Information and Computation 207 (2009) 120–145

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Generating priority rewrite systems for OSOS process languages☆

Irek Ulidowski [a,*], Shoji Yuen [b]

[a] Department of Computer Science, University of Leicester, University Road, Leicester LE1 7RH, UK
[b] Information Engineering Department, Nagoya University, Furo-cho, Chikuka-ku, Nagoya 464-8601, Japan

ARTICLE INFO

Article history:
Received 15 February 2007
Revised 4 October 2007
Available online 7 November 2008

ABSTRACT

We propose an algorithm for generating a Priority Rewrite System (PRS) for an arbitrary process language in the OSOS format such that rewriting of process terms is sound for bisimulation and head normalising. The algorithm is inspired by a procedure which was developed by Aceto, Bloom and Vaandrager and presented in *Turning SOS rules into equations* [L. Aceto, B. Bloom, F.W. Vaandrager, Turning SOS rules into equations, Information and Computation 111 (1994) 1–52.].

For a subclass of OSOS process languages representing finite behaviours the PRSs that are generated by our algorithm are strongly normalising (terminating) and confluent, where termination is proved using the dependency pair and dependency graph techniques. Additionally, such PRSs are complete for bisimulation on closed process terms modulo associativity and commutativity of the choice operator of CCS. We illustrate the usefulness of our results, and the benefits of rewriting with priorities in general, with several examples.

1. Introduction

Structural Operational Semantics (SOS) [31,3] is a method for assigning operational meaning to operators of process languages. The main components of SOS are transition rules, or simply SOS rules, which describe how the behaviour of a composite process depends on the behaviour of its component processes. A general syntactic form of transition rules is called a *format*. A process operator is in a format if all its SOS rules are in the format, and a process language, often abbreviated by PL, is in a format if all its operators are in the format. Many general formats have been proposed and a wealth of important results and specification and verification methods for PLs in these formats have been developed [3].

The motivation and rationale for working with general PLs (via their formats) rather than with specific PLs such as, for example, CCS [27], CSP [20] and ACP [10], is that one can define and use new application-specific operators and features on top of the standard PLs [11,12]. In order to realise the potential of general PLs software tools need to be developed. Such tools would accept general PLs as input languages and perform tasks such as simulation, model checking and equivalence checking, refinement and testing. Several such tools already exist. For example, we can use *Process Algebra Compiler* [36] to change the input PL to the *Concurrency Workbench of New Century* [16]. The Process Algebra Compiler can accept any general PL in the *positive GSOS* format [13] and it produces a "front-end" to the Concurrency Workbench for that PL.

Alternatively, we can utilise the existing term rewriting and theorem prover software tools to analyse properties of processes of general PLs. To this end several procedures for automatic derivation of axiom systems and term rewriting systems for PLs in several formats were proposed [2,1,14,38,8]. The present paper continues this research, particularly on the generation of term rewriting systems for bisimulation originated by Aceto et al. [2] and Bosscher [14], and extends and

☆ An extended abstract of this work appeared at CONCUR 2003 as [39].

* Corresponding author.
E-mail address: iu3@mcs.le.ac.uk (I. Ulidowski).

0890-5401/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2008.10.002

generalises it further. We propose a new procedure for deriving *Priority Rewrite Systems* for bisimulation. Having considered many examples of operators we believe that our work delivers the following improvements: (a) priority rewrite rules are no more complicated and are sometimes simpler than the rewrite rules produced from the axioms as in [2,1,14], (b) they employ no more than and sometimes fewer auxiliary operators (see Remark 5.2), and (c) the priority order that we use increases the effectiveness of term rewriting by reducing the number of critical pairs and thus reducing the nondeterminism inherent in rewriting (see Section 7). We work with *Ordered* SOS PLs [40], or OSOS PLs for short, instead of the GSOS PLs [13] which have the same expressiveness [40]. The proposed procedure generates term rewriting systems with a priority order on rewrite rules instead of axiom systems or ordinary term rewriting systems as in [2,14]. We illustrate this with an example. Consider the priority operator "θ" [6]. For a given irreflexive partial order $\gg$ on actions process $\theta(p)$ is a restriction of p such that, in any state of p, action a can happen only if no action b with $b \gg a$ is possible in that state. If $B_a = \{b \mid b \gg a\}$, then θ is defined in a natural fashion by the following GSOS rules, one for each action a, where expressions of the form $X \stackrel{b}{\not\rightarrow}$ in the premises are called *negative premises*:

$$\frac{X \stackrel{a}{\rightarrow} X' \quad \{X \stackrel{b}{\not\rightarrow}\}_{b \in B_a}}{\theta(X) \stackrel{a}{\rightarrow} \theta(X')}$$

The second procedure in [2], also described in [1], produces the following axioms for θ where the basic operators of CCS, namely "+", prefixing and "**0**", are used. Since a typical rule for θ may have several copies of the argument X in the premises an auxiliary binary operator "$\triangle$", defined below, is used [2].

$$\frac{X \stackrel{a}{\rightarrow} X' \quad \{Y \stackrel{b}{\not\rightarrow}\}_{b \in B_a}}{X \triangle Y \stackrel{a}{\rightarrow} \theta(X')}$$

The following axioms for θ consist of the axiom that makes copies of X and uses the auxiliary operator $\triangle$, and the axioms for $\triangle$ consisting of the distributivity axiom, peeling axioms and inaction axioms:

$$\begin{array}{rcll}
\theta(X) & = & X \triangle X & \\
(X+Y)\triangle Z & = & X\triangle Z + Y \triangle Z & \\
a.X \triangle (b.Y + Z) & = & a.X \triangle Z & \text{if } \neg(b > a) \\
a.X \triangle (b.Y + Z) & = & \mathbf{0} & \text{if } b > a \\
a.X \triangle \mathbf{0} & = & a.\theta(X) & \\
\mathbf{0} \triangle X & = & \mathbf{0} &
\end{array}$$

The priority operator can be defined equivalently, and perhaps more intuitively, by *positive* GSOS rules equipped with an *ordering* to represent the priority order on actions: the ordering has the corresponding effect to negative premises in rules. This is the idea behind the *Ordered* SOS format [40]. The rules for the OSOS version of θ are, one for each a,

$$\frac{X \stackrel{a}{\rightarrow} X'}{\theta(X) \stackrel{a}{\rightarrow} \theta(X')} \; r_a$$

and the ordering $>$ is such that $r_b > r_a$ whenever $b \gg a$. The ordering prescribes that rule r_a can be applied to derive transitions of $\theta(p)$ if no higher priority rule, e.g. r_b, can be applied to $\theta(p)$. This suggests an axiomatisation procedure: derive the axioms from the SOS rules similarly to [2,1], and then "order" them appropriately according to the ordering on the SOS rules. More precisely, we orientate the axioms from left to right to obtain the rewrite rules, then define a *priority* ordering which is an irreflexive partial order (irreflexive and transitive) on the rewrite rules, and then introduce a new type of rewrite rule to deal with the priority ordering. What we obtain is an example of a *Priority Rewrite System*, or PRS for short, originated by Baeten, Bergstra, Klop and Weijland [7]. Our procedure generates the following PRS for the operator θ. We have one rewrite rule θ^b_{pr} for each pair of a and b such that $b \gg a$, and one θ^a_{act} rule for each action a:

$$\begin{array}{lrcl}
\theta^b_{pr}: & \theta(a.X + b.Y + Z) & \rightarrow & \theta(b.Y + Z) \\
\theta_{dn}: & \theta(X + \mathbf{0}) & \rightarrow & \theta(X) \\
\theta_{ds}: & \theta(X + Y) & \rightarrow & \theta(X) + \theta(Y) \\
\theta^a_{act}: & \theta(a.X) & \rightarrow & a.\theta(X) \\
\theta_{nil}: & \theta(X) & \rightarrow & \mathbf{0}
\end{array}$$

The priority ordering on the rewrite rules is defined as follows: $\theta^b_{pr} \succ \theta_{dn}$ for all rewrite rules θ^b_{pr}, $\theta_{dn} \succ \theta_{ds}$ and $\{\theta_{ds},\} \cup \{\theta^a_{act} \mid \text{all } a\} \succ \theta_{nil}$. We can represent this ordering more pictorially. Below, $r \succ r'$ if and only if there is an arrow from r to r':