Innocuous Constructor-Sharing Combinations

Nachum Dershowitz

Department of Computer Science
Hebrew University, Givat Ram
Jerusalem 91904
Israel
nachum@uiuc.edu

Abstract. We investigate conditions under which confluence and/or termination are preserved for constructor-sharing and hierarchical combinations of rewrite systems, one of which is left-linear and convergent.

1 Introduction

In recent years there has been a spate of results on properties of rewrite systems that are preserved when two systems, each possessing the property in question, are combined. Unfortunately, these results often do not allow the two systems to share even constructor symbols (in which case they are called "disjoint") and also impose harsh syntactic conditions on both systems.

Our interest in this paper is in confluence and termination in the constructor-sharing case, with an emphasis on minimizing the conditions imposed on one of the systems, usually at the expense of more severe conditions on the other. Our motivation is the notion that an "application programmer" Alice composes a query from functions she has defined in a system R, functions S provided by a "systems programmer" Bob, and constructors. We have in mind for system-provided programs things like streams of natural numbers and pseudo-higher-order functions like "mapcar". In writing R, Alice is responsible for correctness of R, including termination and determinism, but should not have to concern herself with the possibility that something about S will sabotage correctness of R, or vice-versa. We mainly address the case where S is terminating and confluent, but R is not necessarily left-linear.

The issue of modularity of rewrite systems was first raised by Bidoit [4]. An early work dealing with combining terminating systems was [5]; confluence of disjoint systems was considered in [28]; many other properties were dealt with in [17].

Definitions and notation are as in [7] and [13]. See Table 1. We will use R and $\rightarrow_R$ interchangeably for relation R and indicate composition of relations by their juxtaposition.

2 Counterexamples

The following examples $\frac{R}{S}$ serve to demonstrate the necessity of various conditions we impose on the two systems. The first [11] is not confluent though its constructor-sharing components are (see [15]):

R^-	The inverse of R, also denoted ${}_R\!\leftarrow$.
$R^=$	The reflexive closure of R.
R^+	The transitive closure of R.
R^*	The reflexive-transitive closure of R.
$R^{\equiv}$	The symmetric-reflexive-transitive closure of R. Also $\leftrightarrow_R^*$.
$R^!$	The normalization relation: $x \to_R^! y$ if $x \to_R^* y$ and there is no z such that $y \to_R z$.
$R^\downarrow$	The join relation: $x \downarrow_R y$ if $x \to_R^* {}_R^*\!\leftarrow y$.
R/S	Rewriting modulo: $R/S = S^*RS^*$.
$WN(R)$	R is (weakly) normalizing: for all x there is a y such that $x \to_R^! y$.
$SN(R)$	R is terminating (strongly normalizing), or, equivalently, R^+ is a well-founded ordering.
$SC(R)$	R is strongly confluent: $R^-R \subseteq R^=(R^=)^-$; also denoted $WCR^{\leq 1}$.
$CR(R)$	R is Church-Rosser, or confluent; this is equivalent to $SC(R^*)$.

Table 1. Definitions and Notations.

$$\frac{\begin{array}{c} g(x,x) \to 0 \\ g(x,c(x)) \to 1 \end{array}}{a \to c(a)} \tag{A}$$

The following constructor-sharing system (cf. [5]) is not terminating, though its components are:

$$\frac{\begin{array}{c} a \to 0 \\ a \to 1 \end{array}}{f(0,1,x) \to f(x,x,x)} \tag{B}$$

In the following similarly nonterminating combination, the first system is confluent and (left- and right-) linear (the second is not, however):

$$\frac{\begin{array}{c} g(2) \to 0 \\ g(3) \to 1 \end{array}}{f(0,1,x) \to f(x,x,x)} \qquad \begin{array}{c} a \to 2 \\ a \to 3 \end{array} \tag{C}$$

The following nonterminating example (from [30]) has shared constructors on the right only, but is not left-linear:

$$\frac{\begin{array}{c} a \to 0 \\ a \to 1 \end{array}}{f(x,y,x,y,z) \to f(0,1,z,z,z)} \tag{D}$$

Toyama [27] devised an example with no shared symbols at all:

$$\frac{\begin{array}{c} g(x,y) \to x \\ g(x,y) \to y \end{array}}{f(0,1,x) \to f(x,x,x)} \tag{E}$$

The following is a combination of non-left-linear confluent systems that share no symbols, one of which is non-overlapping (has no critical pairs) and the other has only a trivial overlap:

$$\frac{\begin{array}{c} g(x,x,y) \to y \\ g(x,y,y) \to x \end{array}}{\begin{array}{c} f(x,y,x,y,z) \to f(a,b,z,z,z) \\ a \to 0 \\ b \to 0 \end{array}} \tag{F}$$

Drosten [9] composed the following nonterminating combination of confluent systems, only one of which is not left-linear:

$$\frac{\begin{array}{c} g(x,x,y) \to y \\ g(x,y,y) \to x \end{array}}{\begin{array}{c} f(a,b,x) \to f(x,x,x) \\ f(x,y,z) \to 0 \\ a \to 0 \\ b \to 0 \end{array}} \tag{G}$$

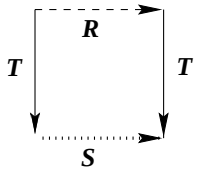
Ohlebusch [20] has shown that termination is also not preserved for confluent non-erasing systems. (An *erasing* system has variables on the left that do not appear on the right.)

3 The Pentagon Property

Before turning to rewrite relations, we look at some abstract properties relating two binary relations R and S.

Suppose T is normalizing and confluent. Then it defines unique normal forms. That is, there is a unique t, denoted $T(s)$, such that $s \to_T^! t$. These normal forms can be used to establish properties of the union $R \cup S$.

Define the *square property* $Q(R,S,T)$ as the inclusion $T^- R \subseteq S T^-$. Figuratively, that is:



Clearly,

Fact 1.

$$CR(R) \wedge CR(S) \wedge Q(R^*, R^*, S^*) \quad \Rightarrow \quad CR(R \cup S)$$

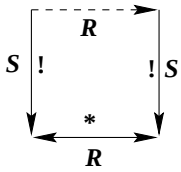
(just by tiling the diamond).

It is well-known that:

Fact 2.

$$CR(R) \wedge CR(S) \wedge WN(S) \wedge Q(R, R^{\equiv}, S^!) \quad \Rightarrow \quad CR(R \cup S)$$

(See, for example, [1].) The square property here looks like:



Proof. If $s \rightarrow_S t$, then $S(s) = S(t)$, since normal forms are unique. If $s \rightarrow_R t$, then the given square property means that $S(s) \leftrightarrow_R^* S(t)$. So whenever $s \leftrightarrow_{R \cup S}^* t$ we have $S(s) \leftrightarrow_R^* S(t)$. By confluence of R we have $s \rightarrow_S^* S(s) \downarrow_R S(t) \,{}_S^*\!\leftarrow t$. Thus, $R \cup S$ is confluent.

We say that R *preserves* S-normal forms if $s \rightarrow_S^! t \rightarrow_R u$ whenever $s \rightarrow_S^! t \rightarrow_R \rightarrow_S^! u$. We will use $PNF(R, S)$ to denote this property.

It should be obvious that normal forms are preserved if S quasi-commutes over R:

Fact 3 [26].

$$RS \subseteq SR^* \quad \Rightarrow \quad PNF(R, S)$$

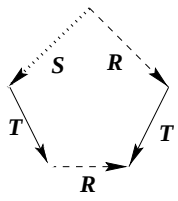
It is also not hard to see that preservation of normal forms allows one to extend the square property from T^* to $T^!$ (when $T^!$ is defined):

Lemma 4.

$$Q(R, S, T^*) \wedge PNF(S, T) \wedge WN(T) \quad \Rightarrow \quad Q(R, S, T^!)$$

Preservation of normal forms is also used to such ends in [6] and [8].

Define the *pentagon property* $P(R, S, T)$ as the inclusion $S^- R \subseteq TRT^-$. (This property is similar to coherence, as explored in [12].) Figuratively, that is:



Obviously:

Fact 5.

$$Q(R, R, S^!) \wedge WN(S) \quad \Rightarrow \quad P(R, S^*, S^!)$$

and

Fact 6.

$$P(R, S^*, S^!) \quad \Rightarrow \quad Q(R, R, S^!) \quad \Rightarrow \quad Q(R, R^{\equiv}, S^!)$$

Accordingly, we look for conditions under which $P(R, S^*, S^!)$ holds.

If S is convergent (confluent and terminating), then the pentagon property extends from a single S step to arbitrarily many:

Lemma 7.

$$P(R, S, S^*) \wedge CR(S) \wedge SN(S) \quad \Rightarrow \quad P(R, S^*, S^*)$$

Proof. The proof is a straightforward induction with respect to the well-founded relation S: Suppose $u \rightarrow_S v \rightarrow_S^* s$ and $u \rightarrow_R t$. It is given that there are u', t' such that $v \rightarrow_S^* u' \rightarrow_R t'$ and $t \rightarrow_S^* t'$. By $CR(S)$ there is an s' such that $s \rightarrow_S^* s'$ and $u' \rightarrow_S^* s'$. Thus, $u' \rightarrow_S^* s'$ and $u' \rightarrow_R t'$. Since u is greater than u' with respect to S, by induction, $s' \rightarrow_S^* \rightarrow_R w$ and $t' \rightarrow_S^* w$ for some w. Putting everything together, we have $s \rightarrow_S^* s' \rightarrow_S^* \rightarrow_R w$ and $t \rightarrow_S^* t' \rightarrow_S^* w$, which is $P(R, S^*, S^*)$. (See the Appendix.)

Similarly:

Lemma 8.

$$P(R, S, S^*) \wedge SC(S) \quad \Rightarrow \quad P(R, S^*, S^*)$$

What we really want is $P(R, S^*, S^!)$:

Lemma 9.

$$P(R, S^!, S^*) \wedge PNF(R, S) \wedge WN(S) \quad \Rightarrow \quad P(R, S^*, S^!)$$

Lemma 10.

$$WN(S) \wedge P(R, S^!, S^!) \quad \Rightarrow \quad P(R, S^*, S^!)$$

Note that:

Lemma 11.

$$WN(S) \wedge P(R, S^!, S^*) \quad \Rightarrow \quad P(R, S^*, S^*)$$

It is easy to see that:

Lemma 12.

$$P(R, S^!, S^*) \wedge PNF(R, S) \wedge CR(S) \quad \Rightarrow \quad Q(S^{\equiv} R S^{\equiv}, R, S^!)$$

Proof. See the Appendix.

Theorem 13.

$$P(R, S^!, S^*) \wedge PNF(R, S) \wedge CR(R) \wedge CR(S) \wedge WN(S) \quad \Rightarrow \quad CR(R \cup S)$$

Proof. String together Fact 1, Lemma 9, and Fact 6.

Lemma 14.

$$P(R, S^*, S^*) \wedge SC(R) \wedge CR(S) \wedge WN(S) \quad \Rightarrow \quad SC(S^* R)$$

Strong confluence and confluence of R^* are the same; strong confluence of $S^* R^*$ implies confluence of $R \cup S$.

Proof. See the Appendix.

Theorem 15.

$$P(R, S^!, S^*) \wedge SC(R) \wedge CR(S) \wedge WN(S) \quad \Rightarrow \quad CR(R \cup S)$$

Proof. This follows, by tiling, from Lemmas 11 and 14.

To summarize, we have two approaches to showing confluence of the union of two confluent systems R and S, when S is terminating. They require the pentagon property $P(R, S, S^*)$ (for Lemma 7) and either (a) strong confluence of R (Theorem 13) or (b) preservation of S-normal forms (Theorem 15).

4 Confluence

Turning to rewrite systems, we look for sufficient conditions under which the pentagon property $P(R, S, S^*)$ holds. The difficulty is that preservation of normal forms is hard to achieve for practical systems, since constructor-topped right sides of R can easily overlap constructor patterns of left-sides of S.

Toyama [28] proved that the union of two confluent systems that share no symbols is confluent (for example, System G), but we are interested in systems that at least share constructor symbols. System (A) shows that such a union need not be confluent; one result for the constructor sharing case is that unique-normalization is preserved [22]; a hierarchical combination need not be confluent, even if the union is normalizing [G. Sivakumar, private communication, 1986]:

$$\frac{\begin{array}{c} g(x,x) \to c(x) \\ g(x, c(x)) \to 1 \end{array}}{f(0,y) \to g(f(y,0), f(y,y))} \qquad \text{(H)}$$

Hereon, R and S will be rewrite systems for which there are no critical pairs between left sides of one and the other. We denote this $R_l \perp S_l$ and say that their left sides do not overlap. When we impose the condition $R_r \perp S_l$, we mean that there are no critical pairs between right sides of R and left sides of S. In particular, R could not even have, in this case, a variable right-hand side (i.e. it must be *non-collapsing*), since that variable unifies with all (renamed) left sides of S. We do allow shared constructors and non-constructors, unless otherwise indicated. To indicate that neither side of R overlaps S on the left, we write $R \perp S_l$.

If the union is terminating, then as a well-known consequence of Knuth's Critical Pair Lemma, the union is also confluent:

Lemma 16.

$$CR(R) \wedge CR(S) \wedge SN(R \cup S) \wedge R_l \perp S_l \quad \Rightarrow \quad CR(R \cup S)$$

The union need not be terminating, however (System B). Even if no symbols are shared, it need not (System E).

We use $LL(R)$ to indicate that R is left-linear. A left-linear system R is *orthogonal* if $R_l \perp R_l$, that is, if it has no critical pairs between its left sides (except for a left side with all of itself). Orthogonal systems are confluent:

Fact 17.

$$LL(R) \wedge R_l \perp R_l \quad \Rightarrow \quad SC(R^{\|})$$

The parallel rewriting relation $R^{\|}$ applies one rule of R at any number of disjoint positions in a term. In general, one can show that R is confluent by showing $CR(R')$ for any R', like $R^{\|}$, whose reflexive-transitive closure is the same.

The following is standard (by considering the relative positions of redexes):

Lemma 18.

$$LL(S) \wedge R_l \perp S_l \quad \Rightarrow \quad P(R^{\|}, S, S^*)$$

Clearly, the union of two orthogonal systems is orthogonal, hence, confluent. This has been weakened to allow critical pairs within each system:

Theorem 19 [24].

$$LL(R) \wedge LL(S) \wedge CR(R) \wedge CR(S) \wedge R_l \perp S_l \quad \Rightarrow \quad CR(R \cup S)$$

System (A) demonstrates the need for left linearity.

We can allow non-collapsing R to be non-left-linear, by imposing termination on S. Specifically:

Theorem 20.

$$CR(R) \wedge CR(S) \wedge LL(S) \wedge SN(S) \wedge R_l \perp S_l \wedge PNF(R, S) \quad \Rightarrow \quad CR(R \cup S)$$

Again, System (A) demonstrates the need for left-linearity (or preservation) and termination.

Proof. We use the abstract properties of the previous section. By Lemma 18, we have $P(R^{\|}, S, S^*)$, which by Lemma 7 gives $P(R^{\|}, S^*, S^*)$. Since R preserves normal forms, so does $R^{\|}$, and by Lemma 9, we get $P(R^{\|}, S^*, S^!)$. This implies $Q(R^{\|}, R^{\|}, S^!)$ and, by Fact 2, yields confluence.

The following should be clear:

Fact 21 [26].

$$LL(S) \wedge R_r \perp S_l \quad \Rightarrow \quad PNF(R, S)$$

Indeed, it is virtually imperative that S be left-linear (or else, every redex of a non-left-linear rule in S must be reducible by some other rule in S).

Hence:

Corollary 22.

$$SN(S) \wedge LL(S) \wedge CR(R) \wedge CR(S) \wedge R \perp S_l \quad \Rightarrow \quad CR(R \cup S)$$

In particular, since orthogonal systems are confluent, the union of an orthogonal system R that has defined symbols at the top of every right side with any constructor-sharing left-linear convergent system S is also confluent.

Unfortunately, this does not allow the right sides of R (which might ordinarily be variables or constructor terms) to overlap left sides of S, as in

$$\begin{array}{r} f(x) \to g(a) \\ g(x) \to x \\ \hline h(x,x) \to 0 \\ \hline a \to 1 \end{array} \qquad \text{(I)}$$

Here

$$h(f(0),1) \underset{S}{\longleftarrow} h(f(0),a) \underset{R}{\longrightarrow} h(g(a),a) \underset{R}{\longrightarrow} h(a,a) \underset{R}{\longrightarrow} 0$$

but not

$$h(f(0),1) \overset{*}{\underset{S}{\longrightarrow}} \overset{*}{\underset{R}{\longrightarrow}} \overset{*}{\underset{S}{\longleftarrow}} 0$$

When R is collapsing, S needs to have very simple left-hand sides, or else, R could easily destroy its normal forms. Suppose S is a rule of the form $f(x_1,\ldots,x_n) \to r$, where the x_i are distinct variables. Any (finite or infinite) system of such rules, each for a different defined symbol f, is confluent and is called a *recursive program scheme*. The following is easy:

Lemma 23. *Every system R preserves normal forms of any recursive program scheme S.*

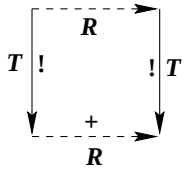
It follows from Theorem 20 that:

Corollary 24. *The union of a terminating recursive program scheme with any confluent system not having the scheme's defined symbols on its left side is confluent.*

5 Termination

Turning to termination, numerous results are known in the disjoint, shared constructor, and hierarchical cases. We are primarily interested in the shared constructor case and in results that impose minimal restrictions on R (other than termination, of course). There are few such: for constructor-based systems in [19]; for right-linear and non-duplicating S in [21] (for the disjoint case, it conjectured by [25] and proved in [16]). Some results for the hierarchical case are contained in [6] and [14].

A standard method of proving termination of a union $R \cup S$ is to find a convergent transformation function T, containing S, such that $Q(R, R^+, T^!)$:



Thus, any derivation containing infinitely many R (and S) steps would map via T to an infinite derivation in R alone. In other words,

Fact 25.

$$SN(R) \wedge SN(S) \wedge S \subseteq T \wedge CR(T) \wedge Q(R, R^+, T^!) \quad \Rightarrow \quad SN(R \cup S)$$

(Cf. [2, Theorem 4].)

For example, let S be a *symbolic interpretation* (as defined in [2]), that is, S consists of a rule $f(x_1, \ldots, x_n) \to r$ such that all x_i, but not f, appear in r. By the above considerations, and using S for T, the combined system is terminating, since any R redex above an f is still a redex after normalizing by S and any redex below an f occurs at least once in r. Applying R can only duplicate redexes of S, but cannot create new ones. The extra restriction on the occurrences of x_i in r are not, however, necessary, as we will see below.

One does not really need *every* R step to map to a strict decrease vis-a-vis R, only that were there an infinite derivation in the union, the decrease would be strict infinitely often.

For example, we have already seen (Fact 18) that $P(S^*, R^{\|}, S^*)$ holds whenever the left sides of R and left-linear S do not overlap, whence it follows that

$$S(s) \xrightarrow[R]{*} u \xleftarrow[S]{*} t$$

whenever $s \to_R t$. When R preserves S-normal forms, we have in fact that u is an S-normal form (of t). Hence, $S(s)$ rewrites to $S(t)$ in zero or more R steps. These conditions hold, for example, in the case considered in [29, Appendix B], where R and S share no symbols, S is left-linear, confluent and terminating, and R is non-collapsing. Since every right side of R contains a symbol that does not appear in S, any R step below and preceding an S step must be in the variable part of the left side of the rule of S. To establish termination, one needs the added consideration that an infinite derivation with fewest alternations of layers of symbols from R and S must have infinitely many steps in the uppermost layer.

Theorem 26. *The union of a terminating system S consisting of one rule of the form $f(x_1, \ldots, x_n) \to r$, where the x_i are distinct variables, and a terminating system R that does not contain the symbol f is terminating.*

Proof. As we have stated in Lemma 23, R preserves S-normal forms. Consider an infinite derivation $t_0 \to t_1 \to t_2 \to \cdots$ in the union, initiated by the smallest possible term t_0. Since R preserves normal-forms, we have $S(t_0) \to_R^* S(t_1) \to_R^* S(t_2) \to_R^* \cdots$. Clearly, $S(t) = S(t')$ if $t \to_S t'$, but there must be infinitely many R steps as well. The only way one could have $S(t) = S(t')$ when $t \to_R t'$ would be if there were an f above the R redex. Were that S-redex to be present throughout the infinite derivation, then

there would be a "smaller" infinite derivation with that redex omitted. It could not be that an f were the top-most symbol throughout the derivation, since then either the top is never rewritten, in which case one of its proper subterms has an infinite derivation, or else the top is rewritten infinitely by S, with R steps interspersed below, contradicting the fact that S is terminating. Were the uppermost S redex to be the result of an R step, then too there must be a smaller infinite derivation initiated by the ancestor of that S redex in the first term of the derivation, since the only way an application of R can replace a redex with a term headed by an f is if that f is a descendent of one already occurring in the initial term of the infinite derivation (R has no f's of its own).

In particular:

Corollary 27 [2]. *The union of a terminating (non-erasing) symbolic interpretation S and a terminating system R that does not contain the interpretation's defined symbols is terminating.*

Corollary 28. [1] *The union of a projection rule $f(x_1, \ldots, x_n) \to x_i$ and a terminating system R that does not contain f is terminating.*

System (E) shows that one cannot have more than one rule per f.

The above theorem can be trivially iterated to allow any number of such rules in S for different f, yielding a new proof of:

Corollary 29 [18]. *The union of a terminating recursive program scheme S and a terminating system R that does not contain the scheme's defined symbols is terminating.*

Systems (B,E) show what happens if one has constructor terms rather than variables in a scheme for f. All the same, we show now how to allow constructor-based linear left sides $f(c_1, \ldots, c_n)$, where the c_i are constructor terms (built from free constructors and variables), by insisting that R be consistent—in the sense that it does not equate instances of distinct constructor terms.

Theorem 30. *The union of two constructor-sharing terminating systems R and S is terminating if S is confluent, left-linear, and constructor-based and any R-unifier of constructor terms appearing on the left of S is also an ordinary unifier.*

Systems (B,E) are inconsistent; (G) shows what happens when S is not constructor-based; (C) shows the need for confluence; and (D,F) show the need for left-linearity.

Proof. The proof is similar to the previous, but uses the extended relation $T = R \backslash S$, instead of plain S, to take normal forms. This rewrite relation is the subset of S/R that allows R steps only below the redex p prior to applying a rule in S:

$$t[s]_p \to^p_{R \backslash S} t' \quad \equiv \quad t[s']_p \to^p_S t' \wedge s \leftrightarrow^*_R s'$$

[1] This result, and suggested proofs, were the object of discussions with Aart Middeldorp, Albert Rubio, and Hans Zantema, at the Termination Workshop in La Bresse, France, 1995.

This relation T is normalizing by innermost rewriting, as is the case for constructor-sharing systems, in general. Furthermore, T is locally confluent, since the conditions on R prevent it from introducing new critical pairs. It is, therefore, terminating (since it is overlaying; see [10]) and confluent. So T gives unique normal forms and $T^!$ is well-defined. Also, R preserves T-normal forms, since it does not introduce any fs, and any effect it can have to create a pattern c_i is taken into account by the R steps that are anyway allowed in applying the extended relation T.

Consider, as before, a minimal infinite derivation $t_0 \to t_1 \to t_2 \to \cdots$, for which we have $T(t_0) \to_R^* T(t_1) \to_R^* T(t_2) \to_R^* \cdots$. Clearly, $T(t) = T(t')$ if $t \to_S t'$, but there must be infinitely many R steps as well. This time, we cannot have infinitely many S rewrites at the top, with R steps interspersed below, since that would contradict the fact that T is terminating. So, there must be infinitely many R steps on top, leading to infinitely many strict decreases $T(t_{i_0}) \to_R^+ T(t_{i_1}) \to_R^+ \cdots$.

This applies to the system

$$\begin{array}{c} mapf(\epsilon) \to \epsilon \\ mapf(x : y) \to f(x) : mapf(y) \end{array} \tag{J}$$

conjectured in [6] to be terminating in conjunction with arbitrary rewrite system R (ϵ and : are constructors). If R is inconsistent, however, as for example

$$\begin{array}{cc} f(0) \to 1 & f(x) \to g(a, a, x) \\ g(\epsilon, 1 : \epsilon, x) \to g(x, x, x) & g(x, y, z) \to 1 \\ a \to \epsilon & \\ a \to 0 : \epsilon & \end{array} \tag{K}$$

then the union need not be terminating. In this case,

$$\begin{array}{l} g(mapf(a), mapf(a), mapf(a)) \to g(mapf(\epsilon), mapf(a), mapf(a)) \to \\ g(mapf(\epsilon), mapf(0 : \epsilon), mapf(a)) \to g(\epsilon, mapf(0 : \epsilon), mapf(a)) \to \\ g(\epsilon, f(0) : mapf(\epsilon), mapf(a)) \to g(\epsilon, f(0) : \epsilon, mapf(a)) \to \\ g(\epsilon, 1 : \epsilon, mapf(a)) \to g(mapf(a), mapf(a), mapf(a)) \to \cdots \end{array}$$

The rules on the right are not needed for nontermination; they are included only to make it have unique constructor normal forms.

Corollary 31. *The union of two constructor-sharing confluent and terminating rewrite systems, one of which is constructor-based and left-linear, is terminating and confluent.*

Proof. Confluence provides consistency. Confluence of the union follows from the Critical Pair Lemma.

6 Remarks

A *stream system* S (for f) is a confluent set of left-linear constructor-based rules of the form $f(c_1, \ldots, c_n) \to r$, with r headed by a constructor. (We can always make such a system terminating—but nonconfluent—by adding an extra argument to each

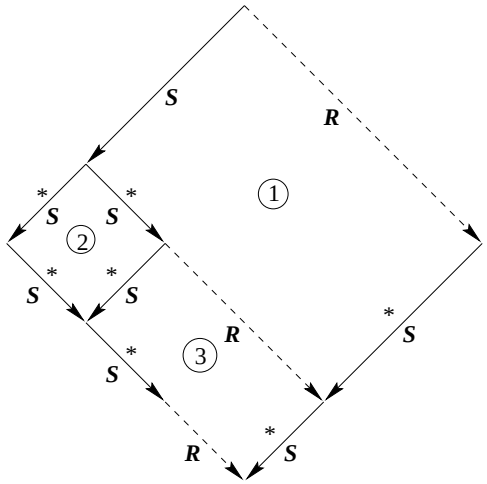
occurrence of f in a rule. The f on the left becomes $f_{s(n)}$ and those in r become f_n, where n is a new variable.) The ideas of this paper can be extended to show that such streams do not destroy confluence and termination of R/S.

We have tended towards the abstract in this paper in the hope that additional applications might surface. We are aware that some of the sufficient conditions we give can be refined to demand only exactly what is required by the proof. It would also be nice to have counterexamples for the necessity of each condition. Finally, we should mention that critical pair conditions for the pentagon property could be formulated, along the lines of the work in [12, 3, 23].

Appendix

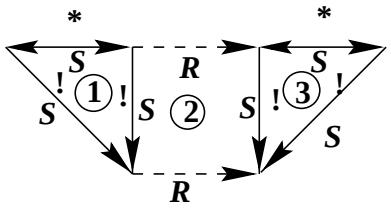
This section contains diagrammatic proofs of several key lemmas. We use dashed arrows for steps in R and solid for S.

Proof of Lemma 7. The proof of the pentagon property $P(R, S^*, S^*)$ is by induction with respect to the terminating relation S:



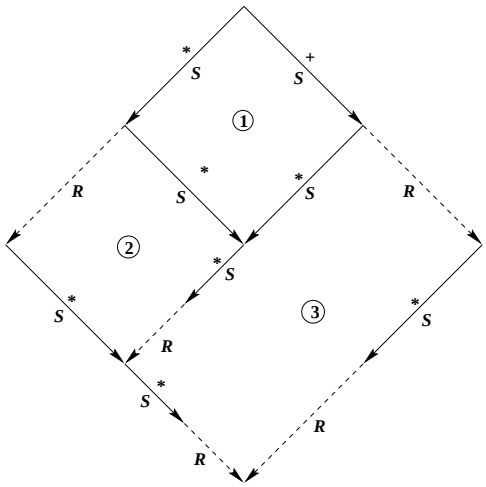
1. Pentagon property $P(R, S, S^*)$.
2. Confluence of S.
3. By induction.

Proof of Lemma 12. The proof is straightforward



1. S has unique normal forms.
2. Pentagon property $P(R, S^!, S^*)$ and preservation.
3. S has unique normal forms.

Proof of Lemma 14. The proof of strong confluence of S^*R is by induction with respect to terminating S.



1. Confluence of S.
2. Pentagon property $P(R, S^*, S^*)$.
3. Induction.

References

1. Y. Akama. On Mints' reductions for ccc-calculus. In *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*, pages 1–12, Berlin, 1993. Springer-Verlag.
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.
3. Françoise Bellegarde and Pierre Lescanne. Termination by completion. *Applied Algebra on Engineering, Communication and Computer Science*, 1(2):79–96, 1990.
4. Michel Bidoit. *Une méthode de présentation de types abstraits: Applications.* PhD thesis, Université de Paris-Sud, Orsay, France, June 1981. Rapport 3045.
5. 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.
6. Nachum Dershowitz. Hierarchical termination. In N. Dershowitz and N. Lindenstrauss, editors, *Proceedings of the Fourth International Workshop on Conditional and Typed Rewriting Systems (Jerusalem, Israel, July 1994)*, volume 968 of *Lecture Notes in Computer Science*, pages 89–105, Berlin, 1995. Springer-Verlag.

7. 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.
8. Roberto di Cosmo. On the power of simple diagrams. In *Proceedings of the Seventh International Conference on Rewriting Techniques and Applications (New Brunswick, NJ)*, volume 1103 of *Lecture Notes in Computer Science*, pages 200–214. Springer-Verlag, July 1996.
9. K. Drosten. *Termersetzungssysteme*. PhD thesis, Universitat Passau, Berlin, Germany, 1989. Informatik Fachberichte 210, Springer-Verlag.
10. Bernhard Gramlich. Relating innermost, weak, uniform and modular termination of term rewriting systems. In A. Voronkov, editor, *Proceedings of the Conference on Logic Programming and Automated Reasoning (St. Petersburg, Russia)*, volume 624 of *Lecture Notes in Artificial Intelligence*, pages 285–296, Berlin, July 1992. Springer-Verlag.
11. 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.
12. Jean-Pierre Jouannaud and Hélène Kirchner. Completion of a set of rules modulo a set of equations. *SIAM J. on Computing*, 15:1155–1194, November 1986.
13. 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.
14. M. R. K. Krishna Rao. Modular proofs for completeness of hierarchical term rewriting systems. *Theoretical Computer Science*, 151:487–512, 1995.
15. M. Kurihara and A. Ohuchi. Modularity of simple termination of term rewriting systems with shared constructors. *Theoretical Computer Science*, 103:273–282, 1992.
16. 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.
17. Aart Middeldorp. *Modular Properties of Term Rewriting Systems*. PhD thesis, Vrije Universiteit, Amsterdam, The Netherlands, 1990.
18. Aart Middeldorp, Hitoshi Ohsaki, and Hans Zantema. Transforming termination by self-labelling. In M. A. Robbie and J. K. Slaney, editors, *Proceedings of the Thirteenth International Conference on Automated Deduction*, volume 1104 of *Lecture Notes in Artificial Intelligence*, pages 373–387, Berlin, July/August 1996. Springer-Verlag.
19. 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.
20. Enno Ohlebusch. Termination is not modular for confluent variable-preserving term rewriting systems.
21. Enno Ohlebusch. On the modularity of termination of term rewriting systems. Report 11, Abteilung Informationstechnik, Universität Bielefeld, Bielefeld, Germany, 1993.
22. Enno Ohlebusch. On the modularity of confluence of constructor-sharing term rewriting systems. In *Proceedings of the Colloquium on Trees in Algebra and Programming*, volume 787 of *Lecture Notes in Computer Science*, pages 261–275, Berlin, 1994. Springer-Verlag.
23. Christian Prehofer. On modularity in term rewriting and narrowing. In J.-P. Jouannaud, editor, *Proceedings of the First International Conference on Constraints in Computational Logics*, volume 845 of *Lecture Notes in Computer Science*, pages 253–268, Berlin, 1994. Springer-Verlag.

24. 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.
25. Michael Rusinowitch. On termination of the direct sum of term-rewriting systems. *Information Processing Letters*, 26:65–70, 1987.
26. Karl Stroetmann. The union of rewrite systems. Cited in [23].
27. Yoshihito Toyama. Counterexamples to termination for the direct sum for the direct sum of term rewriting systems. *Information Processing Letters*, 25:141–143, 1987.
28. 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.
29. Yoshihito Toyama, Jan Willem Klop, and Hendrik Pieter Barendregt. Termination for direct sums of left-linear complete term rewriting systems. *J. of the Association for Computing Machinery*, 42(6):1275–1304, November 1995.
30. Xubo Zhang. Overlap closures do not suffice for termination of general term rewriting systems. *Information Processing Letters*, 37(1):9–11, 1991.