On the Numerical Verification of Probabilistic Rewriting Systems

Jounaidi Ben Hassen and Sofiène Tahar
Department of Electrical and Computer Engineering
Concordia University
Montreal, Quebec, Canada
{jounaidi,tahar}@ece.concordia.ca

Abstract

We present in this paper a technique for the formal verification of probabilistic systems described in PMAUDE*, a probabilistic extension of the rewriting system Maude. Our methodology is based on a numerical verification using the probabilistic symbolic model checking tool PRISM. In particular, we show how we can construct an abstract system from the runs of a model that preserve all the probabilistic properties of the latter. Then we deduce the probabilistic matrix that will be used for the verification in PRISM.*

1. Introduction

The usefulness of rewrite systems is proved by the large range of their application to distributed real-time systems [6]. Since the *natural* behavior of such processes is inherently stochastic, rewriting techniques have been extended to represent this probabilistic behavior [4]. Due to the increasing complexity of those systems, probabilistic formal verification methods [5] are now being deployed. Ironically, even if probabilistic rewriting techniques can offer a unifying logic for stochastic systems, there are very limited approaches that allow the verification of systems described using these techniques. To the best of our knowledge, the only proposed method is the one introduced in [8] where a tool called VESTA [8] is introduced. This tool supports statistical model-checking algorithms for the transient part of probabilistic computation tree logic (PCTL) [3] and continuous stochastic logic (CSL) [2]. When we consider statistical approaches, we generally have to tolerate that any test procedure we use accepts a false hypothesis. Hence, results can be inaccurate and highly dependent on the samples simulated. Thus, the work that we present in this paper will be the first of its kind to describe a technique for the formal verification of a probabilistic rewriting system using a numerical approach.

Informally, probabilistic rewriting systems are rewriting systems where a probability information is added to the rules that describe the system. In this paper, we will make use of the probabilistic rewriting tool PMAUDE [1], which is an extension of Maude [7]. Kumar *et al.* [4] proposed probabilistic rules of the form:

$$\ell : \alpha(\vec{x}) \longrightarrow \alpha'(\vec{x};\vec{y}) \text{ if } \mathbf{C}(\vec{x}) \text{ with probability } \pi(\vec{x}).$$

$\mathbf{C}(\vec{x})$ is called the condition of ℓ and $\pi(\vec{x})$ is the probability associated to ℓ. The rule will match a state fragment if there is a substitution θ for the variables $\vec{x}$ that makes $\theta(\alpha)$ equal to that state fragment and satisfies $\theta(\mathrm{C}(\vec{x}))$. Kumar *et al.* [4] proposed the probabilistic rewriting temporal logics PRTL and PRTL*, where properties of probabilistic rewrite theories can be expressed.

In probabilistic model checking, the model constructed for analysis is probabilistic. This is usually achieved by labelling transitions between states with information about the likelihood that they will occur. By introducing the probabilistic verification, we can verify probabilistic properties such as: "the message will be delivered with probability 0.98". One of the most performing tools in probabilistic model checking is PRISM [5]. It allows the verification of systems which exhibit probabilistic behavior. Properties to be checked against the constructed model are specified using the PCTL [3] or the CSL [2] temporal logics.

2. Proposed Verification Technique

We consider a PMAUDE module $\mathcal{M} = (\Sigma, E \cup A, L, R, \pi)$ describing a probabilistic system. Σ is a ranked alphabet of function symbols. E is a confluent, terminating and sort-decreasing modulo A. A is a set of equational axioms. L is a set of distinctive labels. R is a set of rules coherent with E modulo A. π is a set of probabilities associated to each rule. For verification purposes, we suppose that the number of system's states is finite and that execution paths can be generated through rewriting derivation steps. Execution paths will be sequences of the form

3-9810801-0-6/DATE06 © 2006 EDAA

$\Pi = s_0 \xrightarrow{p_{0,1}} s_1 \xrightarrow{p_{1,2}} s_2 \ldots$, where each s_i is a state given by $\theta([\alpha_i]_{E/A})$. $[\alpha_i]_{E/A}$ is the canonical form of a term α_i. θ is a substitution on $[\alpha_i]_{E/A}$. To avoid nondeterminism, we suppose also that at most, only one θ can be applied to a term $[\alpha_i]_{E/A}$ at one time for a given rule $r \in R$. The probability $p_{i,i+1} \in [0,1]$ is the probability to move from the state s_i to s_{i+1}. Therefore, we suppose that a probability space can be defined on the execution paths of the model in such a way that the paths satisfying any path formula in our concerned logic (PRTL) is measurable. Our technique is based on three steps: definition of an abstract model, translation of the PRTL formula and the verification using PRISM.

Model Abstraction: R can be considered as the union of two disjoint sets R_l and R_g. R_l is the set of rules that have a *local* transformation on the system and R_g is the set of rules that when executed transform the global state of the system. This set should include all the probabilistic rules. If a rule $r_g \in R_g$ is non probabilistic, we assume that its associated probability equals 1. A step derivation is said to be *local* (respectively *global*) if it is obtained from the execution of a rule $r \in R_l$ (respectively $r \in R_g$). We should emphasize that by definition, a rule $r : \alpha \longrightarrow \alpha'$ is interpreted as a *transition* in a concurrent system.
We reduce in $\mathcal{M}$ a ground term t_0. This term represents the initial state of $\mathcal{M}$. We construct an abstract state-transition system from the runs of the initial model by *hiding* all the local step derivations of R_l and all the simplifications and reductions of E/A. The obtained runs describe a state-transition system $\widehat{\mathcal{M}} = (\widehat{S}, \widehat{T}, \widehat{\mathbf{P}})$. A state $\widehat{s_i} \in \widehat{S}$ represents a *global* state in $\mathcal{M}$. It describes one or more local derivations and simplifications over a term α_i. Thus, $\widehat{s_i} \equiv [\alpha_i]_{E/A} \xrightarrow{*}_{(R_l)_{E/A}} \widehat{[\alpha_i]_{E/A}}$. Every transition $\widehat{t_{i,j}}$ represents an evolution of the system from a state $\widehat{s_i}$ to a state $\widehat{s_{i+1}}$. In $\mathcal{M}$, this can be viewed as an application of a *probabilistic* rule $r_{g_i} \in R_g$ over the term $\widehat{[\alpha_i]_{E/A}}$ with a probability $\pi(\theta(\widehat{[\alpha_i]_{E/A}}))$. Therefore, the probability $\widehat{p_{i,j}} \in \widehat{\mathbf{P}}$ associated to the transition $\widehat{t_{i,j}}$ is equal to the probability associated to r_{g_i} when applied on the term $\widehat{[\alpha_i]_{E/A}}$, i.e, $\pi(\theta(\widehat{[\alpha_i]_{E/A}}))$. By distinguishing the local derivations from the global ones, we were able to construct an abstract system $\widehat{\mathcal{M}}$ with only the states and transitions that have an impact on the global behavior of our system.

Formula Transformation: Kumar *et al.* [4] showed how to represent probabilistic temporal logics such as PCTL and CSL under PRTL. Since in our case the verification will be done on the abstract model $\widehat{\mathcal{M}}$, an extra *mapping* should be performed over the variables in the initial property P_{PRTL} in order to obtain the corresponding property $\widehat{P}$. This translation replaces every α_i in P_{PRTL} by its associated state $\widehat{s_i}$ (=$\widehat{[\alpha_i]_{E/A}}$) of the model $\widehat{\mathcal{M}}$ in $\widehat{P}$.

Verification with PRISM: From the probabilistic matrix associated to $\widehat{\mathcal{M}}$, we generate a PRISM model, then, we introduce the property $\widehat{P}$ to be verified. PRISM performs the numerical probabilistic model checking to ascertain whether or not this property is satisfied by the model $\widehat{\mathcal{M}}$. If the property is verified by $\widehat{\mathcal{M}}$ then it is verified by our module $\mathcal{M}$ since $\widehat{\mathcal{M}}$ preserves all the probabilistic properties of $\mathcal{M}$.

3. Conclusion

We presented a technique for the formal verification of probabilistic systems described as probabilistic rewriting theories. Our methodology is based on the construction of an abstract model that preserves all the global transitions and the probabilistic properties of the initial model. This abstraction reduces the size of the system to be verified. Then, PRISM can be used for the numerical verification of the abstract model. Our methodology constitutes the first proposal of its kind attempting to verify a probabilistic rewriting system using a numerical method. Further work should be done especially to define the sublanguage of PMAUDE that can be verified using this method, the implementation of the proposed methodology and its application on several real world case studies.

References

[1] G. Agha, J. Meseguer, and K. Sen. PMAUDE: Rewrite-based specification language for probabilistic object systems. In *QAPL'05*, Edinburgh, Scotland, April 2005.

[2] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-checking continuous-time markov chains. *ACM Transactions in Computer Logic*, 1(1):162–170, 2000.

[3] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *FST & TCS'95*, volume 1026 of *LNCS*, pages 499–513, 1995.

[4] N. Kumar, K. Sen, J. Meseguer, and G. Agha. Probabilistic rewrite theories: Unifying models, logics and tools. Technical Report UIUCDCS-R-2003-2347, Department of computer science, University of Illinois at Urbana-Champaign, 2003.

[5] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 2.0: A tool for probabilistic model checking. In *QEST'04*, pages 322–323, Enschede, The Netherlands, September 2004.

[6] N. Martí-Oliet and J. Meseguer. Rewriting logic: roadmap and bibliography. *Theoretical Computer Science*, 285(2):121–154, 2002.

[7] J. Meseguer. A logical theory of concurrent objects and its realization in the maude language. In *Research Directions in Concurrent Object-Oriented Programming*, pages 314–390. MIT Press, 1993.

[8] K. Sen, M. Viswanathan, and G. Agha. VESTA: A statistical model checker and analyzer for probabilistic systems. In *QEST'05*, Torino, Italy, September 2005.