Reprinted in *Automation of Reasoning 2*, Springer-Verlag, Berlin, pp. 342–376 (1983).

[Krishna Rao, 1992] M. R. K. Krishna Rao. Modular proofs for completeness of hierarchical systems. Unpublished report, December 1992.

[Krishna Rao, 1993] M. R. K. Krishna Rao. Completeness of hierarchical combinatins of term rewriting systems. In *Proceedings of the Thirteenth Conference on Foundations of Software Technology and Theoretical Computer Science (Bombay, India)*, volume 761 of *Lecture Notes in Computer Science*, pages 125–138, Berlin, 1993. Springer-Verlag.

[Kurihara and Kaji, 1990] Masahito Kurihara and Ikuo Kaji. Modular term rewriting systems and the termination. *Information Processing Letters*, 34:1–4, February 1990.

[Kurihara and Ohuchi, 1990] Masahito Kurihara and Azuma Ohuchi. Modularity of simple termination of term rewriting systems. *Journal of Information Processing Society*, 34:632–642, 1990.

[Middeldorp, 1989] Aart Middeldorp. A sufficient condition for the termination of the direct sum of term rewriting systems. In *Proceedings of the Fourth Symposium on Logic in Computer Science*, pages 396–401, Pacific Grove, CA, 1989. IEEE.

[Middeldorp, 1990] Aart Middeldorp. *Modular Properties of Term Rewriting Systems*. PhD thesis, Vrije Universiteit, Amsterdam, The Netherlands, 1990.

[Middeldorp and Toyama, 1991] Aart Middeldorp and Yoshihito Toyama. Completeness of combinations of constructor systems. In R. Book, editor, *Proceedings of the Fourth International Conference on Rewriting Techniques and Applications (Como, Italy)*, volume 488 of *Lecture Notes in Computer Science*, pages 174–187, Berlin, April 1991. Springer-Verlag.

[O'Donnell, 1977] Michael J. O'Donnell. *Computing in systems described by equations*, volume 58 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1977.

[Ohlebusch, 1993] Enno Ohlebusch. On the modularity of termination of term rewriting systems. Report 11, Abteilung Informationstechnik, Universität Bielefeld, Bielefeld, Germany, 1993.

[Pettorossi, 1981] Alberto Pettorossi. Comparing and putting together recursive path orderings, simplification orderings and non-ascending property for termination proofs of term rewriting systems. In *Proceedings of the Eighth EATCS International Colloquium on Automata, Languages and Programming (Acre, Israel)*, volume 115 of *Lecture Notes in Computer Science*, pages 432–447, Berlin, July 1981. Springer-Verlag.

[Plaisted, 1993] David A. Plaisted. Equational reasoning and term rewriting systems. In D. Gabbay, C. Hogger, J. A. Robinson, and J. Siekmann, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, chapter 5, pages 273–364. Oxford University Press, Oxford, 1993.

[Raoult and Vuillemin, 1980] Jean-Claude Raoult and Jean Vuillemin. Operational and semantic equivalence between recursive programs. *J. of the Association for Computing Machinery*, 27(4):772–796, October 1980.

[Rusinowitch, 1987] Michael Rusinowitch. On termination of the direct sum of term-rewriting systems. *Information Processing Letters*, 26:65–70, 1987.

[Toyama, 1987a] Yoshihito Toyama. Counterexamples to termination for the direct sum for the direct sum of term rewriting systems. *Information Processing Letters*, 25:141–143, 1987.

[Toyama, 1987b] Yoshihito Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *J. of the Association for Computing Machinery*, 34(1):128–143, January 1987.

[Toyama *et al.*, 1989] Yoshihito Toyama, Jan Willem Klop, and Hendrik Pieter Barendregt. Termination for the direct sum of left-linear term rewriting systems. In Nachum Dershowitz, editor, *Proceedings of the Third International Conference on Rewriting Techniques and Applications (Chapel Hill, NC)*, volume 355 of *Lecture Notes in Computer Science*, pages 477–491, Berlin, April 1989. Springer-Verlag.

This article was processed using the LaTeX macro package with LLNCS style