Term rewriting properties of SOS axiomatisations

D.J.B. Bosscher

Computer Science/Department of Software Technology

Report CS-R9405 January 1994

Term Rewriting Properties of SOS Axiomatisations

D.J.B. Bosscher
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
e-mail: `doeko@cwi.nl`

Abstract

In [Aceto, Bloom and Vaandrager, '92] two strategies are presented to produce axiomatisations of strong bisimulation equivalence for languages whose operational semantics can be expressed in the GSOS format of [Bloom, Istrail and Meyer, '90]. In [Aceto et al.] it is stated that if the GSOS systems satisfy certain finiteness conditions, one of these axiomatisations is strongly normalising and confluent. We show that their claim as a whole is wrong, but prove confluency and weak normalisation by presenting a normalising rewrite strategy. We can however prove strong normalisation for the axiomatisations of a decidable class of such systems. The analysis of the term rewriting properties of the axiomatisations is modulo the associativity and commutativity of the choice operation.

AMS Subject Classification (1991): 68Q42, 68Q55, 68Q60, 68Q65.
CR Subject Classification (1991): D.1.3, D.3.1, F.3.2, I.2.2.
Keywords & Phrases: Structural Operational Semantics (SOS), GSOS format, bisimulation equivalence, process algebra, axiomatisations, term rewriting.
Note: The author was sponsored by the ESPRIT Basic Research Action 7166, Concur2.

Report CS-R9405
ISSN 0169-118X
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

1 Introduction

Recent years have shown an enormous interest in formulating operational semantics in the style of Plotkin's SOS [11]. Several formats have emerged which allow the description of various programming languages and the properties of these are studied extensively [3],[12],[4],[13]. One of these formats is the GSOS format in which the operational semantics of many process calculi of interest can be expressed [3].

In [1] two strategies are presented to derive automatically an axiomatisation for bisimulation equivalence for languages whose operational semantics is expressed in the GSOS format. This research gives an unambiguous method for generating sound and complete axiom systems. The axiomatisations produced by the strategies are often rather close to existing "human invented" axiomatisations. A practical application of this theory lies in the simplification with the axioms, i.e. how certain operations can be eliminated. The authors claim that the axiomatisations produced by their so-called alternative strategy has nice term rewriting properties: for a class of GSOS systems in which only finite, non-cyclic transition systems can be expressed the axiomatisations are supposed to be strongly normalising and confluent on closed terms. In this paper we will show that their claim as a whole is wrong, but will show weak normalisation and confluency. We will show strong normalisation for the axiomatisations of a decidable class of GSOS systems.

Term rewriting of SOS-style axiomatisations is of interest for the Concur2 project, which aims at the integration of tools and techniques for process algebras. For a well-defined class of languages, axiomatisations can be generated automatically, which can be turned into a term rewriting system at the drop of a hat. The option *simplify* of the ECRIN-tool [8], which simplifies process terms by means of user provided term rewriting rules, can be extended with a default automatically generated alternative. The Process Algebra Manipulator [7] can be equipped with an option to generate axiomatisations automatically from the transition rules.

The organisation of this paper is as follows. In section 2 we will introduce the axiomatisation produced by the alternative strategy and two finiteness conditions on GSOS systems used throughout this paper. In section 3 we will show how to turn the axioms into a term rewriting system and argue briefly why we have to work modulo the associativity and commutativity of the choice operation. In section 4 we will show that an (undecidable) class of GSOS systems in which only finite, non-cyclic systems can be expressed, is weakly, but not strongly normalising. We will do this by presenting a counter example and a normalising strategy from which confluency easily follows. In section 5 we limit ourselves to a decidable subclass of these systems. We will show that with a small proviso the term rewriting systems obtained from these axiomatisations are strongly normalising. We end this paper with the conjecture that even this last proviso can be dropped.

Interestingly enough all results were obtained *without* the theory of recursive path orderings for rewriting modulo ac of [5]. Even more so, we do not know how to obtain the strong normalisation result with this theory.

2 Preliminaries

For the reader unfamiliar with [1] we will give an overview of the alternative strategy used to produce complete and sound axiomatisations for strong bisimulation equivalence of finite GSOS systems. In this paper we will give no definitions of (strong) bisimulation or notions from term rewriting. We think all these are standard for which the reader is referred to e.g. [10] for bisimulation and [6] for term rewriting.

Let us assume as input for the alternative strategy a GSOS system with signature Σ_G. The result is an axiomatisation with a signature Σ_A, which is an extension of Σ_G with possible auxiallary operations and a set of axioms E_A over Σ_A. The first output of the strategy is four axioms, called the FINTREE axioms. FINTREE is an auxilliary language defined as a fragment of CCS [9] expressing all finite, non-cyclic trees. It is defined inductively as 1. $\mathbf{0} \in$ FINTREE 2. $p_1, p_2 \in$ FINTREE $\Rightarrow a \cdot p_1, p_1 + p_2 \in$ FINTREE. We will refer to choice (+), action prefixing (a.) and $\mathbf{0}$ as FINTREE operations.

Definition 2.1 *The FINTREE axioms are given by the following equations*

$$x + y = y + x \quad (1)$$
$$(x + y) + z = x + (y + z) \quad (2)$$
$$x + x = x \quad (3)$$
$$x + \mathbf{0} = x \quad (4)$$

The remaining GSOS operations are then divided in the three classes the *good*, the *bad* and the *ugly*[1]. For the good operations the strategy can generate the axioms describing their behaviour. The other operations are linked with copying and distinctifying axioms to good operations.

First each ugly operations is linked to either a bad or a good operation with a so called copying axiom. "Copying" refers to the possible multiplication of arguments going from left-to-right.

Definition 2.2 *Let $f \in \Sigma_G$ be an ugly operation and $A \in E_A$ an axiom of the form*

$$f(x_1, ..., x_{ar(f)}) \quad = \quad f^c(x'_1, ..., x'_{ar(f^c)}), \tag{5}$$

where $x'_i \equiv x_j$ for some j, and f^c is a good or bad operation. Then A is called a copying *axiom.*

In the proofs in subsequent sections we will use the maximum number an argument is copied by a copy axiom.

Definition 2.3 *Let $cf : \Sigma_A \times N \rightarrow N$ be a function defined as follows. Let $f \in \Sigma_G$ and $i \in N$. If f has no copy axiom in E_A or $i > ar(f)$ then $cf(f,i) = 1$. If f has a copy axiom and $i \leq ar(f)$ then $cf(f,i) = p$, if the argument x_i occurs p times in $f^c(x'_1, ..., x'_{ar(f^c)})$. The maximum copy factor is defined as $max(\{cf(f,i) | f \in \Sigma_G \ \& \ i \in N\})$.*

Second all bad operations are linked to good operations with so-called distinctifying axioms.

Definition 2.4 *Let $f \in \Sigma_A$ be a bad operation and $A \in E_A$ an axiom of the form*

$$f(x_1, ..., x_{ar(f)}) \quad = \quad \Sigma_{i=1}^{p} f_i(x_1, ..., x_{ar(f)}), \tag{6}$$

where all f_i's are good operations. Then A is called a distinctifying *axiom.*

In proofs in subsequent sections we need the (maximum) distinctivity factor of operations.

[1]We try keep from the reader technical SOS Definitions which are not necessary to understand this paper. The three classes are respectively smooth-discarding-distinctive, smooth-discarding non distinctive, and non smooth and discarding, of which the Definitions can be found in [1].

Definition 2.5 *Let $df : \Sigma_A \rightarrow N$ be a function defined as follows. If f has no distinctivity axiom in E_A then $df(f) = 1$. If $f(x_1, ..., x_{ar(f)}) = \sum_{i=1}^{p} f_i(x_1, ..., x_{ar(f)})$ is a distinctivity axiom then $df(f) = p$. The maximum distinctivity factor* df *is defined as $max(\{df(f) | f \in \Sigma_A\})$.*

Finally for all (introduced) good operations, axioms are generated describing their interaction with the FINTREE operations. Intuitively the action axiom describes the result after the process has taken one step.

Definition 2.6 *Let $f \in \Sigma_A$ be a good operation and $A \in E_A$ an axiom of the form*

$$f(P_1, ..., P_{ar(f)}) = a.C[x_1, ..., x_{ar(f)}], \tag{7}$$

where P_i is of the form $a_i.x_i, x_i$ or 0 and x_i appears only in $C[x_1, ..., x_{ar(f)}]$ if $P_i \not\equiv 0$. Then A is called an action *axiom.*

The distributivity axiom describes the interaction between GSOS operations and the + operation.

Definition 2.7 *Let $f \in \Sigma_A$ be a good operation and $A \in E_A$ an axiom of the form*

$$\begin{aligned} f(x_1, ..., x_i + y_i, ..., x_{ar(f)}) = {} & f(x_1, ..., x_i, ..., x_{ar(f)}) \\ & + f(x_1, ..., y_i, ..., x_{ar(f)}). \end{aligned} \tag{8}$$

Then A is called a distributivity *axiom.*

The inaction axiom identifies the GSOS terms which have no behaviour with the constant 0.

Definition 2.8 *Let $f \in \Sigma_A$ be a good operation and $A \in E_A$ an axiom of the form*

$$f(P_1, ..., P_{ar(f)}) = 0, \tag{9}$$

where P_i is of the form $a_i.x_i$, $b_i.x_i+y_i$, x_i or 0. Then A is called an inaction *axiom.*

The peeling axiom "peeles" of parts of a term, which cannot influence its behaviour in any way.

Definition 2.9 *Let f be a good operation and $A \in E_A$ an axiom of the form*

$$f(P_1, ..., b_i.x_i + y_i, ..., P_{ar(f)}) \quad = \quad f(P_1, ..., y_i, ..., P_{ar(f)}) \qquad (10)$$

where P_j is of the form $a_j.x_j$ or x_j. Then A is called a **peeling** *axiom.*

Remark 2.1 The attentive reader may have noticed that we have specified in some detail what the syntactical form is of the different axioms of the axiomatisation. We have not specified which axioms are actually *included* in the axiomatisation produced by the alternative strategy. A precise description can be found in the original paper. However, twice we will make a subtle use of the "completeness" in some sense of the axiomatisation. In Lemma 4.7 we will use a fact for all smooth, distinctive and discarding operations f. A term t with principal operation f, *not* a FINTREE operation, with all its arguments in head normal form is an action, distributivity, (extra) inaction or (extra) peeling redex. In Lemma 4.11 we will use that the axioms presented in this subsection are sound with respect to strong bisimulation equivalence.

2.1 Well-foundedness

In [1] two different definitions of well-foundedness in the setting of GSOS are developed. Semantic well-founded GSOS systems are a class in which it is only possible to express finite, non-cyclic transition systems. We will translate these notions to the setting of axiomatisations.

Definition 2.10 *The axiomatisation of a GSOS system G is* **semantically well-founded** *if for every term $P \in T(\Sigma_G)$ there is a term $Q \in$ FINTREE so that $P = Q$ is provable.*

Since semantic well-foundedness of a GSOS system is in general not decidable, a subclass of GSOS systems is identified which is decidable. Therefore the notion of syntactic well-foundedness is introduced.

Definition 2.11 *The axiomatisation of a GSOS system G is* **syntactically well-founded** *if a function $w : \Sigma_A \rightarrow N$ exists so that the following conditions hold.*

- *For each operation f which is the principal operation of a copying or distinctifying axiom holds $w(f) = w(f^c)$ and $w(f) = w(f_i)$ for all i.*
- *For each action prefixing operation $a.$ holds $w(a.) \geq 1$.*
- *For each action axiom $f(P_1, ..., P_{ar(f)}) = a.C[x_1, ...x_{ar(f)}]$ the following conditions hold.*
  - *$W(C[x_1, ..., x_{ar(f)}]) \leq w(f)$, if $f(P_1, ..., P_{ar(f)})$ has an argument $P_i \equiv a_i.x_i$ for some i,*
  - *$W(C[x_1, ..., x_{ar(f)}]) < w(f)$, otherwise*

  where $W : \mathbb{T}(\Sigma_A) \to \mathbb{N}$ is given by

$$\begin{array}{lcl} W(x) & = & 0 \\ W(f(P_1, ..., P_{ar(f)})) & = & w(f) + \Sigma_{i=1}^{ar(f)} W(P_i). \end{array}$$

As is shown in the original article, the check for syntactic well-foundedness comes down to the (decidable) problem of solving a linear system of diophantine equations. It is proved there that syntactically well-founded GSOS systems which are also linear, are semantically well-founded.

Definition 2.12 *The axiomatisation of GSOS system G is linear iff for every action axiom $lhs = rhs \in E_A$ holds*

- *$FV(rhs) \subseteq FV(lhs)$ and,*
- *every variable in lhs appears not more then once in rhs.*

3 Term Rewriting with the Axioms

The subject of this article is the term rewriting properties of the axiomatisation produced by the alternative strategy of [1]. In the previous section the axiomatisation is described in the detail needed for our purposes.

Rewriting with axioms starts with orienting the axioms of the previous section from left-to-right, or right-to-left, for which we will use the term *rulifying.* The TRS we will use will consist of the axioms 3...10 oriented from left-to-right. We do not include an associativity and commutativity rewrite rule, for reasons we will explain later. The rewrite rules will be named after the axioms, i.e. *action* rewrite rules, if they stem from action axioms, *copy* rewrite rules, if they stem from copy axioms, etc.

For obvious reasons we have not oriented the FINTREE axioms $x+0=x$ and $x+x=x$ as $x \rightarrow x+0$ and $x \rightarrow x+x$, because then all terms lose the strong normalisation property: $s \rightarrow s+0 \rightarrow (s+0)+0 \ldots$. To maintain confluency we now have to introduce "extra" inaction and peeling rewrite rules, mimicking the effect of the combination of the two FINTREE and inaction and peeling rewrite rules, without unnecessary spoiling of rewriting properties.

Definition 3.1 *Let $f \in \Sigma_A$ be a good operation* [2]. *If f has an (extra) inaction rewrite rule of the form*

$$f(P_1, ..., b_i.x_i + y_i, ..., P_{ar(f)}) \rightarrow 0$$

then

$$f(P_1, ..., b_i.x_i, ..., P_{ar(f)}) \rightarrow 0$$

is an extra inaction *rewrite rule. If f has a (extra) peeling rewrite rule of the form*

$$f(P_1, ..., b_i.x_i + y_i, ..., P_{ar(f)}) \rightarrow f(P_1, ..., y_i, ..., P_{ar(f)})$$

then

$$f(P_1, ..., b_i.x_i, ..., P_{ar(f)}) \rightarrow f(P_1, ..., 0, ..., P_{ar(f)})$$

is an extra peeling *rewrite rule.*

The signature of a GSOS system, extended with the auxiliary operations plus the collection of oriented axioms we will call (the TRS of) *the rulified axiomatisation*, which we denote by $< \Sigma_A, R_A >$. Frequently we will use the rulified axiomatisation without the action rewrite rules, which we will refer to as the (TRS of) *the non action rewrite rules.*

We have not oriented the associativity and commutativity axiom for the simple reason that they cannot be oriented without losing the normalising

[2]We chose to be informal and not use a inductive definition or fixed point construction to define the extra inaction and peeling rewrite rules.

property: $s + t \rightarrow t + s \rightarrow s + t$ ... Therefore we cannot use "ordinary" term rewriting, but have to use the more complex rewriting modulo the commutativity of the $+$. However, it is well-known that this is not enough for the associativity still spoils the normalising property and so we have to work modulo the associativity of the $+$ as well. We will denote the commutativity and associativity axioms by ac and the equivalence class of s under ac as $[s]_{ac}$.

Definition 3.2 *If $s,t \in T(\Sigma)$ so that $s \in [s']_{ac}$, $t \in [t']_{ac}$ and $s' \rightarrow t'$ then $[s]_{ac} \rightarrow_{ac} [t]_{ac}$.*

In the sequel we will drop all subscripts ac.

4 Confluency and Weak Normalisation

In [1] is claimed that the rulified axiomatisation has nice term rewriting properties. The authors conjecture that the axiomatisation is strongly normalising and confluent for GSOS systems which are so-called *semantically well-founded.* The next small example disproves their claim: it shows a semantically well-founded GSOS system with an axiomatisation produced by the alternative strategy which is not strongly normalising. However, we will prove that a normalising strategy exists which implies weak normalisation and confluency.

Example 4.1 *Suppose G'' is the GSOS system which is the disjoint extension of FINTREE with two operations f and g and only one rule*

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{a} g(f(a.y))}$$

Then the action rule $f(a.y) \rightarrow a.g(f(a.y))$ and inaction rule $g(x) \rightarrow 0$ are obtained by rulifying the axioms produced by the alternative strategy. G'' is semantically well-founded, as can be verified easily, but $[f(a.0)]$ is not strongly normalising modulo ac, because
$[f(a.0)] \rightarrow [a.g(f(a.0))] \rightarrow [a.g(a.g(f(a.0)))]$ ...

The following strategy however is normalising for the rulified axiomatisation of a semantically well-founded GSOS system.

Definition 4.2 *The strategy Normalise contracts redexes in the following way.*

1. *Contract all non action redexes.*
2. *Contract the outermost action redex surrounded* [3] *by the least number of action prefixing operations.*
3. *Repeat until no redexes are left.*

Remark. Notice that the strategy is non deterministic, e.g. the non action redexes can be contracted in any order. Furthermore the strategy is Markov, in the sense that there is no need to keep the history of the reduction.

To prove that Normalise is indeed normalising, we will first prove that all non action rules decrease the weight of terms of some weight function. The weight function was inspired by the one used in [2].

Definition 4.3 *Let G be a GSOS system with rulified axiomatisation $< \Sigma_A, R_A >$. Let $w : \Sigma_G \longrightarrow N$ be a function which assigns a value to each operation. Let $\hat{}\ : \Sigma_A \longrightarrow \Sigma_G$ be the origin function which is defined as $\hat{f^c} = \hat{f_i} = \hat{f} = f \in \Sigma_G$. Let $\bar{i} : \Sigma_A \times N \longrightarrow N$ be a function which assigns the maximum copy factor cf to every not tested index. If f has an action rewrite rule so that the i-th argument of f in the left-hand side is x_i then $\bar{i}(f,i) = cf$, otherwise $\bar{i}(f,i) = 1$. The weight function $|-| : T(\Sigma_A) \longrightarrow N$ is defined as follows. Let $f \in \Sigma_A$ and $s, s', s_1, ..., s_{ar(f)} \in T(\Sigma_A)$, then*

$$\begin{array}{lcll}
|0| & = & 2 & \\
|s+s'| & = & |s| + |s'| & \\
|a.s| & = & (cf+2).|s| & \\
|f(s_1,...,s_{ar(f)})| & = & 1 + |f^c({s_1}',...,{s_{ar(f^c)}}')| & \textit{if } f \textit{ has a copy axiom} \\
|f(s_1,...,s_{ar(f)})| & = & 1 + \sum_{i=1}^{df(f)} |f_i(s_1,...,s_{ar(f)})| & \textit{if } f \textit{ has a distinctifying} \\
 & & & \textit{axiom} \\
|f| & = & w(\hat{f}) & \textit{if } f \not\equiv 0 \textit{ is a constant} \\
|f(s_1,...,s_{ar(f)})| & = & w(\hat{f})^{\sum_{i=1}^{ar(f)} \bar{i}(f,i).|s_i|} & \textit{otherwise,}
\end{array}$$

where ${s_i}' = s_j$ for some j.

[3]We say that a subterm or a redex t' of a term t is *surrounded* by an action prefixing operation iff t has a subterm of the form $a.C[t']$, $C[]$ possibly the trivial context.

Remark. That $|-|$ is well-defined can be verified with an argument using that the weight of a copy or a distinctifying redex, but a distinctifying redex is calculated from a set of terms which are no longer copy or distinctifying redexes.

The reader may be blurred by the abundance of implicit functions in the definition of $|-|$[4]. The implicit functions $\hat{\ }, \bar{i}$ and constants *cd*, *df* are determined solely by the rulified axiomatisation, as one can easily verify. The implicit function w depends only partially on the rulified axiomatisation.

In this section we will restrict ourselves to the weight functions $|-|$, where $w = w_0$ so that $w_0(f) \geq 3$ for all $f \in \Sigma_A$.

We will start with some easy facts about $|-|$.

Lemma 4.4 *Let G be a GSOS system and* $< \Sigma_A, R_A >$ *its rulified axiomatisation. If* $f \in \Sigma_A$ *and* $s \not\equiv 0 \in T(\Sigma_A)$ *then* $|s| > 2$.

Proof. Trivial, since $w_0(f) \geq 3$ for all f □

Lemma 4.5 *Let G be a GSOS system and* $< \Sigma_A, R_A >$ *its rulified axiomatisation. If s rewrites to t with a non action rewrite rule then* $|s| > |t|$.

Proof. Let $s \equiv C[s']$ where s' is a non action redex. We proceed by an induction on the complexity of $C[]$. We will present one case of the Base Step, which explains the use of w_0. The induction step is straightforward.

- Suppose $s \equiv f(s_1, ..., s_{ar(f)})$ is an (extra) inaction redex and $ar(f) > 0$, then

$$\begin{array}{lcl} |f(s_1, ..., s_{ar(f)})| & = & \{ \ Def. \ |-| \ \} \\ w_0(f)\Sigma_{i=1}^{ar(f)} |s_i| & > & \{ \ w_0(f) \geq 3, \ Lemma\ 4.4 \ \} \\ 2 & = & \{ \ Def. \ |-| \ \} \\ |0|. & & \end{array}$$

Lemma 4.6 *The TRS of non action rewrite rules is strongly normalising modulo ac for closed terms.*

Proof. It is a straightforward induction proof to verify that for all $s' \in [s]$, $|s'| = |s|$ for all functions w. So we can extend $|-|$ in a natural way to *ac* equivalence classes by $|[s]| = |s|$. Now with Lemma 4.5 the result follows □

[4]The Definition of $|-|$ could be simpler, if we chose to prove normalisation alone. We will use its full strength in proving strong normalisation in section 5.

Second we prove the head normalisation property of Normalise. We say that a term is in **head normal form** iff it is of the form $\Sigma_{i=1}^{n} a_i.P_i$.

Lemma 4.7 *Normalise is head normalising on closed terms for the rulified axiomatisation.*

Proof. Let s be a closed term of some GSOS system. We prove by an induction on the complexity of s that Normalise is head normalising for $[s]$. The Base Step is trivial. Let the Induction Hypothesis be that Normalise is head normalising for $[s_1], ..., [s_{ar(f)}]$. Let $s \equiv f(s_1, ..., s_{ar(f)})$ and the head symbol f be the only *marked* symbol. The (marked) copies of this f will be introduced solely by copy, distinctifying or distributivity rewrite rules.

First suppose that during the reduction an *infinite* number of marked f's is created. From Lemma 4.6 we get that after a finite number of rewrite steps all marked f's present in the reduct can no longer be head symbols of copy or distinctifying redexes. So to create an infinite number of marked f's an infinite number of distributivity redexes are contracted. With König's Lemma this implies that at least one of the arguments $s_1, ..., s_{ar(f)}$ has an infinite reduction, which is *not* head normalising. Contradiction with the Induction Hypothesis. Thus the number of marked f's created during the reduction with Normalise is *finite*.

Now suppose the number of marked f's created is finite. At any time in the reduction, the marked f's have a finite number of arguments, derived from $s_1, ..., s_{ar(f)}$. By the Induction Hypothesis these arguments (or their derivations, to be more precise) are reduced to head normal form. Consequently after finitely many steps the reduct is a sum of terms which are action or (extra) inaction redexes (see Remark 2.1). With Normalise all (extra) inaction redexes are rewritten to the head normal form $\mathbf{0}$. The order in which action redexes are chosen in step 2 of Normalise guarantees that all outermost action redexes are rewritten to head normal form □

Third a recursive application of the argument that Normalise is head normalising implies normalisation.

Theorem 4.8 *The strategy Normalise is normalising for the rulified axiomatisation of a semantically well-founded GSOS system.*

Proof. Suppose towards a contradiction that there is an *infinite* ac reduction with the strategy Normalise of some term s. Then by Lemma

4.7 we know that s is reduced to the form $\sum_{i=1}^{n} a_i.P_i$, where the P_i's are reduced to the same form ad infinitum. Now the reduct is not bisimilar to a FINTREE term. Contradiction □

Example 4.9 *As opposed to the reduction in Example 4.1 the contraction of the outermost inaction redex is not postponed indefinitely long by the strategy Normalise.*

$$\begin{array}{lll}
[f(a.0)] & \rightarrow & \{\ \mathit{action}\ \} \\
[a.g(f(a.0))] & \rightarrow & \{\ \mathit{inaction}\ \} \\
[a.0]. & &
\end{array}$$

To state our main result we need one Lemma describing uniqueness of normal forms. We omit the routine proof.

Lemma 4.10 *If* $s, t \in$ FINTREE *are strongly bisimilar and in normal form with respect to the* FINTREE *rewrite rules then* $[s] = [t]$.

The uniqueness of normal forms together with the weak normalising property gives us Church-Rosser.

Corollary 4.11 *The rulified axiomatisation of a semantically well-founded GSOS system is weakly normalising and Church-Rosser on closed terms.*

Proof. Weak normalisation follows immediately from the existence of a normalising strategy in Lemma 4.8. Now for the proof of Church-Rosser, suppose $[s] \rightarrow^* [s']$ and $[s] \rightarrow^* [s'']$. By the soundness of the axiomatisation (see Remark 2.1) we know that s and s' are still strongly bisimilar. Then $[s']$ and $[s'']$ are rewritten with Normalise to bisimilar FINTREE terms, not having FINTREE redexes. Now use Lemma 4.10 □

5 A Strongly Normalizing Subclass

The results in the previous section concerned semantically well-founded GSOS systems, which is a non decidable property of GSOS systems. In [1] a decidable subclass of semantically well-founded GSOS systems is defined called *syntactically well-founded.* In this section we will prove the strongly normalising property of axiomatisations of these systems given a proviso: We will demand that a GSOS system G which is syntactically well-founded with map w is (1) linear and has (2) $w(f) \geq 1$ for all $f \in \Sigma_G$ [5].

The proof of strong normalisation comes down to proving that *with* the proviso, action rules diminish the weight as well. In the previous section we saw that the collection of all non action rules respects a weight function $| - |_{w_0}$ [6]. However, in general $| - |_{w_0}$ is *not* respected by action rewrite rules. Due to the possible nested use of function symbols in the right-hand side of action rewrite rules, there is an exponentiation, which spoils a decreasing of weight. To overcome this problem, we will prove that based on the function w_0 a new "exponentiation proof" map e can be constructed, so that all rewrite rules respect $| - |_e$. The idea is that we compute for all operations in the left-hand side of action rules a new value, *based* on w_0. The maximum of the right-hand sides is then assigned. Because the values of operations depend on each other, we calculate the values recursively.

The map A calculates the maximum values of the right-hand sides with a minimal filling (i.e. 0's) for a given weight function.

Definition 5.1 *Let G be a GSOS system and $w : \Sigma_G \rightarrow N$ a function. Let $A : \Sigma_G \rightarrow N$ be a function defined as follows. Let $f \in \Sigma_G$, then*

$$A(f) \quad = \quad max(\{w(f)\} \cup \{|rhs[\vec{x} := \vec{0}]|_w | rhs \in Rhs_f\})$$

where Rhs_f is defined as

$$\{rhs \quad | \quad r : lhs \rightarrow rhs \in R_A \text{ is an action rule } \& \text{ the principal operation of } lhs \text{ is } g \ \& \ \hat{g} = f\}.$$

With the map w of a syntactically well-founded GSOS system and the auxiliary function A we can construct the "exponentiation proof" map e.

Definition 5.2 *Let G be a syntactically well-founded GSOS system with map w and $< \Sigma_A, R_A >$ its rulified axiomatisation. Let n be defined as $max(\{w(f) | f \in \Sigma_G\})$. The functions $e_0, ..., e_n : \Sigma_G \rightarrow N$ are defined inductively as follows. Let $f \in \Sigma_G$, then e_0 is defined as*

$$e_0(f) \quad = \quad (cf + 2).df.3.w(f)$$

[5]Part 1 of the proviso seems reasonable enough, since it was proved in [1] that linear, syntactically well-founded GSOS systems are semantically well-founded.

[6]In this section we will mention explicitly which w is used in the Definition of $| - |$.

and e_{i+1} *as*

$$\begin{array}{rcll} e_{i+1}(f) & = & e_i(f) & if\ w(f)\ \leq i \\ e_{i+1}(f) & = & 1 + max(\{A_{e_i}(g) | w(g) = w(f)\}) & otherwise. \end{array}$$

Then $e : \Sigma_G \rightarrow N$ *is defined as* e_n.

Remark. To see that e is well-defined it is enough to notice that Σ_G is finite.

The proof that action rules respect $| - |_e$ is very detailed. For clarity we have extracted the following technical fact needed in the proof.

Lemma 5.3 *Let* G *be a GSOS system and* $e : \Sigma_G \rightarrow N$ *a function so that* $e(f) \geq 2$ *for all* f. *Let* $C[]$ *be an* n*-ary context with* $n \geq 1$. *Let* $s_1, ..., s_n \in T(\Sigma)$, *then*

$$max(cf + 3, |C[\vec{0}]|_e)^{\sum_{i=1}^{n} |s_i|_e} \quad > \quad |a.C[s_1, ..., s_n]|_e.$$

Proof. By an induction on the complexity of $C[]$.

Base step. Let $C[]$ be the trivial context. Then

$$\begin{array}{lll} (cf + 3)^{|s_1|} & > & \{Lemma\ 4.4\} \\ (cf + 2).|s_1| & = & \\ |a.s_1|. & & \end{array}$$

Induction Hypothesis. If $C_i[]$ is of smaller complexity then $C[]$, then $max(cf + 3, |C[\vec{0}]|_e)^{\sum_{i=1}^{n} |s_i|} > |a.C[s_1, ..., s_n]|_e|$.

Induction Step. Let $C[s_1, ...s_n] \equiv f(C_1[s_1, ...s_n], ..., C_{ar(f)}[s_1, ...s_n])$ and let the Induction Hypothesis hold for $C_1[], ..., C_{ar(f)}[]$. We only treat the cases that $|C_i[\vec{0}]|_e \geq cf + 3$, the other cases are analogous. Now distinguish three cases.

1. Suppose $f \equiv a.$. Then

$$\begin{array}{lll} |a.C_1[\vec{0}]|^{\sum_{i=1}^{n} |s_i|} & = & \\ (cf + 2)^{\sum_{i=1}^{n} |s_i|}.C_1[\vec{0}]^{\sum_{i=1}^{n} |s_i|} & > & \{I.H., n \geq 1\} \\ (cf + 2).|a.C_1[s_1, ...s_n]| & = & \\ |a.a.C_1[s_1, ..., s_n]|. & & \end{array}$$

2. Suppose $f \equiv +$. Then

$$\begin{array}{lll}
|C_1[\vec{0}]| + |C_2[\vec{0}]|^{\sum_{i=1}^{n}|s_i|} & \geq & \{Binomium, n \geq 1\} \\
|C_1[\vec{0}]|^{\sum_{i=1}^{n}|s_i|} + |C_2[\vec{0}]|^{\sum_{i=1}^{n}|s_i|} & \geq & \{I.H\} \\
|a.C_1[s_1, ..., s_n]| + |a.C_2[s_1, ..., s_n]| & > & \\
|a.(C_1[s_1, ..., s_n] + C_2[s_1, ..., s_n])|. & &
\end{array}$$

3. Suppose $f \not\in \{a., +\}$. Then

$$\begin{array}{lll}
|f(C_1[\vec{0}], ..., C_{ar(f)}[\vec{0}])|^{\sum_{i=1}^{n}|s_i|} & = & \\
\cancel{(}e(\hat{f})^{|C_1[\vec{0}]|+...+|C_{ar(f)}[\vec{0}]|})^{\sum_{i=1}^{n}|s_i|} & = & \\
e(\hat{f})^{|C_1[\vec{0}]|^{\sum_{i=1}^{n}|s_i|}+...+|C_{ar(f)}[\vec{0}]|^{\sum_{i=1}^{n}|s_i|}} & > & \{I.H\} \\
e(\hat{f})^{|a.C_1[s_1,...,s_n]|+...+|a.C_{ar(f)}[s_1,...,s_n]|} & = & \\
e(\hat{f})^{(cf+2).|C_1[s_1,...,s_n]|+...+(cf+2).|C_{ar(f)}[s_1,...,s_n]} & > & \\
|a.f(C_1[s_1, ..., s_n], ..., C_{ar(f)}[s_1, ..., s_n])| & & \square
\end{array}$$

Lemma 5.4 *Let G be a syntactically well-founded GSOS system with map w and $< \Sigma_A, R_A >$ its rulified axiomatisation. If $s, t \in T(\Sigma_A)$ and s rewrites to t with an action rule of R_A, then $|s|_e > |t|_e$.*

Proof. Let $s \equiv C[s']$ where s' is an action redex. The result follows with an induction on the complexity of $C[]$. As before we omit the whole proof and present only the Base Step, the rest of the induction is trivial. Let $r : lhs \longrightarrow rhs \in R_A$ be an action rewrite rule so that s is a head redex of r. By construction of action rewrite rules, lhs is of the form

$$f(P_1, ..., P_{ar(f)}),$$

where P_i is of the form $a_i.x_i, x_i$ or $\mathbf{0}$. Now let P be the set of indexes i so that $P_i \equiv a_i.x_i$, Q so that $P_i \equiv x_i$ and R the rest (i.e. the $\mathbf{0}$'s). Let (3) P' be defined as $P \cap FV(rhs)$[7] and likewise Q' as $Q \cap FV(rhs)$. To prove that e is indeed the requested function, distinguish two cases.

1. Suppose rhs contains an operation g so that (1) $w(g) = w(\hat{f})$ (notice that $g \in \Sigma_G$). Because r is syntactically well-founded and $w(f) \not> w(g)$, rhs is of the form

$$a.g(x'_1, ..., x'_{ar(g)}),$$

[7] Of course the set of *indexes* of the free variables is meant here.

where $x'_i \equiv x_j$ for some j (maybe once, maximally cf times !). Now it is crucial to realise that (2) for *at least one* index i, P_i is of the form $a_i.x_i$, which we will call i'. Now let $s_1, ..., s_{ar(f)} \in T(\Sigma_A)$, then [8]

$$\begin{array}{lll}
|f(P_1, ..., P_{ar(f)})|_e & = & \\
e(\hat{f})^{\sum_{i\in P} |a_i.x_i|_e + \sum_{i \in Q} cf.|x_i|_e + \#R.|0|_e} & \geq & \\
e(\hat{f})^{\sum_{i\in P}(cf+2).|x_i|_e + \sum_{i\in Q} cf.|x_i|_e} & > & \{(2), |x_i|_e \geq 2\} \\
e(\hat{f}).e(\hat{f})^{\sum_{i\in P-\{i'\}}(cf+2).|x_i|_e + (cf+1).|x_{i'}|_e + \sum_{i\in Q} cf.|x_i|_e} & > & \left\{e(\hat{f}) > df.(cf+2)\right\} \\
df.(cf+2).e(\hat{f})^{\sum_{i\in P} cf.|x_i|_e + \sum_{i\in Q} cf.|x_i|_e} & \geq & \{(3)\} \\
df.(cf+2).e(\hat{f})^{\sum_{i\in P'} cf.|x_i|_e + \sum_{i\in Q'} cf.|x_i|_e} & \geq & \{(1), df \geq df(g)\} \\
df(g).(cf+2).e(g)^{\sum_{i\in P'} cf.|x_i|_e + \sum_{i\in Q'} cf.|x_i|_e} & \geq & \\
|a.g(x'_1, ..., x'_{ar(g)})|_e. & &
\end{array}$$

2. Suppose the right-hand side rhs of r contains *no* operation g so that $w(g) = w(\hat{f})$. Because G is syntactically well-founded, rhs contains also no operations g', with $w(g') > w(\hat{f})$ and so by Definition of e, (4) $e(\hat{f}) \geq 1 + |rhs[\vec{x} := \vec{0}]|_e$. Now the only interesting case is if f is *not* a constant. Let $s_1, ..., s_{ar(f)} \in T(\Sigma_A)$, then [9]

$$\begin{array}{lll}
|f(P_1, ..., P_{ar(f)})|_e & = & \\
e(\hat{f})^{\sum_{i\in P} |a_i.x_i|_e + \sum_{i\in Q} cf.|x_i|_e + \#R.|0|_e} & \geq & \\
e(\hat{f})^{\sum_{i\in P'\cup Q'} |x_i|_e} & > & \{(4), Lemma\ 5.3\} \\
|rhs[x_1, ..., x_{ar(f)}]|_e & & \Box
\end{array}$$

Now we have a weight function which is respected by all rewrite rules.

Lemma 5.5 *Let G be a linear, syntactically well-founded GSOS system with map w, where $w(f) \geq 1$ for all $f \in \Sigma_G$. Let $< \Sigma_A, R_A >$ be its rulified axiomatisation. If s rewrites to t with a rewrite rule of R_A, then $|s|_e > |t|_e$.*

Proof. Notice that by construction of e, $e(f) \geq 3$ for all f. Now use Lemmas 4.5 and 5.4 $\Box$

This results extends to rewriting modulo associativity and commutativity of the $+$ using the argument of Lemma 4.6.

[8]We omit the explicit substitution $[x_1 = s_1, ..., x_{ar(f)} = s_{ar(f)}]$.

[9]Same as 8.

Theorem 5.6 *Let G be a syntactically well-founded GSOS system with map w, where $w(f) \geq 1$ for all $f \in \Sigma_G$. Then the rulified axiomatisation of G is strongly normalising modulo ac on closed terms.*

In the beginning of this section we presented the proviso for syntactically well-founded GSOS systems. Although we have no proof for this at the moment, we think that the demand $w(f) \geq 1$ can be dropped. Unfortunately our method of proving a decreasing of weight for action rewrite rules then fails: using 0 as a base in the exponentiation spoils the argument.

Example 5.7 *The condition $w(f) \geq 1$ for all f excludes the (linear, syntactically well-founded) GSOS system G which is the disjoint extension of FINTREE with the rule*

$$\frac{x \xrightarrow{a} y}{f(x) \xrightarrow{a} f(f(y))}$$

Then the rulified axiomatisation consists of the following rules,

$$\begin{array}{lcl} f(a.x) & = & a.f(f(x)) \\ f(x_1 + x_2) & = & f(x_1) + f(x_2) \\ f(0) & = & 0, \end{array}$$

which can be proved SN.

Conjecture 5.8 *The rulified axiomatisation of a linear, syntactically well-founded GSOS system is strongly normalising modulo ac for closed terms.*

Acknowledgement

I thank Frits Vaandrager for the idea, critical reading and the stimulating discussions on this paper. I thank Wan Fokkink for his careful proof-reading.

References

[1] L. Aceto, B. Bloom, and F.W. Vaandrager. Turning SOS rules into equations. In *Proceedings 7th Annual Symposium on Logic in Computer Science,* Santa Cruz, California, pages 113–124. IEEE Computer Society Press, 1992. Full version available as CWI Report CS-R9218, June 1992, Amsterdam. To appear in the LICS 92 Special Issue of *Information and Computation.*

[2] G.J. Akkerman and J.C.M. Baeten. Term rewriting analysis in process algebra. Report P9006, Programming Research Group, University of Amsterdam, 1990.

[3] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced: Preliminary report. In *Conference Record of the* 15th *ACM Symposium on Principles of Programming Languages,* San Diego, California, pages 229–239, 1988. Full version available as Technical Report 90-1150, Department of Computer Science, Cornell University, Ithaca, New York, August 1990. Accepted to appear in *Journal of the ACM.*

[4] J.F. Groóte and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, October 1992.

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

[6] J.W. Klop. Term rewriting systems. In *Handbook of Logic in Computer Science, Volume II.* Oxford University Press, 1992. To appear.

[7] Huimin Lin. PAM: A Process Algebra Manipulator (Version 1.0). Report 4/93, Computer Science, University of Sussex, Brighton, February 1993.

[8] E. Madelaine, R. de Simone, and D. Vergamini. *ECRINS V2-1, USERS MANUAL*, 1989.

[9] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer-Verlag, 1980.

[10] R. Milner. *Communication and Concurrency.* Prentice-Hall International, Englewood Cliffs, 1989.

[11] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[12] R. de Simone. Higher-level synchronising devices in MEIJE–SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

[13] C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. Computing Science Notes 93/18, Eindhoven University of Technology, 1993.