Information and Computation 249 (2016) 76–109

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

A general SOS theory for the specification of probabilistic transition systems

Pedro R. D'Argenio [a,*], Daniel Gebler [b], Matias David Lee [a,1]

[a] FaMAF, Universidad Nacional de Córdoba – CONICET, Ciudad Universitaria, X5000HUA – Córdoba, Argentina
[b] Dept. of Computer Science, VU University Amsterdam, De Boelelaan 1081a, NL-1081 HV Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 2 February 2015
Received in revised form 11 November 2015
Available online 17 March 2016

Keywords:
SOS
Probabilistic transition systems
Bisimulation
Congruence
Rule format
Full abstraction

ABSTRACT

This article focuses on the formalization of the structured operational semantics approach for languages with primitives that introduce probabilistic and non-deterministic behavior. We define a general theoretic framework and present the $nt\mu f\theta/nt\mu x\theta$ rule format that guarantees that bisimulation equivalence (in the probabilistic setting) is a congruence for any operator defined in this format. We show that the bisimulation is fully abstract w.r.t. the $nt\mu f\theta/nt\mu x\theta$ format and (possibilistic) trace equivalence in the sense that bisimulation is the coarsest congruence included in trace equivalence for any operator definable within the $nt\mu f\theta/nt\mu x\theta$ format (in other words, bisimulation is the smallest congruence relation guaranteed by the format). We also provide a conservative extension theorem and show that languages that include primitives for exponentially distributed time behavior (such as IMC and Markov automata based language) fit naturally within our framework.

1. Introduction

Structural operational semantics (SOS for short) [1] is a powerful tool to provide semantics to programming languages. In SOS, process behavior is described using transition systems and the behavior of a composite process is given in terms of the behavior of its components. SOS has been formalized using an algebraic framework as *Transition Systems Specifications (TSS)* [2–6, etc.]. Basically, a TSS contains a signature, a set of actions or labels, and a set of rules. The signature defines the terms in the language. The set of actions represents all possible activities that a process (i.e., a term over the signature) can perform. The rules define how a process should behave (i.e., perform certain activities) in terms of the behavior of its subprocesses, that is, the rules define compositionally the transition system associated to each term of the language. A particular focus of these formalizations was to provide a meta-theory that ensures a diversity of semantic properties by simple inspection on the form of the rules. Thus, there are results on congruences and full abstraction, conservative extension, security, etc. (See [7,6,8] for overviews and references therein.)

In this article we focus on congruence and full abstraction. A congruence theorem guarantees that whenever the rules of a TSS are in a particular format, then a designated equivalence relation is preserved by every context in the signature of such TSS. Thus, for instance, strong bisimulation equivalence [9] is a congruence on any TSS in the *ntyft/ntyxt* format [4]. Full abstraction is a somewhat dual result. An equivalence relation is fully abstract with respect to a language and a given equivalence relation $\equiv$ if it is the largest relation included in $\equiv$ that is a congruence for all operators in the language [10].

* Corresponding author.
E-mail addresses: dargenio@famaf.unc.edu.ar (P.R. D'Argenio), e.d.gebler@vu.nl (D. Gebler), lee@famaf.unc.edu.ar, matias-david.lee@ens-lyon.fr (M.D. Lee).
[1] Current address: LIP, Université de Lyon, CNRS, Ecole Normale Supérieure de Lyon, INRIA, Université Claude-Bernard Lyon 1, France.

http://dx.doi.org/10.1016/j.ic.2016.03.009
0890-5401/

This notion can be straightforwardly extended to a TSS format by considering *all* definable languages in such format: an equivalence relation is fully abstract with respect to a particular format and an equivalence relation $\equiv$ if it is the largest relation included in $\equiv$ that is a congruence for all operators whose semantics is defined by a TSS in that format [10]. For example, strong bisimilarity is fully abstract w.r.t. the *ntyft/ntyxt* format [4] but not w.r.t. the *tyft/tyxt* format [2] or the GSOS format [3].

The introduction of probabilistic process algebras [11–13, etc.] motivated the need for a theory of structural operational semantics to define *probabilistic* transition systems. Previous to the introduction of our format [14], few results have appeared in this direction [15–18] presenting congruence theorems for (probabilistic) bisimilarity [19,20], but no full abstraction result. All previously mentioned studies consider transitions in the form of a quadruple denoted by $t \xrightarrow{a,q} t'$, where t and t' are terms in the language, a is an action or label, and $q \in (0, 1]$ is a probability value. A transition of that form denotes that term t can perform an action a and with probability q continue with the execution of t'. Moreover, it is required that $\pi_{t,a}$, defined by $\pi_{t,a}(t') = \sum_{t \xrightarrow{a,q} t'} q$, is a probability distribution. (This interpretation corresponds to the reactive view, it varies under the generative view [12].) This notation introduces several problems. The first one is that the transition relation cannot be treated as a set because two different derivations may yield the same quadruple. This requires artifacts like multisets or bookkeeping indexes. The second problem is that formats need to be defined jointly on a set of rules rather than a single rule to ensure that $\pi_{t,a}$ is a probability distribution. (Notice that $\pi_{t,a}$ depends on a *set* of transitions which are obtained using different rules.)

Rather than following this approach, we directly represent transitions as a triple $t \xrightarrow{a} \pi_{t,a}$. Thus, a single triple contains the complete information of the probabilistic jump. Moreover, this representation also allows for non-determinism in the sense that if $t \xrightarrow{a} \pi$ and $t \xrightarrow{a} \pi'$ not necessarily $\pi = \pi'$ as requested by reactive systems. Hence, our *probabilistic transition system specifications (PTSS)* define objects very much like Segala's probabilistic automata [21]. More precisely we represent transitions using two different sorts, one that represent states and the other distributions. Thus a labeled transition $t \xrightarrow{a} \theta$ goes from one *state term* t to a *distribution term* θ. If θ is a closed term, then its interpretations $[\![\theta]\!]$ is a probability distribution on state terms. By having a two-sorted signature, operations can be parameterized on distributions, and moreover, we can neatly express open terms in the rules of the PTSS. So, each (probabilistic) transition $t \xrightarrow{a} \theta$ is obtained as a consequence of a single derivation in our PTSSs, and hence formats focus on single rules (as it is the case for non-probabilistic TSSs). This significantly eases the inspection of the format. In addition, a byproduct of this choice is that the proof strategies for the majority of the lemmas and theorems of this article are much the same as those for their non-probabilistic relatives. We observe that this way of representing transitions in rules for process algebra has already appeared in [22], it is also used in the Segala-GSOS format [16] and it is pretty much related to bialgebraic approaches to SOS [16,23].

More precisely, the contributions on this article are:

1. We introduce PTSS with negative and quantitative premises and the possibility of lookahead (Section 3). It uses a two-sorted term algebra to represent the language, where the *distribution* sort is aimed to be interpreted as a probability distribution on the *state* sort. (Section 2.)
2. We use here the most general method to give meaning to PTSSs: we adapt the definition of least 3-valued stable models [5,24] to our setting and only limit to complete PTSSs when a 2-valued model is required. (Section 3.)
3. We introduce the $nt\mu f\theta/nt\mu x\theta$ format in Section 4 and show through carefully crafted examples that each of the restrictions of the format is effectively needed to ensure that bisimulation is a congruence. Moreover, we present a shorthand notation that significantly simplifies the verification of the restrictions, making the format almost as easy to check as the *ntyft/ntyxt* format [4,5].
4. We give a detailed proof that bisimulation is a congruence for any operator defined within the $nt\mu f\theta/nt\mu x\theta$ format as long as every rule is well-founded. (Section 5.)
5. We adapt the concept of *conservative extension* [2,8] to our probabilistic setting. Conservative extensions allow to modularly extend a language preserving all the behavioral properties of the original terms. We also provide a general theorem that guarantees that an extension is conservative. This is presented in Section 6.
6. We show that bisimulation equivalence is fully abstract with respect to the $nt\mu f\theta/nt\mu x\theta$ format and trace equivalence, that is, it is the coarsest congruence w.r.t. any operator definable in (complete) $nt\mu f\theta/nt\mu x\theta$ PTSSs (Section 7).
7. Finally, we discuss some expressiveness issues. Notably, we show that the theory extends immediately to IMCs [25] and Markov automata [26]. We also provide a format that guarantees that the model of the PTSS is indeed a Markov automaton. (Section 8.)

Besides reporting the full proofs, this article extends, improves and correct the work already presented in [14]. In fact, several mistakes were inadvertently introduced there. Some modifications have already been introduced in [27] but also there we have introduced some error. That is why we briefly report the differences and corrections in Appendix A.

2. Preliminaries

Let $S = \{s, d\}$ be a set denoting two sorts. Elements of sort $s \in S$ are intended to represent states in the transition system, while elements of sort $d \in S$ will represent distributions over states. We let σ range over S.