ORDERED SOS RULES AND WEAK BISIMULATION

IAIN PHILLIPS
Department of Computing, Imperial College, London SW7 2BZ.

IREK ULIDOWSKI
Research Institute for Mathematical Sciences, Kyoto University, Kyoto, Japan

One of the main reasons for developing formats of rules with negative antecedents, for example the GSOS format, was to enable natural description of process operators like the sequential composition, priority and broadcast parallel. We propose an alternative method for defining these and similar operators, which does not rely on rules with negative antecedents. Instead, we use ordinary rules but with an ordering on them which indicates the order of their application. We show that operators which are definable by our method preserve a version of weak bisimulation.

1 Introduction

Structured Operational Semantics (SOS) is considered to be the standard method for defining operational meaning of process operators in an arbitrary process language. It was originated by Milner for CCS [1, 2] and formalised by Plotkin [3]. The meaning of each operator on processes is given by a set of transition rules. Each of the rules describes how the behaviour of a process, constructed with the operator and some subprocesses, depends on the behaviour of these subprocesses. For example, the following is one of the rules for $\boxplus$, the CSP external choice operator,

$$\frac{X \xrightarrow{a} X'}{X \boxplus Y \xrightarrow{a} X'}.$$

The rule allows to infer that a process $a.\mathbf{0} \boxplus b.\mathbf{0}$ may perform action a since its subprocess $a.\mathbf{0}$ may itself perform a.

Process operators can be classified accordingly to the form of rules defining their operational meaning. A format of rules is a collection of forms of rules. We say that an operator is in a certain format if its rules belong to that format. Most of the popular process operators are in the De Simone format [4].

However, there is a number of process operators which cannot be adequately defined by De Simone rules alone. The most important examples of these are the sequential composition, priority and broadcast parallel operators. These operators are usually defined by rules with *negative antecedents*, i.e. expressions like $X \stackrel{a}{\nrightarrow}$. Consider, for example, the rules for the sequential composition operator ;

$$\frac{X \xrightarrow{\alpha} X'}{X;Y \xrightarrow{\alpha} X';Y} \qquad\qquad \frac{\forall\alpha.\, X \stackrel{\alpha}{\nrightarrow} \quad Y \xrightarrow{\beta} Y'}{X;Y \xrightarrow{\beta} Y',}$$

where α and β are any visible or silent actions. Given a process $\mathbf{0}; a.\mathbf{0}$ we can deduce, using the second rule for a and the fact that $a.\mathbf{0}$ can perform a, that the process itself can perform a.

Transition rules with negative antecedents as means to give operational semantics to general process operators were originally considered by Bloom, Istrail and Meyer [5, 6, 7]. They proposed a general format of rules called the GSOS format. Groote [22] proposed a more general format called the *ntyft/nxyft* format.

Another issue concerning the formats of transition rules is the use of silent actions. The original De Simone and GSOS formats treated both silent and visible actions in the same way, i.e. as visible. This resulted in many difficulties when *weak equivalences* over De Simone or GSOS process languages were considered. The problem was studied by Bloom [8, 9], Vaandrager [10] and the second author [11, 12]. He proposed the ISOS format, which offers a sound and intuitive treatment of silent actions as well as use of negative antecedents and copying.

In this paper we will only consider those formats of rules which treat silent actions as invisible; for example the formats in [8, 10, 11, 9, 12]. A number of important results were established for these formats. Firstly, certain weak equivalences were shown to be preserved by all process operators in these formats [10, 9, 12]. Secondly, (completed) trace congruences with respect to these formats were discovered [10, 11]. Thirdly, algorithms for generating complete axiomatisations of some weak equivalences were developed [13, 14, 15].

The main conceptual contribution of the paper is a new method for defining process operators, including the sequential composition and priority operators, by transition rules with *no* negative antecedents. Our method is based on a simple idea of *ordering* the transition rules for each operator. The behaviour of a process can then be determined by examining the rules of its main operator, starting with the rules highest in the ordering and then considering the rules lower in the ordering. As a result, a rule lower in the ordering can only be applied if none of the rules above it can. Intuitively, this has the effect of applying a rule with negative antecedents. A simple example is given to illustrate our method. Consider a priority operator θ, which gives a a priority over b, defined by the following rules.

$$\frac{X \xrightarrow{a} X'}{\theta(X) \xrightarrow{a} \theta(X')} \qquad\qquad \frac{X \stackrel{a}{\not\rightarrow} \quad X \xrightarrow{b} X'}{\theta(X) \xrightarrow{b} \theta(X')}$$

We easily deduce $\theta(b.\mathbf{0} \boxplus c.\mathbf{0}) \xrightarrow{b} \theta(\mathbf{0})$. Employing our method we only need the first rule and a similar rule for b (but with no negative antecedents) together with an ordering which puts the first rule above the second. Clearly, the first rule cannot be applied to the process $\theta(b.\mathbf{0} \boxplus c.\mathbf{0})$. But, since $b.\mathbf{0} \boxplus c.\mathbf{0} \xrightarrow{b} \mathbf{0}$, we obtain $\theta(b.\mathbf{0} \boxplus c.\mathbf{0}) \xrightarrow{b} \theta(\mathbf{0})$ by the second rule.

The main technical problem we address is how to combine orderings on rules with the intended use of silent actions. It is known that all ISOS operators, which include most of popular process operators, preserve a version of weak bisimulation [12]. However, it is not difficult, as will be shown in the forthcoming sections, to come up with a positive ISOS operator and a careless ordering on its rules such that the operator does not preserve weak bisimulation. In order to avoid such operators we propose a number of conditions on orderings such that if they are satisfied then the defined operator is guaranteed to preserve weak bisimulation.

2 Preliminaries

Let Var be a countable set of variables ranged by $X, X_i, Y, Y_i, \ldots$. A signature Σ is a set of pairs (f, n), where $n \in \mathbb{N}$ is an arity of f. An element $(g, 0)$ of Σ is called a constant symbol and it is abbreviated as g. The set of open terms over Σ with variables in $V \subseteq \mathsf{Var}$, denoted by $\mathbb{T}(\Sigma, V)$, is ranged by $P, Q, \ldots$. The set of closed terms, written as $\mathrm{T}(\Sigma)$, is ranged by $p, q, \ldots$. $var(T)$ denotes the set of variables appearing in a term T. We use $\equiv$ for syntactic equality. Σ context with n holes $C[X_1, \ldots, X_n]$ is a member of $\mathbb{T}(\Sigma, \{X_1, \ldots, X_n\})$. A *proper* context is a context where each variable occurs at most once. If $t_1, \ldots, t_n$ are Σ terms then $C[t_1, \ldots, t_n]$ is a term obtained by substituting X_i by t_i, $1 \leq i \leq n$. An equivalence $\sim$ on $\mathrm{T}(\Sigma)$ is the congruence if for all Σ contexts $C[\boldsymbol{X}]$ we have $\boldsymbol{t} \sim \boldsymbol{t}'$ implies $C[\boldsymbol{t}] \sim C[\boldsymbol{t}']$. Similarly, an operator (f, n) preserves $\sim$ if for all n-ary vectors of closed terms $\boldsymbol{t}$ and $\boldsymbol{t}'$ with $\boldsymbol{t} \sim \boldsymbol{t}'$ we have $f(\boldsymbol{t}) \sim f(\boldsymbol{t}')$. A *substitution* ρ is a mapping from Var to $\mathbb{T}(\Sigma)$. The substitution ρ extends to a mapping $\mathbb{T}(\Sigma) \rightarrow \mathbb{T}(\Sigma)$ in a standard way. $\mathsf{Act} = \mathsf{Vis} \cup \{\tau\}$, ranged by $\alpha, \ldots$, is a finite set of actions, where Vis is the set of visible actions, ranged by $a, b, \ldots$, and τ is a silent, internal action which is not in Vis.

In this paper we only consider *weak* equivalence on processes. The main characteristic of weak equivalences is the type of process behaviour which is used to determine which processes are related—only visible behaviour is taken into account. In particular this means that only the visible actions of processes are recorded and the silent actions are abstracted away. If one is interested in using weak equivalences and if one searches for a general format of rules with a suitable treatment of silent actions then, we believe, it is reasonable to have the following minimal requirement concerning the form of rules: *All operators defined in the format should preserve the chosen weak equivalence.*

The first attempt to accommodate silent actions in formats of rules is due to Bloom [8]. The important idea of silent rules, called there *patience* rules, was introduced there but the overall treatment of silent actions was not completely satisfactory: one could define operators which would count the number of silent actions processes can perform. Vaandrager in [10] modified the original De Simone format [4] by using the silent rules and showed that failure equivalence is preserved by all operators definable in the modified format. Then, the second author introduced in [11] the ISOS format—a general format of rules with negative antecedents and a suitable treatment of silent actions. He proved that ISOS operators preserve a version of weak bisimulation [12] (sometime called delay bisimulation) and refusal simulation preorder [11]. Also, similar modifications to those used in the ISOS format and some new ideas were applied to positive GSOS rules in [9]. As a result a number of formats were proposed and a number of weak equivalences, including rooted branching and weak bisimulations, were shown to be preserved by the operators in these formats. Finally, the second author showed in [15] that testing preorder of De Nicola & Hennessy [16] is preserved by De Simone (with silent rules) operators.

In this paper we shall consider only those rules which belong to (a slightly extended) ISOS format as described in [11, 12]. The ISOS rules have two distinctive forms. We will use the following notational agreement: we write $X \stackrel{\widetilde{\alpha}}{\rightarrow} X$ instead of $X \stackrel{\tau}{\not\rightarrow}\stackrel{\alpha}{\not\rightarrow}$.

Definition 1 An *action rule* has the following form

$$\frac{\{\ X_i \stackrel{a_i}{\rightarrow} X_i'\ \}_{i \in I} \qquad \{\ X_j \stackrel{\widetilde{\beta_{jk}}}{\rightarrow} X_j\ \}_{j \in J, k \in K_j}}{f(X_1, \ldots, X_n) \stackrel{\alpha}{\rightarrow} C[\boldsymbol{Y}]},$$

where X_i, X_i' are all different variables, I and $J \subseteq \{1, \ldots, n\}$, all K_j are finite subsets of natural numbers and $C[\boldsymbol{Y}]$ contains at most the variables $Y_1, \ldots, Y_n$, where $Y_i = X_i'$ if $i \in I$ and $Y_i = X_i$ otherwise. Note that each Y_i may appear in $C[\boldsymbol{Y}]$ a number of time or not at all. The notions of the *operator* of the rule, the *action, positive* and *negative antecedents* and the *consequent* are as usual. The action rule is *positive* if $J = \emptyset$ or, in other words, if it does not have any negative antecedents. We say that (f, n) *tests* its m-th argument and the

m-th argument for (f,n) is *active* if there is an action rule for (f,n) with X_m in the antecedents. The antecedents and the consequent of a rule r are written as $ante(r)$ and $cons(r)$ respectively.

A *silent rule*, or τ-rule, for an operator f is any of the following rules for $1 \leq i \leq n$, hereafter written as τ_{f_i} or simply as τ_i when f is known from the context,

$$\frac{X_i \xrightarrow{\tau} X_i'}{f(X_1,\ldots,X_i,\ldots,X_n) \xrightarrow{\tau} f(X_1,\ldots,X_i',\ldots,X_n).}$$

An action rule is τ-producing if $\alpha = \tau$, otherwise it is Vis-producing. Let $rules(f)$ be the set of all action rules and τ-rules with $f(\boldsymbol{X})$ in the consequent. The ith argument is *active* in some action rule r if X_i appears in the antecedents of r, and it is active in a set $S \subseteq rules(f)$ if it is active in some action rule in S. Denote the set of such i by $active(S)$, and write $active(f)$ instead of $active(rules(f))$. Let $tau(S)$ be $\{i \mid \tau_i \in S\}$, and write $tau(f)$ instead of $tau(rules(f))$. A τ-rule τ_i is *associated* with f if the ith argument of f is active.

Definition 2 With the notations as in the previous definition, a set of rules $S \subseteq rules(f)$ is in the *ISOS* format (*positive ISOS* format) if it satisfies $active(S) = tau(S)$ (and all its action rules are positive). A process operator is an ISOS (positive ISOS) operator if the set of its defining rules is in the ISOS (positive ISOS) format. An ISOS (positive ISOS) process language is a triple $(\Sigma, \mathsf{A} \cup \{\tau\}, R)$, where Σ is a finite set of ISOS (positive ISOS) operators, $\mathsf{A} \subseteq \mathsf{Vis}$, A is finite and R is a finite set of rules for operators in Σ.

Most of the operators in commonly used process languages are positive ISOS except, for example, for the CCS and ACP choice operator $+$ and the ACP left-merge operator. They are not positive ISOS because they do not have the associated τ-rules. But, there are positive ISOS versions of these operators which are appropriate for most of the purposes. For example, the CSP external choice operator $\boxplus$ is one of the positive ISOS choice operators. It is defined by the following rule schema

$$\frac{X \xrightarrow{a} X'}{X \boxplus Y \xrightarrow{a} X'} \qquad\qquad \frac{Y \xrightarrow{a} Y'}{X \boxplus Y \xrightarrow{a} Y'}$$

together with two appropriate τ-rules. Also, the CSP internal choice operator and the mixed choice defined in [12] are positive ISOS operators.

Given an ISOS process language $(\Sigma, \mathsf{A} \cup \{\tau\}, R)$, a *labeled transition system* can be defined in a standard way; the details can be found, for example, in [5, 17, 7, 12]. A labeled transition system for the above language is a structure

$(\mathrm{T}(\Sigma), \mathsf{A} \cup \{\tau\}, \rightarrow)$, where $\mathrm{T}(\Sigma)$ is a set of *process terms* or *processes* and $\rightarrow \subseteq \mathrm{T}(\Sigma) \times \mathsf{A} \cup \{\tau\} \times (\mathrm{T}(\Sigma)$ is the unique *transition relation* generated by the language. We will often call this transition system the basic transition system for a given language. Expressions of the form $t \xrightarrow{\alpha} t'$, where $t, t' \in \mathrm{T}(\Sigma)$ and $\alpha \in \mathsf{A} \cup \{\tau\}$, are called *transitions*. They are varied by T, T'. With the notation as above, we say that transition $t \xrightarrow{\alpha} t'$ is *valid* (*holds*) in $\rightarrow$ if $(t, \alpha, t') \in \rightarrow$.

We will use some standard notation: for $(p, a, q) \in \rightarrow$ we write $p \xrightarrow{a} q$ and interpret it as process p performs a and in doing so becomes q. The abbreviations we will use are $p \xrightarrow{a}$ for $\exists q \in \mathrm{T}(\Sigma).\ p \xrightarrow{a} q$ and $p \stackrel{a}{\nrightarrow}$ for $\neg \exists q \in \mathrm{T}(\Sigma).\ p \xrightarrow{a} q$. Also, $p \stackrel{\tau}{\Rightarrow} q$ means $p = q$ or $\exists p_1, \ldots, p_n$ such that $p = p_1$, $p_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} p_n$ and $p_n = q$. Using the above we write $p \stackrel{a}{\Rightarrow} q$ to mean $p \stackrel{\tau}{\Rightarrow} p' \xrightarrow{a} q$ for some p'.

Next, we define a version of weak bisimulation relation which is based on weak bisimulation (or observation equivalence) discussed in [18, 19, 12]. This version is often called *delay* bisimulation [20, 21].

Definition 3 Assume the basic transition systems for a given ISOS process language with the set of actions $\mathsf{A} \cup \{\tau\}$.

1. A binary relation R on processes is a *weak bisimulation* if pRq implies, for all $\alpha \in \mathsf{A} \cup \{\tau\}$,

$$\begin{array}{l} \forall p'.\ p \xrightarrow{\alpha} p' \ \text{ implies } \ (\exists q'.\ q \stackrel{\alpha}{\Rightarrow} q' \ \text{ and } \ p'Rq'), \\ \forall q'.\ q \xrightarrow{\alpha} q' \ \text{ implies } \ (\exists p'.\ p \stackrel{\alpha}{\Rightarrow} p' \ \text{ and } \ p'Rq'). \end{array}$$

2. $p \approx q$ if there exists a weak bisimulation R such that pRq.

The difference between our definition of weak bisimulation and the standard notion as defined, for example, in [2] and denoted here by $\approx_B$ lies in the meaning of $p \stackrel{a}{\Rightarrow} q$: in [2] $p \stackrel{a}{\Rightarrow} q$ means $p \stackrel{\tau}{\Rightarrow} p' \xrightarrow{a} q' \stackrel{\tau}{\Rightarrow} q$, for some p', q'. It is worth noting that $\approx$ is strictly finer than $\approx_B$ since the third τ-law is not sound for $\approx$. One of the reasons for choosing this version of weak bisimulation is that the standard version is not preserved by many simple positive ISOS operators. In fact, there are much simpler De Simone operators [15], for example action refinement-like operators, which fail to preserve the standard version.

Finally, we state the congruence result for $\approx$ and ISOS operators.

Proposition 4 Let $G = (\Sigma, \mathsf{A} \cup \{\tau\}, R)$ be any (positive) ISOS process language. Then $\approx$ is preserved by all Σ operators, that is for all $(f, n) \in \Sigma$ we have

$$\forall 1 \leq i \leq n.\ p_i \approx q_i \ \text{ implies } \ f(p_1, \ldots, p_n) \approx f(q_1, \ldots, q_n).$$

The proof is similar to that of Theorem 9.5.1 in [12], thus it is omitted.

3 Ordering on Rules

The antecedents of ISOS rules, unlike those of positive ISOS rules, refer to both the positive and negative behaviour of subprocesses. We will use a priority ordering on positive ISOS rules in order to achieve the effect of rules with negative antecedents. The ordering specifies the order in which the rules are to be applied when deriving transitions of terms.

Example 5 Consider an operator f such that $f(p)$ behaves like p except that whenever p cannot perform b then $f(p)$ can perform a new action $\tilde{b}$. Operator f can be defined by the following ISOS rules.

$$\frac{X \xrightarrow{\alpha} X'}{f(X) \xrightarrow{\alpha} f(X')} \qquad\qquad \frac{X \overset{\tau}{\not\rightarrow} \overset{b}{\not\rightarrow}}{f(X) \xrightarrow{\tilde{b}} f(X)}$$

But, f can also be defined as follows, for $\alpha \neq b$,

$$\frac{X \xrightarrow{\alpha} X'}{f(X) \xrightarrow{\alpha} f(X')}\ r_1 \qquad\qquad \frac{X \xrightarrow{\tau} X'}{f(X) \xrightarrow{\tau} f(X')}\ \tau_1$$

$$\frac{X \xrightarrow{b} X'}{f(X) \xrightarrow{b} f(X')}\ r_2 \qquad\qquad f(X) \xrightarrow{\tilde{b}} f(X) \qquad r_3$$

with the ordering $r_3 < \tau_1, r_3 < r_2$. We notice that, using ISOS rules, $f(p) \xrightarrow{\tilde{b}}$ if and only if $p \overset{\tau}{\not\rightarrow} \overset{b}{\not\rightarrow}$. Applying the positive ISOS rules with the above ordering we can deduce $f(p) \xrightarrow{\tilde{b}}$ only if rules τ_1 and r_2 cannot be applied for $f(p)$: that is when $p \overset{\tau}{\not\rightarrow}$ and $p \overset{b}{\not\rightarrow}$.

Let $<_f$ be a relation on $rules(f)$. $r <_f r'$ means that r has a lower priority than r' (and r' has a higher priority than r) when deriving the transitions of $f(\boldsymbol{p})$, any $\boldsymbol{p}$. By this we mean that a rule can only be applied when no rules with higher priority can be applied. We do not assume irreflexivity or transitivity of the relation $<_f$. Given positive ISOS language G $<_G$, or simply $<$ if G is clear from the context, is $\bigcup_{f \in \Sigma_G} <_f$.

Definition 6 An ordered process language is a pair $(G, <)$, where G is a positive ISOS process language and $<$ is the ordering on its rules as described above.

Next, we describe how to associate a transition relation with $(G, <)$. Our method is based on that originated in [22]. We will need a function $d : \mathrm{T}(\Sigma) \rightarrow$

$\mathbb{N}$ to specify the depth of operator names in a ground term over a signature Σ. Function d is defined inductively:

- $d(t) = 0$ if t is a constant,
- $d(g(t_1, \ldots, t_r)) = 1 + \max\{d(t_i) \mid 1 \leq i \leq r\}$.

The extension of d to transitions is defined as expected: $d(t \xrightarrow{*} t') = d(t)$. Also, the following will be useful: $higher(r) = \{r' \in rules(f) \mid r <_f r'\}$, and correspondingly $lower(r)$.

Definition 7 Given $(G, <)$ we associate a transition relation, $\rightarrow \subseteq \mathrm{T}(\Sigma) \times \mathsf{A} \cup \{\tau\} \times \mathrm{T}(\Sigma)$, defined as follows: $\rightarrow = \bigcup_{l<\omega} \rightarrow^l$, where transition relations $\rightarrow^l \subseteq \mathrm{T}(\Sigma) \times \mathsf{A} \cup \{\tau\} \times \mathrm{T}(\Sigma)$ are:

$$\begin{aligned} T \in \rightarrow^l \text{ iff } \quad & d(T) = l \;\; \& \;\; \exists\, r \in R,\ \rho : V \rightarrow \mathrm{T}(\Sigma).\ [\ \rho(cons(r)) = T \\ & \qquad \& \ \rho(ante(r)) \in \textstyle\bigcup_{k<l} \rightarrow^k \\ & \qquad \& \ \forall r' \in higher(r).\ \rho(ante(r')) \notin \textstyle\bigcup_{k<l} \rightarrow^k\ ]. \end{aligned}$$

Similarly as in [22] it is easy to show that the transition relation $\rightarrow$ associated with a given $(G, <)$ *agrees* with it. Intuitively, it means that all transitions in $\rightarrow$ are derivable by rules of G under the ordering $<$. In general, there may be a number of transition relations which agree with $(G, <)$. But, we claim that the transition relation associated with $(G, <)$ is the unique transition relation which agrees with $(G, <)$. The definitions of the basic and derived transition systems for $(G, <)$ are then straightforward.

4 Ordered Rules with Silent Actions

One may wonder if all process operators which are defined by ordered positive ISOS rules preserve our version of weak bisimulation. The answer is negative. We will give a number of examples to prove our claim. Moreover, these examples will help us to find a number of conditions such that all operators, which can be defined by rules satisfying these conditions, preserve $\approx$.

Notation: In order to shorten the presentation of the forthcoming conditions we will leave out the outermost universal quantifiers binding $f \in \Sigma$ and $r \in rules(f)$. Also, all the rules mentioned in the conditions, i.e. τ_i, r and r', are understood to belong to $rules(f)$.

Consider an interleaving parallel operator $\|$ defined as follows:

$$\frac{X \xrightarrow{a} X'}{X \| Y \xrightarrow{a} X' \| Y}\ r_{a*} \qquad \frac{X \xrightarrow{\tau} X'}{X \| Y \xrightarrow{\tau} X' \| Y}\ \tau_1$$

$$\frac{Y \xrightarrow{a} Y'}{X \| Y \xrightarrow{a} X \| Y'}\ r_{*a} \qquad \frac{Y \xrightarrow{\tau} Y'}{X \| Y \xrightarrow{\tau} X \| Y'}\ \tau_2$$

The names of rules and rule schemas accompany them on the right. The ordering is $r_{a*} > \tau_2$. Now we have $b \approx \tau b$ and $a \| b \stackrel{b}{\Rightarrow}$, but $a \| \tau b \stackrel{b}{\not\Rightarrow}$.

Thus, we need to find a conditions on the ordering relation which would not allow the above ordering. The first condition might be as follows.

$$\text{if } r \text{ is a } \tau\text{-rule then } \mathit{higher}(r) = \emptyset$$

The intuition is that τ-rules should not have lower priority. But, although the condition is quite natural it is also restrictive. For example, let f be a binary operator. When deriving the behaviour of $f(p, q)$ one may want to give priority to the first subprocess (like with the sequential composition operator). This may result in some rules associated with the first subprocess being above τ_2. We can allow such orderings provided that all the rules, which are above τ_2, are above all the rules with active second argument. This can be written as follows.

$$\begin{array}{lll} \text{if } r \in \mathit{higher}(\tau_i) & \text{then} & \mathit{lower}(\tau_i) = \emptyset \text{ and} \\ & & \forall r'.\ \text{if } i \in \mathit{active}(r') \text{ then } r \in \mathit{higher}(r') \end{array} \qquad (1)$$

The condition implies that no action rule can be above any of its τ-rules.

There are operators, definable by positive ISOS rules satisfying condition (1), which do not preserve weak bisimulation. Consider the priority operator θ which gives d priority over b. It is defined by the following rule schema, for $\alpha \in \{b, d, \tau\}$,

$$\frac{X \xrightarrow{\alpha} X'}{\theta(X) \xrightarrow{\alpha} \theta(X').}$$

The action rules are denoted by r_b and r_d, and the τ-rule is τ_1. Suppose $r_d > r_b$. Let $p = b \| \tau d$ and $q = b \| d$, where $\|$ is the usual interleaved parallel composition operator. Then, $p \approx q$ and $\theta(p) \stackrel{b}{\Rightarrow}$, but $\theta(q) \stackrel{b}{\not\Rightarrow}$. To repair this we additionally require $\tau_1 > r_b$.

This example leads to the following condition.

$$\text{if } i \in active(higher(r)) \text{ then } \tau_i \in higher(r) \tag{2}$$

The intuition here is that in order to apply the rule r we need to make sure that no other rules with higher priority (and thus their τ-rules) can be applied.

Next, we claim our main result.

Theorem 8 Let $(G, <)$ be an ordered process language, where $G = (\Sigma, \mathsf{A} \cup \{\tau\}, R)$ is any positive ISOS process language and the ordering $<$ satisfies conditions (1) and (2). Then $\approx$ defined over the derived transition system for $(G, <)$ is preserved by all Σ operators.

The theorem is proved similarly as, for example, in [17, 6, 12]. Due to the lack of space the proof is omitted.

5 Applications

Sequential composition is often defined by rules with negative antecedents as, for example in [5, 7]. However, it can be defined by the following ordered rule schemas, where $a, b \in \mathsf{A}$:

$$\frac{X \xrightarrow{a} X'}{X;Y \xrightarrow{a} X';Y}\ r_{a*} \qquad\qquad \frac{Y \xrightarrow{b} Y'}{X;Y \xrightarrow{b} Y'}\ r_{*b}$$

together with the appropriate τ-rules τ_1 and τ_2. The ordering on the rules satisfies $r_{a*} > r_{*b}, \tau_2$ for all a, b and the conditions (1) and (2).

Priorities are employed in process languages to represent such phenomena as time-outs and interrupts. For a given partial order $<$ on visible actions $\theta(X)$ is a restriction of X such that action a can only happen only if no $b > a$ is possible. This can be defined operationally by the first of the following rules.

$$\frac{X \xrightarrow{\alpha} X' \quad X \stackrel{\beta}{\nrightarrow}\ (\forall \beta > \alpha)}{\theta(X) \xrightarrow{\alpha} \theta(X')} \qquad\qquad \frac{X \xrightarrow{a} X'}{\theta(X) \xrightarrow{a} \theta(X')}\ r_a$$

However, θ can be defined by a set of ordered positive ISOS rules, like the second rule above for each $a \in \mathsf{A}$ and the τ-rule τ_1. The ordering requires $r_a < r_b$ whenever $a < b$, and $\tau_1 > r_b$ whenever there exist an action rule r_a such that $r_a \in higher(r_b)$. In [23], the first author surveyed a number of approaches to priorities in process algebras. He showed how these approaches might be fitted into a common operational framework based on ordered rules.

The **broadcast parallel** operator $\|$ can be defined by the following positive ISOS rules

$$\frac{X \xrightarrow{a} X' \quad Y \xrightarrow{a} Y'}{X \| Y \xrightarrow{a} X' \| Y'} \qquad \frac{X \xrightarrow{a} X'}{X \| Y \xrightarrow{a} X' \| Y} \qquad \frac{Y \xrightarrow{a} Y'}{X \| Y \xrightarrow{a} X \| Y'},$$

denoted by r_{aa}, r_{a*} and r_{*a} respectively, with the ordering $r_{aa} > r_{a*}, r_{*a}$.

6 Conclusions

We have introduced a new method for operationally defining arbitrary process operators which are normally defined by transition rules with negative antecedents. Our method relies on rules without negative antecedents but we equip the rules with an ordering. It specifies the order in which the rules are applied when deriving transitions of process terms. We have also proposed two conditions on rule orderings which guarantee that all operators definable by rules satisfying these conditions preserve weak bisimulation.

References

1. R. Milner. *A Calculus for Communicating Systems.* Springer-Verlag, Berlin, 1980. LNCS 92.
2. R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.
3. G. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, 1981.
4. R. de Simone. Higher-level synchronising devices in MEIJE-SCCS. *Theoretical Computer Science*, Vol. 37, pp. 245–267, 1985.
5. 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, 1988.
6. B. Bloom. *Ready simulation, bisimulation, and the semantics of CCS-like languages.* PhD thesis, MIT, 1990.
7. B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *Journal of ACM*, Vol. 42(1),, 1995. Also appeared as Technical Report TR 90-1150, Cornell, 1990.
8. B. Bloom. Strong process equivalence in the presence of hidden moves. Preliminary report, MIT, 1990.
9. B. Bloom. Structural operational semantics for weak bisimulations. *Theoretical Computer Science*, Vol. 146, pp. 27–68, 1995. Also appeared as Technical Report TR 93-1373, Cornell, 1993.

10. F.W. Vaandrager. On the relationship between process algebra and input/output automata. In *Proceedings of the 6th Annual IEEE Symposium on Logic in Computer Science*, Amsterdam, 1991.
11. I. Ulidowski. Equivalences on observable processes. In *Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science*, Santa Cruz, California, 1992.
12. I. Ulidowski. *Local Testing and Implementable Concurrent Processes.* PhD thesis, Imperial College, University of London, 1994.
13. L. Aceto, B. Bloom, and F.W. Vaandrager. Turning SOS rules into equations. In *Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science*, Santa Cruz, California, 1992.
14. I. Ulidowski. Axiomatisations of weak equivalences for De Simone languages. In I. Lee and S.A. Smolka, editors, *Proceedings of the 6th International Conference on Concurrency Theory* CONCUR'95, Philadelphia, 1995. Springer-Verlag. LNCS 962.
15. I. Ulidowski. Finite axiom systems for testing preorder and De Simone process languages. In *Proceedings of the 5th International Conference on Algebraic Methodology and Software Technology AMAST'96*, München, Germany, 1996.
16. R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, Vol. 34, pp. 83–133, 1984.
17. J.F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, Vol. 100, pp. 202–260, 1990.
18. R. Milner. A modal characterisation of observable machine behaviours. In G. Astesiano and C. Böhm, editors, *CAAP 81*, pp. 25–34, Berlin, 1981. Springer-Verlag. LNCS 112.
19. S. Abramsky. Observation equivalence as a testing equivalence. *Theoretical Computer Science*, Vol. 53, pp. 225–241, 1987.
20. W.P. Weijland. *Synchrony and Asynchrony in Process Algebra.* PhD thesis, University of Amsterdam, 1989.
21. R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. In G.X. Ritter, editor, *Information Processing 89*, pp. 613–618. Elsevier Science Publishers B. V. (North-Holland), 1989. To appear in JACM.
22. J.F. Groote. Transition system specifications with negative premises. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings of* CONCUR'90, Amsterdam, 1990. Springer-Verlag. LNCS 458.
23. I.C.C. Phillips. Approaches to priority in process algebras. Unpublished manuscript, 1994.