Conservative Extension in Structural Operational Semantics

Luca Aceto[*] Wan Fokkink[†] Chris Verhoef[‡]

1 Introduction

Structural operational semantics (SOS) [44] provides a framework to give an operational semantics to programming and specification languages. In particular, because of its intuitive appeal and flexibility, SOS has found considerable application in the study of the semantics of concurrent processes. SOS generates a labelled transition system, whose states are the closed terms over an algebraic signature, and whose transitions are supplied with labels. The transitions between states are obtained inductively from a transition system specification (TSS), which consists of so-called transition rules of the form $\frac{\text{premises}}{\text{conclusion}}$. A typical example of a transition rule is

$$\frac{x \xrightarrow{a} x'}{x\|y \xrightarrow{a} x'\|y}$$

stipulating that if $t \xrightarrow{a} t'$ holds for closed terms t and t', then so does $t\|u \xrightarrow{a} t'\|u$ for each closed term u. In general, validity (or invalidity) of the positive (or negative) premises of a transition rule, under a certain substitution, implies validity of the conclusion of this rule under the same substitution.

This column is an excerpt from [2], which gives an overview of recent results in the field of SOS, with an emphasis on existing formats for TSSs.

[*]**BRICS** (**B**asic **R**esearch **in** **C**omputer **S**cience), Centre of the Danish National Research Foundation, Department of Computer Science, Aalborg University, Fredrik Bajers Vej 7-E, DK-9220 Aalborg Ø, Denmark. Email: `luca@cs.auc.dk`. Partially supported by a grant from the Italian CNR, Gruppo Nazionale per l'Informatica Matematica (GNIM).

[†]CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. Email: `wan@cwi.nl`. Partially supported by a grant from the Nuffield Foundation.

[‡]University of Amsterdam, Department of Computer Science, Programming Research Group, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands. Email: `x@wins.uva.nl`.

Each of these formats comes equipped with a rich body of results that are guaranteed to hold for any process calculus whose TSS is within that format.

Over and over again, process calculi such as CCS [40], CSP [47], and ACP [11] have been extended with new features, and the TSSs that provide the operational semantics for these process algebras were extended with transition rules to describe these features; see, e.g., [10] for a systematic approach. A question that arises naturally is whether or not the the original and the extended TSS induce the same transitions $t \stackrel{a}{\to} t'$ for closed terms t in the original domain. Usually it is desirable that an extension is operationally conservative, meaning that the provable transitions for an original term are the same both in the original and in the extended TSS.

Groote and Vaandrager [34, Thm. 7.6] proposed syntactic restrictions on a TSS, which automatically yield that an extension of this TSS with transition rules that contain fresh function symbols in their sources is operationally conservative. Bol and Groote [18, 33] supplied this conservative extension format with negative premises. Verhoef [49] showed that, under certain conditions, a transition rule in the extension can be allowed to have an original term as its source. D'Argenio and Verhoef [22, 23] formulated a generalization in the context of inequational specifications. Fokkink and Verhoef [25] relaxed the syntactic restrictions on the original TSS, and lifted the operational conservative extension result to higher-order languages. This column contains an exposition on existing conservative extension formats for SOS, and their applications with respect to term rewriting systems and completeness of axiomatizations.

Predicates in SOS semantics can be coded as binary relations [34]. Moreover, negative premises can often be expressed positively using predicates [9]. However, in the literature SOS definitions are often decorated with predicates and/or negative premises. For example, predicates are used to express matters like (un)successful termination, convergence, divergence [3], enabledness [14], maximal delay, and side conditions [42]. Negative premises are used to describe, e.g., deadlock detection [38], sequencing [17], priorities [7, 21], probabilistic behaviour [39], urgency [19], and various real [37] and discrete time [6, 35] settings. Since predicates and negative premises are so pervasive, and often lead to cleaner semantic descriptions for many features and constructs of interest, we deal explicitly with these notions.

The organization of this column is as follows. Sect. 2 gives an overview of the basics of SOS. Sect. 3 presents syntactic constraints to ensure that an extension of a TSS is operationally conservative. Sect. 4 and 5 contain applications of conservative extension in equational specification and term rewriting. Sect. 6 finishes with some conclusions.

2 Structural Operational Semantics

In this section we present the basic notions from process theory that are needed in the remainder of this column.

2.1 Labelled Transition Systems

We begin by reviewing the model of labelled transition systems [36, 44], which are used to express the operational semantics of many process calculi.

Definition 2.1 *A* labelled transition system *(LTS) is a quadruple*

$$(\mathsf{Proc}, \mathsf{Act}, \left\{ \xrightarrow{a} \mid a \in \mathsf{Act} \right\}, \mathsf{Pred}),$$

where:

- Proc *is a set of* states, *ranged over by* s*;*
- Act *is a set of* actions, *ranged over by* a, b*;*
- $\xrightarrow{a} \subseteq \mathsf{Proc} \times \mathsf{Proc}$ *for each* $a \in \mathsf{Act}$*. We use the more suggestive notation* $s \xrightarrow{a} s'$ *in lieu of* $(s, s') \in \xrightarrow{a}$*, and write* $s \stackrel{a}{\nrightarrow}$ *if* $s \xrightarrow{a} s'$ *for no state* s'*;*
- $P \subseteq \mathsf{Proc}$ *for every* $P \in \mathsf{Pred}$*. We write* sP *(resp.* $s\neg P$*) if state* s *satisfies (resp. does not satisfy) predicate* P*.*

Relations $s \xrightarrow{a} s'$ *and predicates* sP *in an LTS are called* transitions.

In what follows, an LTS is often identified with its collection of transitions. We trust that the meaning will always be clear from the context.

LTSs describe the operational behaviour of processes in great detail. In order to abstract away from irrelevant information on the way that processes compute, a wealth of notions of behavioural equivalence over the states of an LTS have been studied in the literature on process theory. A systematic investigation of these notions is presented in [28], where van Glabbeek presents the linear time/branching time spectrum. This lattice contains all the known behavioural equivalences over LTSs, ordered by inclusion.

2.2 Term Algebras

We start from a countably infinite set Var of variables, ranged over by x, y, z.

Definition 2.2 *A* signature Σ *is a set of* function symbols, *disjoint from* Var, *together with an* arity *mapping that assigns a natural number* $ar(f)$ *to each function symbol* f. *A function symbol of arity zero is called a* constant, *while function symbols of arity one and two are called* unary *and* binary, *respectively.*

The arity of a function symbol represents its number of arguments.

Definition 2.3 *The set* $\mathbb{T}(\Sigma)$ *of* (open) terms *over a signature* Σ, *ranged over by* t, u, *is the least set such that:*

- *each* $x \in \mathsf{Var}$ *is a term;*
- $f(t_1, \ldots, t_{ar(f)})$ *is a term, if* f *is a function symbol and* $t_1, \ldots, t_{ar(f)}$ *are terms.*

$\mathrm{T}(\Sigma)$ *denotes the set of* closed terms *over* Σ, i.e., *terms that do not contain variables.*

For a constant a, the term $a()$ is abbreviated to a. By convention, whenever we write a term-like phrase (e.g., $f(t,u)$), we intend it to be a term (i.e., f is binary).

Definition 2.4 *A* substitution *is a mapping* $\sigma : \mathsf{Var} \to \mathbb{T}(\Sigma)$. *A substitution is* closed *if it maps each variable to a closed term in* $\mathrm{T}(\Sigma)$. *A substitution extends to a mapping from terms to terms as usual; the term* $\sigma(t)$ *is obtained by replacing occurrences of variables* x *in* t *by* $\sigma(x)$.

2.3 Transition System Specifications

We proceed to introduce the main objects of study in the field of SOS, viz. transition system specifications.

Definition 2.5 *Let* Σ *be a signature, and let* t *and* t' *range over* $\mathbb{T}(\Sigma)$. *A* transition rule ρ *is of the form* H/α, *with* H *a collection of* positive premises $t \xrightarrow{a} t'$ *and* tP, *and of* negative premises $t \stackrel{a}{\nrightarrow}$ *and* $t\neg P$. *Moreover, the* conclusion α *is of the form* $t \xrightarrow{a} t'$ *or* tP. *The left-hand side of the conclusion is the* source *of* ρ. *A transition rule is* closed *if it does not contain variables.*

A transition system specification *(TSS) is a collection of transition rules. A TSS is* positive *if its transition rules do not contain negative premises.*

For the sake of clarity, we will often display transition rules in the form $\frac{H}{\alpha}$. The first systematic study of TSSs may be found in [48], while the first study of TSSs with negative premises appeared in [16].

We proceed to define when a transition is provable from a TSS. The following notion of a proof from [31] generalizes the standard definition (see, e.g., [34]) by allowing for the derivation of closed transition rules. The derivation of a transition α corresponds to the derivation of the closed transition rule H/α with $H = \varnothing$. The case $H \neq \varnothing$ corresponds to the derivation of α under the assumptions in H.

Definition 2.6 Positive literals *are transitions* $t \xrightarrow{a} t'$ *and* tP*, while* negative literals *are expressions* $t \stackrel{a}{\nrightarrow}$ *and* $t\neg P$*, where* t *and* t' *range over the collection of closed terms. A* literal *is a positive or negative literal.*

Definition 2.7 *Let* T *be a TSS. A* proof *of a closed transition rule* H/α *from* T *is a well-founded, upwardly branching tree whose nodes are labelled by literals, where the root is labelled by* α*, and if* K *is the set of labels of the nodes directly above a node with label* β*, then*

1. *either* $K = \varnothing$ *and* $\beta \in H$*,*
2. *or* K/β *is a closed substitution instance of a transition rule in* T*.*

If a proof of H/α *from* T *exists, then* H/α *is* provable *from* T*.*

2.4 Three-Valued Stable Models

In the presence of negative premises, the meaning of a TSS is sometimes ambiguous. For example, one can express that a transition holds if it does not hold. In order to associate an LTS to each TSS, we use the notion of a (least) three-valued stable model, introduced by Przymusinski [46] in logic programming. A three-valued stable model partitions the collection of transitions into three disjoint sets: the set C of transitions that are *certainly* true, the set U of transitions for which it is *unknown* whether or not they are true, and the set F of remaining transitions that are *false*.

Definition 2.8 *A disjoint pair of sets of transitions* $\langle C, U\rangle$ *constitutes a* three-valued stable model *for a TSS* T *if:*

- *a transition* α *is in* C *iff* T *proves a closed transition rule* N/α *where* N *contains only negative literals and* $C \cup U \models N$*;*
- *a transition* α *is in* $C \cup U$ *iff* T *proves a closed transition rule* N/α *where* N *contains only negative literals and* $C \models N$*.*

Each TSS has one or more three-valued stable models. For example, the TSS

$$\frac{a\neg P_2}{aP_1} \quad \frac{a\neg P_1}{aP_2}$$

has $\langle\{aP_1\},\varnothing\rangle$, $\langle\{aP_2\},\varnothing\rangle$, and $\langle\varnothing,\{aP_1,aP_2\}\rangle$ as its three-valued stable models. Each TSS T affords a unique (information-)least three-valued stable model $\langle C,U\rangle$, in the sense that the set U is maximal. Przymusinski [46] showed that this least three-valued stable model coincides with the so-called well-founded model that was introduced by van Gelder, Ross, and Schlipf [27] in logic programming. It is advocated in, e.g., [18, 32] that a TSS is meaningful if and only if its least three-valued stable model does not contain unknown transitions. In particular, each TSS that does not contain negative premises in its transition rules satisfies this requirement. The reader is referred to [31, 32] for more information on three-valued stable models and related notions.

3 Operational Conservative Extension

Often one wants to add new operators and rules to a given TSS. Therefore, a natural operation on TSSs is to take their componentwise union. The following definition stems from [34].

Definition 3.1 *Let T_0 and T_1 be TSSs whose signatures Σ_0 and Σ_1 agree on the arity of the function symbols in their intersection. We write $\Sigma_0 \oplus \Sigma_1$ for the union of Σ_0 and Σ_1. The* sum *of T_0 and T_1, notation $T_0 \oplus T_1$, is the TSS over signature $\Sigma_0 \oplus \Sigma_1$ containing the rules in T_0 and T_1.*

An *operational conservative extension* requires that an original TSS and its extension prove exactly the same closed transition rules that have only negative premises and an original closed term as their source. (This notion of an operational conservative extension is related to an equivalence notion for TSSs in [24, 32]: two TSSs are equivalent if they prove exactly the same closed transition rules that have only negative premises.)

Definition 3.2 *A TSS $T_0 \oplus T_1$ is an* operational conservative extension *of TSS T_0 if for each closed transition rule N/α such that:*

- *N contains only negative literals;*
- *the left-hand side of α is in $\mathrm{T}(\Sigma_0)$;*
- *N/α is provable from $T_0 \oplus T_1$;*

we have that N/α is provable from T_0.

3.1 Guaranteeing Operational Conservative Extension

We start by defining the notion of a source-dependent variable [25, 30], which will be an important ingredient of a rule format to ensure that an extension of a TSS is operationally conservative. In order to conclude that an extended TSS is operationally conservative over the original TSS, we need to know that the variables in the original transition rules are source-dependent. In the literature this criterion is sometimes neglected. For example, in [43] an extended TSS is considered in which each transition rule in the extension contains a fresh operator in its source, and from this fact alone it is concluded that the extension is operationally conservative. In general, however, this characteristic is not sufficient, as is shown in the next example.

Example: Let a and b be constants. Consider the TSS over signature $\{a\}$ that consists of the transition rule xP/aP. Extend this TSS with the TSS over signature $\{b\}$ that consists of the transition rule $\varnothing/bP$, which contains the fresh constant b in its source. The transition aP can be proved in the extended TSS, but not in the original one, so this extension is not operationally conservative. □

Definition 3.3 *The* source-dependent *variables in a transition rule ρ are defined inductively as follows:*

- *all variables in the source of ρ are source-dependent;*
- *if $t \xrightarrow{a} t'$ is a premise of ρ and all variables in t are source-dependent, then all variables in t' are source-dependent.*

A transition rule is source-dependent *if all its variables are.*

Note that the transition rule xP/aP from the example above is not source-dependent, because its variable x is not.

Thm. 3.4 below, which stems from [25], formulates sufficient criteria for a TSS $T_0 \oplus T_1$ to be an operational conservative extension of TSS T_0. We say that a term in $\mathbb{T}(\Sigma_0 \oplus \Sigma_1)$ is *fresh* if it contains a function symbol from $\Sigma_1 \backslash \Sigma_0$. Similarly, an action or predicate symbol in T_1 is *fresh* if it does not occur in T_0.

Theorem 3.4 *Let T_0 and T_1 be TSSs over signatures Σ_0 and Σ_1, respectively. Under the following conditions, $T_0 \oplus T_1$ is an operational conservative extension of T_0.*

1. *Each $\rho \in T_0$ is source-dependent.*

2. For each $\rho \in T_1$,

- *either the source of ρ is fresh,*
- *or ρ has a premise of the form $t \xrightarrow{a} t'$ or tP, where:*
  - $t \in \mathbb{T}(\Sigma_0)$;
  - *all variables in t occur in the source of ρ;*
  - *t', a, or P is fresh.*

3.2 Applications to TSSs

We apply Thm. 3.4 to some TSSs from the literature.

Basic Process Algebra with Empty Process The signature of *basic process algebra* with *empty process* [50], denoted by $\mathrm{BPA}_\epsilon(\mathsf{Act})$, consists of the following operators:

- a set Act of constants, representing indivisible behaviour;
- a constant ϵ, called empty process, representing successful termination;
- a binary operator $+$, called *alternative composition*, where a term $t_1 + t_2$ represents the process that executes either t_1 or t_2;
- a binary operator $\cdot$, called *sequential composition*, where a term $t_1 \cdot t_2$ represents the process that executes first t_1 and then t_2.

So the BNF grammar [5] for $\mathrm{BPA}_\epsilon(\mathsf{Act})$ is (with $a \in \mathsf{Act}$):

$$t ::= \ a \mid \epsilon \mid t_1 + t_2 \mid t_1 \cdot t_2 \ .$$

The intuition for the operators in $\mathrm{BPA}_\epsilon(\mathsf{Act})$ is formalized by the transition rules in Table 1 from [11], which constitute the TSS for $\mathrm{BPA}_\epsilon(\mathsf{Act})$. This TSS defines transitions $t \xrightarrow{a} t'$ to express that term t can evolve into term t' by the execution of action $a \in \mathsf{Act}$, and transitions $t\surd$ to express that term t can terminate successfully. The variables x, x', y, and y' in the transition rules range over the collection of closed terms, while the a ranges over Act.

The transition rules for $\mathrm{BPA}_\epsilon(\mathsf{Act})$ are source-dependent. For example, consider the third transition rule for sequential composition in Table 1:

$$\frac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y}$$

$$\frac{}{a \xrightarrow{a} \epsilon} \qquad \frac{}{\epsilon\surd}$$

$$\frac{x\surd}{x+y\surd} \quad \frac{x \xrightarrow{a} x'}{x+y \xrightarrow{a} x'} \quad \frac{y\surd}{x+y\surd} \quad \frac{y \xrightarrow{a} y'}{x+y \xrightarrow{a} y'}$$

$$\frac{x\surd \quad y\surd}{x \cdot y\surd} \quad \frac{x\surd \quad y \xrightarrow{a} y'}{x \cdot y \xrightarrow{a} y'} \quad \frac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y}$$

Table 1: Transition Rules for $\mathrm{BPA}_\epsilon(\mathsf{Act})$.

The variables x and y are source-dependent, because they occur in the source. Moreover, since x is source-dependent, the premise $x \xrightarrow{a} x'$ ensures that x' is source-dependent. Since the three variables x, x', and y in this transition rule are source-dependent, the transition rule is source-dependent.

Extending the Set of Actions Suppose that Act is extended to a set Act^{ext}. The TSS for $\mathrm{BPA}_\epsilon(\mathsf{Act}^{ext})$ is the TSS for $\mathrm{BPA}_\epsilon(\mathsf{Act})$ in Table 1, with the proviso that a ranges over Act^{ext}. We make the following observations concerning the extra transition rules in the TSS for $\mathrm{BPA}_\epsilon(\mathsf{Act}^{ext})$:

- the source of the transition rule $a \xrightarrow{a} \epsilon$ for $a \in \mathsf{Act}^{ext} \backslash \mathsf{Act}$ contains the fresh constant a;
- each transition rule concerning an a-transition of an alternative or sequential composition with $a \in \mathsf{Act}^{ext} \backslash \mathsf{Act}$, such as

$$\frac{x \xrightarrow{a} x'}{x+y \xrightarrow{a} x'}$$

  contains a premise with the fresh relation symbol $\xrightarrow{a}$ and with as left-hand side a variable from the source.

So, since the transition rules for $\mathrm{BPA}_\epsilon(\mathsf{Act})$ are source-dependent, it can be concluded from Thm. 3.4 that $\mathrm{BPA}_\epsilon(\mathsf{Act}^{ext})$ is an operational conservative extension of $\mathrm{BPA}_\epsilon(\mathsf{Act})$.

Priorities The language $\mathrm{BPA}_{\epsilon\theta}(\mathsf{Act})$ is obtained by adding the priority operator θ from [7] to $\mathrm{BPA}_{\epsilon}(\mathsf{Act})$. This function symbol assumes a partial order $<$ on Act. Intuitively, the process $\theta(t)$ is obtained by eliminating all transitions $s \xrightarrow{a} s'$ from the process t for which there is a transition $s \xrightarrow{b} s''$ with $a < b$. For example, if $a < b$ then $\theta(a+b)$ can execute the action b but not the action a. The TSS for $\mathrm{BPA}_{\epsilon\theta}(\mathsf{Act})$ consists of the transition rules in Tables 1 and 2, where the transition rules in the latter table capture the operational semantics of the priority operator. This TSS has a unique least three-valued stable model, which does not contain unknown transitions. (This follows from the fact that the TSS is *stratifiable* [33, 45].)

$$\frac{x\surd}{\theta(x)\surd} \qquad\qquad \frac{x \xrightarrow{a} x' \qquad x \stackrel{b}{\nrightarrow} \ \text{for } a < b}{\theta(x) \xrightarrow{a} \theta(x')}$$

Table 2: Transition Rules for the Priority Operator.

The two transition rules for the priority operator in Table 2 contain the fresh function symbol θ in their sources. So, since the transition rules for $\mathrm{BPA}_{\epsilon}(\mathsf{Act})$ are source-dependent, Thm. 3.4 implies that $\mathrm{BPA}_{\epsilon\theta}(\mathsf{Act})$ is an operational conservative extension of $\mathrm{BPA}_{\epsilon}(\mathsf{Act})$.

3.3 Implications for Three-Valued Stable Models

In [25] it was noted that the operational conservative extension notion as formulated in Def. 3.2 implies a conservativity property for three-valued stable models. If an extended TSS is operationally conservative over the original TSS, in the sense of Def. 3.2, and if a three-valued stable model of the extended TSS is restricted to those transitions that have an original term as left-hand side, then the result is a three-valued stable model of the original TSS.

Proposition 3.5 *Let $T_0 \oplus T_1$ be an operational conservative extension of T_0. If $\langle C, U\rangle$ is a three-valued stable model of $T_0 \oplus T_1$, then*

$$\begin{aligned} C' &\triangleq \{\alpha \in C \mid \textit{the left-hand side of } \alpha \textit{ is in } \mathrm{T}(\Sigma_0)\} \\ U' &\triangleq \{\alpha \in U \mid \textit{the left-hand side of } \alpha \textit{ is in } \mathrm{T}(\Sigma_0)\} \end{aligned}$$

is a three-valued stable model of T_0.

The converse of Prop. 3.5 also holds, in the following sense. If an extended TSS is operationally conservative over the original TSS, then each three-valued stable model of the original TSS can be obtained by restricting some three-valued stable model of the extended TSS to those transitions that have an original term as left-hand side.

Proposition 3.6 *Let $T_0 \oplus T_1$ be an operational conservative extension of T_0. If $\langle C, U\rangle$ is a three-valued stable model of T_0, then there exists a three-valued stable model $\langle C', U'\rangle$ of $T_0 \oplus T_1$ such that*

$$\begin{array}{rcl} C & \triangleq & \{\alpha \in C' \mid \textit{the left-hand side of } \alpha \textit{ is in } \mathrm{T}(\Sigma_0)\} \\ U & \triangleq & \{\alpha \in U' \mid \textit{the left-hand side of } \alpha \textit{ is in } \mathrm{T}(\Sigma_0)\} \ . \end{array}$$

Corollary 3.7 *Let $T_0 \oplus T_1$ be an operational conservative extension of T_0. If $\langle C, U\rangle$ is the least three-valued stable model of $T_0 \oplus T_1$, then*

$$\begin{array}{rcl} C' & \triangleq & \{\alpha \in C \mid \textit{the left-hand side of } \alpha \textit{ is in } \mathrm{T}(\Sigma_0)\} \\ U' & \triangleq & \{\alpha \in U \mid \textit{the left-hand side of } \alpha \textit{ is in } \mathrm{T}(\Sigma_0)\} \end{array}$$

is the least three-valued stable model of T_0.

4 Applications to Axiomatizations

This section discusses how operational conservative extension can be used to derive that an extension of an axiomatization is so-called axiomatically conservative, or that an axiomatization is complete or ω-complete with respect to some behavioural equivalence.

4.1 Axiomatic Conservative Extension

Definition 4.1 *A* (conditional) axiomatization *over a signature Σ consists of a set of (conditional) equations, called* axioms*, of the form $t_0 = u_0 \Leftarrow t_1 = u_1, \ldots, t_n = u_n$ with $t_i, u_i \in \mathbb{T}(\Sigma)$ for $i = 0, \ldots, n$.*

An axiomatization gives rise to a binary equality relation $=$ on $\mathbb{T}(\Sigma)$ thus:

- if $t_0 = u_0 \Leftarrow t_1 = u_1, \ldots, t_n = u_n$ is an axiom, and σ a substitution such that $\sigma(t_i) = \sigma(u_i)$ for $i = 1, \ldots, n$, then $\sigma(t_0) = \sigma(u_0)$;
- the relation $=$ is closed under reflexivity, symmetry, and transitivity;

- if f is a function symbol and $u = u'$, then

$$f(t_1, \ldots, t_{i-1}, u, t_{i+1}, \ldots, t_{ar(f)}) = f(t_1, \ldots, t_{i-1}, u', t_{i+1}, \ldots, t_{ar(f)}).$$

Definition 4.2 *Assume an axiomatization $\mathcal{E}$, together with an equivalence relation $\sim$ on $\mathrm{T}(\Sigma)$.*

1. *$\mathcal{E}$ is* sound *modulo $\sim$ iff $t = u$ implies $t \sim u$ for all $t, u \in \mathrm{T}(\Sigma)$.*

2. *$\mathcal{E}$ is* complete *modulo $\sim$ iff $t \sim u$ implies $t = u$ for all $t, u \in \mathrm{T}(\Sigma)$.*

Note that the above definitions of soundness and completeness, albeit standard in the literature on process algebras, are weaker than the classic ones in logic and universal algebra, where they are required to apply to arbitrary open expressions.

Definition 4.3 *Let $\mathcal{E}_0$ and $\mathcal{E}_1$ be axiomatizations over signatures Σ_0 and $\Sigma_0 \oplus \Sigma_1$, respectively. The axiomatization $\mathcal{E}_0 \cup \mathcal{E}_1$ is an* axiomatic conservative extension *of $\mathcal{E}_0$ if every equality $t = u$ with $t, u \in \mathrm{T}(\Sigma_0)$ that can be derived from $\mathcal{E}_0 \cup \mathcal{E}_1$ can also be derived from $\mathcal{E}_0$.*

The next theorem from [49] can be used to derive that an extension of an axiomatization is axiomatically conservative.

Theorem 4.4 *Let $\sim$ be an equivalence relation on $\mathrm{T}(\Sigma_0 \oplus \Sigma_1)$. Assume axiomatizations $\mathcal{E}_0$ and $\mathcal{E}_1$ over Σ_0 and $\Sigma_0 \oplus \Sigma_1$, respectively, such that:*

1. *$\mathcal{E}_0 \cup \mathcal{E}_1$ is sound over $\mathrm{T}(\Sigma_0 \oplus \Sigma_1)$ modulo $\sim$;*

2. *$\mathcal{E}_0$ is complete over $\mathrm{T}(\Sigma_0)$ modulo $\sim$.*

Then $\mathcal{E}_0 \cup \mathcal{E}_1$ is an axiomatic conservative extension of $\mathcal{E}_0$.

The idea behind Thm. 4.4 is as follows. Suppose that $t = u$ can be derived from $\mathcal{E}_0 \cup \mathcal{E}_1$ for $t, u \in \mathrm{T}(\Sigma_0)$. Soundness of $\mathcal{E}_0 \cup \mathcal{E}_1$ (requirement 1) yields $t \sim u$. Hence, completeness of $\mathcal{E}_0$ (requirement 2) yields that $t = u$ can be derived from $\mathcal{E}_0$.

Thm. 4.4 is particularly helpful in the case of an operational conservative extension of a TSS. Assume TSSs T_0 and T_1 over signatures Σ_0 and $\Sigma_0 \oplus \Sigma_1$, respectively, where $T_0 \oplus T_1$ is an operational conservative extension of T_0. Moreover, let $\sim$ be an equivalence relation on states in LTSs. Since the states in the LTSs associated with T_0 and $T_0 \oplus T_1$ are closed terms, the equivalence relation $\sim$ carries over to $\mathrm{T}(\Sigma_0)$ and $\mathrm{T}(\Sigma_0 \oplus \Sigma_1)$, respectively.

Owing to operational conservativity, the equivalence relation $\sim$ on $\mathrm{T}(\Sigma_0)$ as induced by T_0 agrees with this equivalence relation on $\mathrm{T}(\Sigma_0)$ as induced by $T_0 \oplus T_1$. Applications of Thm. 4.4 in process algebra, in the presence of an operational conservative extension of a TSS, are abundant in the literature; we give a typical example.

Example: Using Thm. 3.4 it is not hard to see that the process algebra ACP_θ [7] is an operational conservative extension of ACP. Baeten, Bergstra, and Klop introduced in *op. cit.* an axiomatization $\mathcal{E}_0$ that is complete over ACP modulo bisimulation equivalence, and an axiomatization $\mathcal{E}_0 \cup \mathcal{E}_1$ that is sound over ACP_θ modulo bisimulation equivalence. Hence, Thm. 4.4 says that $\mathcal{E}_0 \cup \mathcal{E}_1$ is an axiomatic conservative extension of $\mathcal{E}_0$. (In [7], fifteen pages were needed to prove this fact for the more general case of open terms, by means of a term rewriting analysis.) □

4.2 Completeness of Axiomatizations

The next theorem from [49] can be used to derive that an axiomatization is complete.

Theorem 4.5 *Let $\sim$ be an equivalence relation on $\mathrm{T}(\Sigma_0 \oplus \Sigma_1)$. Assume axiomatizations $\mathcal{E}_0$ and $\mathcal{E}_1$ over Σ_0 and $\Sigma_0 \oplus \Sigma_1$, respectively, such that:*

1. *$\mathcal{E}_0 \cup \mathcal{E}_1$ is sound over $\mathrm{T}(\Sigma_0 \oplus \Sigma_1)$ modulo $\sim$;*
2. *$\mathcal{E}_0$ is complete over $\mathrm{T}(\Sigma_0)$ modulo $\sim$;*
3. *for each $t \in \mathrm{T}(\Sigma_0 \oplus \Sigma_1)$ there is a $t' \in \mathrm{T}(\Sigma_0)$ such that $t = t'$ can be derived from $\mathcal{E}_0 \cup \mathcal{E}_1$.*

Then $\mathcal{E}_0 \cup \mathcal{E}_1$ is complete over $\mathrm{T}(\Sigma_0 \oplus \Sigma_1)$ modulo $\sim$.

The idea behind Thm. 4.5 is as follows. Let $t, u \in \mathrm{T}(\Sigma_0 \oplus \Sigma_1)$ with $t \sim u$. There exist terms $t', u' \in \mathrm{T}(\Sigma_0)$ such that $\mathcal{E}_0 \cup \mathcal{E}_1$ proves $t = t'$ and $u = u'$ (requirement 3). Soundness of $\mathcal{E}_0 \cup \mathcal{E}_1$ (requirement 1) yields $t \sim t'$ and $u \sim u'$, which together with $t \sim u$ implies $t' \sim u'$. Finally, owing to completeness of $\mathcal{E}_0$ over $\mathrm{T}(\Sigma_0)$ (requirement 2), we may derive $t' = u'$, and thus $t = t' = u' = u$.

Thm. 4.5 is particularly helpful in the case of an operational conservative extension of a TSS. Assume TSSs T_0 and T_1 over signatures Σ_0 and $\Sigma_0 \oplus \Sigma_1$, respectively, where $T_0 \oplus T_1$ is an operational conservative extension of T_0. Moreover, let $\sim$ be an equivalence relation on states in LTSs. Since the states in the LTSs associated with T_0 and $T_0 \oplus T_1$ are closed terms, the equivalence relation $\sim$ carries over to $\mathrm{T}(\Sigma_0)$ and $\mathrm{T}(\Sigma_0 \oplus \Sigma_1)$, respectively. Owing to operational conservativity, the equivalence relation $\sim$ on $\mathrm{T}(\Sigma_0)$ as induced by T_0 agrees with this equivalence relation on $\mathrm{T}(\Sigma_0)$ as induced by $T_0 \oplus T_1$. Applications of Thm. 4.5 in process algebra, in the presence of an operational conservative extension of a TSS, are abundant in the literature; we give a typical example.

Example: Using Thm. 3.4 it is not hard to see that the process algebra ACP [12] is an operational conservative extension of BPA_δ. Bergstra and Klop presented in *op. cit.* an axiomatization $\mathcal{E}_0$ that is complete over BPA_δ modulo bisimulation equivalence, and an axiomatization $\mathcal{E}_0 \cup \mathcal{E}_1$ that is sound over ACP modulo bisimulation equivalence, and that satisfies requirement 3 above. Hence, Thm. 4.5 says that $\mathcal{E}_0 \cup \mathcal{E}_1$ is complete over ACP modulo bisimulation equivalence. $\square$

For the precise proofs of Thm. 4.4 and Thm. 4.5, and for more detailed information such as generalizations of these results to axiomatizations based on inequalities, the reader is referred to [22, 23, 49].

4.3 ω-Completeness of Axiomatizations

Definition 4.6 *An axiomatization $\mathcal{E}$ over a signature Σ is* ω-complete *if an equation $t = u$ with $t, u \in \mathbb{T}(\Sigma)$ can be derived from $\mathcal{E}$ whenever $\sigma(t) = \sigma(u)$ can be derived from $\mathcal{E}$ for all closed substitutions σ.*

Milner [41] introduced a technique to derive ω-completeness of an axiomatization using SOS. The idea is to give a semantics to open (as opposed to closed) terms; in particular, variables need to be incorporated in the transition rules. See, e.g., [1, 29] for further applications of this technique in the realm of process algebra.

The next theorem can be used to derive that an axiomatization is ω-complete.

Theorem 4.7 *Let $\sim$ be an equivalence relation on $\mathbb{T}(\Sigma)$. Suppose that for all $t, u \in \mathbb{T}(\Sigma)$, $t \sim u$ whenever $\sigma(t) \sim \sigma(u)$ for all closed substitutions σ. If $\mathcal{E}$ is an axiomatization over Σ such that*

1. *$\mathcal{E}$ is sound over $\mathrm{T}(\Sigma)$ modulo $\sim$, and*

2. *$\mathcal{E}$ is complete over $\mathbb{T}(\Sigma)$ modulo $\sim$,*

then $\mathcal{E}$ is ω-complete.

The idea behind Thm. 4.7 is as follows. Let $t, u \in \mathbb{T}(\Sigma)$ and suppose that $\sigma(t) = \sigma(u)$ can be derived from $\mathcal{E}$ for all closed substitutions σ. Soundness of $\mathcal{E}$ over $\mathrm{T}(\Sigma)$ modulo $\sim$ (requirement 1) yields $\sigma(t) \sim \sigma(u)$ for all closed substitutions σ, so $t \sim u$. Then completeness of $\mathcal{E}$ over $\mathbb{T}(\Sigma)$ modulo $\sim$ (requirement 2) yields that $t = u$ can be derived from $\mathcal{E}$.

Assume a TSS T_0 over a signature Σ, and let T_0 be extended with a TSS T_1 that provides semantics to variables; thus, $T_0 \oplus T_1$ gives semantics to open terms in $\mathbb{T}(\Sigma)$. Suppose that $T_0 \oplus T_1$ is an operational conservative extension of T_0. Moreover, let $\sim$ be an equivalence relation on states in LTSs. Since the states in the LTSs associated with T_0 and $T_0 \oplus T_1$ are closed and open terms, respectively, the equivalence relation $\sim$ carries over to $\mathrm{T}(\Sigma_0)$ and $\mathbb{T}(\Sigma_0)$. Owing to operational conservativity, the equivalence relation $\sim$ on $\mathrm{T}(\Sigma_0)$ as induced by T_0 agrees with this equivalence relation on $\mathrm{T}(\Sigma_0)$ as induced by $T_0 \oplus T_1$. Applications of Thm. 4.7 in process algebra are abundant in the literature; we give a typical example.

Example: Extend the TSS for $\mathrm{BPA}_\epsilon(\mathsf{Act})$ in Table 1 by letting the symbol a range not only over Act, but also over Var. In a sense this means that variables are considered to be constants. This extension is operationally conservative, which follows from Thm. 3.4 by the following facts:

- the transition rules for $\mathrm{BPA}_\epsilon(\mathsf{Act})$ are source-dependent;
- the sources of transition rules $z \xrightarrow{z} \epsilon$ for variables z are fresh;
- each transition rule for alternative or sequential composition with z-transitions, such as

$$\frac{x \xrightarrow{z} x'}{x + y \xrightarrow{z} x'}$$

  contains a premise with the fresh relation symbol $\xrightarrow{z}$ and as left-hand side a variable from the source.

Furthermore, the following properties can be derived for the axiomatization $\mathcal{E}$ of $\mathrm{BPA}_\epsilon(\mathsf{Act})$ in [50]:

1. $\mathcal{E}$ is sound over $\mathrm{BPA}_\epsilon(\mathsf{Act})$ modulo bisimulation equivalence;
2. open terms t and u in $\mathrm{BPA}_\epsilon(\mathsf{Act})$ are bisimilar whenever $\sigma(t)$ and $\sigma(u)$ are bisimilar for all closed substitutions σ;
3. $\mathcal{E}$ is complete over the open terms in $\mathrm{BPA}_\epsilon(\mathsf{Act})$ modulo bisimulation.

So Thm. 4.7 implies that $\mathcal{E}$ is ω-complete over $\mathrm{BPA}_\epsilon(\mathsf{Act})$ modulo bisimulation equivalence. $\square$

5 Applications to Rewriting

This section discusses how operational conservative extension can be used to derive that an extension of a conditional term rewriting system is so-called rewrite conservative, or that a conditional term rewriting system is ground confluent.

5.1 Rewrite Conservative Extension

Definition 5.1 *Assume a signature Σ. A* conditional term rewriting system *(CTRS) [4, 13] over Σ consists of a collection of* rewrite rules

$$t_0 \to u_0 \Leftarrow t_1 \to^* u_1, \ldots, t_n \to^* u_n$$

with $t_i, u_i \in \mathbb{T}(\Sigma)$ for $i = 0, \ldots, n$.

Intuitively, a rewrite rule is a directed axiom that can only be applied from left to right. A CTRS induces a binary rewrite relation $\to^*$ on terms, similar to the way that an axiomatization induces an equality relation on terms (the only difference is that the rewrite relation is not closed under symmetry), thus:

- if $t_0 \to u_0 \Leftarrow t_1 \to^* u_1, \ldots, t_n \to^* u_n$ is a rewrite rule, and σ a substitution such that $\sigma(t_i) \to^* \sigma(u_i)$ for $i = 1, \ldots, n$, then $\sigma(t_0) \to^* \sigma(u_0)$;
- the relation $\to^*$ is closed under reflexivity and transitivity;
- if f is a function symbol and $u \to^* u'$, then

  $$f(t_1, \ldots, t_{i-1}, u, t_{i+1}, \ldots, t_{ar(f)}) \to^* f(t_1, \ldots, t_{i-1}, u', t_{i+1}, \ldots, t_{ar(f)}).$$

The definition of sum of TSSs (cf. Def. 3.1) applies equally well to CTRSs.

Definition 5.2 *Let R_0 and R_1 be CTRSs over signatures Σ_0 and $\Sigma_0 \oplus \Sigma_1$, respectively. $R_0 \oplus R_1$ is a* rewrite conservative extension *of R_0 if every rewrite relation $t \to^* u$ with $t \in \mathrm{T}(\Sigma_0)$ that can be derived from $R_0 \oplus R_1$ can also be derived from R_0.*

The conservative extension theorem for TSSs, Thm. 3.4, applies to CTRSs just as well; see [26] for more details. Note that the definition of source-dependent variables in transition rules, Def. 3.3, also applies to rewrite rules (where, in a rewrite rule $t_0 \to u_0 \Leftarrow t_1 \to^* u_1, \ldots, t_n \to^* u_n$, the expression $t_0 \to u_0$ is the conclusion and the $t_i \to^* u_i$ for $i = 1, \ldots, n$ are the premises).

Theorem 5.3 *Let R_0 and R_1 be CTRSs over signatures Σ_0 and $\Sigma_0 \oplus \Sigma_1$, respectively. Under the following conditions, $R_0 \oplus R_1$ is a rewrite conservative extension of R_0.*

1. *Each $\rho \in R_0$ is source-dependent.*
2. *For each $\rho \in T_1$,*
   - *either the source of ρ is fresh,*
   - *or ρ has a premise of the form $t \to t'$ where:*
     - *$t \in \mathbb{T}(\Sigma_0)$;*
     - *all variables in t occur in the source of ρ;*
     - *t' is fresh.*

5.2 Ground Confluence of CTRSs

Definition 5.4 *A CTRS is ground confluent if for all $t, t_0, t_1 \in \mathrm{T}(\Sigma)$ with $t \to^* t_0$ and $t \to^* t_1$ there is a $u \in \mathrm{T}(\Sigma)$ with $t_0 \to^* u$ and $t_1 \to^* u$.*

Ground confluence is an important property, for instance, to prove that an axiomatization is complete modulo some behavioural equivalence relation.

The next theorem from [49] can be used to derive that a CTRS is ground confluent. We say that a CTRS R is *sound* modulo an equivalence relation $\sim$ on $\mathrm{T}(\Sigma)$ if $t \to^* u$ implies $t \sim u$ for all $t, u \in \mathrm{T}(\Sigma)$.

Theorem 5.5 *Let $\sim$ be an equivalence relation on $\mathrm{T}(\Sigma_0 \oplus \Sigma_1)$. Assume CTRSs R_0 and R_1 over Σ_0 and $\Sigma_0 \oplus \Sigma_1$, respectively, such that:*

1. *$R_0 \oplus R_1$ is sound over $\mathrm{T}(\Sigma_0 \oplus \Sigma_1)$ modulo $\sim$;*
2. *if $t, t' \in \mathrm{T}(\Sigma_0)$ with $t \sim t'$, then there is a $u \in \mathrm{T}(\Sigma_0)$ such that $t \to^* u$ and $t' \to^* u$ can be derived from R_0;*
3. *for each $t \in \mathrm{T}(\Sigma_0 \oplus \Sigma_1)$ there is a $t' \in \mathrm{T}(\Sigma_0)$ such that $t \to^* t'$ can be derived from $R_0 \oplus R_1$.*

Then $R_0 \oplus R_1$ is ground confluent over $\mathrm{T}(\Sigma_0 \oplus \Sigma_1)$.

The idea behind Thm. 5.5 is as follows. Let $t \in \mathrm{T}(\Sigma_0 \oplus \Sigma_1)$ such that $t \to^* t_0$ and $t \to^* t_1$ can be derived from $R_0 \oplus R_1$. There exist $t_0', t_1' \in \mathrm{T}(\Sigma_0)$ such that $t_0 \to^* t_0'$ and $t_1 \to^* t_1'$ can be derived from $R_0 \oplus R_1$ (requirement 3). Soundness of $R_0 \oplus R_1$ (requirement 1) yields $t \sim t_0 \sim t_0'$ and $t \sim t_1 \sim t_1'$, so $t'_0 \sim t'_1$. Then there exists a $u \in \mathrm{T}(\Sigma_0)$ such that $t'_0 \rightarrow^* u$ and $t'_1 \rightarrow^* u$ (requirement 2). Hence, $t_0 \rightarrow^* u$ and $t_1 \rightarrow^* u$.

Thm. 5.5 is particularly helpful in the case of an operational conservative extension of a TSS. Assume TSSs T_0 and T_1 over signatures Σ_0 and $\Sigma_0 \oplus \Sigma_1$, respectively, where $T_0 \oplus T_1$ is an operational conservative extension of T_0. Moreover, let $\sim$ be an equivalence relation on states in LTSs. Since the states in the LTSs associated with T_0 and $T_0 \oplus T_1$ are closed terms, the equivalence relation $\sim$ carries over to $\mathrm{T}(\Sigma_0)$ and $\mathrm{T}(\Sigma_0 \oplus \Sigma_1)$, respectively. Owing to operational conservativity, the equivalence relation $\sim$ on $\mathrm{T}(\Sigma_0)$ as induced by T_0 agrees with this equivalence relation on $\mathrm{T}(\Sigma_0)$ as induced by $T_0 \oplus T_1$. Applications of Thm. 5.5, in the presence of an operational conservative extension of a TSS, are abundant in the literature; we give a typical example.

Example: Using Thm. 3.4 it is not hard to see that the process algebra ACP [12] is an operational conservative extension of BPA_δ. Bergstra and Klop presented in *op. cit.* an (unconditional) CTRS $R_0 \oplus R_1$ for the process algebra ACP, which reduces each closed term in ACP to a closed term in BPA_δ. Moreover, $R_0 \oplus R_1$ is sound over ACP modulo bisimulation equivalence, and it is easily shown that R_0 can reduce bisimilar closed terms in BPA_δ to the same closed term in BPA_δ. Hence, Thm. 4.4 says that $R_0 \oplus R_1$ is ground confluent. (In [12, p. 122], an analysis of about 400 cases was needed to prove this fact for the more general case of open terms.) $\square$

6 Conclusion

Operational conservativity of an extension of a TSS can in general be concluded in a straightforward fashion from the syntactic form of the transition rules. Operational conservative extension seems such a natural notion that in the literature this property is often a hidden assumption: its formulation and proof are omitted without justification. For example, this happens in the design of process algebras, and in applications of the strategy to prove ω-completeness mentioned in Sect. 4.3.

Paying attention to operational conservative extension not only leads to more accurate contemplations on concurrency theory, but is also beneficial in other respects. Namely, operational conservative extension can be applied to derive useful results in the realm of equational reasoning, which are much harder to obtain using more classical term rewriting approaches or customized techniques.

References

[1] L. Aceto, W. Fokkink, R. van Glabbeek, and A. Ingólfsdóttir, *Axiomatizing prefix iteration with silent steps*, Information and Computation, 127 (1996), pp. 26–40.

[2] L. Aceto, W. Fokkink, and C. Verhoef, *Structural operational semantics*, in Handbook of Process Algebra, J. Bergstra, A. Ponse, and S. Smolka, eds., Elsevier, 1999. To appear.

[3] L. Aceto and M. Hennessy, *Termination, deadlock and divergence*, J. Assoc. Comput. Mach., 39 (1992), pp. 147–187.

[4] F. Baader and T. Nipkow, *Term Rewriting and All That*, Cambridge University Press, 1998.

[5] J. Backus, *The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference*, in Proceedings ICIP, Unesco, 1960, pp. 125–131.

[6] J. Baeten and J. Bergstra, *Discrete time process algebra*, in Cleaveland [20], pp. 401–420.

[7] J. Baeten, J. Bergstra, and J. W. Klop, *Syntax and defining equations for an interrupt mechanism in process algebra*, Fundamenta Informaticae, IX (1986), pp. 127–168.

[8] J. Baeten and J. W. Klop, eds., *Proceedings 1st Conference on Concurrency Theory,* Amsterdam, The Netherlands, vol. 458 of Lecture Notes in Computer Science, Springer-Verlag, 1990.

[9] J. Baeten and C. Verhoef, *A congruence theorem for structured operational semantics with predicates*, in Best [15], pp. 477–492.

[10] ——, *Concrete process algebra*, in Handbook of Logic in Computer Science, S. Abramsky, D. Gabbay, and T. Maibaum, eds., vol. IV, Oxford University Press, 1995, pp. 149–268.

[11] J. Baeten and P. Weijland, *Process Algebra*, Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

[12] J. Bergstra and J. W. Klop, *Process algebra for synchronous communication*, Information and Control, 60 (1984), pp. 109–137.

[13] ——, *Conditional rewrite rules: Confluence and termination*, J. Comput. System Sci., 32 (1986), pp. 323–362.

[14] J. Bergstra, A. Ponse, and J. van Wamel, *Process algebra with backtracking*, in Proceedings REX School/Symposium on A Decade of Concurrency: Reflections and Perspectives, Noordwijkerhout, The Netherlands, J. de Bakker, W. d. Roever, and G. Rozenberg, eds., vol. 803 of Lecture Notes in Computer Science, Springer-Verlag, 1994, pp. 46–91.

[15] E. Best, ed., *Proceedings 4th Conference on Concurrency Theory,* Hildesheim, Germany, vol. 715 of Lecture Notes in Computer Science, Springer-Verlag, 1993.

[16] B. Bloom, S. Istrail, and A. Meyer, *Bisimulation can't be traced: preliminary report*, in Conference Record 15th ACM Symposium on Principles of Programming Languages, San Diego, California, 1988, pp. 229–239. Preliminary version of [17].

[17] ——, *Bisimulation can't be traced*, J. Assoc. Comput. Mach., 42 (1995), pp. 232–268.

[18] R. Bol and J. F. Groote, *The meaning of negative premises in transition system specifications*, J. Assoc. Comput. Mach., 43 (1996), pp. 863–914.

[19] T. Bolognesi and F. Lucidi, *Timed process algebras with urgent interactions and a unique powerful binary operator*, in Proceedings REX Workshop on Real-Time: Theory in Practice, Mook, The Netherlands, June 1991, J. de Bakker, C. Huizing, W. d. Roever, and G. Rozenberg, eds., vol. 600 of Lecture Notes in Computer Science, Springer-Verlag, 1992, pp. 124–148.

[20] R. Cleaveland, ed., *Proceedings 3rd Conference on Concurrency Theory,* Stony Brook, NY, vol. 630 of Lecture Notes in Computer Science, Springer-Verlag, 1992.

[21] R. Cleaveland and M. Hennessy, *Priorities in process algebras*, Information and Computation, 87 (1990), pp. 58–77.

[22] P. D'Argenio, *A general conservative extension theorem in process algebras with inequalities*, in Proceedings 2nd Workshop on the Algebra of Communicating Processes, Eindhoven, The Netherlands, A. Ponse,

C. Verhoef, and B. van Vlijmen, eds., Report CS-95-14, Eindhoven University of Technology, 1995, pp. 67–79.

[23] P. D'Argenio and C. Verhoef, *A general conservative extension theorem in process algebras with inequalities*, Theoretical Comput. Sci., 177 (1997), pp. 351–380.

[24] W. Fokkink and R. van Glabbeek, *Ntyft/ntyxt rules reduce to ntree rules*, Information and Computation, 126 (1996), pp. 1–10.

[25] W. Fokkink and C. Verhoef, *A conservative look at operational semantics with variable binding*, Information and Computation, 146 (1998), pp. 24–54.

[26] ——, *Conservative extension in positive/negative conditional term rewriting with applications to software renovation factories*, in Proceedings 2nd Conference on Fundamental Approaches to Software Engineering, Amsterdam, The Netherlands, J.-P. Finance, ed., vol. 1577 of Lecture Notes in Computer Science, Springer-Verlag, 1999, pp. 98–113.

[27] A. van Gelder, K. Ross, and J. Schlipf, *The well-founded semantics for general logic programs*, J. Assoc. Comput. Mach., 38 (1991), pp. 620–650.

[28] R. van Glabbeek, *The linear time – branching time spectrum*, in Baeten and Klop [8], pp. 278–297.

[29] ——, *A complete axiomatization for branching bisimulation congruence of finite-state behaviours*, in Proceedings 18th Symposium on Mathematical Foundations of Computer Science 1993, Gdansk, Poland, A. Borzyszkowski and S. Sokołowski, eds., vol. 711 of Lecture Notes in Computer Science, Springer-Verlag, 1993, pp. 473–484.

[30] ——, *Full abstraction in structural operational semantics (extended abstract)*, in Proceedings 3rd Conference on Algebraic Methodology and Software Technology, Enschede, The Netherlands, M. Nivat, C. Rattray, T. Rus, and G. Scollo, eds., Workshops in Computing, Springer-Verlag, 1993, pp. 77–84.

[31] ——, *The meaning of negative premises in transition system specifications II*, in Automata, Languages and Programming, 23rd Colloquium, F. Meyer auf der Heide and B. Monien, eds., vol. 1099 of Lecture Notes in Computer Science, Paderborn, Germany, 1996, Springer-Verlag, pp. 502–513.

[32] ——, *The meaning of negative premises in transition system specifications II*, Report STAN-CS-TN-95-16, Department of Computer Science, Stanford University, 1996.

[33] J. F. Groote, *Transition system specifications with negative premises*, Theoretical Comput. Sci., 118 (1993), pp. 263–299.

[34] J. F. Groote and F. Vaandrager, *Structured operational semantics and bisimulation as a congruence*, Information and Computation, 100 (1992), pp. 202–260.

[35] M. Hennessy and T. Regan, *A process algebra for timed systems*, Information and Computation, 117 (1995), pp. 221–239.

[36] R. Keller, *Formal verification of parallel programs*, Comm. ACM, 19 (1976), pp. 371–384.

[37] S. Klusener, *Completeness in real time process algebra*, in Proceedings 2nd Conference on Concurrency Theory, Amsterdam, The Netherlands, J. Baeten and J. F. Groote, eds., vol. 527 of Lecture Notes in Computer Science, Springer-Verlag, 1991, pp. 376–392.

[38] K. G. Larsen, *Modal specifications*, Tech. Rep. R 89-09, Institute for Electronic Systems, University of Aalborg, 1989.

[39] K. G. Larsen and A. Skou, *Compositional verification of probabilistic processes*, in Cleaveland [20], pp. 456–471.

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

[41] ——, *A complete axiomatisation for observational congruence of finite-state behaviors*, Information and Computation, 81 (1989), pp. 227–247.

[42] F. Moller and C. Tofts, *A temporal calculus of communicating systems*, in Baeten and Klop [8], pp. 401–415.

[43] X. Nicollin and J. Sifakis, *The algebra of timed processes,* **ATP***: theory and application*, Information and Computation, 114 (1994), pp. 131–178.

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

[45] T. Przymusinski, *On the declarative semantics of deductive databases and logic programs*, in Foundations of Deductive Databases and Logic Programming, J. Minker, ed., Morgan Kaufmann Publishers, Inc., Los Altos, California, 1988, pp. 193–216.

[46] ——, *The well-founded semantics coincides with the three-valued stable semantics*, Fundamenta Informaticae, 13 (1990), pp. 445–463.

[47] A. Roscoe, *The Theory and Practice of Concurrency*, Prentice-Hall International, 1998.

[48] R. de Simone, *Calculabilité et Expressivité dans l'Algebra de Processus Parallèles* Meije, thèse de 3^e cycle, Univ. Paris 7, 1984.

[49] C. Verhoef, *A general conservative extension theorem in process algebra*, in Proceedings IFIP Working Conference on Programming Concepts, Methods and Calculi, San Miniato, Italy, E.-R. Olderog, ed., IFIP Transactions A-56, Elsevier, 1994, pp. 149–168.

[50] J. Vrancken, *The algebra of communicating processes with empty process*, Theoretical Comput. Sci., 177 (1997), pp. 287–328.