# A Hierarchy of SOS Rule Formats

(Preliminary abstract)

Jan Friso Groote [1] MohammadReza Mousavi [2]
Michel A. Reniers [3]

*Department of Computer Science, Eindhoven University of Technology (TU/e), P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands*

**Abstract**

In 1981 Structural Operational Semantics (SOS) was introduced as a systematic way to define operational semantics of programming languages by a set of rules of a certain shape [56]. Subsequently, the rule format became object of study. Using so-called Transition System Specifications (TSSs) several authors syntactically restricted the format of rules and showed several useful properties about the semantics induced by any TSS adhering to the format. This has resulted in a line of research proposing several syntactical rule formats and associated meta-theorems. Properties that are guaranteed by such rule formats range from well-definedness of the operational semantics and compositionality of behavioral equivalences to security- and probability-related issues. In this paper, we provide an initial hierarchy of SOS rules formats and meta-theorems formulated around them.

*Key words:* Formal Semantics, Structural Operational Semantics, Rule Formats, Framework.

## 1 Introduction

Structural Operational Semantics has become the common way to define operational semantics. Operational semantics defines the possible *transitions* that a piece of syntax can make during its "execution". Each transition may be *labelled* by a message to be communicated to the outside world. Transitions of a composed piece of syntax can usually be defined in a generic way, in terms

[1] Email: J.F.Groote@tue.nl
[2] Email: M.R.Mousavi@tue.nl
[3] Email: M.A.Reniers@tue.nl

of the transitions of its constituting parts. This forms the central idea behind Structural Operational Semantics.

Transition System Specifications (TSSs), as introduce by Groote and Vaandrager in [32], are a formalization of structural operational semantics. By imposing syntactic restrictions on TSSs one can deduce several interesting properties about their induced operational semantics. These properties range from well-definedness of the operational semantics [31,14,28] to security- [61,62] and probability-related issues [9,36]. The syntactic restrictions imposed by these meta-theorems usually suggest particular forms of deduction rules to be safe for a particular purpose and hence these meta-theorems usually define what is called a *rule format*. The excellent overview [3] provides existing rule formats to its date of publication (2001). Since then, even more formats have been proposed and we felt that in order to keep track, this field of formats required some structure.

Therefore, we attempt to present an overview of all SOS rule formats defined in the literature, together with the meta-theorems formulated around them. All the results are put in a lattice to easy locate the most suited format for a certain application. To do this, we define the concept of a TSS, in a far more general setting than [32], including the concepts of multi-sorted signatures and variable binding, inspired by the definition of [20]. This general definition of TSS serves as a unifying framework and paves the way for studying semantic meta-theorems for SOS and comparing their underlying frameworks.

The rest of this paper is organized as follows. In Section 2 the hierarchy of formats is given. In Section 3, we list different syntactic features that an SOS rules can have. In Section 4, we review semantic meta-theorems about different SOS frameworks.

**Disclaimer.** This paper is a preliminary draft. This means that formats can still be missing (we are very thankful if notified of such omissions), not all relevant theorems have been added and several other points need to be settled.

## 2 A hierarchy of operational formats

Since the first formats have been defined for TSS rules, many have been added. In order to keep track of them we made an overview of the existing rule formats in Figure 1. The lattice presented there has SOS frameworks as nodes, ordered by syntactic inclusion (mainly based on the syntactic features). The most general format can be found at the top and more specific formats can at lower positions. The arrows indicate syntactical inclusion. This is a preliminary version of the paper, in which we did not have time to verify all inclusions. Those that are still unconfirmed are indicated with dotted lines.

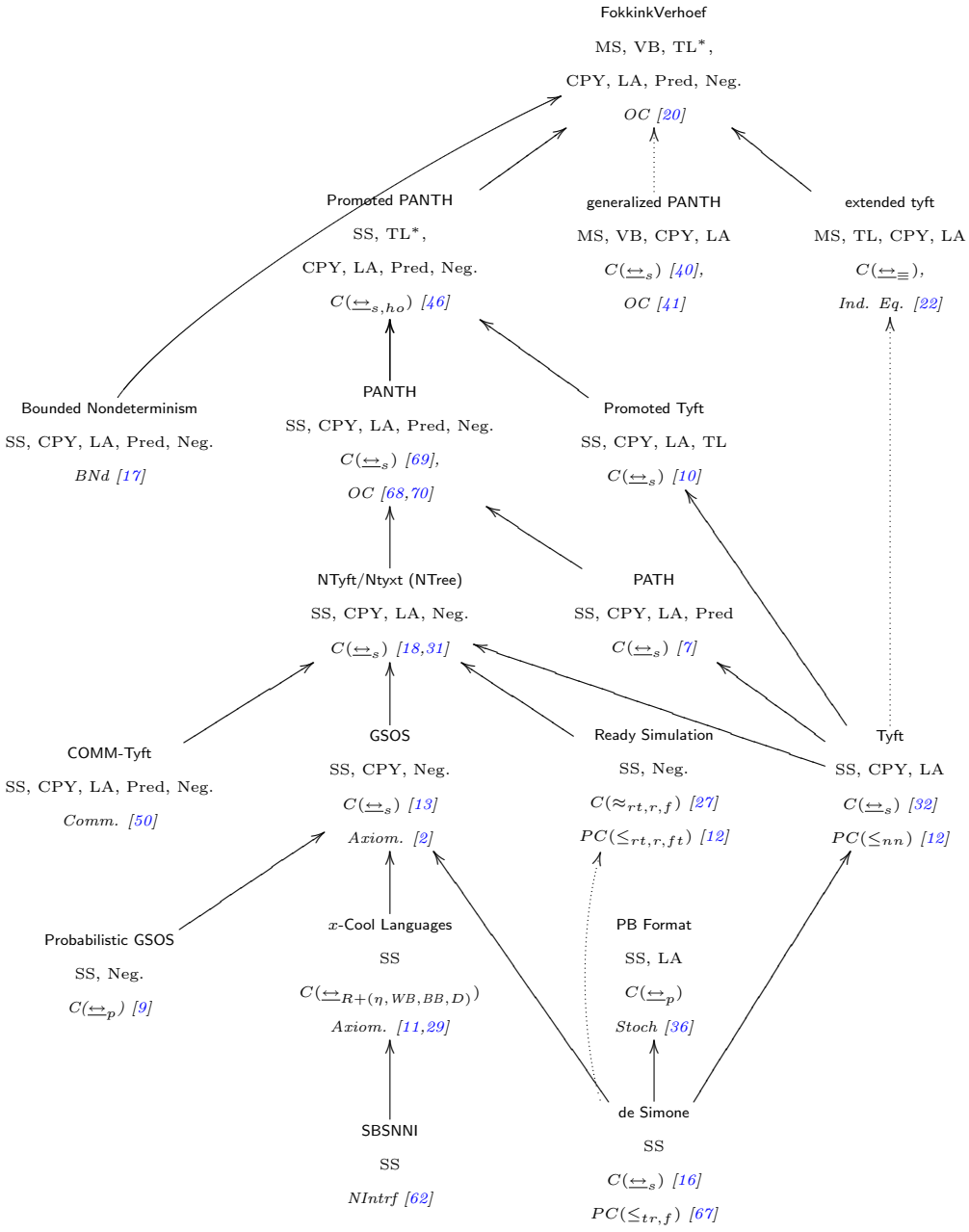


Fig. 1. A hierarchy of existing SOS formats

A node in this lattice has the following structure. At the right an example is shown, taken from the lattice. The syntactic features of each format are described in Section 3. The abbreviations can be found in Table 1. The theorems holding for TSSs in each format are described in Section 4 and the abbreviations can be found in Table 2.

| Format Name | NTyft/NTyxt (NTree) |
|---|---|
| Syntactic Features | SS, CPY, LA, Neg. |
| *Semantic Meta-Theorems [references]* | $C(\underline{\leftrightarrow}_s)$ *[18,31]* |

## 3 Syntactic Features of TSSs

In this section we list the distinctive syntactical features of the different rule formats. The typical features for each format are listed in the diagram.

| Shorthand | Syntactic Feature |
|---|---|
| SS vs. MS | Single- vs. Multi-Sorted Terms |
| VB | Variable Binding |
| TL vs. TL* | A Term vs. List of Terms as Labels |
| CPY | Copying Variables |
| LA | Look Ahead |
| Pred | Predicates |
| Neg | Negative Premises |

Table 1
Syntactic features of rule formats (cf. Fig. 1)

### *3.1 Labels*

Labels are terms that may appear as parameters of transition relations and predicates in the deduction rules. SOS frameworks can be classified with respect to the kind of labels they afford as follows.

**Open Terms as Labels.** Many SOS frameworks assume a special sort for labels and only allow for constants (alternatively, closed terms) of this sort to appear as labels. Such SOS frameworks thus forbid any correlation between valuation of terms and labels through the use of common variables. Frameworks defined in [20,22,10,47] allow for arbitrary terms as labels. All other SOS frameworks reviewed in this paper only allow for constant labels.

Open terms are used as labels in a number of cases in transition system specifications. Higher-order process calculi [15,60,59] are examples of formalisms exploiting this feature.

**Lists of Terms as Labels.** Most SOS frameworks only allow for a single term as label. The only existing exceptions are those of [20,47].

### *3.2 Signatures*

**Names and Binders.** In many contemporary process algebras and calculi, concepts of names, (actual and formal) variables and name abstraction (binding) are present and even serve as a basic ingredient. For example, in the $\pi$-calculus of Milner [43,44,45], names are first-class citizens and the whole calculus is built around the notion of passing names among concurrent agents. Less central, yet important, instances of these concepts appear in different process algebras in the form of the recursion operator, the infinite sum operator and the time-integration operator (cf., for example, [43], [39,57] and [6], respectively). Hence, it is interesting to accommodate the concept of names in the TSS framework.

There have been a few attempts in this direction. Proposals for modeling names and binders are formulated in [20,40,41]. Apart from the introduction of parameterized variables, the TSS framework of [40,41] is more restricted than that of [20] in that it does not allow for arbitrary terms as labels.

Apart from [20,40,41], all other rule formats we mention in the remainder of this paper do not support names and binders.

**Multi-Sorted States.** Based on the number of sorts allowed in the signature, an SOS framework may be classified in the following three categories:

(i) Multi-sorted TSS's: In such frameworks, there is no restriction on the sorts allowed for constructing terms.

(ii) $N$-sorted TSS's: A framework may only allow for a fixed number of sorts participating in the signature. An example of such frameworks is the process-tyft format of [49] where there are two distinguished sorts of processes and data. Apart from these three sorts that are used to define the states of the semantics, there is a sort for constant labels.

(iii) Single-sorted: This is the most common framework in the literature. It has a single sort for operational states which is usually called the sort of *processes* (and terms from this sort are process terms). In this framework, there is usually a sort for constant labels, as well.

The TSS of [22] has a special status with respect to its allowed signatures. Namely, it requires a special sort for processes and at least one (necessarily different) sort for labels. Furthermore, it requires that process sorts should not participate in function symbols with label sorts as targets.

**Look Ahead.** A framework allows for look ahead if a deduction rule in the framework may have two premises with a variable in the target of one of the premises being present in the source of the other. An example of a deduction rule with look ahead is the following.

$$\frac{x \xrightarrow{\tau} y \quad y \xrightarrow{l} z}{x \xrightarrow{l} z}$$

The above rule from [24] is used to combine silent ($\tau$) and ordinary transitions in order to implement a weak semantics (by ignoring silent steps) inside a strong semantic framework.

**Well-foundedness.** Here, we formally define the concept of well-foundedness which is a useful concept in the proof of several meta-results.

**Definition 3.1 (Variable Dependency Graph and Well-foundedness)** *The* variable dependency graph *of a deduction rule is a graph of which the nodes are variables and there is an edge between two variables if one appears in the source and the other in the target of (the same) positive premise in the deduction rule. A deduction rule is* well-founded *when all the backward chains of variables in the variable dependency graph are finite. A TSS is well-founded when all its deduction rules are.*

All practical instances of SOS specifications are well-foundedned. Well-foundedness also comes very handy in the proof of semantic meta-results for SOS frameworks. Hence, it is only of theoretical interest whether a framework allows for non-well-founded deduction rules or not.

**Copying** A framework has the *copying* feature if it allows for repetition of variables in the target of the conclusion. A simple example of copying is the second rule in the following TSS which defines the semantics of the `while` construct.

$$\frac{\neg \mathit{Hold}[b, M]}{\langle \texttt{while}\ (b)\ \texttt{do}\ P\ \texttt{od}, M\rangle \downarrow}$$

$$\frac{\mathit{Hold}[b, M]}{\langle \texttt{while}\ (b)\ \texttt{do}\ P\ \texttt{od}, M\rangle \rightarrow \langle P; \texttt{while}\ (b)\ \texttt{do}\ P\ \texttt{od}, M\rangle}$$

**Infinite Premises.** It is an interesting theoretical question whether a framework allows for infinite number of premises or not. Also practically, when dealing with infinite domains (e.g., infinite basic actions, data or time domains), it is sometimes useful to have deduction rules with infinite premises. The following example from [52] illustrates a possible use of deduction rules

with infinite premises:

$$\frac{x \xrightarrow{a} y \quad a \notin H}{\partial_H(x) \xrightarrow{a} \partial_H(x')} \qquad \frac{\forall_{a \in A \setminus H} x \xrightarrow{a} \!\!\!\!\!\!/}{\partial_H(x) \xrightarrow{\chi} \delta}$$

The above deduction rules define the semantics of the encapsulation operator $\partial_H(\_)$ which forbids its parameter from performing actions in $H$. If the parameter cannot perform any ordinary action allowed by $\partial_H$ then it makes a transition to the deadlocking process $\delta$. If the set of basic actions is infinite, then for some finite $H$, the deduction rule in the right-hand side has infinite (negative) premises.

### *3.4 Negative Premises*

Negative premises are a complicating factor in SOS frameworks. They cause several complications with respect to the semantics of TSS's which are rather difficult to solve. In other words, it is not immediately clear what can be considered a "proof" for a negative formula.

To our knowledge, in practice, the first example of negative premises in SOS appeared in [5] in the specification of the semantics of the following priority operator $\theta(\_)$.

$$\frac{x \xrightarrow{a} x' \quad \forall_{b>a} x \xrightarrow{b} \!\!\!\!\!\!/}{\theta(x) \xrightarrow{a} \theta(x')}$$

The above deduction rule states that a parameter of $\theta(\_)$ can perform a transition with label $a$ if no transition with a label $b$ of higher priority can be performed (according to a given ordering $>$). In addition to negative premises, the above deduction rule may have infinite premises if there is a chain of priorities with infinitely many (different) basic actions.

### *3.5 Predicates*

Predicates are useful syntactic features which are used to specify phenomenon such as termination or divergence.

### *3.6 Other Syntactic Features*

**Ordering the Deduction Rules.** One way to avoid the use of negative premises (and sometimes predicates) is by defining an order among deduction rules. Then, a deduction rule of a lower order may be applied to prove a formula only when there is no deduction rule with a higher order applicable. For example, the semantics of the priority operator defined in Section 3.4 can

be expressed in terms of a number of rules of the following form

$$\frac{x \xrightarrow{a} x'}{\theta(x) \xrightarrow{a} \theta(x')}$$

with an ordering among such rules based on the ordering among labels. The semantics of the sequential composition operator can also be defined as follows.

$$\frac{x \xrightarrow{l} x'}{x; y \xrightarrow{l} x'; y} \qquad \frac{y \xrightarrow{l} y'}{x; y \xrightarrow{l} y'}$$

with the rule in the left-hand side being ordered above the right-hand side rule. This way, the second argument of sequential composition can take over, only when the first part cannot make a transition, i.e., has terminated (we do not consider unsuccessful termination or deadlock in this simple setting). The implications of introducing an order among deduction rules and its possible practical use are investigated in [65,55]

**Equational Specifications.** Structural congruences are equational addenda to SOS specification which can define inherent properties of function symbols or define some function symbols in terms of the others. For example, the following equation specifies that the order of arguments in a parallel composition does not matter or in other words, that parallel composition is commutative.

$$x \,||\, y \equiv y \,||\, x$$

In [48], the addition of equational specifications to SOS specifications is studied in detail.

## 4 Semantic Meta-Results

In this section we list a number of meta-results. Each meta-result is abbreviated (see Table 2) and indicated in each format in Figure 1.

### *4.1 Congruence for Behavioral Equivalences*

Given an operational semantics, it is interesting to observe when two systems show the same behavior. It is also interesting to check whether a particular system is a restricted implementation of the other. To check these, we have to have notions of *behavioral equivalence* and *behavioral pre-order*, respectively. It is very much desired for a notion of behavioral equivalence (pre-order) to be compositional or in technical terms to be a *congruence* (pre-congruence). There is a myriad of notions of behavioral equivalence and pre-order in the literature [26,25]. Correspondingly, there are a number of rule formats guaranteeing these notions to be a (pre-)congruence [32,12,11,29]. In the remainder, we confine ourselves to single-sorted frameworks with constant labels. In such

Semantic Meta-Theorems

| Shorthand | Semantic Meta-Theorems |
|---|---|
| $C(\underline{\leftrightarrow}_x)$ | Congruence for $x$-Bisimulation |
| $C(\approx_x)$ | Congruence for $x$-equality |
| $PC(\leq_x)$ | Pre-congruence for $x$-pre-order |
| OC | Operational Conservativity |
| Axiom. | Deriving Sound and Complete Axiomatization |
| Ind. Eq. | Comparison of Induced Equality Classes |
| NIntrf | Non Intereference (Security-related [58]) |
| BNd | Bounded Non-determinism |
| Stoch | Stochasticity |

Table 2
Short-hands for theorems used in Figure 1

frameworks arity of a function symbol can be conveniently expressed by a natural number (representing the number of parameters in the left-hand side of the arrow). The only congruence meta-theorems for multi-sorted frameworks are those of [22,40,20] and with open terms as labels are [22,20,47].

We start with defining the notion of congruence.

**Definition 4.1 ((Pre-)Congruence)** *An equivalence (pre-order) $R \subseteq \mathcal{T} \times \mathcal{T}$ is a* (pre-)congruence *with respect to a signature $\Sigma$ if and only if for all $(f, ar(f)) \in \Sigma$ and all $\overrightarrow{p}_{ar(f)-1}, \overrightarrow{q}_{ar(f)-1} \in \mathcal{T}$, if $\overrightarrow{p}_{ar(f)-1}$ $R$ $\overrightarrow{q}_{ar(f)-1}$ then $f(\overrightarrow{p}_{ar(f)-1})$ $R$ $f(\overrightarrow{q}_{ar(f)-1})$.*

The first congruence formats were defined for the notion of strong bisimilarity, defined below.

**Definition 4.2 (Bisimulation and Bisimilarity [54])** *A relation $R \subseteq \mathcal{C} \times$*

$\mathcal{C}$ *is a* bisimulation *relation with respect to a set of transition relations* $Rel$ *and a set of predicates* $Pr$ *if and only if* $\forall_{p,q \in \mathcal{C}}\ pRq \Rightarrow \forall_{r \in Rel, P \in Pr}$ *such that* $r$ *and* $P$ *are of arbitrary arities of* $n$ *and* $m$*,*

*(i)* $\forall_{p', \overrightarrow{p}_n \in \mathcal{C}}\ p \xrightarrow{\overrightarrow{p}_r^n} p' \Rightarrow \exists_{q' \in \mathcal{C}}\ q \xrightarrow{\overrightarrow{p}_r^n} q' \wedge (p', q') \in R$*;*

*(ii)* $\forall_{q', \overrightarrow{p}_n \in \mathcal{C}}\ q \xrightarrow{\overrightarrow{p}_r^n} q' \Rightarrow \exists_{p' \in \mathcal{C}}\ p \xrightarrow{\overrightarrow{p}_r^n} p' \wedge (p', q') \in R$*;*

*(iii)* $P(\overrightarrow{p}_m) \Leftrightarrow P(\overrightarrow{q}_m)$*.*

*Two closed terms* $p$ *and* $q$ *are are* bisimilar *if and only if there exists a bisimulation relation* $R$ *with respect to* $Rel$ *such that* $(p, q) \in R$*. Two closed terms* $p$ *and* $q$ *are* bisimilar *with respect to a transition system specification* $tss$*, denoted by* $tss \vdash p \leftrightarrows q$*, if and only if they are bisimilar with respect to the semantics of* $tss$*.*

There are good reasons for considering strong bisimilarity as an important notion of behavioral equivalence. Here, we mention a few.

(i) Strong bisimilarity usually gives rise to elegant and neat theories and it turns out that congruence formats for it are also much more elegant and compact than those for other (weaker) notions;

(ii) For finite state processes, strong bisimilarity can be checked very efficiently in practice [53] while some weaker notions are intractable [35];

(iii) Other notions can often be coded in strong bisimilarity [24].

### *4.2 Well-definedness of the Semantics*

If there are negative premises in the semantical rules it is not self evident anymore whether the rules define a transition relation in an unambiguous way. In [31], a first criterion using which a TSS in the ntyft/ntyxt format is guaranteed to have a well-defined semantics. This criterion, defined below, is called (strict) stratification and is originally due to [23] in logic programming. It is an important property of a format when it guarantees that every set of rules equivocally defines a transition relation.

### *4.3 Conservativity of Language Extensions*

Operational semantics of languages may be extended by adding new pieces of syntax to the signature and new rules to the set of deduction rules. A number of meta-theorems have been proposed to check whether extensions do not change the behavior of the old language and whether they preserve equalities among old terms. Two general instances of such meta-theorems are formulated in [20,41]. We review the results of [20] in this section, which gives the most detailed account of this issue.

To extend a language defined by a TSS, one may have to combine an existing signature with a new one. However, not all signatures can be combined into one as the arities of the function symbols may clash. To prevent this, we define two signatures to be *consistent* when they agree on the arity of the shared function symbols. In the remainder, we always assume that extended and extending TSS's are consistent. The following definition formalizes the concept of operational extension.

**Definition 4.3 (Extension of a TSS)** *Consider TSS's* $tss_0 = (\Sigma_0, L_0, D_0)$ *and* $tss_1 = (\Sigma_1, L_1, D_1)$. *The extension of* $tss_0$ *with* $tss_1$, *denoted by* $tss_0 \cup tss_1$, *is defined as* $(\Sigma_0 \cup \Sigma_1, L_0 \cup L_1, D_0 \cup D_1)$.

Next, we define when an extension of a TSS is called operationally conservative.

**Definition 4.4 (Operational Conservativity [68])** *Consider TSS's* $tss_0 = (\Sigma_0, L_0, D_0)$ *and* $tss_1 = (\Sigma_1, L_1, D_1)$. *If* $\forall_{p \in C(\Sigma_0)}\ \forall_{p' \in C(\Sigma_0 \cup \Sigma_1)}\ \forall_{l \in L_0 \cup L_1}$ $tss_0 \cup tss_1 \vDash p \xrightarrow{l} p' \Leftrightarrow tss_0 \vDash p \xrightarrow{l} p'$, *then* $tss_0 \cup tss_1$ *is an* operationally conservative extension *of* $tss_0$.

Next, we formulate sufficient conditions to prove operational conservativity. But before that, we need a few auxiliary definitions.

**Definition 4.5 (Source Dependency)** *All variables appearing in the source of the conclusion of a deduction rule are called* source dependent. *A variable of a deduction rule is* source dependent *if it appears in a target of a premise of which all the variables of the source are source dependent. A premise is* source dependent *when all the variables appearing in it are source dependent. A rule is* source dependent *when all its variables are. A TSS is* source dependent *when all its rules are.*

**Definition 4.6 (Reduced Rules)** *For a deduction rule* $d = (H, c)$, *the reduced rule with respect to a signature* $\Sigma$ *is defined by* $\rho(d, \Sigma) \doteq (H', c)$ *where* $H'$ *is the set of all premises from* $H$ *which have a* $\Sigma$*-term as a source.*

The following result, from [20], gives sufficient conditions for an extension of a TSS to be operationally conservative.

**Theorem 4.7 (Operational Conservativity Meta-Theorem [20])** *Given two TSS's* $tss_0 = (\Sigma_0, L_0, D_0)$ *and* $tss_1 = (\Sigma_1, L_1, D_1)$, $tss_0 \cup tss_1$ *is an operationally conservative extension of* $tss_0$ *if:*

*(i)* $tss_0$ *is source dependent;*

*(ii) for all* $d \in D_1$ *at least one of the following holds:*

- *(a) the source of the conclusion has a function symbol in* $\Sigma_1 \setminus \Sigma_0$, *or*
- *(b)* $\rho(d, \Sigma_0)$ *has a source-dependent positive premise* $t \xrightarrow{l} t'$ *such that* $l \notin \Sigma_0$ *or* $t' \notin T(\Sigma_0)$.

In [51], slightly more liberal notions of operational conservativity, called *orthogonality*, is introduced and a meta-theorem about it is proposed. Orthogonality allows for the addition of new transitions from / predicates on the old syntax provided that these new transitions and predicates do not change the behavioral equalities among old terms.

### *4.4 Generating Equational Theories*

Equational theories are central notions to process algebras [4,34,42]. They capture the basic intuition behind the algebra, and the models of the algebra are expected to respect this intuition (e.g., the models induced by operational semantics). One of the added-values of having equational theories is that they enable reasoning at the level of syntax without committing for particular models of the semantics. When the semantic model of behavior (e.g., the transition system associated to a term) is infinite, these techniques may come very handy.

To establish a reasonable link between the operational model and the equational theory of the algebra, a notion of behavioral equality should be fixed. Ideally, the notion of behavioral equivalence should coincide with the closed derivations of the equational theory. One side of this coincidence is captured by the *soundness* theorem which states that all closed derivations of the equational theory are indeed valid with respect to the particular notion of behavioral equality. The other side of the coincidence, called *completeness*, phrases that all induced behavioral equalities are derivable from the equational theory, as well. These concepts are formalized in the remainder.

**Definition 4.8 (Equational Theory)** *An* equational theory *or* axiomatization $(\Sigma, V, E)$ *is a set of equalities $E$ on a signature $\Sigma$ of the form $t = t'$, where $t, t' \in \mathcal{T}$. A closed instance $p = p'$, for some $p, p' \in \mathcal{C}$, is derivable from $E$, denoted by $E \vdash p = p'$ if and only if it is in the smallest congruence relation on closed terms induced by the equalities of $E$.*

*An equational theory $(\Sigma, V, E)$ is* sound *with respect to a TSS $tss$ (also on signature $\Sigma$) and a particular notion of behavioral equality $\sim$ if and only if for all $p, p' \in \mathcal{C}$, if $E \vdash p = p'$, then it holds that $tss \vdash p \sim p'$. It is* complete *if the implication holds in the other direction.*

An equational theory is called sound with respect to a particular semantics and notion of equivalence, if all derivable closed equalities also hold in the semantic model. It is called complete if for

In [2,1], an automatic method for generating sound and complete equational theories from GSOS specifications is presented. This technique was extended in [8] to cater for explicit termination of processes. This approach, although more complicated in nature, gives rise to more intuitive and more compact sets of equations compared to the original approach of [2]. Axiom

systems for pre-orders have been generated, too, see for instance [64].

### *4.5 Other Meta-Results*

**Non-Interference.** Confidentiality is an important aspect of security and non-interference [30] is an well-studied means to guarantee end-to-end confidentiality. Non-interference means that a user with a lower confidentiality level cannot infer anything about the higher level information by interacting with the system (using lower-level methods that has in hand). In [61,62] a rule format for non-interference is proposed which is based on the Cool languages format (in order to guarantee compositionality of non-intereference) and imposes further restrictions to assure that the lower-level behavior of the system does not change as a result of performing higher-level transitions.

**Decomposition of Logical Formulae.** In [33], a logical framework, called the *Henessy-Milner logic* after the authors' names, is proposed. The Hennessy-Milner logic can be used to reason about processes and characterize their equalities. In [37,38] a meta-theory is developed that allows for decomposing Hennesy-Milner formulae using the structure of terms in a generic way by examining deduction rules of the process language in the De Simone format. This result has been improved in [19] and extended to the ready simulation format (ntyft/ntyxt format without look ahead).

**Stochasticity.** For probabilistic transition systems, it is essential to make sure that the sum of all probabilities belonging to the same distribution amounts to 1 (or zero). This is called (semi-)stochasticity. In [36], a restricted form of the De Simone format is proposed that guarantees semi-stochasticity. To avoid dealing with negative premises the format of [36] supports ordering on rules.

**Bounded Non-determinism.** In [66], a rule format, by imposing restrictions on the De Simone format, is proposed which guarantees that the induced semantics affords only bounded non-determinism, i.e., each closed term has only finite number of outgoing transitions. Fokkink and Duong Vu in [21] generalize the result of [66] to a far more general SOS framework.

**Timing properties.** In [63] a set of syntactic conditions have been defined that guarantee properties for timed systems, such as time determinism, persistence, maximal progress and patience.

## References

[1] Luca Aceto. Deriving complete inference systems for a class of GSOS languages generating regular behaviours. In Bengt Jonsson and Joachim Parrow, editors,

*Proceedings of the fifth International Conference on Concurrency Theory (CONCUR'94)*, volume 836 of *Lecture Notes in Computer Science*, pages 449–464. Springer-Verlag, Berlin, Germany, 1994.

[2] Luca Aceto, Bard Bloom, and Frits W. Vaandrager. Turning SOS rules into equations. *Information and Computation (I&C)*, 111:1–52, 1994.

[3] Luca Aceto, Wan J. Fokkink, and Chris Verhoef. Structural operational semantics. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra, Chapter 3*, pages 197–292. Elsevier Science, Dordrecht, The Netherlands, 2001.

[4] Jos C. M. Baeten and Jan A. Bergstra. Real Time Process Algebra. *Formal Aspects of Computing*, 3:142–188, 1991.

[5] Jos C. M. Baeten and Rob J. van Glabbeek. Merge and termination in process algebra. In Kesav V. Nori, editor, *Proceeding of the Seventh Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'05)*, volume 287 of *Lecture Notes in Computer Science*, pages 153–172. Springer-Verlag, Berlin, Germany, 1987, 1987.

[6] Jos C. M. Baeten and Cornelis A. Middelburg. *Process Algebra with Timing*. EATCS Monographs. Springer-Verlag, Berlin, Germany, 2002.

[7] Jos C. M. Baeten and Chris Verhoef. A congruence theorem for structured operational semantics with predicates. In Eike Best, editor, *International Conference on Concurrency Theory (CONCUR'93)*, volume 715 of *Lecture Notes in Computer Science*, pages 477–492. Springer-Verlag, Berlin, Germany, 1993.

[8] Jos C.M. Baeten and Erik P. de Vink. Axiomatizing GSOS with termination. *Journal of Logic and Algebraic Programming (JLAP)*, 60-61:323–351, 2004.

[9] Falk Bartels. GSOS for probabilistic transition systems. In *Proceedings of the 5th International Workshop on Coalgebraic Methods in Computer Science (CMCS'02)*, volume 65 of *Electronic Notes in Theoretical Computer Science*, pages 1–25, 2002.

[10] Karen L. Bernstein. A congruence theorem for structured operational semantics of higher-order languages. In *IEEE Symposium on Logic In Computer Science (LICS'98)*, pages 153–164. IEEE Computer Society, Los Alamitos, CA, USA, 1998.

[11] Bard Bloom. Structural operational semantics for weak bisimulations. *Theoretical Computer Science (TCS)*, 146:25–68, 1995.

[12] Bard Bloom, Wan Fokkink, and Rob J. van Glabbeek. Precongruence formats for decorated trace semantics. *ACM Transactions on Computational Logic*, 5(1):26–78, 2004.

[13] Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *Journal of the ACM (JACM)*, 42(1):232–268, January 1995.

[14] Roland Bol and Jan Friso Groote. The meaning of negative premises in transition system specifications. *Journal of the ACM (JACM)*, 43(5):863–914, September 1996.

[15] Gérard Boudol. Towards a lambda-calculus for concurrent and communicating systems. In Josep Díaz and Fernando Orejas, editors, *Proceedings of the International Joint Conference on Theory and Practice of Software Development (TAPSOFT'89)*, volume 351 of *Lecture Notes in Computer Science*, pages 149–161. Springer-Verlag, Berlin, Germany, 1989.

[16] Robert de Simone. Higher-level synchronizing devices in MEIJE-SCCS. *Theoretical Computer Science (TCS)*, 37:245–267, 1985.

[17] Wan Fokkink and Thuy Duong Vu. Structural operational semantics and bounded nondeterminism. *Acta Informatica*, 39(6-7):501–516, 2003.

[18] Wan J. Fokkink and Rob J. van Glabbeek. Ntyft/ntyxt rules reduce to ntree rules. *Information and Computation (I&C)*, 126(1):1–10, 1996.

[19] Wan J. Fokkink, Rob J. van Glabbeek, and Paulien de Wind. Compositionality of Hennessy-Milner logic through structural operational semantics. In Andrzej Lingas and Bengt J. Nilsson, editors, *Proceedings of the 14th Symposium on Fundamentals of Computation Theory (FCT'03)*, volume 2751 of *Lecture Notes in Computer Science*, pages 412–422. Springer-Verlag, Berlin, Germany, 2003.

[20] Wan J. Fokkink and Chris Verhoef. A conservative look at operational semantics with variable binding. *Information and Computation (I&C)*, 146(1):24–54, 1998.

[21] Wan J. Fokkink and Thuy Duong Vu. Structural operational semantics and bounded nondeterminism. *Acta Informatica*, 39(6–7):501–516, 2003.

[22] Vashti Galpin. A format for semantic equivalence comparison. *Theoretical Computer Science (TCS)*, 309(1-3):65–109, 2003.

[23] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Proceedings of the 5th International Conference on Logic Programming (ICLP'88)*, pages 1070–1080. MIT Press, Cambridge, MA, USA, 1988.

[24] Rob J. van Glabbeek. Bounded nondeterminism and the approximation induction principle in process algebra. In Franz-Josef Brandenburg, Guy Vidal-Naquet, and Martin Wirsing, editors, *Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science (STACS'87)*, volume 247 of *Lecture Notes in Computer Science*, pages 336–347. Springer-Verlag, Berlin, Germany, 1987.

[25] Rob J. van Glabbeek. The linear time - branching time spectrum I. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra, Chapter 1*, pages 3–100. Elsevier Science, Dordrecht, The Netherlands, 2001.

[26] Rob J. van Glabbeek. The linear time - branching time spectrum II. In Eike Best, editor, *International Conference on Concurrency Theory (CONCUR'93)*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer-Verlag, Berlin, Germany, 1993.

[27] Rob J. van Glabbeek. Full abstraction in structural operational semantics (extended abstract). In Maurice Nivat, Charles Rattray, Teodor Rus, and Giuseppe Scollo, editors, *Proceedings of the Third International Conference on Algebraic Methodology and Software Technology (AMAST '93)*, pages 75–82. Springer-Verlag, Berlin, Germany, 1993.

[28] Rob J. van Glabbeek. The meaning of negative premises in transition system specifications II. *Journal of Logic and Algebraic Programming (JLAP)*, 60-61:229–258, 2004.

[29] Rob J. van Glabbeek. On cool congruence formats for weak bisimulations (extended abstract). Submitted, draft available from: http://www.cse.unsw.edu.au/~rvg/pub/cool-ea.pdf, 2005.

[30] Joseph A. Goguen and José Meseguer. Security policies and security models. In *IEEE Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, Los Alamitos, CA, USA, 1982.

[31] Jan Friso Groote. Transition system specifications with negative premises. *Theoretical Computer Science (TCS)*, 118(2):263–299, 1993.

[32] Jan Friso Groote and Frits W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation (I&C)*, 100(2):202–260, October 1992.

[33] Mathew C. B. Hennessy and Colin Striling. Algebraic laws for non-determinism and concurrency. *Journal of the ACM*, 32(1):137–161.

[34] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[35] Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation (I& C)*, 86(1):43–68, 1990.

[36] Ruggero Lanotte and Simone Tini. Probabilistic congruence for semistochastic generative processes. In Vladimiro Sassone, editor, *Proceedings of the 8th International Conference on Foundations of Software Science and Computational Structures (FOSSACS'05)*, volume 3441 of *Lecture Notes in Computer Science*, pages 63–78. Springer-Verlag, 2005.

[37] Kim G. Larsen. *Context-Dependent Bisimulation between Processes*. PhD thesis, University of Edinburgh, Edinburgh, Scotland, UK, 1986.

[38] Kim G. Larsen and L. Xixin. Compositionality through an operational semantics of contexts. *Journal of Logic and Computation*, 1(6):761–795, 1991.

[39] Sebastian P. Luttik. *Choice quantification in process algebra*. PhD thesis, Department of Computer Science, University of Amsterdam, Amsterdam, The Netherlands, 2002.

[40] Cornelis A. Middelburg. Variable binding operators in transition system specifications. *Journal of Logic and Algebraic Programming*, 47(1):15–45, 2001.

[41] Cornelis A. Middelburg. An alternative formulation of operational conservativity with binding terms. *Journal of Logic and Algebraic Programming (JLAP)*, 55(1-2):1–19, 2003.

[42] Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[43] Robin Milner. The polyadic $\pi$-calculus: a tutorial. In Friedrich L. Bauer, Wilfried Brauer, and Helmut Schwichtenberg, editors, *Logic and Algebra of Specification*, pages 203–246. Springer-Verlag, 1993. An earlier version of this paper appeared as Technical Report ECS-LFCS-91-180 of University of Edinburgh, 1991.

[44] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I. *Information and Computation (I&C)*, 100(1):1–40, 1992. An earlier version of this paper appeared as Technical Report ECS-LFCS-89-85 of University of Edinburgh, 1989.

[45] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part II. *Information and Computation (I&C)*, 100(1):41–77, 1992. An earlier version of this paper appeared as Technical Report ECS-LFCS-89-86 of University of Edinburgh, 1989.

[46] MohammadReza Mousavi, Murdoch J. Gabbay, and Michel Reniers. SOS for higher order processes. Technical report, Department of Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 2005.

[47] MohammadReza Mousavi, Murdoch J. Gabbay, and Michel A. Reniers. SOS for higher order processes. In *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR'05)*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 2005. To appear.

[48] MohammadReza Mousavi and Michel Reniers. Congruence for structural congruences. In *Proceedings of the Eighth International Conference on Foundations of Software Science and Computation Structures (FOSSACS'05)*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 2005. To appear.

[49] MohammadReza Mousavi, Michel Reniers, and Jan Friso Groote. Congruence for sos with data. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS'04)*, pages 302–313. IEEE Computer Society Press, Los Alamitos, CA, USA, 2004.

[50] MohammadReza Mousavi, Michel Reniers, and Jan Friso Groote. A syntactic commutativity format for SOS. *Information Processing Letters (IPL)*, 93:217–223, March 2005.

[51] MohammadReza Mousavi and Michel A. Reniers. Orthogonal extensions in structural operational. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 1214–1225. Springer-Verlag, Berlin, Germany, 2005.

[52] Xavier Nicollin and Joseph Sifakis. The algebra of timed processes ATP: theory and application. *Information and Computation (I&C)*, 114(1):131–178, October 1994.

[53] Robert Paige and Robert Endre Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[54] David M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, Berlin, Germany, 1981.

[55] Iain Phillips and Irek Ulidowski. Ordered sos rules and weak bisimulation. In Abbas Edalat, Sofia Jourdan, and Guy McCusker, editors, *Advances in Theory and Formal Methods of Computing*, pages 300–311. Imperial College Press, London, UK, 1996.

[56] Gordon D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark, September 1981.

[57] Michel A. Reniers, Jan Friso Groote, Mark B. van der Zwaag, and Jos van Wamel. Completeness of timed $\mu CRL$. *Fundamenta Informaticae*, 50(3-4):361–402, 2002.

[58] Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.

[59] Davide Sangiorgi. Bisimulation for higher-order process calculi. *Information and Computation (I&C)*, 131(2):141–178, 1996.

[60] Bent Thomsen. A theory of higher order communicating systems. *Information and Computation (I& C)*, 116:38–57, 1995.

[61] Simone Tini. Rule formats for non interference. In Pierpaolo Degano, editor, *Proceedings of the 12th European Symposium on Programming, Programming*

*Languages and Systems (ESOP'03)*, volume 2618 of *Lecture Notes in Computer Science*, pages 129–143. Springer-Verlag, Berlin, Germany, 2003.

[62] Simone Tini. Rule formats for compositional non-interference properties. *Journal of Logic and Algebraic Progamming (JLAP)*, 60:353–400, 2004.

[63] I. Ulidowski and S. Yuen. Process languages with discrete time based on the Ordered SOS format and rooted eager bisimulation. *Journal of Logic and Algebraic Programming*, 60-61:401–461, 2004.

[64] Irek Ulidowski. Finite axiom systems for testing preorder and De Simone process languages. *Theoretical Computer Science (TCS)*, 239(1):97–139, 2000.

[65] Irek Ulidowski and Iain Phillips. Formats of ordered sos rules with silent actions. In Michel Bidoit and Max Dauchet, editors, *Proceedings of 7th International Joint Conference on Theory and Practice of Software Development (TAPSOFT'97)*, volume 1214 of *Lecture Notes in Computer Science*, pages 297–308. Springer-Verlag, Berlin, Germany, 1997.

[66] Frits W. Vaandrager. Expressiveness results for process algebras. In Jaco W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *Proceedings of the REX Workshop on Semantics*, volume 666 of *Lecture Notes in Computer Science*, pages 609–638. Springer-Verlag, Berlin, Germany, 1993.

[67] Frits W. Vaandrager. On the relationship between process algebra and input/output automata. In *Proceedings of the Sixth Annual IEEE Symposium onLogic in Computer Science (LICS'91)*, pages 387–398. IEEE Computer Society, 1991.

[68] Chris Verhoef. A general conservative extension theorem in process algebra. In Ernst-Rüdiger Olderog, editor, *Proceedings of third IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET'94)*, volume A-56 of *IFIP Transactions*, pages 274–302. Elsevier Science Publishers, 1994.

[69] Chris Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2(2):274–302, 1995.

[70] Chris Verhoef, Luca Aceto, and Wan Fokkink. Conservative extension in structural operational semantics. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 69:110–132, 1999.