1991

G.J. Akkerman, J.C.M. Baeten

Term rewriting analysis in process algebra

Computer Science/Department of Software Technology Report CS-R9130 June

Term Rewriting Analysis in Process Algebra

G.J. Akkerman
Department of Philosophy, University of Utrecht,
Heidelberglaan 2, 3584 CS Utrecht, The Netherlands

J.C.M. Baeten
Department of Software Technology, CWI,
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands
and
Programming Research Group, University of Amsterdam
P.O. Box 41882, 1009 DB Amsterdam, The Netherlands

Abstract

We discuss the application of term rewriting analysis to theories of communicating concurrent systems. Turning such a theory into a canonical term rewriting system yields decidability and is essential for implementation. We do this for the theory ACP with a silent step in the setting of branching bisimulation. It is necessary to consider rewriting modulo equalities.

The article has the following structure: After the introduction and a list of definitions, we consider in section 3 Knuth-Bendix completion. In section 4, we consider a Peterson-Stickel complete term rewriting system which has the same term algebra as the fragment of ACP^{τ} it corresponds with. In section 5, we prove termination of this rewrite system.

1980 Mathematics Subject Classification (1985 Revision):
68Q45, 68Q55, 68Q65, 68Q50.
1987 CR Categories: F.4.3, D.2.10, D.3.1, D.3.3.
Key words & Phrases: process algebra, term rewriting, silent step, branching bisimulation, Knuth-Bendix completion.
Note1: When we started this article, and wrote much of it, the first author was employee of the Software Engineering Research Centrum, Utrecht, The Netherlands.
Note2: The second author received partial support from ESPRIT basic research action 3006, CONCUR, and from RACE project 1046, SPECS. This document does not necessarily reflect the views of the SPECS consortium.

1 Introduction

In this section, we motivate our investigations, list our two main sources, and give a brief overview of the process algebra system that we consider.

1.1 Motivation for a term rewriting analysis

In this subsection, we explain why we did a term rewriting analysis of ACP^{τ}. The foremost reason for doing a term rewriting analysis is that it yields a canonical term rewriting system. A *canonical term rewriting system* is a term rewriting system that is *strongly terminating* and has *unique* normal forms. Canonical term rewriting systems can be used for the following purposes:

Report CS-R9130
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

1. It can be verified by inspection whether a certain class of function symbols (called *defined functions*) can be eliminated from a term by means of the rules of the rewriting system. (See [HH82, JK89].)
2. It can be used for proving that certain algebras are conservative extensions of others. (See [JK89].)
3. It yields a decision procedure for equality of terms: Two terms have the same interpretation in every model of an algebra if they have the same normal form.
4. A canonical term rewriting system yields a well defined implementation of the algebra for closed terms.

1.2 Origins

We apply the completion method of [PS81] to the process algebra system ACP^τ of [BW90]. One of the authors has considered this subject earlier, in [Akk87]. The version of ACP^τ considered in that report resisted completion. We now succesfully consider a modern version of ACP^τ.

1.3 Algebraical preliminaries

In this section we consider some notions from abstract algebra.

1.3.1 Term algebras

In this paper we consider the *closed term algebra* and *open term algebra* associated with an axiom system:

- The *closed term algebra* (or just *term algebra*) is constructed by considering the set of all closed terms over the signature of the axiom system, and partitioning them in equivalence classes, where terms are in the same equivalence class if and only if they can be shown equal by means of the equations.
- The *open term algebra* is constructed in the same way, but starting from the set of all terms, including those containing variables.

An equation is considered valid in the open term algebra if its left hand side and right hand side are in the same equivalence class, and valid in the closed term algebra if for every ground substitution instance of the equation, the left and right hand side of this instance are in the same equivalence class.

1.3.2 Conservative extensions

An axiom system E_1 is called a *conservative extension* of an axiom system E_2 if it allows all deductions of E_1, and does not allow any new deductions on old terms. Usually consideration is restricted to ground terms only.

1.4 Process Algebra

In this section, we will describe the axiomatization of process algebra that we consider. See table 1. This is a fragment of ACP^τ which can among other variants of process algebra be found in [BW90].

The axioms describe the possible behaviour of a process in terms of choices "+" and sequences (denoted by a usually invisible infix dot, as in ax for $a \cdot x$) of actions. Those actions are in the axioms referred to by action variables "a", "b" and "c," ranging over some set A of actions. The specification is parametrized over this set A of actions together with a partial communication

$x + y = y + x$	A1	$x\tau = x$	B1
$x + (y + z) = (x + y) + z$	A2	$x(\tau(y + z) + y) = x(y + z)$	B2
$x + x = x$	A3		
$(x + y)z = xz + yz$	A4		
$(xy)z = x(yz)$	A5		
$x + \delta = x$	A6	$a \mid b = \gamma(a, b)$ if $\gamma(a, b) \downarrow$	CF1
$\delta x = \delta$	A7	$a \mid b = \delta$ otherwise	CF2
$x \parallel y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x \mid y$	CM1	$ax \mid b = (a \mid b)x$	CM5
$a \lfloor\!\lfloor x = ax$	CM2	$a \mid bx = (a \mid b)x$	CM6
$ax \lfloor\!\lfloor y = a(x \parallel y)$	CM3	$ax \mid by = (a \mid b)(x \parallel y)$	CM7
$(x + y) \lfloor\!\lfloor z = x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$	CM4	$(x + y) \mid z = x \mid z + y \mid z$	CM8
		$x \mid (y + z) = x \mid y + x \mid z$	CM9

Table 1: Considered fragment of ACP^{τ}.

function $\gamma : A \times A \to A$. We take $\delta, \tau \in A$, and we assume that for all $a, b, c \in A$, $a|b = b|a$, $a|(b|c) = (a|b)|c$, $a|\tau = \delta$ and $a|\delta = \delta$.

For the meaning of the axioms in table 1, the reader is referred to [BW90].

Missing in table 1 are the axioms for ∂_H and τ_I. We do not consider those function symbols because their axiomatization depends on the actions in H and I being actions from A, and this does not fit our analysis using open terms.

A successful term rewriting analysis for ACP was carried out in [BK84] (although the treatment of rewriting modulo commutativity and associativity was inexact) and used to establish basic results about the theory. In [BK85] however, it appeared that such an analysis for the theory with Milner's silent step τ is not possible, due to the fact that the second and third τ law of Milner cannot be ordered (i.e., given a direction). Therefore, in [BK85] graph rewriting techniques and a notion of saturation were used instead, to establish the basic theorems for ACP with silent step. In [GW89] an alternative formulation of the laws for the silent step was proposed (having as semantics the so-called "branching bisimulation") and advantages over Milner's approach discussed. The question arises whether ACP with branching bisimulation allows a successful term rewriting analysis. In [BW90] it is claimed that this is indeed the case. The present article provides a full proof of this fact. A similar claim for CCS with branching bisimulation was made in [DIN90].

2 Definitions

In this section we define our concepts. We hope that most of our readers already are familiar with the concepts involved, otherwise it would be useful to consult [HO80].

Definition: (symbols)
We assume a set of uninterpreted *symbols.*

Definition: (signature)
A *signature* is a pair $\langle F, V\rangle$ of two disjoint sets of symbols. Symbols in F are called *function symbols.* Symbols in V are called *variables.* We assume the existence of a function $arity : F \to \mathbb{N}$.

Definition: (terms)
The set $T(F, V)$ of *terms* over F, V is defined as the smallest set such that $V \subseteq T(F, V)$ and if $t_1, \ldots, t_n \in T(F, V)$ and *arity*$(f) = n$ then $f(t_1, \ldots, t_n) \in T(F, V)$. We will usually denote $T(F, V)$ by T.

Definition: (equality of terms)

We use "$\equiv$" to denote *syntactical equality* on terms.

Definition: (occurrences, replacement)
Occurrences are finite sequences of integers. The set of occurrences $O(t)$ of a term t is defined by: $O(v) = \langle\rangle$ for $v \in V$ and $O(f(t_1, \ldots, t_n)) = \{\langle\rangle\} \cup \{\bigcup_{i=1}^{n} \langle i \rangle \frown O(t_i)\}$, where $u \frown v$ denotes the concatenation of sequences u and v. For $u \in O(t)$ the subterm t/u at u in t is defined by $t/\langle\rangle = t$ and $f(t_1, \ldots, t_n)/\langle i, u_1, ..., u_l\rangle = t_i/\langle u_1, ..., u_l\rangle$. We define $O(s,t)$, the occurrences of s in t as $\{u \in O(t) | t/u = s\}$. For $u \in O(t)$ replacement $t[u \leftarrow s]$ of the subterm in t at u by s is defined by $t[\langle\rangle \leftarrow s] = s$ and $f(t_1, \ldots, t_n)[\langle i \rangle \frown u \leftarrow s] = f(t_1, \ldots, t_{i-1}, t_i[u \leftarrow s], t_{i+1}, \ldots, t_n)$.

Definition: (closure under contexts)
We say that a relation $R : T \times T$ is *closed under contexts* if whenever $t_1 = s_1, \ldots, t_{i-1} = s_{i-1}, t_i R s_i, t_{i+1} = s_{i+1}, \ldots, t_n = s_n$ and *arity*$(f) = n$ then $f(t_1, \ldots, t_n) R f(s_1, \ldots, s_n)$.

Definition: (term rewriting system, axiom system)
We define both a *term rewriting system* and an *axiom system* as finite sets of pairs of terms. The difference between them is their intended use. We express this intended use by writing a pair of terms $\langle s, t\rangle$ in a term rewriting system as $s \rightarrow t$, and the same pair in an axiom system as $s = t$.

Definition: (substitution)
A *substitution* is a total function σ from V to T. Substitutions are extended to maps from T to T by $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$. We say that a relation R on terms is *closed under substitution* if for all terms s and t and all substitutions σ whenever sRt also $\sigma(s) R \sigma(t)$.

Definition: (Equational term rewriting system)
An *equational term rewriting system (ETRS)* is a pair $\langle R, E\rangle$ where R is an term rewriting system, and E an axiom system.

Definition: (equality and rewrite relations)
The *one step equality relation* $=_E^1$ of an equational term rewriting system $\langle R, E\rangle$ is defined as the smallest symmetrical relation on terms containing E that is closed under substitutions and contexts. The *equality relation* $=_E$ of a term rewriting system is defined as the reflexive transitive closure of the one step equality relation. The *one step rewrite relation* $\rightarrow_R^1$ of a term rewriting system R is defined by: $s \rightarrow_R^1 t$ if there exist an occurrence $u \in O(s)$, a substitution σ and a rule $l \rightarrow r \in R$ such that $s/u \equiv \sigma(l)$ and $t \equiv s[u \leftarrow \sigma(r)]$. The one step rewrite relation is the smallest relation on terms containing R that is closed under substitutions and contexts.

Definition: (Unification, matching)
We say that a substitution σ is an *E-unifier* of the two terms s and t if $\sigma(s) =_E \sigma(t)$. We say that two terms s and t are *E-unifiable* if there exists an E-unifier of s and t. We say that a substitution σ *E-matches* s to t if $\sigma(s) =_E t$.

Definition: (Rewriting relations modulo)
The one step rewrite relation $\rightarrow_{R/E}^1$ is defined as $=_E \, . \rightarrow_R^1 . =_E$. (Here $R_1.R_2$ is the composition $\{\langle x, y\rangle | \exists u(x R_1 u \wedge u R_2 y\}$ of the relations R_1 and R_2.) The rewrite relation $\rightarrow_{R/E}$ is defined as the reflexive transitive closure of $\rightarrow_{R/E}^1$. Notice that this relation allows equality steps between the rewrite steps.

Definition: (Noetherian)
A relation $\rightarrow$ is called *Noetherian* (or *well-founded*) if there are no infinite sequences $s_1 \rightarrow s_2 \rightarrow \ldots$.

Definition: (confluence of a pair of terms)
We call a pair $\langle s_1, s_2\rangle$ of terms *confluent* for a relation $\rightarrow$ if there exists a term t such that $s_1 \rightarrow^* t$ and $s_2 \rightarrow^* t$. (Here we use R^* for the transitive closure of R.)

3 Knuth-Bendix completion considered briefly.

Note: In this section, we will use the word *rewriting* for a reduction, because we will use the word *reduction* for the transformation of a problem to another (simpler) problem.

In this report we consider the confluence of a term rewriting system presenting ACP^τ. A well-known technique for constructing (often infinite) term rewriting systems for an equational theory is Knuth-Bendix completion. We will consider Knuth-Bendix completion in its original form [KB70, Hue80], and in its variant by Peterson and Stickel [PS81]. Let us first consider confluence.

A term rewriting system is ***confluent*** if for every pair of divergent reductions $t \rightarrow^* t_1$ and $t \rightarrow^* t_2$, the resulting pair of terms $\langle t_1, t_2 \rangle$ is confluent. Knuth-Bendix completion is based on a technique for finding divergent reductions $t \rightarrow^* t_1$ and $t \rightarrow^* t_2$ for which there are no convergent reductions. This technique for finding such divergent rewritings is based on the following two reductions:

- First, the problem is reduced to the consideration of all divergent pairs of one step rewritings, under the assumption that the rewriting relation is well-founded.
- Next, this problem is reduced to the consideration of all one step rewritings from terms obtained by overlapping two left hand sides of rewrite rules.

Since there are only finitely many (most general) overlappings of the left hand side of a rewrite rule, these reductions make the test for confluence decidable, and in fact yield a counterexample if they fail.

This theory of Knuth-Bendix completion sketched above has to be modified if we allow a sequence of rewrite steps interspersed with equality steps. The required modifications are not obvious if some rewrite rules have a variable that occurs more than once in its left hand side. We will use a variant of Knuth-Bendix completion by Peterson and Stickel, because this variant can indeed be viewed as a modification of Knuth-Bendix completion. (Other, more general, variants exist, e.g.,[JK86] and [BD89], but in as far as those deviate from the framework of Peterson and Stickel, they develop more theory, and require, e.g., a test for *coherence*.) Peterson and Stickels completion differs in the following respects from Knuth-Bendix completion:

- Unification in the overlap test is done modulo associativity and commutativity.
- Termination of the rewrite rules must be proven modulo associativity and commutativity. (That is, there must be no infinite descending chains of rewrite steps interspersed with equality steps.)
- The set of rewrite rules has to be increased: For every rule $l \rightarrow r$ where l is governed by a both associative and commutative function symbol (say $+$), we require that a rule $l + x \rightarrow r + x$ be added, where x is a variable not occurring in l or r. We notice that addition of such *extensions* is equationally sound, but it may give rise to new overlaps.

4 Results

A natural term rewriting system associated with ACP^τ would have the equations of ACP^τ oriented from left to right. Unfortunately, that rewrite system would not be confluent. In this section, we motivate the construction of a rewrite system by means of a sequence of divergent reductions. We will start by considering some divergent reductions of ACP^τ considered as a rewrite system. Later we will also consider a divergent reduction based on the rules we added to ACP^τ.

All term rewriting systems that we consider have the same term algebra as our fragment of ACP^τ: No equations were deleted, and only equations valid in the term algebra of our fragment of ACP^τ were added, as can easily be proved by means of structural induction.

Note: It is *not at all clear* whether the steps that we took to resolve the divergences are the only reasonable ones, though we cannot think of others that work. (Just orienting the outermost terms to rules did not seem very promising.)

1	$x + x = x$	14	$\delta \vert x = \delta$
2	$x\tau = x$	15	$\tau \vert x = \delta$
3	$x + \delta = x$	16	$x \lfloor\!\lfloor (\tau y) = x \lfloor\!\lfloor y$
4	$x \lfloor\!\lfloor \tau = x$	17	$a \vert b = \gamma(a, b)$
5	$\delta \vert a = \delta$	18	$x((\tau y) + y) = xy$
6	$\tau \vert a = \delta$	19	$x \Vert y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x \vert y$
7	$(xy)z = x(yz)$	20	$(\tau x) \vert y = \delta$
8	$a \lfloor\!\lfloor x = ax$	21	$x \lfloor\!\lfloor (\tau y + y) = x \lfloor\!\lfloor y$
9	$x(\tau y) = xy$	22	$(ax) \vert b = (a \vert b)x$
10	$(x + y)z = xz + yz$	23	$x(\tau(y + z) + y) = x(y + z)$
11	$\delta x = \delta$	24	$x \lfloor\!\lfloor (\tau(y + z) + y) = x \lfloor\!\lfloor (y + z)$
12	$(x + y) \lfloor\!\lfloor z = x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$	25	$(ax) \lfloor\!\lfloor y = a(x \Vert y)$
13	$(x + y) \vert z = x \vert z + y \vert z$	26	$(ax) \vert (by) = (a \vert b)(x \Vert y)$

Table 2: The associated rewrite system.

We will now present the sequence of divergences based on the new rewrite rules that they led to: rise to:

4.1 Rules [9] and [18]

Rules [9] and [18] are consequences of other rules:

- Rule [9]: $x(\tau y) =^{[7^{-1}]} (x\tau)y =^{[2]} xy$.
- Rule [18]: $x((\tau y) + y) =^{[3^{-1}]} x((\tau(y + \delta)) + y) =^{[23]} x(y + \delta) =^{[3]} xy$

The other rules present in table 2 but not in ACP^{τ} are not equational consequences of the rules used thus far, but are valid in the (*closed*) *term algebra*. (And the other rewrite rules can be used for proving this by structural induction.) They were added to avoid the problems noted below.

4.2 Rule [14]

Because: $x \vert y =^{[3^{-1}]} (x + \delta) \vert y =^{[13]} (x \vert y) + (\delta \vert y)$ Since the two outermost terms of this deduction form an equation of two terms in normal form, and this equation is not present as a rewrite rule or the converse of one, we must add rules to rewrite at least one of both terms. In this case, we decided to add rule [14]. The terms resulting after application of these rules are then equal by rule [3]

4.3 Rules [4] and [15]

Because: $ay =^{[8^{-1}]} a \lfloor\!\lfloor y =^{[2^{-1}]} (a\tau) \lfloor\!\lfloor y =^{[25]} a(\tau \Vert y) =^{[19]} a(\tau \lfloor\!\lfloor y + y \lfloor\!\lfloor \tau + y \vert \tau) =^{[8]} a(\tau y + y \lfloor\!\lfloor \tau + y \vert \tau)$ As before, we must add rules to rewrite at least one of both terms. In this case, we decided to add rules [4] and [15]. The terms resulting after application of these rules are then equal by rule [18]

Remark:
Let us note explicitly, that for resolving this equation, which was produced by equations obtained from left linear rules only (viz. rules [2], [8], [19] and [25]), we need rule [18], which is *not* left linear, (or its converse, which is nonterminating). Therefore we suspect that we *cannot* do a similar term rewriting analysis of the left linear fragment of ACP^{τ} using techniques (given in [Hue80]) only applicable to left linear rules. Consequently, we think that Peterson-Stickel completion provides the simplest (but also least powerful) working approach to term rewriting in process algebra.

4.4 Rules [16] and [20]

$a(x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x|y) =^{[19^{-1}]} a(x\|y) =^{[25^{-1}]} (ax) \lfloor\!\lfloor y =^{[9^{-1}]} a(\tau x) \lfloor\!\lfloor y =^{[25]} a(\tau x\|y) =^{[19]} a((\tau x \lfloor\!\lfloor y) + (y \lfloor\!\lfloor \tau x) + (x|\tau y)) =^{[25]} a(\tau(x\|y) + (y \lfloor\!\lfloor \tau x) + (x|\tau y)) =^{[19]} a(\tau(x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x|y) + (y \lfloor\!\lfloor \tau x) + (x|\tau y))$
As before, we must add rules to rewrite at least one of both terms. In this case, we decided to add rules [16] and [20]. The terms resulting after application of these rules are then equal by rule [23]

4.5 Commutativity of "|"

In table 2, we have already used commutativity of "|" by having only one version of [15] and [20]. Therefore, we assume for the moment that we have both versions of each rule. We consider now the following deduction: $a(x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x|y) =^{[19^{-1}]} a(x\|y) =^{[25^{-1}]} (ax) \lfloor\!\lfloor y =^{[16^{-1}]} (ax) \lfloor\!\lfloor (\tau y) =^{[25]} a(x\|(\tau y) =^{[19]} a(x \lfloor\!\lfloor (\tau y) + (\tau y) \lfloor\!\lfloor x + x|(\tau y)) =^{[16]} a(x \lfloor\!\lfloor y + (\tau y) \lfloor\!\lfloor x + x|(\tau y)) =^{[25]} a(x \lfloor\!\lfloor y + \tau(y\|x) + x|(\tau y)) =^{[20]} a(x \lfloor\!\lfloor y + \tau(y\|x) + \delta) =^{[3]} a(x \lfloor\!\lfloor y + \tau(y\|x) =^{[19]} a(x \lfloor\!\lfloor y + \tau(y \lfloor\!\lfloor x + x \lfloor\!\lfloor y + y|x)) =^{[23]} a(y \lfloor\!\lfloor x + x \lfloor\!\lfloor y + y|x)$. (Notice that this divergent reduction is based on the rules 16 and 20 that we have added.) To resolve it, we decide to add an equation "$x|y = y|x$", expressing the commutativity of "|".

4.6 Rules [21] and [24]

These can now be proven by the axioms presented thus far:

- Rule [21]: $x \lfloor\!\lfloor (\tau y + y) =^{[16^{-1}]} x \lfloor\!\lfloor \tau(\tau y + y) =^{[18]} x \lfloor\!\lfloor \tau y =^{[16]} x \lfloor\!\lfloor y$
- Rule [24]: $x \lfloor\!\lfloor (\tau(y+z) + y) =^{[16^{-1}]} x \lfloor\!\lfloor \tau(\tau(y+z) + y) =^{[23]} x \lfloor\!\lfloor \tau(y+z) =^{[16]} x \lfloor\!\lfloor (y+z)$

4.7 Main theorem

We can now state our main theorem, the confluence of our ACP^{τ} subset.

Theorem:
The system in table 2 is confluent.

Proof:
We verified this by computing all E-critical pairs, and checking their E-confluence, using the implementation of Peterson-Stickel completion described in [Akk87]. Under the assumption of well-foundedness of the rewrite relation, this suffices. It may be appropriate to check the confluence of table 2 with another implementation as well.

5 Proving termination

First we state a definition from [De79]:

Definition: (Simplification ordering)
A transitive and irreflexive relation $>$ is a *simplification ordering* on a set of terms T if for any terms t, t', $f(\cdots t \cdots)$, $f(\cdots t' \cdots) \in T$

1. $t > t'$implies $f(\cdots t \cdots) > f(\cdots t' \cdots)$ and
2. $f(\cdots t \cdots) > t$.

For the proof of termination of our term rewriting system, we use the following theorem (from [De79]):

Theorem:

$[x \Vert y]$	$2^{[x]+[y]+2}$
$[x \mathbin{\lfloor\!\lfloor} y]$	$2^{[x]+[y]}$
$[x \mid y]$	$2^{[x]+[y]}$
$[xy]$	$[x][y]+[x]$
$[x+y]$	$[x]+[y]+1$
$[a]$	2

Table 3: Interpretation of the function symbols

A term rewriting system $P = \{l_i \rightarrow r_i\}_{i=1}^{n}$ terminates if there exists a simplification ordering $>$ over T such thar $l_i > r_i$ for any assignment of terms in T to the variables of l_i.

Since this theorem considers ordinary rewriting, while we consider rewriting modulo associativity and commutativity, we cannot immediately apply it. However, if we require that the ordering is compatible with associativity and commutativity of the associative and commutative function symbols, the proof in [De79] can easily be adapted to rewriting modulo associativity and commutativity.

We will now describe the simplification ordering that we use to prove termination:

Definition: (Numbers greater than 1)
We define $\mathbf{N}_{\geq 2}$ as $\{m \in \mathbf{N} | m \geq 2\}$.

Definition: (extended polynomial terms)
We define extended polynomial terms inductively by:

- All $n \in \mathbf{N}_{\geq 2}$ are extended polynomials.
- All $v \in \mathbf{V}$ are extended polynomials.
- If x_1 and x_2 are extended polynomials, then $x_1 + x_2$ is an extended polynomial.
- If x_1 and x_2 are extended polynomials, then $x_1.x_2$ is an extended polynomial.
- If x is an extended polynomial, then 2^x is an extended polynomial.

We denote the set of extended polynomials over V by $X[V]$. Assigments of $\mathbf{N}_{\geq 2}$ to V are extended to $X[V]$ in the expected way.

Definition: (Majorizing)
We define the ordering $s > t$ ("s majorizes t") as $\forall_{m:V \rightarrow \mathbf{N}_{\geq 2}}.m(s) > m(t)$. Notice that majorizing is closed under substitution.

Theorem:
ACP^{τ} is terminating

Proof:
We make the following observations:

- Using the interpretation given in table 3, we define an ordering on T by $s > t$ if $[s] > [t]$.
- In table 3, associative and commutative function symbols are interpreted in such a way that compatibility modulo associativity and commutativity is ensured. (Our interpretation satisfies $[(x+y)+z] = [x+(y+z)]$, $[x+y] = [y+x]$, and $x|y = y|x$.)
- After noting that all the basic functions $2^{\cdot}$, $.+.$, $.\cdot.$ and 2 used to construct this interpretation already satisfy the conditions for a simplification ordering, it is easy to prove that $>$ on T is a simplification ordering.
- Some easy calculations show that for every rule $l \rightarrow r$ in table 2, $[l] > [r]$, hence $l > r$.

Remark: Remarkable about [De79] is that it, when it is used as we do here, does not depend on the well-foundedness of $\mathbf{N}_{\geq 2}$, in the sense that e.g., $\mathbf{R}_{\geq 2}$, the real numbers greater than 2, would work just as well.

References

[Akk87] G.J. Akkerman. *Knuth-Bendix Completions of Process Algebra Axiomatizations.* Technical Report IR-135, Free University, Amsterdam, October 1987.

[BD89] L. Bachmair and N. Dershowitz. Completion for rewriting modulo a congruence. *Theoretical Computer Science*, 67:173–201, 1989.

[BK84] J. A. Bergstra and J. W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1/3):109–137, 1984.

[BK85] J. A. Bergstra and J. W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, 1985.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge University Press, 1990.

[De79] N. Dershowitz A note on simplification orderings. *Information Processing Letters*, 9(5), 1979.

[DIN90] R. De Nicola, P. Inverardi and M. Nesi. Using the Axiomatic Presentation of Behavioural Equivalences for Manipulating CCS Specifications. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems.* LNCS 407, Springer Verlag, 1990.

[Fag84] F Fages. *Associative Commutative Unification.* Technical Report 287, INRIA, April 1984.

[GW89] R.J. Glabbeek and W. P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. (Extended Abstract.) In G. X. Ritter, editor, *Information Processing 89, IFIP World Congress*, San Francisco, pages 613-618 North-Holland, Amsterdam 1989.

[HH82] G. Huet and J.-M. Hullot. Proofs by induction in equational theories with constructors. *Journal of Computer and System Sciences*, 25:239–266, 1982.

[HO80] G. Huet and D. Oppen. Equations and rewrite rules: a survey. In R. Book, editor, *Formal Languages: Perspectives and Open Problems*, pages 349–405, Academic Press, 1980.

[Hue80] G. Huet. Confluent reductions: abstract properties and applications to term rewriting systems. *Journal of the Association for Computing Machinery*, 27(4), 1980.

[Hue81] G. Huet. A complete proof of correctness of the knuth bendix completion algorithm. *Journal of Computer and Systems Sciences*, 23(1), 1981.

[JK86] J.-P. Jouannaud and H. Kirchner Completion of a set of rules modulo a set of equations. *SIAM Journal on Computing*, 15:1155–1194, 1986.

[JK89] J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82, 1989.

[KB70] D. E. Knuth and P. B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebras*, pages 263–297, Pergamon Press, 1970.

[PS81] G. E. Peterson and M. E. Stickel. Complete sets of reductions for some equational theories. *Journal of the Association for Computing Machinery*, 28(2), 1981.