Termination of rewriting systems by polynomial interpretations and its implementation

Ahlem Ben Cherifa, Pierre Lescanne

INRIA

UNITÉ DE RECHERCHE
INRIA-LORRAINE

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

Rapports de Recherche

N° 677

TERMINATION OF REWRITING SYSTEMS BY POLYNOMIAL INTERPRETATIONS AND ITS IMPLEMENTATION

Ahlem Ben CHERIFA
Pierre LESCANNE

JUIN 1987

TERMINATION OF REWRITING SYSTEMS BY POLYNOMIAL INTERPRETATIONS AND ITS IMPLEMENTATION

TERMINAISON DES SYSTEMES DE REECRITURE PAR LES INTERPRETATIONS POLYNOMIALES ET SON IMPLANTATION

PAR

AHLEM BEN CHERIFA

ET

PIERRE LESCANNE

Résumé : Cet article décrit une implantation concrète, dans le laboratoire de réécriture REVE, d'une procédure élémentaire permettant de prouver des inégalités entre des polynômes sur les entiers et qui est utilisée pour la preuve de la terminaison des Systèmes de Réécriture, essentiellement dans le cas de la théorie associative-commutative pour laquelle nous présentons une caractérisation des interprétations polynômiales des opérateurs associatifs-commutatifs.

Abstract : This paper describes the actual implementation in the rewrite rule laboratory REVE, of an elementary procedure that checks inequalities between polynomials and is used for proving termination of rewriting systems, especially in the more difficult case of associative-commutative rewriting systems, for which a complete characterization is given.

Termination of Rewriting Systems by Polynomial Interpretations and its Implementation[1].

Ahlem Ben Cherifa

Pierre Lescanne

Centre de Recherche en Informatique de Nancy

Campus Scientifique BP 239

54506 Vandoeuvre, FRANCE

ABSTRACT

This paper describes the actual implementation in the rewrite rule laboratory REVE, of an elementary procedure that checks inequalities between polynomials and is used for proving termination of rewriting systems, especially in the more difficult case of associative-commutative rewriting systems, for which a complete characterization is given.

---

[1] This work was supported by the Greco de Programmation.

1. The origin of the problem

Termination is central in programming and in particular in term rewriting systems, the latter being both a theoretical and a practical basis for functional and logic languages. Indeed the problem is not only a key for ensuring that a program and its procedures eventually produce the expected result, it is also important in concurrent programming where liveness results rely on termination of the components. Term rewriting systems are also used for proving equational theorems and are a basic tool for checking specifications of abstract data types. Again, the termination problem is crucial in the implementation of the Knuth-Bendix algorithm, which tests the local confluence and needs the termination to be able to infer the total confluence. Termination is also necessary to direct equations properly. Until now, methods based on recursive path ordering were satisfactory [Dershowitz-82,Jouannaud,etal.-82], but when we recently ran experiments on transformation of **FP** programs [Bellegarde 84], we were faced with a problem that the recursive path ordering could not handle. The problem, motivated by a simple example of code optimization, is just Associativity + Endomorphism.

$$(x_1 * x_2) * x_3 = x_1 * (x_2 * x_3)$$
$$f(x_1 * x_2) = f(x_1) * f(x_2)$$

The variables are functions, * is the composition and **f** is a mapcar-like operator. In order to optimize the program, the user wants to decrease the number of uses of **f** and to orient the equation

$$f(x_1 * x_2) = f(x_1) * f(x_2)$$

into the rule

$$f(x_1) * f(x_2) \rightarrow f(x_1 * x_2)$$

what the recursive path ordering accepts by setting $* > f$. It orients the associativity equation in

$$(x_1 * x_2) * x_3 \rightarrow x_1 * (x_2 * x_3)$$

by setting the status of $*$ to be left-to-right, and then the Knuth-Bendix procedure diverges generating the rules:

$$f^n(x_1 * x_2) * x_3 \rightarrow f^n(x_1) * (f^n(x_2) * x_3)$$

This is not what the user expects, since he or she rather wants to get the terminating system

$$f(x_1) * f(x_2) \rightarrow f(x_1 * x_2)$$
$$(x_1 * x_2) * x_3 \rightarrow x_1 * (x_2 * x_3)$$
$$f(x_1) * (f(x_2) * x_3) \rightarrow f(x_1 * x_2) * x_3$$

where the third rule also decreases the number of occurrences of f. Fortunately, thanks to Lankford [Lankford 79] this system can be easily proved terminating by using the polynomial interpretation $[f](X_1) = 2X_1$ and $[*](X_1,X_2) = X_1X_2 + X_1$, and this motivates our attempt for implementing a nice and efficient method for mechanically proving termination based on polynomial interpretations. Our purpose is not to extend his method in any way, but strictly implement what is presented by Huet and Oppen in their survey[1] [Huet-Open-80]. We also characterize the polynomial interpretations for associative-commutative operators, restricting drastically the

---

[1] The authors say: "the proof of (inequality of the form $[i(x*y)](X,Y) > [i(y)*i(x)](X,Y)$ (see example 1)) is not straightforward, since it involves showing for instance $(\forall\ x,y \in \mathbf{N})$" $x^2(1+2y)^2 > y^2(1+2x^2)$" ([Huet & Oppen 80] p. 367).

space of interpretations for such operators. We get, that way, the unique safe method for associative commutative rewriting which is implemented. Finally we extend the method to Cartesian product providing a technique for proving termination of associative commutative systems, such as a simple specification of the Natural numbers. This example is especially interesting since it is important and no other method for proving its termination works [Bachmair-Plaisted-85,Gnaedig-Lescanne-86].

This paper rather emphasizes the actual implementation in REVE, and the examples that were mechanically proved by REVE. It describes procedures for checking polynomial inequalities, which are both efficient and general. It was said such a method was already proposed by Lankford, but we do not know if it was actually implemented and we did not have access to published matter on the subject. In the absence of such information we have made this work independently. By the way, the reader will find no new result on theoretical aspects of the polynomial interpretation method, but those dealing with actual implementation. We feel indeed it is important to propose algorithms, and people who have used our software REVE are usually grateful to notice the system performs all tedious computations required by the polynomial interpretation method at their place.

2. Interpretation by Functions over the Naturals and Termination

Let $T(F, \{x_1,\ldots,x_m\})$ be the set of terms on $\{x_1,\ldots,x_m\}$ and $\mathbf{N}^m \to \mathbf{N}$ the set of m-ary functions on natural numbers. Suppose that for each k-ary function $f \in F_k \subseteq F$, we define an interpretation of f as a polynomial with k variables. The interpretation of f will be written $\lambda X_1 \ldots X_k [f](X_1,\ldots,X_k)$ and often we simply write $[f](X_1,\ldots,X_k)$. We use the systematic convention

of associating variables $X_1,\ldots, X_k$, Y, Z, U for the polynomials with variables $x_1,\ldots, x_k$, y, z, u for the terms. This interpretation allows us to define on $\mathbf{N}^{\blacksquare} \rightarrow \mathbf{N}$ an F-algebra $\mathbf{N}_{\blacksquare}$, in the following way: if $p_1,\ldots,p_k \in \mathbf{N}^{\blacksquare} \rightarrow \mathbf{N}$, then

$$f_{N_m}(p_1,\ldots,p_k)(X_1,\ldots,X_m) = [f](p_1(X_1,\ldots,X_m),\ldots,p_k(X_1,\ldots,X_m))$$

If we define $[x_i](X_1,\ldots,X_m) = X_i$, then there exists a unique extension to a morphism from $T(F,\{x_1,\ldots,x_m\})$ to the F-algebra $\mathbf{N}_{\blacksquare}$ imposed on $\mathbf{N}^{\blacksquare} \rightarrow \mathbf{N}$ that we will also write [.].

Example 1:

Suppose $F_0=\{e\}$, $F_1=\{i\}$ and $F_2=\{*\}$, and define $[e]=2$, $[i](X_1)=X_1^2$,

$$[*](X_1,X_2)= 2X_1X_2 + X_1.$$

Then

$$[(x_1*i(x_2))*x_2](X_1,X_2)= 4X_1X_2^3 + 2X_1X_2^2 + 2X_1X_2 + X_1$$

There exists on $\mathbf{N}^{\blacksquare} \rightarrow \mathbf{N}$ a natural partial strict ordering which is defined as $h < k$ if and only if $(\forall a_1 \in \mathbf{N})\ldots(\forall a_m \in \mathbf{N})$ $h(a_1,\ldots,a_m) < k(a_1,\ldots,a_m)$. This ordering is obviously well-founded, otherwise a sequence $h_1 > \ldots > h_n > \ldots$ would exist which by instantiation would produce an infinite sequence $h_1(a_1,\ldots,a_m) > \ldots > h_n(a_1,\ldots,a_m) >\ldots$

On $T(F, \{x_1,\ldots,x_m\})$, we define an ordering $<_{[.]}$ by $s <_{[.]} t$ if and only if $[s] < [t]$. This ordering is well-founded by definition. It is also stable by instantiation, which means that for all substitution σ, $\sigma(s) <_{[.]} \sigma(t)$. Indeed since [.] is a morphism, $[\sigma(s)](X_1,\ldots,X_m) < [\sigma(t)](X_1,\ldots,X_m)$ is equivalent to the following inequality between polynomials $[s]([\sigma(x_1)],\ldots,[\sigma(x_m)]) < [t]([\sigma(x_1)],\ldots,[\sigma(x_m)])$ and this inequality holds if the inequality $[s] < [t]$ holds. It will be said to be compatible

if $s <_{[.]} t$ implies $f(\ldots,s,\ldots) <_{[.]} f(\ldots,t,\ldots)$. Let us recall now a main termination criterion:

Proposition 1: [Manna & Ness 70]

A term rewriting system R terminates on a set $T(F,\{x_1,\ldots,x_m\})$ of terms if there exists a well-founded and compatible ordering $>'$ such that, for all rules $g \rightarrow d$ in R and for all substitutions σ , $\sigma(g) > \sigma(d)$

So, any interpretation associated with a compatible ordering could be used to prove termination.

3. Polynomial Interpretation for proving Termination

Given a term rewriting system $\{g_1 \rightarrow d_1,\ldots,g_n \rightarrow d_n\}$, the problem consists of first guessing a compatible interpretation [.] and then proving that $g_1 >_{[.]} d_1,\ldots,g_n >_{[.]} d_n$. In this paper we give no good solution to the first problem since currently nobody, at our knowledge, has good heuristics. Instead, we make suggestions in Section 6 and we expect that computer experiments will lead to a progress in this direction. So, in this paper, we will essentially focus on proving the inequalities between the interpretations of terms. Then, polynomial interpretations can be used as that is shown in [Huet-Open-80], although it is known that there are term rewriting systems that cannot be proven to terminate using this method. However, we feel that our method could be extended to other classical recursive functions like the exponentials.

3.1. An overview of the problem

Notice first there is no algorithm to decide inequalities between n polynomials over **N** [Lankford 79]. Otherwise this algorithm could be used

to solve the tenth Hilbert Problem. Indeed to decide inequalities between polynomials over **N** is equivalent to decide inequalities between polynomials over **Z** (an inequality between polynomials over **Z** is transformed into several inequalities between polynomials over **N**, according to the positiveness or the negativeness of the variables). So, if we could decide n arbitrary inequalities, we could in particular decide whether n polynomials are positive. This in turn could be used for deciding whether n polynomial equations $P_1=0 \wedge \ldots \wedge P_n=0$ have solutions, since it is equivalent to decide $P_1^2 + \ldots + P_n^2 = 0$ [Davis 73].

However we know after Tarski [Tarski 51,Cohen 69] that if $Q_1,R_1,\ldots,Q_n,R_n$ are considered as polynomials over **R**, there exists an algorithm to decide first-order formulas like $Q_1 > R_1 \wedge \ldots \wedge Q_n > R_n$. The most elaborate version of such an algorithm was proposed by Collins [Collins 75] and actually used to prove termination of simple term rewriting systems. Anyway this algorithm is a very large piece of code, it has an exponential complexity and is not really efficient (at least for our purpose, where we want to be able to handle thirty, up to one hundred comparisons in a reasonable time). Thus we have chosen a much simpler view of the problem.

1. We do not want to decide inequalities and leave open the problem of guessing an adequate interpretation **[.]**. Our aim is a procedure that simply checks properties that insure the wanted inequalities between polynomials and thereby directs equations into rules. Experience has shown that proving these inequalities is rather tedious, even when one has a good intuition on how to select the polynomials and computer experiments gave us surprises, (see Example 2).

2. We want to base these computations on really elementary and basic principles such that a simple and efficient implementation can be easily devised.

Our first idea will be to restrict the domain of polynomials to $\mathbf{N}-\{0,1\}$, in other words to consider the set $(\mathbf{N}-\{0,1\})^{\blacksquare} \rightarrow \mathbf{N}$. Now the idea behind this becomes easy, and generalizes the fact that $XY > Y$ if $X > 1$. To guarantee the stability by instantiation, we have also to be sure that the values $[\sigma(x_i)](X_1,\ldots,X_n)$ are in $\mathbf{N}-\{0,1\}$. Thus we have to check that the interpretation of each term is a function in $(\mathbf{N}-\{0,1\})^{\blacksquare} \rightarrow \mathbf{N}-\{0,1\}$. This will be easily satisfied, if each time we have $X_i > 1$, for all i in [1..m], then $[f](X_1,\ldots,X_m) > 1$, and this will be true if [f] satisfies the inequalities $[f](a_1,\ldots,a_m) \geq a_i$ on $\mathbf{N}-\{0,1\}$. This last monotonicity condition is quite similar to the subterm property[1] of the simplification orderings and both conditions can obviously be checked with our algorithm. Usually it is enough to ensure that the coefficients of the interpretations are natural numbers. With the degree of each variable greater than 1, this implies the compatibility[2].

3.2. An example

Before explaining our method let us look at the termination of Associ-

---

[1] An ordering > on a set of terms T possesses the subterm property if,

$$f(\ldots t \ldots) > t$$

for all terms in T.

[2] It is surprising to notice that, in general, the interpretations suggested by the authors in the literature satisfy these conditions (restriction on the domain and subterm property), which show they are not so restrictive.

ativity + Endomorphism. Let us take

$$[f](X_1)=2X_1$$

$$[*](X_1,X_2)= X_1X_2 + X_1$$

It is easy to check that this interpretation satisfies the subterm property. Now we may compute the values of the left-hand and right-hand sides of the rules. For the associativity rule, we obtain

$$[(x_1*x_2)*x_3](X_1,X_2,X_3)= X_1X_2X_3 + X_1X_2 + X_1X_3 + X_1$$

$$[x_1*(x_2*x_3)](X_1,X_2,X_3)= X_1X_2X_3 + X_1X_2 + X_1$$

so that we have to prove the inequality

$$X_1X_3 > 0$$

It is true because of $X_1 > 0$ and $X_3 > 0$. Similarly, as

$$[f(x_1)*(f(x_2)*x_3)](X_1,X_2,X_3)=4X_1X_2X_3 + 4X_1X_2 + 2X_1$$

$$[f(x_1*x_2)*x_3](X_1,X_2,X_3)=2X_1X_2X_3 + 2X_1X_2 + 2X_1X_3 + 2X_1,$$

this yields the inequality

$$2X_1X_2X_3 + 2X_1X_3 - 2X_1X_2 > 0 \quad \text{which is true since}$$

$$2X_1X_2X_3 > 2X_1X_2 \text{ and } 2X_1X_3 > 0$$

The latter follows from the fact that all its coefficients are positive. The former from the fact that $X_3 > 1$. The test for the rule $f(x_1)*f(x_2) \rightarrow f(x_1*x_2)$ is left to the reader.

3.3. The Principles of the procedure used for proving positiveness

Here is the key of our method. We prove inequalities one at a time and starting from a polynomial $P^{[0]}$ we build a sequence of inequalities such that "$P^{[0]} \geqq \ldots P^{[n-1]} \geqq P^{[n]} > 0$". The positiveness of $P^{[n]}$ is supposed

to be checked by a basic principle like "all coefficients are positive". At each step we transform some coefficients of $P^{[i]}$ (actually two) such that $P^{[i]}=P^{[i+1]} + Q^{[i]}$ where $Q^{[i]}$ is a positive monomial. More precisely, we propose the algorithm shown in Figure 1.

Positive = **proc**(P: polynomial) **returns**(**string**)

  while there exists a negative coefficient **do**

    if there exist $a_{p_1,\ldots,p_m} > 0$ and $a_{q_1,\ldots,q_m} < 0$,

      with $p_i \geq q_i$ for all $i \in [1..m]$

    then choose($a_{p_1,\ldots,p_m}$, $a_{q_1,\ldots,q_m}$)

      change($a_{p_1,\ldots,p_m}$, $a_{q_1,\ldots,q_m}$)

    else return("no-answer")

    end

  end

return("positive")

end

Figure 1: A procedure for checking positiveness of a polynomial

---

We suppose that

$$P^{[i]} = \sum_{p_1,\ldots,p_m \in \mathbf{N}^m} a^{[i]}_{p_1\cdots p_m} X_1^{p_1}\ldots X_m^{p_m}$$

where $a^{[i]}_{p_1\cdots p_m}$ is the coefficient of $X_1^{p_1}\ldots X_m^{p_m}$ in $P^{[i]}$.

The main idea of the procedure **Positive** is to consider a monomial with a negative coefficient, say ν and to try to find a monomial with a positive coefficient, say π, which bounds it. This means that for all value greater than or equal to 2, the value of π will be greater than the value of ν. This comes from consideration on the degree of the monomials, i.e. the degree of π has to be greater than the degree of ν. When such a monomial

is found, one removes from π as few part we can to bound as much part of ν. The procedure will now rely on how we choose the function **change**. We propose two solutions for the body of **change**($a_{p_1 \cdots p_m}$, $a_{q_1 \cdots q_m}$) where $a_{p_1 \cdots p_m}$ is positive and $a_{q_1 \cdots q_m}$ is negative. The first one is the most straightforward and uses the fact that the values of the variables are greater than 1. Therefore a monomial like X^n is greater than 1 and if n is greater than m, a monomial like X^nY is greater than X^mY. We indicate precisely the transformation to perform on the coefficients of π and ν according to the previous conventions.

Solution 1:

if $a^{[i]}_{p_1 \cdots p_m} > |a^{[i]}_{q_1 \cdots q_m}|$

then

$$a^{[i+1]}_{p_1 \cdots p_m} := a^{[i]}_{p_1 \cdots p_m} + a^{[i]}_{q_1 \cdots q_m}$$
$$a^{[i+1]}_{q_1 \cdots q_m} := 0$$

else

$$a^{[i+1]}_{p_1 \cdots p_m} := 0$$
$$a^{[i+1]}_{q_1 \cdots q_m} := a^{[i]}_{q_1 \cdots q_m} + a^{[i]}_{p_1 \cdots p_m}$$

The other coefficients are unaltered.

The example of Associativity + Endomorphism illustrates an application of this technique. In order to understand the second solution let us look at a realistic example that cannot be solved by the previous transformation.

Example 2. Using a Knuth-Bendix procedure, the rewriting system

$$x + (y + z) \rightarrow (x + y) + z$$
$$x * (y + z) \rightarrow (x * y) + (x * z)$$

can be completed into itself plus the rule

$$(u + (x * y)) + (x * z) \rightarrow u + (x * (y + z))$$

by using the interpretation $[+](X, Y) = XY + Y$ and the $[*](X, Y) = XY$. A completion algorithm which orients the third rule has to prove that

$$[(u + (x * y)) + (x * z)](X,Y,Z,U) > [u + (x * (y + z))](X,Y,Z,U)$$

i.e.,

$$UX^2YZ + X^2YZ + XZ > UXYZ + UXZ + XYZ + XZ.$$

Since all the coefficients are equal to 1, in order to apply Solution 1, we should have at least as many monomials with positive coefficients as those with negative coefficients. However we can take advantage of the fact that X is greater than or equal to 2, thus

$$UX^2YZ \geq 2UXYZ > UXYZ + UXZ.$$

This remark can be generalized to the power of 2. Indeed

$$a^{[i]}_{p_1\cdots p_m} X_1^{p_1}\cdots X_m^{p_m} + a^{[i]}_{q_1\cdots q_m} X_1^{q_1}\cdots X_m^{q_m}$$
$$= (a^{[i]}_{p_1\cdots p_m} + a^{[i]}_{q_1\cdots q_m} X_1^{q_1-p_1}\cdots X_m^{q_m-p_m}) X_1^{p_1}\cdots X_m^{p_m}$$

Since $X_i \geq 2$, $p_i \geq q_i$ and $a^{[i]}_{q_1\cdots q_m} \leq 0$, the expression below is

$$\geq (a^{[i]}_{p_1\cdots p_m} + a^{[i]}_{q_1\cdots q_m} 2^{q_1-p_1}\cdots 2^{q_m-p_m}) X_1^{p_1}\cdots X_m^{p_m}$$

The idea is now to use this last coefficient as the new coefficient of $X_1^{p_1}\cdots X_m^{p_m}$ in $P^{[i+1]}$ and to delete the coefficient of $X_1^{q_1}\cdots X_m^{q_m}$. This removes a smaller part of $a^{[i]}_{p_1\cdots p_m}$ than Solution 1. For instance, with

$X^2Y - X$ we get $3/4\ X^2Y$ instead of 0.

Solution 2:

if $a^{[i]}_{p_1\ldots p_m} > |a^{[i]}_{q_1\ldots q_m} 2^{q_1-p_1}\ldots 2^{q_m-p_m}|$

then

$$a^{[i+1]}_{p_1\ldots p_m} := a^{[i]}_{p_1\ldots p_m} + a^{[i]}_{q_1\ldots q_m} 2^{q_1-p_1}\ldots 2^{q_m-p_m}$$
$$a^{[i+1]}_{q_1\ldots q_m} := 0$$

else

$$a^{[i+1]}_{p_1\ldots p_m} := 0$$
$$a^{[i+1]}_{q_1\ldots q_m} := a^{[i]}_{q_1\ldots q_m} + a^{[i]}_{p_1\ldots p_m} 2^{p_1-q_1}\ldots 2^{p_m-q_m}$$

The other coefficients are unaltered.

We may now prove the two following propositions.

Proposition 2: Using Solution 1, **change** transforms a polynomial into a less polynomial.

Proof: Take the first **change** function and consider only the first case of Solution 1.

$$P^{[i+1]} = \ldots + a^{[i]}_{k_1\ldots k_m} X_1^{k_1}\ldots X_m^{k_m} + \ldots + 0X_1^{q_1}\ldots X_m^{q_m} + \ldots + (a^{[i]}_{p_1\ldots p_m} + a^{[i]}_{q_1\ldots q_m})X_1^{p_1}\ldots X_m^{p_m}$$

$$= \ldots + a^{[i]}_{k_1\ldots k_m} X_1^{k_1}\ldots X_m^{k_m} + \ldots + a^{[i]}_{p_1\ldots p_m} X_1^{p_1}\ldots X_m^{p_m} + \ldots + a^{[i]}_{q_1\ldots q_m} X_1^{q_1}\ldots X_m^{q_m} X_1^{p_1-q_1}\ldots X_m^{p_m-q_m}.$$

Therefore, if $a^{[i]}_{p_1 \ldots p_m} > |a^{[i]}_{q_1 \ldots q_m}|$, putting $P^{[i]}$ in the left hand side, we get:

$$P^{[i]} = P^{[i+1]} - a^{[i]}_{q_1 \ldots q_m} X_1^{q_1} \ldots X_m^{q_m} (X_1^{p_1 - q_1} \ldots X_m^{p_m - q_m} - 1)$$

and

$$P^{[i]} = P^{[i+1]} + a^{[i]}_{p_1 \ldots p_m} X_1^{q_1} \ldots X_m^{q_m} (X_1^{p_1 - q_1} \ldots X_m^{p_m - q_m} - 1)$$

otherwise

Proposition 3: Using Solution 2, **change** transforms a polynomial into a less polynomial.

Proof: Remember than $X_i \geq 2$, so any power of X_i / 2 is greater than 1. Thus, when one takes the second **change** function,

if $a^{[i]}_{p_1 \ldots p_m} > |a^{[i]}_{q_1 \ldots q_m} 2^{q_1 - p_1} \ldots 2^{q_m - p_m}|$

then

$$P^{[i]} = P^{[i+1]} - a^{[i]}_{q_1 \ldots q_m} X_1^{q_1} \ldots X_m^{q_m} [(X_1/2)^{p_1 - q_1} \ldots (X_m/2)^{p_m - q_m} - 1]$$

else

$$P^{[i]} = P^{[i+1]} + a^{[i]}_{p_1 \ldots p_m} X_1^{q_1} \ldots X_m^{q_m} [(X_1/2)^{p_1 - q_1} \ldots (X_m/2)^{p_m - q_m} - 1]$$

Notice that the polynomials $P^{[i]}$ have their coefficients in dyadic numbers, i.e. numbers of the form $n/2^k$, even though $P^{[0]}$ has coefficients in **N**.

We may now state the following theorem about the procedure **Positive**.

Theorem: Whatever **change** function is taken, the procedure **Positive** always terminates and returns "positive", only if the polynomial "input" is positive for all natural greater than or equal to 2.

Proof: The correction comes from the correction of **change** and the termina-

tion comes from the fact that each call to **change** decreases the absolute value of the coefficient of one monomial of a number at least equal to 2^{-k} where k is the maximum of $p_1+\ldots+p_m$ for all monomials in $P^{[0]}$.

When the first method is used, the second one can always be used and the difference between $P^{[i]}$ and $P^{[i+1]}$ is always smaller and has more chance to succeed as shown in Example 2. For this reason, we have implemented the second method in REVE and we have kept the presentation of the first one, because of its simplicity. In the procedure **Positive**, the choice of the coefficients to compare is a difficult part of its implementation. Some choices could lead to a failure when others do not, as illustrated by the following example see [BenCherifa-86] for a full discussion.

Example 3: Let

$$P^{[0]}(X, Y) = X^2Y + X^2 - X - 4Y$$

and suppose we choose $a_{2,1}$ and $a_{1,0}$, then

$$P^{[1]}(X, Y) = 3/4\ X^2Y + X^2 - 4Y.$$

Now choose $a_{2,1}$ and $a_{0,1}$

$$P^{[2]}(X, Y) = X^2 - Y$$

which leads to a failure. On the other hand, we could have chosen first $a_{2,0}$ and $a_{1,0}$,

$$P^{[1]}(X, Y) = X^2Y + 1/2X^2 - 4Y$$

and $a_{2,1}$ and $a_{0,1}$

$$P^{[2]}(X, Y) = 1/2X^2,$$

which is positive.

The current implementation in REVE tries to compare a $a^{[i]}_{p_1 \cdots p_m}$ and a $a^{[i]}_{q_1 \cdots q_m}$ with the smallest difference and seems to work well. A study should be made to know if it is good.

On the other hand, notice that other properties of polynomials could be used like

$$X_1^2 + X_2^2 > 2X_1X_2,$$

or

$$X_1^4 + 6X_1^2X_2^2 + X_2^4 > 4X_1^3X_2 + 4X_1X_2^3.$$

It could be also possible to take values greater than or equal to 3 and to include this property into the definition of a new function **change**.

4. Polynomial Interpretations of Associative-Commutative Operators

Associative-commutative operators often occur in rewriting system and it is really important to have methods to prove termination of associative-commutative rewriting systems. In the presence of associative-commutative equations (written AC), one interprets the quotient algebra $T(F,\{x_1,\ldots,x_m\})/AC$. Therefore any interpretation has to be consistent with the laws. The purpose of this section is to give a characterization of when polynomial interpretations are consistent in this sense. It turns out that this criterion is very simple and can be tested simply. Thus if a polynomial Q interprets an associative-commutative operator, it satisfies the two conditions:

$$Q(X, Y) = Q(Y, X)$$

$$Q(Q(X, Y), Z) = Q(X, Q(Y, Z)).$$

The first equation says that Q is symmetric, the second one gives a bound on the degree. Indeed if the highest degree of X in Q is m, then m has to satisfy the identity $m^2 = m$, since m^2 is the highest degree of X in Q(Q(X,Y),Z). Therefore m=0,1 and the general form of Q is

$$Q(X,Y) = aXY + b(X+Y) + c$$

then

$$Q(Q(X,Y),Z) = a^2XYZ + ab(XY+YZ+ZX) + b^2X + b^2Y + (ac+b)Z + c(b+1).$$

$$Q(X,Q(Y,Z)) = a^2XYZ + ab(XY+YZ+ZX) + (ac+b)X + b^2Y + b^2Z + c(b+1),$$

then

$$Q(Q(X,Y),Z) - Q(X,Q(Y,Z)) = (ac + b - b^2)(Z - X)$$

and we have this criterion:

Proposition 4: The polynomials that satisfy associative-commutative equations, i.e., the polynomials that interpret associative-commutative operators, are the polynomials of the form:

$$aXY + b(X+Y) + c \qquad \text{with} \qquad ac + b - b^2 = 0$$

Surprisingly enough we are not aware of any mention of this criterion in the literature, except for Lankford who gives a non exhaustive list of possible solutions, but no actual characterization and a false conjecture. He says [Lankford 79]: "The only polynomials that we have found have one of the following three forms c, cXY, c + x + y + dxy. We conjecture that these are the only forms that such polynomials can have". His list does not include polynomials like 2XY + 2(X + Y) + 1, XY + 2(X + Y) + 2, 3XY + 3(X + Y) + 2, 2XY + 3(X + Y) + 3, 6XY + 3(X + Y) + 1, XY + 3(X + Y) + 6

etc... and his class c + x + y + dxy is too large, without the restriction that c or d is equal to zero.

Proofs of termination of associative commutative rewriting systems based on polynomial interpretations have another property. They do not require to prove the termination of the extensions [Peterson-Stickel-81,Jouannaud-Kirchner-84,Jouannaud-Kirchner-86]. Recall that the extension of a rule $s \rightarrow t$ is the rule $s + x \rightarrow t + x$. Thus if

$$[s] > [t]$$

then

$$[s+x] > [t+x]$$

since this is equivalent to

$$a([s]X) + b([s] + X) + c > a([t]X) + b([t] + X) + c$$

and to

$$(aX + b)\,[s] > (aX + b)\,[t].$$

5. Interpretations by a Cartesian product of polynomials, or why Lankford's example 3 works?

When dealing with a really classical example, namely the Naturals with addition and product defined in terms of the function "successor", we arrived at the rather difficult problem that neither the classical polynomial interpretations nor the other approaches could handle [Bachmair-Plaisted-85]. Indeed, such a specification uses a rewriting system with "+" and "." associative and commutative.

$$0 + x \rightarrow x$$

$$s(x) + y \rightarrow s(x + y)$$

$$0 \cdot x \rightarrow 0$$

$$s(x) \cdot y \rightarrow (x \cdot y) + y$$

$$x \cdot (y + z) \rightarrow (x \cdot y) + (x \cdot z)$$

Because of the associativity and commutativity and the restrictions on the polynomial interpretations of "+" and ".", we have to choose their interpretations of degree one. A simple computation made on the degree of X in the interpretation of the distributivity shows that the interpretation of "+" has to be of the form "(X + Y) + c", but this cannot work with any interpretation proving the termination of the rule:

$$s(x) + y \rightarrow s(x + y).$$

Thus the classical interpretations do not work in this case and we propose to use a p-tuple of polynomials for interpreting the operators instead of a unique one. We will use the notation

$$[t](X_1,\ldots,X_n) = ([t]_1(X_1,\ldots,X_n),\ldots,[t]_p(X_1,\ldots,X_n))$$

where the $[t]_i(X_1,\ldots,X_n)$ are the same kind of polynomials as defined in the previous sections. A lexicographical comparison will allow us to handle scale of ordering that polynomials cannot.

Let us first define this on the previous example. The interpretation of an operator of arity n is a pair of polynomials of same arity n. The interpretation of a term is made component-wise. The comparison of two terms is made lexicographically by first comparing the first components and if they are equal their second components. Since the lexicographical

product of well-founded orderings is well-founded, we obtained that way a well-founded ordering. For instance, let us take:

$[0] = (2, 2)$

$[s](X) = (X + 2, X + 1)$

$[+](X, Y) = (X + Y + 1, XY)$

$[.](X, Y) = (XY, XY)$.

Since the components of the interpretations of "+" and "." satisfy the conditions for the polynomial interpretations of associative and commutative operators, the whole interpretations are constant on each equivalence class modulo associativity and commutativity and can be used for proving the termination of the previous associative commutative rewriting system. We have indeed

for the first rule

$[0 + x](X) = (X + 3, 2X)$

$[x](X) = (X, X)$

and

$(X + 3, 2X) > (X, X)$

for the second rule

$[s(x) + y](X, Y) = (X + Y + 3, XY + Y)$

$[s(x + y)](X, Y) = (X + Y + 3, XY + 1)$

and

$(X + Y + 3, XY + Y) > (X + Y + 3, XY + 1)$

for the third rule

$[0 . x](X) = (2X, 2X)$

$[0](X) = (2, 2)$

and

$$(2X, 2X) > (2, 2)$$

for the fourth rule

$$[s(x) \cdot y](X, Y) = (XY + 2Y, XY + Y)$$

$$[(x \cdot y) + y](X, Y) = (XY + Y + 1, XY^2)$$

and

$$(XY + 2Y, XY + Y) > (XY + Y + 1, XY^2)$$

for the fifth rule

$$[x \cdot (y + z)](X, Y, Z) = (XY + XZ + X, XYZ)$$

$$[(x \cdot y) + (x \cdot z)](X, Y, Z) = (XY + XZ + 1, X^2YZ)$$

and

$$(XY + XZ + X, XYZ) > (XY + XZ + 1, X^2YZ).$$

Therefore the rewriting system is terminating.

As we mention in the title this method is already present in [Lankford 79] in rule (18) and rule (19). However we feel our presentation gives the conceptual framework behind rules that are only given through an example, and allows extension to more than two levels in the Cartesian product. In addition the method is actually implemented in REVE.

6. Finding the right polynomial interpretation: several suggestions

In this section we would like to provide hints for the difficult problem of finding an interpretation which proves the termination of a given rewriting system. During our experiments the main method we used was by "trial and error" and for this the help of a computer was essential, for instance in the Associativity + Distributivity system mentioned in Example 2, the computer behaved better than ourselves. This can be improved by good messages, when the software fails to orient a specific rule. However,

Dave Detlefs noticed that in many examples of associative-commutative rewriting systems a trivial interpretation like $[+](X, Y) = 2XY$ works often and he proposed it as the default. As suggestions for a user faced to the problem of orienting a rewriting system by a polynomial interpretation, we propose experimental facts.

Suggestion 1.

Set the interpretation of your constants to 2.

Suggestion 2.

Look for a hierarchy on your operators and use it in a Cartesian product interpretation. Such a hierarchy could be provided by a precedence on the operators given, for instance, by an incremental ordering like the recursive path ordering or the recursive decomposition ordering [Forgaard-Detlefs-85] run on the rewriting system. This precedence can be topologically sorted to obtain a total hierarchy, that is to extend it to a linear ordering. To use it in a Cartesian product of polynomial interpretations, set the interpretation of the operators with the highest precedence to a highest degree polynomial interpretation on the first component of the Cartesian product (see next section for examples).

Suggestion 3.

For your associative operators, if you want to orient your rule like

$$x + (y + z) \rightarrow (x + y) + z$$

try one of these interpretations, either $[+](X, Y) = XY + Y$ or, $[+](X, Y) = 2XY + Y$ or $[+](X, Y) = X + Y^2$. The idea is always to give more "weight" to the second argument and the choice between these

interpretations will depend on the other rules. Obviously, the interpretations $[+](X, Y) = XY + X$ or, $[+](X, Y) = 2XY + X$ or, $[+](X, Y) = X^2 + Y$ may work if you want to orient the rule in the opposite direction.

Suggestion 4.

If the rule is a definition of an operator i.e., a rule of the form $f(x_1,\ldots, x_n) \rightarrow t(x_1,\ldots, x_n)$, an interpretation like $[f] = [t] + 1$ works.

The appendix gives a demonstration of a computer session run on the term rewriting system laboratory REVE. The example is an axiomatization of groups due to Taussky, and proposed by Knuth and Bendix in their paper [Knuth-Bendix-70]. The reader will notice that the interpretation of $*$, namely $[*](X, Y) = 2XY + Y$ is an application of Suggestion 3 and that the interpretation of g, namely $[f(x,x*i(y))] + 1$, is an application of Suggestion 4. None can be guessed at the beginning of the completion algorithm, but only when the associativity of $*$ and the actual definition of g in terms of f are known. This shows the difficulties in the use of these suggestions.

7. Review of Systems whose termination was proved by our algorithm and actually run on REVE.

The first example we proved was obviously the Associativity-Endomorphism (see Section 1). Notice that the interpretation $[f](X_1)= 2X_1$ and $[*](X_1,X_2)= X_1^2 + X_2$, produces the same diverging system as the Recursive Path Ordering. A similar example is Associativity-Antimorphism. It can be seen as the optimization of inversions in an algebraic system with

an associative law, like in matrix manipulation.

$$x * (y * z) \rightarrow (x * y) * z$$

$$f(x) * f(y) \rightarrow f(y * x)$$

$$(x * f(y)) * f(z) \rightarrow x * f(z * y)$$

The system is convergent, but its termination cannot be proved by a recursive path ordering. The interpretation $[*](X, Y) = XY + Y$ and $[f](X) = X + 1$ works.

Then we studied the equations for the groups

$$e * x == x$$

$$i(x) * x == e$$

$$(x * y) * z == x * (y * z)$$

$$x / y == x * i(y)$$

using an interpretation proposed by Huet [Huet-80].

$$[e] = 2$$

$$[i](X) = X^2$$

$$[*](X, Y) = 2XY + X$$

$$[/](X,Y) = 1 + X + 2XY^2.$$

The interpretation of / is just $[(x * i(y)] + 1$ (Suggestion 4). This interpretation was used to complete these equations into the ten classical Knuth-Bendix rules plus the definition of /. On the first three equations our Suggestion 3 would give the following Cartesian product interpretation to get the ten classical rules.

$$[e] = (2, 2)$$

$$[i](X) = (X^2, X^2)$$

$$[*](X, Y) = (X + Y, X + Y^2)$$

On the four equations for groups another interpretation can be taken

$[e] = 2$

$[i](X) = X^2$

$[*](X, Y) = 1 + X + Y^4$

$[/](X, Y) = X + Y^2$.

It completes them in the non classical set of 10 rules already discovered using the recursive path ordering [Lescanne-83].

$(e / x) \rightarrow i(x)$

$i(e) \rightarrow e$

$(x / e) \rightarrow x$

$(x / x) \rightarrow e$

$i(i(x)) \rightarrow x$

$i((y / x)) \rightarrow x / y$

$(x / y) / i(y) \rightarrow x$

$(x / i(y)) / y \rightarrow x$

$(x / (y / z)) \rightarrow ((x / i(z)) / y)$

$(x * y) \rightarrow (x / i(y))$.

We are also interested by the termination proof of the following system, given by Hsiang [Hsiang-82]

$x + 0 = x$

$x + (-x) = 0$

$x * 1 = x$

$x * x = x$

$(x + y) * z = (x * z) + (y * z)$

$x + x = 0$

using the polynomial interpretations:

$[0] = 2$

$[1] = 2$

$[-](X) = X + 1$

$[+](X,Y) = X + Y + 2$

$[*](X,Y) = 2XY + 1$

The last system presented in this review, deals with the symbolic differentiation with respect to x, given by Dershowitz [Dershowitz-85]

$D_x\ x = 1$

$D_x\ a = 0$

$D_x(\alpha + \beta) = D_x\ \alpha + D_x\ \beta$

$D_x(\alpha - \beta) = D_x\ \alpha - D_x\ \beta$

$D_x(-\alpha) = -D_x\ \alpha$

$D_x(\alpha * \beta) = \beta * D_x\ \alpha + \alpha * D_x\ \beta$

$D_x(\alpha / \beta) = D_x\ \alpha / \beta - \alpha * D_x\ \beta / \beta^2$

$D_x(\ln \alpha) = D_x\ \alpha / \alpha$

$D_x(\alpha^\beta) = \beta * \alpha^{\beta-1} * D_x\ \alpha + \alpha^\beta * (\ln \alpha) * D_x\ \beta$

The following polynomial interpretations can be used to prove termination and to complete the above system:

$[+](\alpha,\beta) = \alpha + \beta$

$[-](\alpha,\beta) = \alpha + \beta$

$[\hat{}](\alpha,\beta) = \alpha + \beta$

$[-](\alpha) = \alpha + 1$

$[a] = [b] = [0] = [1] = [2] = 2$

$[*](\alpha,\beta) = \alpha + \beta$

$[/](\alpha,\beta) = \alpha + \beta$

$[D_x](\alpha) = \alpha^3$

$[\ln](\alpha) = \alpha + 1$

This is a little different from the interpretation proposed by Dershowitz, but agrees with our suggestions, namely the interpretation of the constant

is set to 2.

8. Conclusion

The method based on polynomial interpretations is now proposed in the rewrite rule laboratory REVE. Though this termination check procedure is definitely necessary in order to run some of the examples we know, like Associativity + Endomorphism or Associativity + Antimorphism, we do not think it will replace in each occasion the current methods based on recursive path ordering [Dershowitz-82] or recursive decomposition ordering [Jouannaud,etal.-82], since they have shown to have a large scope and to be really easy to use in many practical cases [Forgaard-Detlefs-85], and to be usable when polynomial interpretation fails as pointed out by Dershowitz [Dershowitz-83]. Let us notice that other general terminations can handle the associativity+ endomorphism example namely the Knuth-Bendix ordering [Martin 87] and the transformation ordering [Bellegarde & Lescanne 87]. Moreover, since the termination of rewriting systems is undecidable [Huet-Lankford-78], there exists obviously no universal method. So, we think we will keep both methods in REVE and let the user choose the method he or she wants. However in the case of associative-commutative operators, the extensions of the recursive path ordering either fail or are not ready for being incorporated in a rewrite rule laboratory like REVE. So, it is the only method currently available and we are incorporating it into REVE-3 (the general equational rewrite rule laboratory) as the mechanism for proving termination in the associative-commutative case.

In conclusion, we would like to talk about the polynomial interpretation limitations. The first one deals with our criterion. As already men-

tioned, it cannot prove the positiveness of the polynomial

$$X_1^2 + X_2^2 - 2X_1X_2 + 1 = (X_1-X_2)^2 + 1$$

Since we never encountered such a polynomial in termination proofs we feel that would not be an obstacle for using it. On the opposite, we do not feel as a restriction that our method checks positiveness of polynomial for values greater than or equal to 2 instead of any natural value, since a change of variable gives a polynomial that satisfies the condition. Thus, if in the polynomial $3X^2 - 7$, which is positive for $X \geq 3$, we change X into $Y + 1$ we get the polynomial $3Y^2 - Y - 4$ that fulfills our requirements. The second kind of restriction is mentioned by [Huet-Open-80,Dershowitz-85]. They say that the general polynomial interpretations impose a polynomial upper bound on the complexity of the computations, we claim this is not true as demonstrated by this one rule term rewriting system

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

which has an exponential computation complexity, since it computes a term related to the Fibonacci numbers and which can be proved to terminate using the interpretation $[f](X, Y) = X + Y$ and $[s](X) = X^2$. We conjecture that such a limitation exists, but a precise formulation has to be stated and proved.

Acknowledgment

We are pleased to thank all the people who gave us helpful suggestions, especially Françoise Bellegarde, Dave Detlefs, Harald Ganzinger, Jean-Pierre Jouannaud, Dallas Lankford, Laurence Puel, Jean-Luc Rémy and René Schott.

References

Bachmair-Plaisted-85.

L. Bachmair and D.A. Plaisted, "Termination Orderings For Associative-Commutative Rewriting Systems," Journal of Symbolic Computation, 1985.

Bellegarde 84.

F. Bellegarde, "Rewriting Systems on FP Expressions to reduce the number of Sequences Yielded," Science of Computer Programming, vol. 6, pp. 11-34, North-Holland, 1986.

Bellegarde & Lescanne 87.

F. Bellegarde and P. Lescanne, "Transformation Ordering," CAAP '87, 1987.

BenCherifa-86.

A. BenCherifa, "Preuves de terminaison de systèmes de réécriture fondées sur des interprétations polynomiales," Thèse, Université de Nancy 1, Nancy (France), 1986.

Cohen 69.

P. J. Cohen, "Decision Procedures for Real And P-Adic Fields," Comm. Pure And Appl. Math. 22, pp. 131-151, 1969.

Collins 75.

G. Collins, "Quantifier Elimination for Real Closed Fields By Cylindrical Algebraic Decomposition," in Proc. 2Nd Gi Conference on Automata And Formal Languages Lecture Notes in Computer Science, Springer-

Verlag, Kaiserslautern, 1975.

Davis 73.

M. Davis, "Hilbert'S Tenth Problem Is Unsolvable," Amer. Math. Monthly, vol. 80, no. 3, pp. 233-269, 1973.

Dershowitz-82.

Nachum Dershowitz, "Orderings for Term-Rewriting Systems," Theoretical Computer Science, vol. 17, pp. 279-301, 1982.

Dershowitz-83.

N. Dershowitz, "Well-Founded Orderings," ATR-83(8478)-3, Information Science Research Office, The Aerospace Corporation, El Segundo, California (USA), May 1983.

Dershowitz-85.

N. Dershowitz, "Termination," in Proc. 1rst Conf. Rewriting Techniques and Applications, Lecture Notes in Computer Science, vol. 202, pp. 180-224, Springer Verlag, Dijon (France), May 1985.

Forgaard-Detlefs-85.

R. Forgaard and D. Detlefs, "An incremental algorithm for proving termination of term rewriting sytems," in Proc. 1rst International Conference on Rewriting Techniques and Applications., Lecture Notes in Computer Science, vol. 202, Springer Verlag, Dijon (France), 1985.

Gnaedig-Lescanne-86.

I. Gnaedig and P. Lescanne, "Proving Termination of Associative Rewriting Systems by Rewriting," in Proc. 8th Conf. on Automated Deduction, Lecture Notes in Computer Science, Springer Verlag, Oxford (England), 1986.

Hsiang-82.

J. Hsiang, "Topics in Automated Theorem Proving And Program Generation," PHD Thesis Unv. of Illinois at Urbana-Campaign, 1982.

Huet-80.

G. Huet, "Confluent reductions: abstract properties and applications to term rewriting systems," J. of ACM, vol. 27, no. 4, pp. 797-821, Oct. 1980.

Huet-Lankford-78.

G. Huet and D.S. Lankford, "On the Uniform Halting Problem for Term Rewriting Systems," Rapport Laboria 283, Iria, Mars 1978.

Huet-Open-80.

G. Huet and D. Oppen, "Equations and Rewrite Rules: A Survey," in Formal Languages: Perspectives And Open Problems, ed. Book R., Academic Press, 1980.

Jouannaud,etal.-82.

J.P. Jouannaud, P. Lescanne, and F. Reinig, "Recursive Decomposition Ordering," in Formal Description of Programming Concepts 2, ed. Bjorner D., pp. 331-346, North Holland, Garmish Partenkirschen, RFA, 1982.

Jouannaud-Kirchner-84.

J.P. Jouannaud and H. Kirchner, "Completion of a set of rules modulo a set of equations," Proceedings 11th ACM Conference of Principles of Programming Languages, Salt Lake City, 1984.

Jouannaud-Kirchner-86.

J.Jouannaud and H.Kirchner, "Completion of a set of rules modulo a set of equations," SIAM J. of Computing, vol. 15(4), 1986.

Knuth-Bendix-70.

D. Knuth and P. Bendix, "Simple Word Problems in Universal Algebras," Computational Problems in Abstract Algebra Ed. Leech J., Pergamon Press, pp. 263-297, 1970.

Lankford 79.

D.S. Lankford, "On Proving Term Rewriting Systems Are Noetherian," Report Mtp-3, Math. Dept., Louisiana Tech University, May 1979.

Lescanne-83.

P. Lescanne, "Computer Experiments with the REVE Term Rewriting System Generator," in 10th ACM Conf. on Principles of Programming Languages, pp. 99-108, Austin Texas, January 1983.

Manna & Ness 70.

Z. Manna and S. Ness, "On the Termination of Markov Algorithms," Third Hawaii International Conference on System Sciences, pp. 789-792, 1970.

Martin 87.

U. Martin, "How to choose the weights in the Knuth Bendix ordering," 2nd Conf. on Rewriting Techniques and Applications, Bordeaux, 1987.

Peterson-Stickel-81.

G. Peterson and M. Stickel, "Complete sets of reduction for equational theories with complete unification algorithms," J. of ACM, vol. 28, no. 2, pp. 233-264, 1981.

Tarski 51.

A. Tarski, "A Decision Method for Elementary Algebra And Geometry," U. of California Press, Berkeley, 1951.

APPENDIX

The appendix is a demonstration of a computer session run on REVE the term rewriting system laboratory. The example is an axiomatization of groups due to Taussky, and given by Knuth and Bendix in their paper

```
-> read taussky

User equations:

1.   x * (y * z) == (x * y) * z
2.   e * e == e
3.   x * i(x) == e
4.   g(x * y, y) == f(x * y, x)
5.   f(e, x) == x

No critical pair equations.

No rewrite rules.

Note that the following identifiers were parsed as nullary operators:
     e

-> ord

The current ordering is `noeq-dsmpos'.

What ordering do you wish to use now? poly

Current Interpretations:

I[x * y] = 2(x)(y)
I[e] = 2
I[i(x)] = 2(x)
I[g(x, y)] = 2(x)(y)
I[f(x, y)] = 2(x)(y)
```

```
I[x * y] = 2(x)(y)
New polynomial, `quit,' or [ret]: ((2*x)*y)+y
I[x * y] = (y) + 2(x)(y)

I[e] = 2
New polynomial, `quit,' or [ret]: 2
I[e] = 2

I[i(x)] = 2(x)
New polynomial, `quit,' or [ret]: (x*x)
I[i(x)] = (x^2)

I[g(x, y)] = 2(x)(y)
New polynomial, `quit,' or [ret]: ((2*x)*(y*y))+(y*y)+x+1
I[g(x, y)] = 1 + (x) + (y^2) + 2(x)(y^2)

I[f(x, y)] = 2(x)(y)
New polynomial, `quit,' or [ret]: x+y
I[f(x, y)] = (y) + (x)

All rewrite rules have been turned back into equations to preserve

the proof of termination.

All operator information has been discarded.

The current ordering is now `poly.'

-> kb
Starting Knuth-Bendix...

There are currently 0 rules and 5 equations in the system.

Ordered the equation:
    x * (y * z) == (x * y) * z
    into the rewrite rule:
    x * (y * z) -> (x * y) * z

Ordered the equation:
    e * e == e
    into the rewrite rule:
    e * e -> e

Ordered the equation:
    x * i(x) == e
    into the rewrite rule:
    x * i(x) -> e

Ordered the equation:
    g(x * y, y) == f(x * y, x)
    into the rewrite rule:
    g(x * y, y) -> f(x * y, x)
```

```
Ordered the equation:
    f(e, x) == x
    into the rewrite rule:
    f(e, x) -> x

There are currently 5 rules and 0 equations in the system.


Starting to compute critical pairs...

Critical pairs between the rule:
    e * e -> e
    and the rule:
    g(x * y, y) -> f(x * y, x)
        are as follows:
        g(e, e) == e

There are currently 5 rules and 1 equation in the system.


Starting to reduce and order equations...

There are currently 5 rules and 1 equation in the system.

Ordered the equation:
    g(e, e) == e
    into the rewrite rule:
    g(e, e) -> e
            .
            .
            .
            .
            .

There are currently 12 rules and 11 equations in the system.


Starting to reduce and order equations...

There are currently 12 rules and 11 equations in the system.

Ordered the equation:
    i(y) * x1 == i(i(x1) * y)
    into the rewrite rule:
    i(i(x1) * y) -> i(y) * x1

There are currently 13 rules and 10 equations in the system.


Starting to compute critical pairs...

Critical pairs between the rule:
    i(i(x)) -> x
```

```
and the rule:
i(i(x1) * y) -> i(y) * x1
    are as follows:
    i(x * y) == i(y) * i(x)

There are currently 13 rules and 11 equations in the system.


Starting to reduce and order equations...

There are currently 13 rules and 11 equations in the system.

Ordered the equation:
    i(x * y) == i(y) * i(x)
    into the rewrite rule:
    i(x * y) -> i(y) * i(x)

        Following 2 left-hand sides reduced:
        i(y1 * y) * y1 -> i(y)
            became:
            (i(y) * i(y1)) * y1 == i(y)
        i(i(x1) * y) -> i(y) * x1
            became:
            i(y) * i(i(x1)) == i(y) * x1

There are currently 12 rules and 10 equations in the system.


Starting to compute critical pairs...


Starting to reduce and order equations...


No user equations.

No critical pair equations.

Rewrite rules:

 1.   x * (y * z) -> (x * y) * z
 2.   x * i(x) -> e
 3.   f(e, x) -> x
 4.   (x * y) * i(y) -> x
 5.   i(e) -> e
 6.   i(i(x)) -> x
 7.   e * x -> x
 8.   x * e -> x
 9.   i(x) * x -> e
10.   (x * i(x1)) * x1 -> x
11.   g(x1, x) -> f(x1, x1 * i(x))
12.   i(x * y) -> i(y) * i(x)
```

```
Your system is complete!


Knuth-Bendix runtime:
  Total: 1:12.
    Unification: 20.00
    Rewriting: 28.35
    Ordering: 3.56
    Overhead: 20.90
  Computed 155 critical pairs and ordered 35 equations into rules.

-> q
```

Some steps in the computer session run on REVE have been skipped, the example being too long to appear exhaustively in this appendix.

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