# Rule formats for bounded nondeterminism in nominal structural operational semantics*†

Álvaro García-Pérez

ICE-TCS, School of Computer Science, Reykjavík University, Iceland
alvarog@ru.is

## 1 Introduction

Structural operational semantics (SOS) [14,15] is a widely used formalism for defining the formal semantics of computer programs and for proving properties of the corresponding programming languages. In the SOS formalism a transition system specification (TSS) [9], which consists of a signature together with a set of inference rules, specifies a labelled transition system (LTS) [11] whose states (i.e., processes) are closed terms over the signature and whose transitions are those that can be proved using the inference rules.

Rule formats [3, 13] are syntactically checkable restrictions on the inference rules of a TSS that guarantee some useful property of the associated LTS. We focus on the finiteness of the number of outgoing transition from a given state, which is referred to as bounded nondeterminism in [18]. Broadly, bounded nondeterminism is taken as a synonym of finite branching [7]. Finite branching breaks down into the more elementary properties of initials finiteness and image finiteness [1].

Nominal structural operational semantics (NoSOS) [5] enriches the SOS formalism by adopting the nominal techniques from [8, 16] to deal with names and variable-binding operations within the SOS framework. The nominal techniques allow one to extend pleasantly structural induction and recursion to languages with variable-binding operations, without the need to redo on a case-by-case basis a large number of routine constructions that deal with renaming of bound variables [8]. The NoSOS framework develops the nominal techniques in the general setting of meta-theory of SOS [3, 13] and makes them applicable to a wide variety of specific languages.

The investigations on rule formats for bounded nondeterminism are far from being new. Vaandrager [17] introduced a rule format for SOS based on the de Simone format [6] that guarantees that the associated LTS is finite branching. Following Vaandrager, Bloom [4] introduced a rule format for his GSOS formalism that also guarantees a finite-branching LTS. Finally, Fokkink and Vu [7] introduced yet another less restrictive rule format for SOS which adapts the notion of strict stratification from [10], and showed that a TSS in this format induces an LTS that is finite branching.

Our work takes this programme further by contributing on three fronts:

(i) We provide syntactic conditions that use *global* information to filter more junk rules (i.e., rules that are never involved in a proof tree) than the conditions of the rule format in [7], which uses *local* information.

*Joint work with Luca Aceto, Ignacio Fábregas and Anna Ingólsfdóttir.

†This research has been supported by the project 'Nominal Structural Operational Semantics' (nr. 141558-051) of the Icelandic Research Fund.

(ii) We extend the applicability of the rule formats to the nominal setting by tackling one of the most prominent challenges there, namely that of allowing variables to occur in the actions that label a transition [2].

(iii) We consider a family of bounded-nondeterminism properties that are more elementary than finite branching, and which include image finiteness and initials finiteness [1].

The examples that follow are representative of each of these three contributions. Recall from [7] that an LTS is finite branching iff for every process $p$, the set $\{(l, p') \mid p \xrightarrow{l} p'\}$ is finite.

**Example 1.1.** *Let the signature $\Sigma$ consist of unary function symbols $f$ and $g$ and constant symbol $c$. Let the set $L$ consist of label $l$. Consider the TSS*

$$\frac{}{g(x) \xrightarrow{l} x} \qquad \frac{g^i(x) \xrightarrow{l} x}{f(x) \xrightarrow{l} x}\,, \qquad i \in \mathbb{N}$$

*where $g^i$ stands for applying $i$ times the function symbol $g$ to its argument. The TSS induces an LTS that is finite branching. Notice that for $g(p)$ and $f(p)$, with $p$ any process, the only provable transitions are respectively $g(p) \xrightarrow{l} p$ and $f(p) \xrightarrow{l} p$ since the axiom on the left allows one to instantiate the rule template on the right only for $i = 1$.*

Previous work on rule formats for finite branching [7] introduces the $\eta$-types, which determine an over-approximation of the set of rules that give rise to transitions. One of the conditions of the rule format in [7] is to require the $\eta$-types to be finitely inhabited. The $\eta$-typing discipline is *local* and thus it is not strong enough to discern whether the instances of the rule template of Example 1.1 would take part in a proof tree or not. For all $i \in \mathbb{N}$, the instances of the rule template above have one and the same $\eta$-type, which is infinitely inhabited. This renders the TSS of Example 1.1 out of the conditions of the rule format.

**Example 1.2.** *Consider the nominal TSS (NTSS for short) for term-for-atom substitution ($a \overset{T}{\mapsto} x$) on page 6 of [5], which includes the rule* $(\mathrm{abs1_{Ts}})$ *that we reproduce next:*

$$\ldots \qquad \frac{x \xrightarrow{y \overset{T}{\mapsto} z} x' \qquad a\#z \qquad a\#y}{[a]x \xrightarrow{y \overset{T}{\mapsto} x} [a]x'}\ (\mathrm{abs1_{Ts}}) \qquad \ldots$$

*Recall from [5] that the nominal term $[a]x$ is an abstraction where the atom $a$ is abstracted in the nominal term $x$, and that $a\#z$ is a freshness assertion that is provable iff the atom $a$ does not appear free in the nominal term $z$. The substitution of term $x$ for atom $a$ in term $t$ is modelled as an LTS with transitions $t \xrightarrow{a \overset{T}{\mapsto} x} t\{a \mapsto x\}$. Notice that $\cdot \overset{T}{\mapsto} \cdot$ is a binary function symbol and that the labels include arbitrary nominal terms, i.e., the $y \overset{T}{\mapsto} z$ that labels the first premiss of rule* $(\mathrm{abs1_{Ts}})$ *contains variables $y$ and $z$.*

The rule format in [7] requires the labels of transitions to be ground. The occurrence of variables is important in order to determine whether a proof tree can introduce spurious variables, which could be unified to any term and could possibly break bounded nondeterminism. Besides, the proof of correctness of the rule format in [7] relies on the TSS having a *strict stratification* (see Definition 4 of [7]) that entails an order relation among processes and enforces

the sources of the premisses in a rule to be less than the source of the rule. This prevents the associated LTS to implement ungarded recursion. The occurrence of variables in the labels opens for the possibility of the LTS to break bounded nondeterminism. In order to regain boundedness, a suitable notion of strict stratification may have to take the labels into account, which poses a non-trivial challenge.

Recall from [1] that an LTS is initials finite iff for every process $p$ the set $\{l \mid \exists p' \text{ s.t. } p \xrightarrow{l} p'\}$ is finite, and it is image finite iff for every process $p$ and every label $l$, the set $\{p' \mid p \xrightarrow{l} p'\}$ is finite. An LTS is finite branching iff it is both initials finite and image finite, and thus the latter properties are more elementary than the property of finite branching.

**Example 1.3** (Example 5.3 of [7]). *Let $r \in \mathbb{R}_{>0}$. Consider the operator for deadlock in real-time Basic Process Algebra [12], which can be expressed by the rule*

$$\frac{}{\delta[r] \xrightarrow{\delta[s]} \checkmark} \qquad 0 < s < r.$$

*Process $\delta[r]$ is infinitely branching and has an uncountable set of initials. However, $\delta[r]$ has a finite set of images since for a given time $s$ the only possible transition labelled by $\delta[s]$ is $\delta[r] \xrightarrow{\delta[s]} \checkmark$. The associated LTS is image finite, but it is not finite branching nor initials finite.*

The $\eta$-types of [7] constrain the cardinality of the actions that label the premisses in a rule. In conjunction with the conditions that ensure that a proof tree cannot introduce spurious variables in the targets of transitions, the $\eta$-types being finitely inhabited and the strict stratification are enough to guarantee finite branching. However, finite branching is only one among many bounded-nondeterminism properties, and it is certainly not the most elementary. For the property of initials finiteness, the process $p$ is fixed and the rule format constrains the cardinality of the labels $l$ in transitions $p \xrightarrow{l} p'$ while allowing the targets $p'$ to be unbounded. For the property of image finiteness, the process $p$ and the label $l$ are fixed and the rule format constrains the cardinality of the targets $p'$ in transitions $p \xrightarrow{l} p'$. In order to guarantee these elementary properties, more refined syntactic conditions than the ones in [7] are needed.

# 2 Contributions

**Filtering junk rules.** We introduce the $S$-types that, differently from the $\eta$-types in [7], rely on the *global* information provided by the order relation between processes that is entailed by the strict stratification. The $S$-types filter out those rules for which the sources of the premisses and the sources of the rule are not in the order relation. For instance, the TSS of Example 1.1 has a strict stratification given by

$$\begin{aligned} S(f(p)) &= 1 \\ S(g(p)) &= 0. \end{aligned}$$

An instantiation of the rule template on the right of Example 1.1 has source $f(x)$ and premiss with source $g^i(x)$ for some $i \in \mathbb{N}$, which unify respectively with processes $f(p)$ and $g^i(p)$ (with $p$ any process). The strict stratification $S$ is undefined for $g^i(p)$ with $i \neq 1$, and thus the source of the rule and the source of the premiss are in the order relation only when $i = 1$, i.e., $S(g(p)) < S(f(p))$. The other instantiations of the rule template do not have a valid $S$-type and can be disregarded because they will never take part in a proof tree. The $S$-type of the instantiation where $i = 1$ is finitely inhabited.

**Variables in labels and elementary properties.** We introduce a transformation for transitions that turns the *triadic* representation $p \xrightarrow{l} p'$ into a *dyadic* representation $o \to d$ ($o$ for *origin* and $d$ for *destination*) that places the label $l$ either in the origin, i.e., $(p, l) \to p'$, or in the destination, i.e., $p \to (l, p')$. (As in [2], we assume that nominal terms are closed under Cartesian product.) The rule format is applied verbatim to the dyadic representation of an NTSS. The transformation serves two purposes:

(i) The 0-fold Cartesian product (unit) is the new *administrative* label in all transitions, while the *real* labels are arbitrary terms either in the origin or in the destination. This circumvents duly the concerns about the proof trees introducing spurious variables and about the strict stratification.

(ii) Different dyadic transformations enforce different bounded-nondeterminism properties, i.e., $p \to (l, p')$ for finite branching and $(p, l) \to p'$ for image finiteness. (More on initials finiteness below.) For example, the dyadic transformation for image finiteness of rule ($\mathrm{abs1}_{\mathrm{Ts}}$) of Example 1.2 reads:

$$\frac{(x, y \overset{T}{\mapsto} z) \longrightarrow x' \qquad a\#z \qquad a\#y}{([a]x, y \overset{T}{\mapsto} x) \longrightarrow [a]x'} \,.$$

The LTS for term-for-atom substitution is initials finite because for a given nominal term $t$ and label $a \overset{T}{\mapsto} x$ ($a$ is the atom to be substituted for and $x$ is the subject of the substitution) there is only one result of the substitution, i.e., $t \xrightarrow{a \overset{T}{\mapsto} x} t\{a \mapsto x\}$.

In order to ensure initials finiteness, we relax certain syntactic conditions of the rule format for finite branching as to unconstrain the cardinality of targets $p'$ in the destinations $(l, p')$. We refer to this relaxation of the rule format as *extrusion*.

**Other bounded-nondeterminism properties.** The dyadic transformation for finite branching applies to yet another elementary bounded-nondeterminism property. By extruding the labels instead of the targets the following property is ensured: for every process $p$, the set $\{p' \mid \exists l \text{ s.t. } p \xrightarrow{l} p'\}$ is finite.

Other dyadic transformations in which the label $l$ is itself the origin or the destination are possible, which we have dubbed $l \uparrow (p, p')$ and $(p, p') \downarrow l$. Together with extrusion, our rule format affords for a family of up to eight bounded-nondeterminism properties based on the dyadic transformations, which include finite branching, initials finiteness and image finiteness. We are not aware whether any of the five other properties has received any particular name in the literature. These properties may have relevance in the nominal setting where labels of transitions have a prominent role.

# 3 Future work

We have considered NTSSs *after stripping away freshness assertions*. We conjecture that, for such NTSSs, the freshness assertions can only restrict the cardinality of provable transitions from *infinite* (all) to *cofinite* (all but finitely many), and hence freshness assertions do not have an impact in the bounded-nondeterminism properties. Proving this result is still work in progress.

# References


[1] Samson Abramsky. *Domain theory and the logic of observable properties*. PhD thesis, Department of Computer Science, Queen Mary College, University of London, 1987.

[2] Luca Aceto, Matteo Cimini, Mohammad Reza Mousavi, Michel A. Reniers, and Murdoch James Gabbay. Nominal SOS. To be submitted.

[3] Luca Aceto, Wan Fokkink, and Chris Verhoef. Structural operational semantics. In A. Ponse J.A. Bergstra and S.A. Smolka, editors, *Handbook of Process Algebra*, chapter 3, pages 197–292. Elsevier, 2001.

[4] Bard Bloom. CHOCOLATE: Calculi of Higher Order COmmunication and LAmbda TErms (preliminary report). In Hans-Juergen Boehm, Bernard Lang, and Daniel M. Yellin, editors, *Conference Record of the 21st ACM Symposium on Principles of Programming Languages, Portland, Oregon*, pages 339–347. ACM Press, 1994.

[5] Matteo Cimini, Mohammad Reza Mousavi, Michel A. Reniers, and Murdoch James Gabbay. Nominal SOS. *Electronic Notes in Theoretical Computer Science*, 286:103–116, 2012.

[6] R. de Simone. Higher-level synchronising devices in Meije–SCCS. *Theoretical Computer Science*, 37(3):245–267, 1985.

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

[8] M. J. Gabbay and A. Pitts. A new approach to abstract syntax involving binders. In G. Longo, editor, *Proceedings of the 14th Symposium on Logic in Computer Science, Trento, Italy*, pages 214–224. IEEE Computer Society Press, 1999.

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

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

[11] R. M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, 1976.

[12] A. S. Klusener. *Models and axioms for a fragment of real time process algebra*. PhD thesis, Department of Mathematics and Computing Science, Technical University of Eindhoven, 1993.

[13] Mohammad Reza Mousavi, Michel A. Reniers, and Jan Friso Groote. SOS formats and meta-theory: 20 years after. *Theoretical Computer Science*, 373(3):238–272, 2007.

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

[15] Gordon D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60-61:17–139, 2004.

[16] Christian Urban, Andrew Pitts, and Murdoch J. Gabbay. Nominal unification. *Theoretical Computer Science*, 323(1-3):473–497, 2004.

[17] F. Vaandrager. Expressiveness results for process algebras. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX Workshop on Semantics: Foundations and Applications, Beekbergen, The Netherlands*, volume 666 of *Lecture Notes in Computer Science*, pages 609–638. Springer, 1993.

[18] Rob J. van Glabbeek. Bounded nondeterminism and the approximation induction principle in process algebra. In F. J. Brandenburg, G. Vidal-Naquet, and M. Wirsing, editors, *Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science, Passau, Germany*, volume 247 of *Lecture Notes in Computer Science*, pages 336–347. Springer, 1987.