Termination by completion

François Bellegarde, Pierre Lescanne

INRIA

UNITE DE RECHERCHE
INRIA-LORRAINE

Rapports de Recherche

N° 1174

Programme 1
Programmation, Calcul Symbolique
et Intelligence Artificielle

TERMINATION BY COMPLETION

Françoise BELLEGARDE
Pierre LESCANNE

Mars 1990

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

RR_1174

Terminaison par Complétion
Termination by Completion

Françoise BELLEGARDE et Pierre LESCANNE

Résumé

Ce rapport présente une procédure de complétion pour prouver la terminaison des systèmes de réécriture. Il fonctionne de la façon suivante: étant donnés un système de réécriture R qui est supposé se terminer et un système de réécriture T utilisé pour transformer R, la complétion construit un système de réécriture auxiliaire S, le système transformé de R par S. La terminaison de S et T associée à une propriété appelée coopération locale implique la terminaison de R. Si le processus se termine cela fournit une preuve automatique de terminaison de R.

Abstract

This paper presents a completion procedure for proving termination of term rewrite systems. It works as follows. Given a term rewrite system R supposed to terminate and a term rewrite system T used to transform R, the completion builds an auxiliary term rewrite system S, the system transformed of R by T. The termination of S and T associated with a property called local cooperation implies the termination of R. If the process terminates this provides a mechanical proof of the termination of R.

Termination by Completion

Françoise BELLEGARDE*, Pierre LESCANNE
Centre de Recherche en Informatique de Nancy, CNRS
and INRIA-Lorraine
Domaine Scientifique Victor Grignard, BP 239,
54506 Vandœuvre-lès-Nancy, France
email: bellegar@cse.ogi.edu, lescanne@loria.crin.fr

Abstract

This paper presents a completion procedure for proving termination of term rewrite systems. It works as follows. Given a term rewrite system R supposed to terminate and a term rewrite system T used to transform R, the completion builds an auxiliary term rewrite system S, the system transformed of R by T. The termination of S and T associated with a property called local cooperation implies the termination of R. If the process terminates this provides a mechanical proof of the termination of R.

Introduction

Among the problems related to rewrite systems, termination is probably one of the most difficult. In particular, it has been proved to be undecidable [HL78] even for a system with one rule [Dau89] and some specific termination proofs are hard problems [HL86]. Since it has many practical applications, especially in completion procedures, many computer aided methods have been proposed. Based on simplification orderings they are usually limited by the fact they do not accept the left-hand side of a rule to be embedded in the right-hand side. The transformation orderings proposed here do not have this limitation.

There are essentially two families of orderings on terms for proving termination. One is called *syntactical* and orders the terms from a careful analysis of their structure. Among these syntactical orderings are the "recursive path orderings" [Der82], the "recursive path of subterm ordering" [Pla78] and the "recursive decomposition ordering" [JLR82,Les89b], see also [Rus87]. These orderings are easy to use since they are based on a concept of precedence which is somewhat natural, especially in the context of abstract data types. A naive view sees the precedence as a measure of the complexity of the definition of the operators if one considers rules as definitions. Another family of orderings contains those called *semantical*. The semantical orderings interpret the terms in another structure where a well-founded ordering is known. There are basically two familiar well-ordered sets: the natural numbers and the terms ordered by a syntactical ordering. Taking the first choice, for instance by considering the size of the terms, is not not general enough and does not provide orderings that are fully invariant, i.e, stable by substitutions. This leads to consideration of functions over the natural numbers to insure full invariance and, since they are simple to manipulate, polynomials [Lan79,Hue80] are often used and an implementation of this

*On leave at Oregon Graduate Center, Beaverton, Oregon

method can be made [BL87b]. The ordering described by Knuth and Bendix in their paper [KB70] is actually a composition of a semantical ordering (an interpretation with polynomials of degree one) and of a syntactical ordering.

From a semantic view, if the set of terms ordered by a syntactical ordering is chosen as a target for the "semantics", this means that one transforms the rewrite system into another one. For convenience this is more suitably described by a rewrite system [BL87a] and properties of commutation have to be checked [BD86]. If we call R the system of which one wants to prove the termination and T the transforming system, the goal is to build a system S with good properties such that the termination of R comes directly from the termination of S. This paper describes a completion procedure that mechanically builds S, given R and T. This procedure is given by inference rules.

This work originated from [BL86] and was described in [BL87a]. Some results were independently proposed by Bachmair and Dershowitz [Bac88,BD86] and their presentation has st strongly influenced our presentation. A generalization was done recently by Geser [Ges89]. This paper contains in addition a description of the completion procedure which was implemented in *CAML* by Bruno Galabertier [Gal88] and many simplification ordering resistant examples all tested with the implementation.

Notations

We suppose the reader is familiar with the basic features and notations of term rewrite systems [DJ89]. Let F be a set of operator symbols and V be a set of variables, $\mathcal{T}(F, V)$ is the set of terms with symbols in F and variables in V.

The relations on terms $\rightarrow^{-1}$, or $\leftarrow$ denote the *converse* of the relation $\rightarrow$ between two terms. $\rightarrow^*$ and $\rightarrow^+$ denote respectively the reflexive and transitive closure and the transitive closure of $\rightarrow$. Given a relation R, we freely write it R or $\rightarrow_R$; the second form is used generaly with rewrite relations. We write $R_1.R_2$ or $\rightarrow_{R_1}.\rightarrow_{R_2}$ for the *composition* of the two relations $\rightarrow_{R_2}$ and $\rightarrow_{R_1}$. We write $R_1 \subseteq R_2$ or $\rightarrow_{R_1} \subseteq \rightarrow_{R_2}$ if $\{(x,y) \mid x \rightarrow_{R_1} y\} \subseteq \{(x,y) \mid x \rightarrow_{R_2} y\}$ as sets.

A (rewrite) *rule* is an oriented pair of terms. $\Sigma(\mathcal{T}(\mathcal{F}, \mathcal{X}))$ is the set of substitutions with range $\mathcal{T}(\mathcal{F}, \mathcal{X})$. We say that a relation $\vdash$ satisfies the *replacement property* if

$$s \vdash t \Rightarrow f(\ldots, s, \ldots) \vdash f(\ldots, t, \ldots).$$

A relation $\vdash$ is *invariant* for a set Σ of substitutions if

$$(\forall \sigma \in \Sigma)\ s \vdash t \Rightarrow s\sigma \vdash t\sigma.$$

A relation is *fully invariant* if it is invariant for $\Sigma(\mathcal{T}(\mathcal{F}, \mathcal{X}))$. A fully invariant relation which has the replacement property is called a *rewrite relation.*

$\rightarrow$ is *noetherian* (or by analogy with rewrite systems one says also that $\rightarrow$ *terminates* and one speaks of the *termination of the relation*) if and only if there is no infinite sequence of terms $t_1, t_2, \ldots$, such that $t_1 \rightarrow t_2 \rightarrow \ldots$. The relation $\rightarrow$ is *confluent* if for all terms t, t_1 and t_2 such that $t \rightarrow^* t_1$ and $t \rightarrow^* t_2$, there exists a term t' such that $t_1 \rightarrow^* t'$ and $t_2 \rightarrow^* t'$. In terms of inclusion we have $\leftarrow^*.\rightarrow^* \subseteq \rightarrow^*.\leftarrow^*$. *Local confluence* is $\leftarrow.\rightarrow \subseteq \rightarrow^*.\leftarrow^*$. The normal form of a term t is a term $t\downarrow_R$ such that $t \rightarrow_R^* t\downarrow_R$ and there is no u such that $t\downarrow_R \rightarrow_R u$. If $\rightarrow_R$ is noetherian and confluent the normal-form exists and is unique. Given two relation R and S, R/S is called R *modulo* S and stands for the relation $S^*.R.S^*$. Note that R/S and R/S^* are the same.

A rewrite rule is an ordered pair of terms, written $l \to r$, such that $V(r) \subseteq V(l)$ where $V(t)$ is the set of variables occurring in t. A rule $l \to r$ is *left-linear* if each variable occurs only once in l and it is *variable preserving* if $V(r) = V(l)$ A rewrite system is a set R of rules. $\to_R$ or $\to$ if there is no ambiguity is the rewrite relation generated by R. We say that the rewrite system R is confluent or noetherian when the rewrite relation $\to_R$ is confluent or noetherian. A noetherian ordering which has the *replacement property*, i.e. $s \to t$ implies $f(...s...) \to f(...t...)$ for all the operator symbols of F, is called a *reduction ordering.*

1 The Transformation Ordering

Let us consider simple examples in order to motivate and illustrate the transformation ordering and related properties. The following example comes from [Bel86,Bel85]

Example 1 : *Associativity and Endomorphism*

Consider the two identities,

$$\begin{array}{ll} associativity: & (x+y)+z = x+(y+z) \\ endomorphism: & f(x)+f(y) = f(x+y). \end{array}$$

Suppose the identities are oriented from left to right by a recursive path ordering [Der87, KL80] based on a left to right lexicographic status (LR) for the symbol + and a precedence $(+ > f)$ for the symbols + and f. They form the set of rewrite rules,

$$\begin{array}{ll} associativity: & (x+y)+z \to x+(y+z) \\ endomorphism: & f(x)+f(y) \to f(x+y) \end{array}$$

Running the Knuth-Bendix procedure one looks for critical pairs and returns the following one

$$endo\_assoc: f(x+y)+z = f(x)+(f(y)+z).$$

This identity is oriented in a rewrite rule from left to right because of the LR status of +:

$$f(x+y)+z \to f(x)+(f(y)+z)$$

and then the procedure diverges generating infinitely many rules of the form

$$f^n(x+y)+z \to f^n(x)+(f^n(y)+z)$$

In addition, the system

$$\begin{array}{ll} (x+y)+z & \to x+(y+z) \\ f(x)+f(y) & \to f(x+y) \\ & \vdots \\ f^n(x+y)+z & \to f^n(x)+(f^n(y)+z) \\ & \vdots \end{array}$$

is not desirable if one does code optimization. First it is infinite which does not make simplifications easy, second it increases the number of calls to f. Indeed one may imagine that f is an expensive function like a Fourrier Transform and + is the naive addition. Considering the cost of evaluating f, it is expected that its number of occurrences reduces.

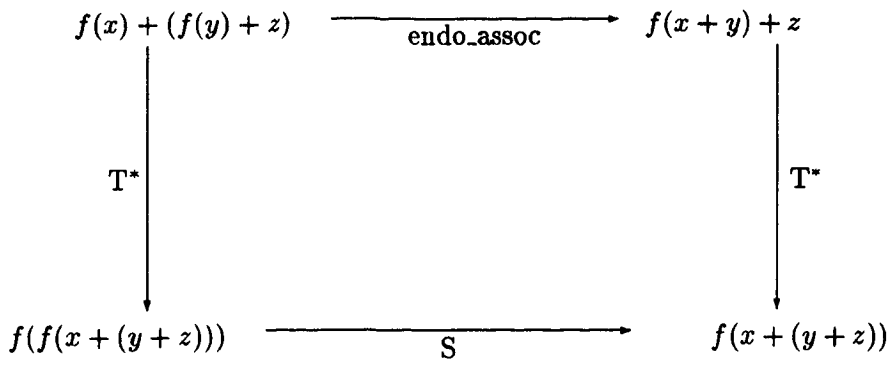

Figure 1: diagram for *endo_assoc*

A better system would be *the rewrite system Assoc+Endo*:

$$\begin{array}{ll} \mathit{associativity}: & (x+y)+z \rightarrow x+(y+z) \\ \mathit{endomorphism}: & f(x)+f(y) \rightarrow f(x+y) \\ \mathit{endo\_assoc}: & f(x)+(f(y)+z) \rightarrow f(x+y)+z \end{array}$$

with the question of finding an ordering for proving its termination. Definitely no precedence based recursive path ordering works. However the two sides of *endo_assoc* could be transformed into two terms where the numbers of occurrences of f is preserved and that can be easily compared by a recursive path ordering $f(f(x+(y+z)))$ for the left-hand side and $f(x+(y+z))$ for the right-hand side seem to be good candidates and a rewrite system that realizes this transformation is T:

$$\begin{array}{ll} \mathit{left\_f\_up}: & f(x)+y \rightarrow f(x+y) \\ \mathit{right\_f\_up}: & y+f(x) \rightarrow f(y+x) \\ \mathit{associativity}: & (x+y)+z \rightarrow x+(y+z) \end{array}$$

T is confluent and terminating and normalizes (or transforms) the sides of *endo_assoc* as wished, namely:

$$\begin{array}{ll} (f(x)+(f(y)+z))\downarrow_T & \equiv f(f(x+(y+z))) \\ (f(x+y)+z)\downarrow_T & \equiv f(x+(y+z)) \end{array}$$

If now we consider *the rewrite system S*:

$$f(f(x)) \rightarrow f(x)$$

we get Figure 1 for the rule *endo_asso* and Figure 2 for the rule *endomorphism*.

The rewrite system S terminates and entails the termination of *Assoc+Endo*.

Example 2 The second example comes from the Cartesian category combinators [Cur86], see also [Bel85,Bel86,HL86] where similar systems are presented. Consider the rewrite system C:

$$\begin{array}{ll} \mathit{assoc}: & (x \circ y) \circ z \rightarrow x \circ (y \circ z) \\ \mathit{dist\_pair}: & \langle x, y\rangle \circ z \rightarrow \langle x \circ z, y \circ z\rangle \\ \mathit{idl}: & x \circ I \rightarrow x \\ \mathit{idr}: & I \circ x \rightarrow x \\ p_1: & \mathit{Fst} \circ \langle x, y\rangle \rightarrow x \\ p_2: & \mathit{Snd} \circ \langle x, y\rangle \rightarrow y \end{array}$$

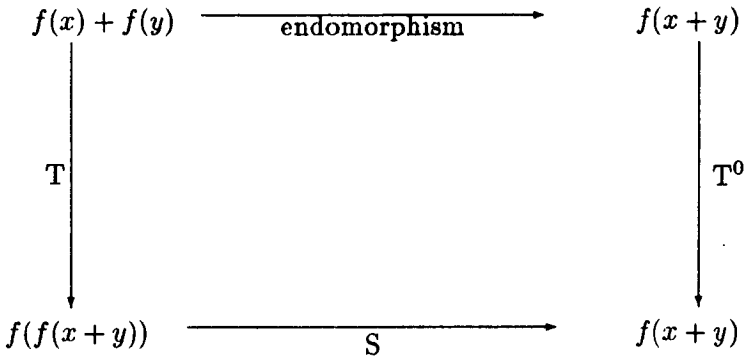

Figure 2: diagram for *endomorphism*

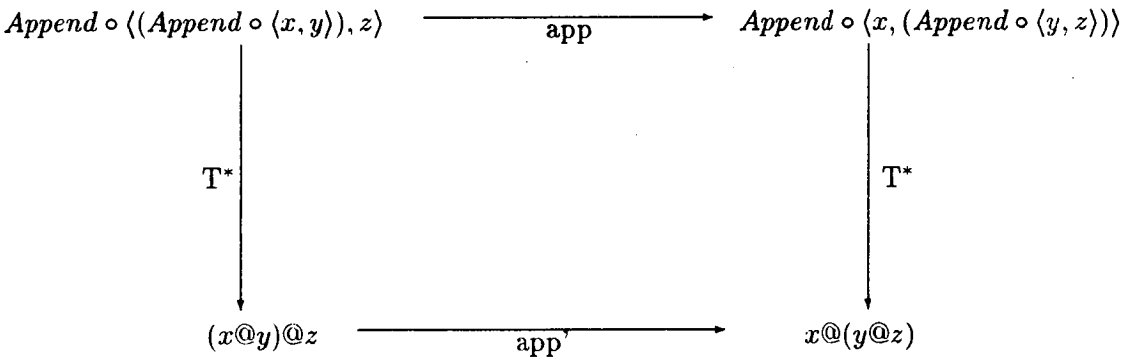

Figure 3: diagram for *app'*

$\circ$ is the composition of combinators and $\langle \_, \_ \rangle$ is the pairing. I is the identity combinator and Fst and Snd are respectively the first projection combinator and the second projection combinator.

C terminates and is confluent. The termination proof uses the recursive path ordering (RPO) with the left to right lexicographic status (LR) for $\circ$ and $\langle \_, \_ \rangle$ and the precedence $(\circ > \langle \_, \_ \rangle)$. Now, let us add another operator *Append* that has arity 0, and, let us consider the following axiom:

$$app : Append \circ \langle (Append \circ \langle x, y \rangle), z \rangle = Append \circ \langle x, (Append \circ \langle y, z \rangle) \rangle.$$

The two sides are incomparable by any recursive path ordering, so no recursive path ordering can prove the termination of the *rewrite system* $R = C \cup \{app\}$.

$$\begin{array}{ll}
assoc: & (x \circ y) \circ z \rightarrow x \circ (y \circ z) \\
dist_pair: & \langle x, y \rangle \circ z \rightarrow \langle x \circ z, y \circ z \rangle \\
idl: & x \circ I \rightarrow x \\
idr: & I \circ x \rightarrow x \\
p_1: & Fst \circ \langle x, y \rangle \rightarrow x \\
p_2: & Snd \circ \langle x, y \rangle \rightarrow y \\
app: & Append \circ \langle Append \circ \langle x, y \rangle, z \rangle \rightarrow Append \circ \langle x, Append \circ \langle y, z \rangle \rangle.
\end{array}$$

similarly no polynomial interpretation works. However, we can think in terms of transformations. The set $\mathcal{T}(F, V)$ of the terms on F and V with $F = \{\circ, \langle \_, \_ \rangle, I, Fst, Snd, Append\}$ is embedded in the set $\mathcal{T}(F', V)$ with $F' = F \cup @$ and the terms are transformed by the rule r_1 of the *rewrite system* T

$$r_1 : Append \circ \langle x, y \rangle \rightarrow x@y$$

this rule rewrites the sides l and r of the rule *app*

$$l \xrightarrow[r_1]{*} (x@y)@z$$

$$r \xrightarrow[r_1]{*} x@(y@z)$$

The rule app' of the *rewrite system* S

$$app' : (x@y)@z \rightarrow x@(y@z),$$

gives the diagram of Figure 3. Since $(x@y)@z >_{RPO} x@(y@z)$ with the left to right lexicographic status for the symbol, the rule app' can be used to prove the termination of R.

Example 3 This example comes from [Der87]. It is interesting because it cannot be proved to terminate using simplification orderings

$$f(f(x)) \rightarrow f(g(f(x)))$$

because it is self embedded which means that the left-hand side is embedded in the right-hand side in the sense of Kruskal's theorem. However thinking in term of transformations, we transform the right-hand side by *the rewrite system T:*

$$f(g(f(x))) \rightarrow f(x)$$

and using a *rewrite system S:*

$$f(f(x)) \to f(x)$$

we get the diagram

$$f(f(x)) \underset{S}{\to} f(x) \underset{T}{\leftarrow} f(g(f(x)))$$

Since $f(f(x)) >_{RPO} f(x)$, this can be used to prove the termination of this self embedded rewrite system.

1.1 The ideas behind the concepts

Suppose given T the transforming rewrite system and S a rewrite system whose well-foundedness is known. Basically we define an ordering that satisfies

$$t_1 > t_2 \text{ if } t_1 \downarrow_T \overset{+}{\underset{S}{\to}} t_2 \downarrow_T .$$

If we say "if and only if" in the former statement, this ordering still works but it is not fully invariant. Indeed suppose t_1 and t_2 are two non ground terms that satisfy $t_1 > t_2$. This does not implies that $\sigma(t_1) > \sigma(t_2)$ even if S is fully invariant. Indeed $\sigma(t_1 \downarrow_T)$ is not always the normal form of $\sigma(t_1)$, it could be the case that $\sigma(t_1 \downarrow_T)$ is reducible. So instead of defining $t_1 > t_2$ to be $t_1 \downarrow_T \to_S^+ t_2 \downarrow_T$, we define $t_1 > t_2$ by the existence of u_1 and u_2 such that $t_1 \to_T^* u_1 \to_S^+ u_2 \leftarrow_T^* t_2$. But now the question is to know whether this defines an ordering. Especially whether the relation is transitive. Recall that a relation $\to$ is transitive if and only if $\to . \to \subseteq \to$. In other words one may wonder whether

$$\overset{*}{\underset{T}{\to}} . \overset{+}{\underset{S}{\to}} . \overset{*}{\underset{T}{\leftarrow}} . \overset{*}{\underset{T}{\to}} . \overset{+}{\underset{S}{\to}} . \overset{*}{\underset{T}{\leftarrow}} \subseteq \overset{*}{\underset{T}{\to}} . \overset{+}{\underset{S}{\to}} . \overset{*}{\underset{T}{\leftarrow}}$$

This would be true if $\leftarrow_T^* . \to_T^*$ is the identity. But this is rarely the case. However if T is confluent, we get

$$\overset{*}{\underset{T}{\leftarrow}} . \overset{*}{\underset{T}{\to}} \subseteq \overset{*}{\underset{T}{\to}} . \overset{*}{\underset{T}{\leftarrow}}$$

and we have to prove:

$$\overset{*}{\underset{T}{\to}} . \overset{+}{\underset{S}{\to}} . \overset{*}{\underset{T}{\to}} . \overset{*}{\underset{T}{\leftarrow}} . \overset{+}{\underset{S}{\to}} . \overset{*}{\underset{T}{\leftarrow}} \subseteq \overset{*}{\underset{T}{\to}} . \overset{+}{\underset{S}{\to}} . \overset{*}{\underset{T}{\leftarrow}}$$

This last inclusion holds if one can shift $\leftarrow_T^*$ through $\to_S^+$ to the right. Suppose this succeeds, we get then:

$$\begin{aligned} & \overset{*}{\underset{T}{\to}} . \overset{+}{\underset{S}{\to}} . \overset{*}{\underset{T}{\to}} . \overset{*}{\underset{T}{\leftarrow}} . \overset{+}{\underset{S}{\to}} . \overset{*}{\underset{T}{\leftarrow}} \\ \subseteq & \overset{*}{\underset{T}{\to}} . \overset{+}{\underset{S}{\to}} . \overset{*}{\underset{T}{\to}} . \overset{+}{\underset{S}{\to}} . \overset{*}{\underset{T}{\leftarrow}} . \overset{*}{\underset{T}{\leftarrow}} \\ = & \overset{*}{\underset{T}{\to}} . \overset{+}{\underset{S}{\to}} . \overset{*}{\underset{T}{\to}} . \overset{+}{\underset{S}{\to}} . \overset{*}{\underset{T}{\leftarrow}} \end{aligned}$$

In order to shift $\to_T^*$ through $\to_S^+$ to the left, it is reasonable and natural to consider the relation $\to_T^* . \to_S (\to_S \cup \to_T^*)^* . \leftarrow_T^*$ which starts by applications of T, then followed by one application of S, followed by applications of T possibly followed by applications of S and eventually followed by $\leftarrow_T^*$ which is the converse of $\to_T^*$ (Figure 4).

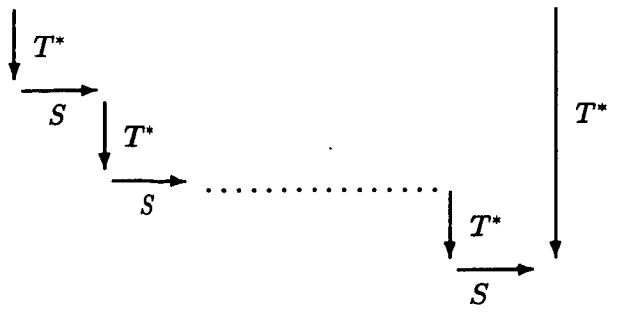

Figure 4: The relation $(S/T)^+.(T^{-1})^*$

This last relation which has several expressions like

$$(\underset{S}{\rightarrow} \cup \underset{T}{\rightarrow})^*. \underset{S}{\rightarrow} . \overset{*}{\underset{T}{\rightarrow}} . \overset{*}{\underset{T}{\leftarrow}}$$

or

$$(\underset{T}{\rightarrow} \cup \underset{S}{\rightarrow})^*. \underset{S}{\rightarrow} (\underset{S}{\rightarrow} \cup \underset{T}{\rightarrow})^*. \overset{*}{\underset{T}{\leftarrow}}$$

or

$$(\overset{*}{\underset{T}{\rightarrow}} . \underset{S}{\rightarrow} . \overset{*}{\underset{T}{\rightarrow}})^+. \overset{*}{\underset{T}{\leftarrow}}$$

or in Bachmair and Dershowitz notations,

$$(S/T)^+ . (T^{-1})^*$$

is a generalization of S by T. Proving its transitivity requires a way to shift $\leftarrow_T^*$ through $\rightarrow_S .(\rightarrow_S \cup \rightarrow_T)^*$ in order to remove subdiagrams of the form

$$\overset{*}{\underset{T}{\leftarrow}} . \underset{S}{\rightarrow} .(\underset{S}{\rightarrow} \cup \underset{T}{\rightarrow})^*.$$

This relies on the local confluence of T

$$\underset{T}{\leftarrow} . \underset{T}{\rightarrow} \subseteq \overset{*}{\underset{T}{\rightarrow}} . \overset{*}{\underset{T}{\leftarrow}}$$

and another property called *local cooperation* of S with T which is the property

$$\underset{T}{\leftarrow} . \underset{S}{\rightarrow} \subseteq (\overset{*}{\underset{T}{\rightarrow}} . \underset{S}{\rightarrow} . \overset{*}{\underset{T}{\rightarrow}})^+ . \overset{*}{\underset{T}{\leftarrow}}$$

Notice that $(\rightarrow_T^* . \rightarrow_S . \rightarrow_T^*)^+ . \leftarrow_T^*$ requires at least one S step and this is a key step in the proof of termination.

1.2 Some basic results

Two relations $\rightarrow_S$ and $\rightarrow_T$ are considered. These relations are general relations not only rewrite relations. This means that when we say $\rightarrow_T$ is confluent or $\rightarrow_S$ terminates, we state properties on abstract relations.

As we have seen the local confluence of $\rightarrow_T$ is the property $\leftarrow_T . \rightarrow_T \subseteq \rightarrow_T^* . \leftarrow_T^*$. We now define the *local cooperation* of $\rightarrow_S$ with $\rightarrow_T$ which is a kind of local confluence.

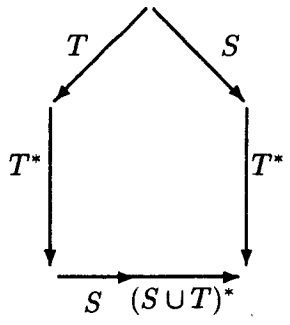

Figure 5: S cooperates locally with T

Definition 1 $\rightarrow_S$ cooperates locally with $\rightarrow_T$ *if and only if*

$$\underset{T}{\leftarrow} \cdot \underset{S}{\rightarrow} \subseteq \underset{S/T}{\overset{+}{\rightarrow}} \cdot \underset{T}{\overset{*}{\leftarrow}}$$

or $T^{-1} \cdot S \subseteq (T^* \cdot S \cdot T^*)^+ \cdot (T^{-1})^*$ *this is equivalent to (Figure 5)*

$$\underset{T}{\leftarrow} \cdot \underset{S}{\rightarrow} \subseteq \underset{T}{\overset{*}{\rightarrow}} \cdot \underset{S}{\rightarrow} (\underset{S}{\rightarrow} \cup \underset{T}{\overset{*}{\rightarrow}})^* \cdot \underset{T}{\overset{*}{\leftarrow}} .$$

The following theorem is a consequence of results in [BD86] and in [BL86].

Theorem 1 *If S locally cooperates with T, $S \cup T$ terminates and T is locally confluent then $(S/(T \cup T^{-1}))^+$ is a well-founded ordering and moreover when S and T satisfy the replacement property and are fully invariant, $(S/(T \cup T^{-1}))^+$ satisfies the replacement property and is fully invariant.*

Since rewrite relations on $\mathcal{T}(F, V)$ satisfy the replacement property and are fully invariant we may state the following corollary:

Corollary 1 *Let $\rightarrow_S$ and $\rightarrow_T$ be two rewrite systems. Suppose $\rightarrow_S$ locally cooperates with $\rightarrow_T$, $\rightarrow_S \cup \rightarrow_T$ is noetherian and $\rightarrow_T$ is confluent then $(S/(T \cup T^{-1}))^+$ is a fully invariant reduction ordering.*

Corollary 2 *With the condition of Theorem 1, a rewrite system that satisfies*

$$l \, (S/(T \cup T^{-1}))^+ \, r$$

for all rules $l \rightarrow r$ terminates.

Usually one proves $l \downarrow_T \rightarrow_S r \downarrow_T$, hence the name "transformation ordering".

2 A criterion for Local Cooperation

In Theorem 1 there are three properties to check, namely local cooperation, local confluence and termination. The two last one can be proved using classical techniques. In this section we use the fact that $\rightarrow_S$ and $\rightarrow_T$ are rewrite relations in order to check local cooperation.

Definition 2 *A* critical pair $\langle p, q\rangle$ *between a rule of* S *and a rule of* T *such that:*

$$p \underset{T}{\leftarrow} \cdot \underset{S}{\rightarrow} q$$

is said to be cooperative *if and only if*

$$p \underset{S/T}{\overset{+}{\rightarrow}} \cdot \underset{T}{\overset{*}{\leftarrow}} q$$

In this section, T is supposed *variable preserving.* If a term t may be rewritten to t' at the position α by a rule $l \rightarrow r$ and is rewritten at the position $\alpha\beta\gamma$, by another rule $g \rightarrow d$, where $\beta \in Pos(l)$ and $l(\beta) = x \in V$, then t' may be rewritten at least once by the rule $g \rightarrow d$ since x occurs at least once in r and thus $t|_{\alpha\beta\gamma}$ occurs at least once in t'. This property for T is important if one does not want to loose track of S rewrites, as required in the local cooperation.

Theorem 2 *If* T *is a variable preserving and left-linear rewrite system. A rewrite system* S *cooperates locally with* T *if and only if all the critical pairs between* S *and* T *are cooperative.*

The proof looks like the proof of the similar theorem on confluent critical pairs.

Proof: The only if part is obvious. For the if part, let us have t such that $t \rightarrow_S t_1$ by a rule $l_1 \rightarrow r_1$ of S applied at the position α_1 and that $t \rightarrow_T t_2$ by a rule $l_2 \rightarrow r_2$ of T at the position α_2.

1. α_1 and α_2 are disjoint positions, we have

   $$t_2 \underset{S}{\rightarrow} \cdot \underset{T}{\leftarrow} t_1$$

   and then $t_2 \rightarrow^+_{S/T} \cdot \leftarrow^*_T t_1$.

2. α_1 (or α_2) is a prefix of α_2 (or α_1) and α_2 (or α_1) corresponds to an internal position i.e., a non variable position in l_1 (or l_2), there exists a critical pair between S and T and we get the result because this critical pair is cooperative.

3. α_1 is a prefix of α_2 and $\alpha_2 = \alpha_1\beta\gamma$ where $\beta \in Pos(l_1)$ and $l_{1|\beta} = x \in V$, the subterms of t_2 corresponding to the other occurrences of x in l_1 can also be rewritten by $l_2 \rightarrow r_2$ at appropriate positions (so doing, we get t'_2). The subterms of t_1 corresponding to the occurrences of x in r_1 can also be rewritten by $l_2 \rightarrow r_2$ (so doing we get t'_1). Then we have $t'_2 \rightarrow_S t'_1$. Thus

   $$t_2 \underset{T}{\overset{*}{\rightarrow}} t'_2 \underset{S}{\rightarrow} t'_1 \underset{T}{\overset{*}{\leftarrow}} t_1$$

   and then $t_2 \rightarrow^+_{S/T} \cdot \leftarrow^*_T t_1$ (Figure 6).

4. α_2 is a prefix of α_1 and $\alpha_1 = \alpha_2\beta\gamma$ where $\beta \in Pos(l_2)$ and $l_{2|\beta} = x \in V$. T is *variable preserving* thus $V(l_2) = V(r_2)$ and we find again x at least once in r_2. Thus it is possible to rewrite t_2 at least once by the rule $l_1 \rightarrow r_1$ of S. If we rewrite all the subterms of t_2 corresponding to the occurrences of x

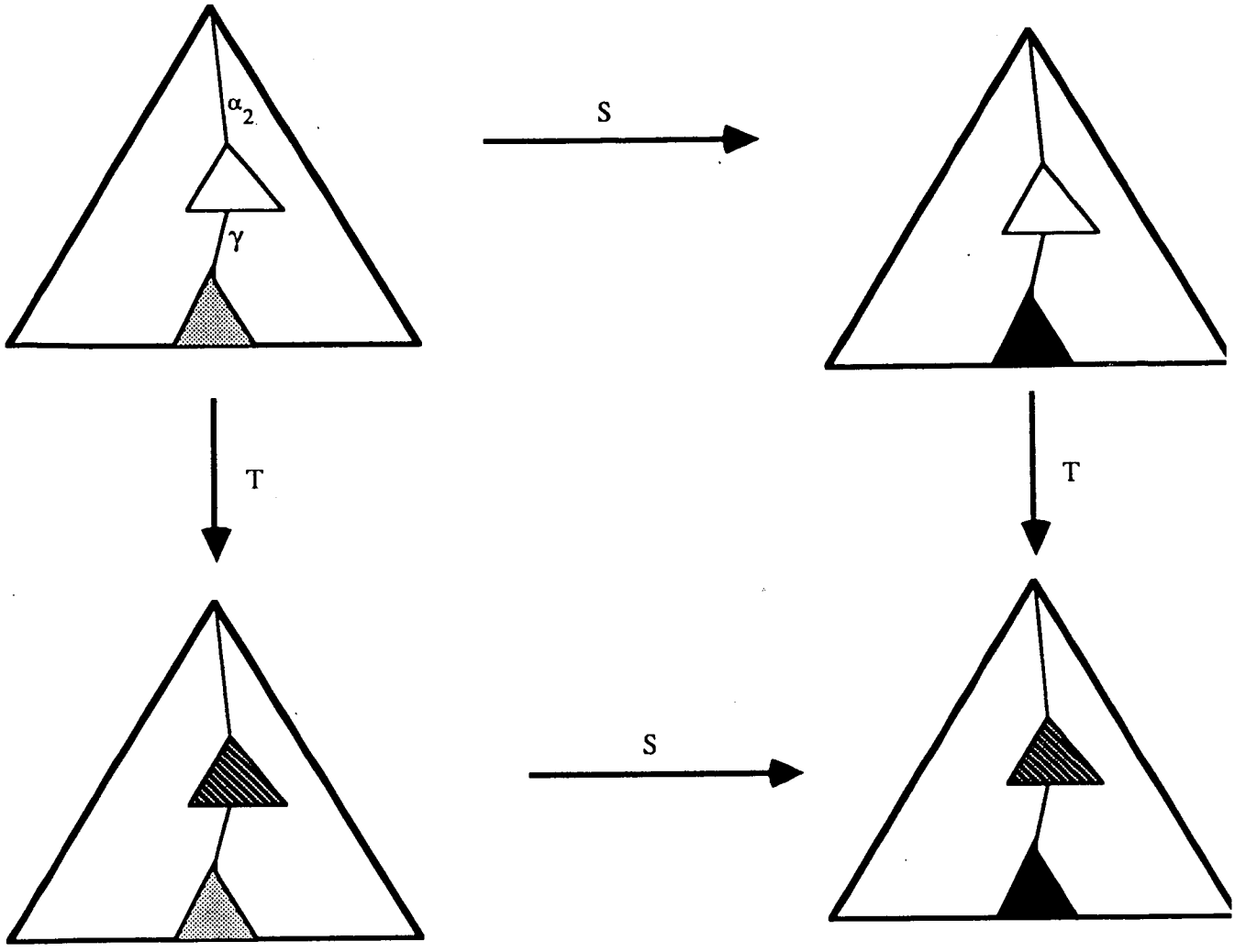

Figure 6: T rewrites "under" S

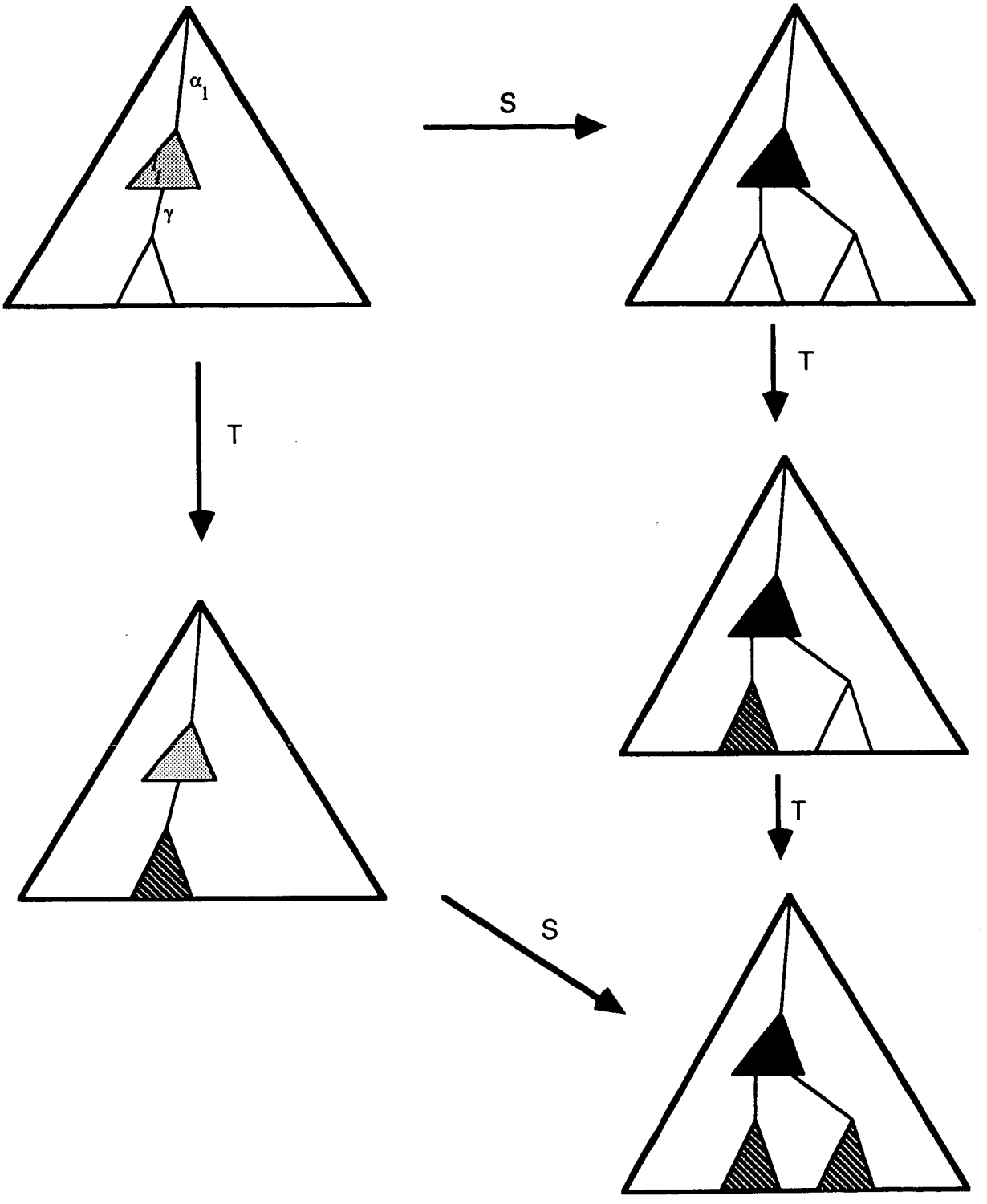

Figure 7: S rewrites "under" T

(perhaps many) by the rule $l_1 \to r_1$, we get t'_2. Thus we have $t_2 \to_S . \to_S^* t'_2$. T is *left-linear* therefore after one replacement of $\sigma(l_1)$ by $\sigma(r_1)$, the rule $l_2 \to r_2$ is still applicable and produces t'_2. Thus we have $t_1 \to_T t'_2$ and get (Figure 7)

$$t_2 \underset{S}{\to} . \overset{*}{\underset{S}{\to}} . \underset{T}{\leftarrow} t_1.$$

□

The previous theorem is the original result in [BL86] and the only necessary here. Geser proves a more general result in [Ges89].

Example 4 Let R be the rewrite system made of the two first rules of $Assoc + Endo$

$$\begin{array}{ll} endomorphism: & f(x) + f(y) \to f(x + y) \\ endo\_assoc: & f(x) + (f(y) + z) \to f(x + y) + z \end{array}$$

We choose T to be:

$$\begin{array}{l} left\_f\_up: f(x) + y \to f(x + y) \\ right\_f\_up: y + f(x) \to f(y + x) \end{array}$$

in order to push up f and put down $+$ and we choose S to be

$$f(f(x)) \to f(x)$$

T is confluent and $S \cup T$ terminates. S and T satisfy the condition of Theorem 2, therefore S cooperates locally with T. Thus we may use $(S/(T \cup T^{-1}))^+$ to prove the termination of R. Let us check that $l\ (S/(T \cup T^{-1}))^+\ r$ for all rules $l \to r$ of R:

Proof: We have to check two inequalities:

- $f(x) + f(y)(S/(T \cup T^{-1}))^+ f(x + y)$ since

$$f(x) + f(y) \underset{T}{\overset{2}{\to}} f(f(x + y)) \underset{S}{\to} f(x + y)$$

- $f(x) + (f(y) + z)(S/(T \cup T^{-1}))^+ f(x + y) + z$ since

$$f(x) + (f(y) + z) \underset{T}{\overset{2}{\to}} f(f(x + y) + z) \underset{S}{\to} f((x + y) + z) \underset{T}{\leftarrow} f(x + y) + z$$

□

3 Strengthen the ordering

The ordering $S/(T \cup T^{-1})$ is not universal. For instance, it was used in the previous example to prove termination of only a subset of system *Assoc + Endo*. In other words, given a system $R = R_1 \cup R_2$, one proves only $R_1 \subseteq S/(T \cup T^{-1})$. However, Bachmair and Dershowitz have proven that if R_1/R_2 and R_2 terminate then $R_1 \cup R_2$ terminates. As we said, $R_1 \subseteq S/(T \cup T^{-1})$, gives the termination of R_1, but it also gives the termination of the stronger relation $R_1/(T \cup T^{-1})$ or $R_1/(T \cup T^{-1})^*$. If we prove that $R_2 \subseteq (T \cup T^{-1})^*$ then R_1/R_2 terminates. If now R_2 terminates then $R_1 \cup R_2$ terminates.

Now let us describe a practical situation. Like with polynomial orderings where the key choice is the interpretation of the operators by polynomial, in the transformation ordering the key choice is the transformation T. Usually we proceed as follows. First, we convince ourselves that a classical method (a recursive path ordering in our case) does not work, usually by identifying resisting rules (for instance, self-embedded). We may take for R_1 this set of resisting rules or a larger one. Second, we try to guess an "interpretation" that can be described by a confluent, variable preserving, left-linear and terminating rewrite system T and that can solve the termination for R_1 and we enlarge T so that

$$R_2 = R - R_1 \subseteq (T \cup T^{-1})^*.$$

Third, we propose a method (a recursive path ordering in our case) for proving the termination of T, R_2 and S. Fourth, we build S such that S cooperates locally with T. Note the paradox that we have to propose a method for proving the termination of a system S that we do not know yet, usually this is not a problem since the recursive path ordering used to prove the termination of T can be extended. Note also that the construction of S can be mechanized (see section 5).

4 More Examples

In this section, we propose many examples to illustrate the power of this method. They all have been experimented with the procedure proposed in the next section. We start first with the three examples presented in the introduction and set the termination problem in terms of T and S.

Example 5 In Example 4, we didn't consider rule *assoc*; to handle it we extend $T = \{left\_f\_up, right\_f\_up\}$ with the left-linear and variable preserving rule *assoc* itself. $S = \{f(f(x)) \rightarrow f(x)\}$ still cooperates with $T' = \{left\_f\_up, right\_f\_up, assoc\}$ and $T' \cup S$ terminates, therefore the system *Assoc+Endo* terminates.

Example 6 In the system $C \cup \{app\}$ of Example 2, *app* is the rule that poses problem, let T be

$$\begin{array}{ll} r_1: & Append \circ \langle x, y\rangle \rightarrow x@y \\ dist\_pair: & \langle x, y\rangle \circ z \rightarrow \langle x \circ z, y \circ z\rangle \end{array}$$

and S be

$$\begin{array}{ll} app': & (x@y)@z \rightarrow x@(y@z) \\ idl: & x \circ I \rightarrow x \\ idr: & I \circ x \rightarrow x \\ p_1: & Fst \circ \langle x, y\rangle \rightarrow x \\ p_2: & Snd \circ \langle x, y\rangle \rightarrow y \\ assoc: & (x \circ y) \circ z \rightarrow x \circ (y \circ z) \\ r_2: & (x@y) \circ z \rightarrow (x \circ z)@(y \circ z) \end{array}$$

T is confluent, variable preserving and left-linear. $S \cup T$ terminates. By Theorem 2, we prove that S cooperates with T by checking whether the critical pairs between S and T are cooperative. For instance, the critical pair

$$(x@y) \circ z = Append \circ (\langle x, y\rangle \circ z)$$

between *assoc* and r_1 is cooperative (Figure 8).

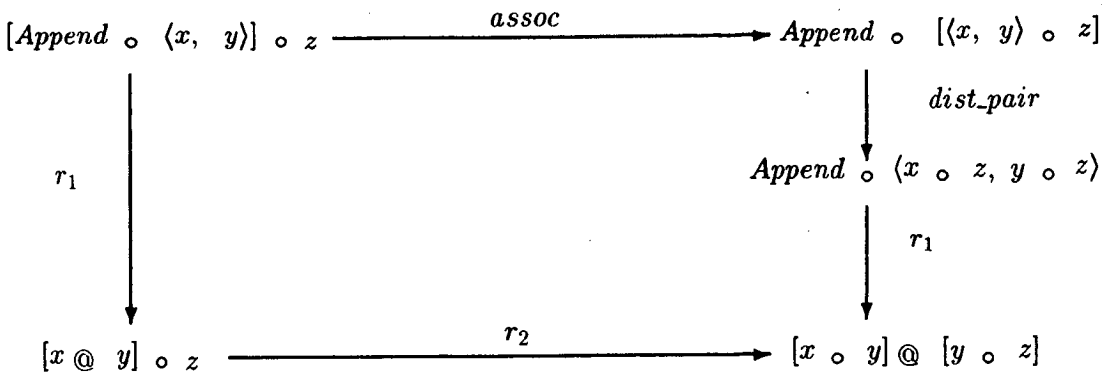

Figure 8: diagram for *assoc*

$$R = \{assoc, idl, idr, p_1, p_2, app\} \cup \{dist\_pair\}$$

terminates.

Proof: The proof requires three steps.

- app, $l \downarrow_T \rightarrow_{app'} r \downarrow_T$ and $app' \in S$.
- $assoc$, idl, idr, p_1 and p_2 are rules of S and $l = l \downarrow_T \rightarrow_S r \downarrow_T = r$.
- $dist\_pair$ is a rule of T.

□

Comment: The rewrite systems S and T are not unique. We could also choose another solution for T,

$$\begin{array}{ll} r_1: & Append \circ \langle x, y\rangle \rightarrow x@y \\ assoc: & (x \circ y) \circ z \rightarrow x \circ (y \circ z) \\ dist\_pair: & \langle x, y\rangle \circ z \rightarrow \langle x \circ z, y \circ z\rangle \\ r_2: & (x@y) \circ z \rightarrow (x \circ z)@(y \circ z) \end{array}$$

and for S,

$$\begin{array}{ll} app': & (x@y)@z \rightarrow x@(y@z) \\ idl: & x \circ I \rightarrow x \\ idl: & I \circ x \rightarrow x \\ p_1: & Fst \circ \langle x, y\rangle \rightarrow x \\ p_2: & Snd \circ \langle x, y\rangle \rightarrow y \end{array}$$

which have also the properties required by Theorem 2.

Example 7 Let us consider now the rewrite system

$$f(f(x)) \rightarrow f(g(f(x)))$$

Since g is just a separator of two occurrences of f, it seems natural to remove it and to transform these two occurrences of f into one f, thus we choose T to be:

$$f(g(f(x))) \rightarrow f(x)$$

and S to be

$$f(f(x)) \rightarrow f(x)$$

T is confluent and terminates, $S \cup T$ terminates. Let us look at the critical pairs between T and S. We have first

$$f(f(x)) \underset{T}{\leftarrow} f(f(g(f(x)))) \underset{S}{\rightarrow} f(g(f(x)))$$

which is cooperative because $f(f(x)) \rightarrow_S f(x) \leftarrow_T f(g(f(x)))$. Second we have

$$f(f(x)) \underset{T}{\leftarrow} f(g(f(f(x)))) \underset{S}{\rightarrow} f(g(f(x)))$$

which has basically the same critical pair. This shows that S cooperates locally with T. T is variable preserving and left-linear. R terminates because

$$f(f(x)) \underset{S}{\rightarrow} f(x) \underset{T}{\leftarrow} f(g(f(x)))$$

We now give other examples. The following one comes from [Der87].

Example 8 The termination of this simple system

$$\begin{aligned} f(a) &\rightarrow f(b) \\ g(b) &\rightarrow g(a) \end{aligned}$$

resists to usual methods because one has to choose between $a > b$ or $b > a$, thus no recursive path ordering based on a precedence works. However we choose T to be:

$$f(b) \rightarrow g(b)$$

and S to be

$$\begin{aligned} f(a) &\rightarrow g(b) \\ g(b) &\rightarrow g(a) \end{aligned}$$

Proof: The relation $S \cup T$ terminates. The proof uses a recursive path ordering, taking $f > g$ and $f > b > a$. T is obviously locally confluent and S cooperates locally with T because there are no critical pairs between S and T. We now prove that R terminates by:

- $f(a) \downarrow_T = f(a)$, $f(b) \downarrow_T = g(b)$ and $f(a) \rightarrow_S g(b)$, thus $f(a) \rightarrow_{S/(T \cup T^{-1})+} f(b)$
- $g(b) \downarrow_T = g(b)$, $g(a) \downarrow_T = g(a)$ and $g(b) \rightarrow_S g(a)$, thus $g(b) \rightarrow_{S/(T \cup T^{-1})+} g(a)$

□

Example 9 The termination of:

$$\begin{aligned} g(x,y) &\rightarrow h(x,y) \\ h(f(x),y) &\rightarrow f(g(x,y)) \end{aligned}$$

cannot be proved by a recursive path ordering, the first rule requires to take the precedence $g > h$ and the second requires $h > f$ and $h >$ (or $\sim$) g. However we choose T to be:

$$g(x,y) \rightarrow h(x,y)$$

and S to be

$$h(f(x),y) \rightarrow f(h(x,y))$$

Proof: There are no critical pairs between S and T. $S \cup T$ terminates taking $g > h > f$. T is confluent and by Theorem 2, S cooperates locally with T. We prove that R terminates:

- $g(x,y) \rightarrow_T h(x,y)$
- $h(f(x),y) \downarrow_T = h(f(x),y)$,
  $f(g(x,y)) \downarrow_T = f(h(x,y))$
  and $h(f(x),y) \rightarrow_S f(h(x,y))$.

□

Example 10 Another self embedded rewrite system is:

$$f(g(x)) \rightarrow f(h(g(x)))$$

Taking a new symbol q, we choose T to be:

$$h(g(x)) \rightarrow q(x)$$

and S to be

$$f(g(x)) \rightarrow f(q(x))$$

Proof: The relation $S \cup T$ are terminating and there are no critical pairs between S and T. T is confluent and S locally cooperates with T. We prove that R terminates:

$f(g(x)) \downarrow_T = f(g(x))$, $f(h(g(x))) \downarrow_T = f(q(x))$ and $f(g(x)) \rightarrow_S f(q(x))$. □

Example 11 This example is quite natural and comes from a specification of binary numbers. It was proposed to us by Alfons Geser as a challenge and required to use our program in order to examine all the details of the proof of termination, namely the cooperation of S with T,

$$Div2(\emptyset) \rightarrow \emptyset \tag{1}$$
$$Div2(S(\emptyset)) \rightarrow \emptyset \tag{2}$$
$$Div2(S(S(x)) \rightarrow S(Div2(x)) \tag{3}$$
$$LastBit(\emptyset) \rightarrow 0 \tag{4}$$
$$LastBit(S(\emptyset)) \rightarrow 1 \tag{5}$$
$$LastBit(S(S(x))) \rightarrow LastBit(x) \tag{6}$$
$$Conv(\emptyset) \rightarrow \epsilon \,\&\, 0 \tag{7}$$
$$Conv(S(x)) \rightarrow Conv(Div2(S(x))) \,\&\, LastBit(S(x)) \tag{8}$$

It is a definition of the mapping of the $S^n(\emptyset)$ representation of the naturals to their binary representation. The main problem of termination in this system is due to the rule:

$$Conv(S(x)) \rightarrow Conv(Div2(S(x))) \,\&\, LastBit(S(x))$$

where the left-hand side

$$Conv(S(x))$$

is embedded in the right-hand side

$$Conv(Div2(S(x))) \,\&\, LastBit(S(x)).$$

The idea to get rid of this embedding is to introduce a new operation, let us call it $Prefix$ with the following property

$$Conv(Div2(x)) \dot{\rightarrow} Prefix(x)$$

that we put in to the transforming rewrite system T. It is also necessary to add to T rule (7) of R

$$Conv(\emptyset) \rightarrow \epsilon \,\&\, 0$$

The termination of the following rewrite system that we call S

$$\begin{aligned}
Div2(\emptyset) &\rightarrow \emptyset && (9)\\
Div2(S(\emptyset)) &\rightarrow \emptyset && (10)\\
Div2(S(S(x))) &\rightarrow S(Div2(x)) && (11)\\
LastBit(\emptyset) &\rightarrow 0 && (12)\\
LastBit(S(\emptyset)) &\rightarrow 1 && (13)\\
LastBit(S(S(x))) &\rightarrow LastBit(x) && (14)\\
Conv(S(x)) &\rightarrow Prefix(S(x)) \,\&\, LastBit(S(x)) && (15)\\
Prefix(\emptyset) &\rightarrow \epsilon \,\&\, 0 && (16)\\
Prefix(S(\emptyset)) &\rightarrow \epsilon \,\&\, 0 && (17)\\
Prefix(S(S(x))) &\rightarrow Conv(S(Div2(x))) && (18)
\end{aligned}$$

can be easily checked by a recursive path ordering, based on the precedence

$$\begin{aligned}
Conv > LastBit > 0\\
LastBit > 1\\
S > Div2\\
S > Conv > Prefix > +\\
Zero > 0\\
Zero > 0
\end{aligned}$$

One sees that each rule in R can be transformed by T into a rule in S, except rule (7); however this rule belongs to T that terminates. Therefore the termination of S implies the termination of R.

5 Mechanizing the construction of the ordering

In Section 3, we described a method for building a transformation ordering and therefore a proof of termination. This method requires three data: a set of rewrite rules R for which we want to give a proof of termination, a set of rewrite rules T, the "transformation" and a proof of termination for T and later S. The ordering for termination of $S \cup T$ will be written

$\succ$ and given by a precedence and a status for the operators since we use the recursive path ordering in our current implementation. One can imagine another method like a polynomial interpretation [BL87b]. The procedure works as a completion and will be described by a set of transition rules as for completion [Bac88,BDH86,Les89a]. The set of rewrite rules T, the precedence and the status are not modified by the procedure, therefore they do not belong to the data structure on which the transition rules work.

The data structure of the procedure is a triple $\langle R, P, S\rangle$. Each component of the triple is a set of ordered pairs of terms. R is a set of rules candidates to be ordered by the transformation ordering. P is a set of cooperative critical pairs S is set of pairs that will eventually form the set previously called S and that will be used with T to build the transformation ordering. During the run of the procedure, R decreases and S increases. An ordered pair is written $l \Rightarrow r$ and a transition rule is written $\langle R, P, S\rangle \vdash \langle R', P', S'\rangle$.

Elimination of orderable pairs, ER

Rules in R that are included in $(T \cup T_{-1})^*$ and orderable by $\succ$ belongs to what we called R_2 and proved to terminate directly by $\succ$ and not by S and T, therefore they can be removed at the beginning of the completion process. Note that, since T is confluent, $l\ (T \cup T^{-1})^*\ r$ is the same as $l \rightarrow_T^* . \leftarrow_T^* r$.

$$\langle R \cup \{l \Rightarrow r\}, P, S\rangle \vdash \langle R, P, S\rangle \quad \text{if } l \underset{T}{\overset{*}{\rightarrow}} . \underset{T}{\overset{*}{\leftarrow}} r \wedge (l \succ r)$$

Moving of a pair, M

One moves from R to S pairs with different T-normal forms for the sides and not orderable by the given recursive path ordering $\succ$.

$$\langle R \cup \{l \Rightarrow r\}, P, S\rangle \vdash \langle R, P \cup \{l \Rightarrow r\}, S\rangle \quad \text{if } \neg(l \underset{T}{\overset{*}{\rightarrow}} . \underset{T}{\overset{*}{\leftarrow}} r) \vee \neg(l \succ r)$$

Elimination of cooperative pairs, EP

As we said, P gets critical pairs obtained by superpositions between a rule in T and a rule S, if such a critical pair is cooperative it can be removed.

$$\langle R, P \cup \{l \Rightarrow r\}, S\rangle \vdash \langle R, P, S\rangle \quad \text{if } l \underset{S/T}{\overset{+}{\rightarrow}} . \underset{T}{\overset{*}{\leftarrow}} r$$

Normalization, N

Here are transition rules for simplifying rules in P

$$\begin{array}{ll} \langle R, P \cup \{l \Rightarrow r\}, S\rangle \vdash \langle R, P \cup \{u \Rightarrow r\}, S\rangle & \text{if } u \equiv l \downarrow_T \\ \langle R, P \cup \{l \Rightarrow r\}, S\rangle \vdash \langle R, P \cup \{l \Rightarrow v\}, S\rangle & \text{if } v \equiv r \downarrow_T \end{array}$$

Orientation and critical pair computation, OC

A pair goes to S only if it can be ordered by $\succ$. If $\mathcal{CP}(s,T)$ is the set of all critical pairs between s and all the terms $t \in T$. Orientation is a good time to compute critical pairs, therefore, when $l \Rightarrow r$ is oriented, $\mathcal{CP}(s,T)$ is added to P.

$$\langle R, P \cup \{l \Rightarrow r\}, S\rangle \vdash \langle R, P \cup \mathcal{CP}(l,T), S \cup \{l \Rightarrow r\}\rangle \quad \text{if } l \succ r$$

To use these rules efficiently requires a control that gives some order in which the rules are invoked. In our implementation, we choose to apply first ER and M on the rules in R, then to loop on EP, EN, N and OC. We can describe the control by

$$\mathsf{ER}^!; \mathsf{M}^!; (\mathsf{EP}; \mathsf{N}; \mathsf{OC})^!$$

where ";" is the sequential composition and $\mathsf{A}^!$ means that the simple or compound rule A is applied while it is applicable. It is easy to see that the procedure stops with two kinds of data structures: either $\langle \emptyset, \emptyset, S \rangle$ which is a success, or with P not empty which means that there is a pair in P that cannot be oriented and which is a failure. It should be noted that the procedure can diverge since computed pairs can be added for ever in P, but there is no real problem of fairness. A procedure that diverges without failure still provides a proof of termination. A CAML implementation of this procedure based on a description of this procedure by transition rules was done [Gal88] and used to check our examples.

6 Conclusion

These techniques allowed us to mechanically prove the termination of many rewrite systems for which no mechanical prove existed yet. Now it should be interesting to integrate this algorithm in a completion to see how it reacts and it should also be interesting to see how it can be extended to handle termination of equational rewrite systems especially of associative and commutative rewrite systems.

We would like here to thank Bruno Galabertier, without his implementation we would not be able to make experiments and to get confidence in our theoretical tool.

References

[Bac88] L. Bachmair. *Proof by Consistency in Equational Theories.* Technical Report, SUNY at Stony Brook USA, June 1988.

[BD86] L. Bachmair and N. Dershowitz. Commutation, transformation and termination. In J. Siekmann, editor, *Proceedings 8th Conf. on Automated Deduction*, pages 5–20, Springer-Verlag, Lecture Notes in Computer Science, 1986.

[BDH86] L. Bachmair, N. Dershowitz, and J. Hsiang. Orderings for equational proofs. In *Proceedings Symp. Logic in Computer Science*, pages 346–357, Boston (Massachusetts USA), 1986.

[Bel85] F. Bellegarde. Utilisation des systèmes de réécriture d'expressions fonctionnelles comme outils de transformation de programmes itératifs. Thèse de doctorat d'Etat, Université de Nancy I, 1985.

[Bel86] F. Bellegarde. Rewriting systems on FP expressions to reduce the number of sequences yielded. *Science of Computer Programming*, 6:11–34, 1986.

[BL86] F. Bellegarde and P. Lescanne. *Termination Proofs Based On Transformation Techniques.* Internal Report 86-R-034, Centre de Recherche en Informatique de Nancy, 1986.

[BL87a] F. Bellegarde and P. Lescanne. Transformation orderings. In *12th Coll. on Trees in Algebra and Programming, TAPSOFT*, pages 69–80, Springer-Verlag, Lecture Notes in Computer Science, 1987.

[BL87b] A. BenCherifa and P. Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *Science of Computer Programming*, 9(2):137–160, October 1987.

[Cur86] P.L. Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming.* Pitman, 1986.

[Dau89] M. Dauchet. Simulation of turing machines by a left-linear rewrite rule. In N. Dershowitz, editor, *Proceedings of the 3rd Conference on Rewriting Techniques and Applications, Chapel Hill, North Carolina, USA*, pages 109–120, Springer-Verlag, April 1989. Lecture Notes in Computer Science, volume 355.

[Der82] N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.

[Der87] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1 & 2):69–116, 1987.

[DJ89] N. Dershowitz and J. P. Jouannaud. *Rewriting systems. Handbook of Computer Science*, North Holland, 1989. To be published.

[Gal88] B. Galabertier. *Implémentation de l'ordre de terminaison par transformation.* Technical Report, CRIN, Septembre 1988.

[Ges89] A. Geser. Termination relative. Internal Report, Universität Passau, West Germany, November 1989.

[HL78] G. Huet and D-S. Lankford. *On the Uniform Halting Problem for Term Rewriting Systems.* Technical Report 283, Laboria, France, 1978.

[HL86] T. Hardin and A. Laville. Proof of termination of the rewriting system SUBST on CCL. *Theoretical Computer Science*, 46:305–312, 1986.

[Hue80] G. Huet. Confluent reductions : abstract properties and applications to term rewriting systems. *Journal of the Association for Computing Machinery*, 27(4):797–821, October 1980. Preliminary version in 18th Symposium on Foundations Of Computer Science, IEEE, 1977.

[JLR82] J. P. Jouannaud, P. Lescanne, and F. Reinig. Recursive decomposition ordering. In Bjørner D., editor, *Formal Description of Programming Concepts 2*, pages 331–348, North Holland, Garmisch-Partenkirchen, RFA, 1982.

[KB70] D.E. Knuth and P.B. Bendix. *Simple Word Problems in Universal Algebras*, pages 263–297. J. Leech, Oxford, U.K., pergamon press edition, 1970. Computational Problems in Abstract Algebra.

[KL80] S. Kamin and J-J. Lévy. Two generalizations of the recursive path ordering. 1980. Unpublished manuscript.

[Lan79] D.S. Lankford. *On Proving Term Rewriting Systems are Noetherian.* Technical Report, Louisiana Tech. University, Mathematics Dept., Ruston LA, 1979.

[Les89a] P. Lescanne. Completion procedures as transition rules + control. In M. Diaz and F. Orejas, editors, *TAPSOFT'89*, pages 28–41, Springer-Verlag, Lecture Notes in Computer Science, 1989. Vol. 351.

[Les89b] P. Lescanne. On the recursive decomposition ordering with lexicographical status and other related orderings. *Journal of Automated Reasoning*, xx(xx):xx–xx, 1989. To be published.

[Pla78] D. Plaisted. *A recursively defined ordering for proving termination of term rewriting systems.* Technical Report R-78-943, U. of Illinois, Dept of Computer Science, 1978.

[Rus87] M. Rusinowitch. Path of subterms ordering and recursive decomposition ordering revisited. *J. of Symbolic Computation*, 3(1 & 2):117–132, 1987.

Imprimé en France
par
. l'Institut National de Recherche en Informatique et en Automatique .