A CTL Model Checker for Stochastic Automata Networks*

Lucas Oleksinski, Claiton Correa, Fernando Luís Dotti, and Afonso Sales**

PUCRS - FACIN, Porto Alegre, Brazil
{lucas.oleksinski,claiton.correa}@acad.pucrs.br,
{fernando.dotti,afonso.sales}@pucrs.br

Abstract. Stochastic Automata Networks (SAN) is a Markovian formalism devoted to the quantitative evaluation of concurrent systems. Unlike other Markovian formalisms and despite its interesting features, SAN does not count with the support of model checking. This paper discusses the architecture, the main features and the initial results towards the construction of a symbolic CTL Model Checker for SAN. A parallel version of this model checker is also briefly discussed.

1 Introduction

Stochastic Automata Networks (SAN) was proposed by Plateau [12], being devoted to the quantitative evaluation of concurrent systems. It is a Markovian formalism that allows modeling a system into several subsystems which can interact with each other. Subsystems are represented by automata and interactions by synchronizing transitions of cooperating automata on same events. Dependencies among automata can also be defined, using functions. Functions evaluate on the global state of the automata network and can be used to specify the behavior of specific automata. The use of functions allows the description of complex behaviors in a very compact way [1]. Quantitative analysis of SAN models is possible using specialized software tools (*e.g.*, PEPS [13] or SAN Lite-Solver [14]), fundamentally allowing one to associate probabilities to the states of the model, using a steady state or transient analysis.

While developing models for involved situations it is highly desirable to reason about their computation histories and thus model checking becomes important. Indeed, many formalisms for quantitative analysis count with the support of specialized model checking tools. In the context of CTMC-based model checking, we can mention PRISM [8], SMART [4] and CASPA [7]. Such support however is lacking for SAN. In this paper we report our results towards the construction of the first SAN model checker. In this initial version, the tool is restricted to CTL model checking opposed to the stochastic verification as offered by the aforementioned tools.

* Paper partially sponsored by CNPq (560036/2010-8) and FAPERGS (PqG 1014867).
** Afonso Sales receives grant from PUCRS (Edital 01/2012 – Programa de Apoio à Atuação de Professores Horistas em Atividades de Pesquisa na PUCRS).

K. Joshi et al. (Eds.): QEST 2013, LNCS 8054, pp. 286–289, 2013.

2 Tool Overview

Fig. 1 illustrates the main processing steps of the SAN model checker. It has as input a model written in the SAN modeling language [13], a CTL (*Computation Tree Logic*) property, and an additional information if a witness or a counterexample to the property is desired. As output it offers the answer whether the property is true or false, and a witness or counterexample as chosen. The tool supports the standard CTL where atomic propositions are assertions about the global state of the automata network according to the SAN language [13].

The compilation of the SAN model generates a Markovian descriptor which is used as the system transition relation, *i.e.*, a set of tensors which operated by generalized Kronecker algebra allows the achievement of next states. The initial states of the model are those considered as reachable in the reachability declaration of the SAN model. Multi-valued Decision Diagrams (MDD) are used to encode the Reachable State Space (RSS) of the SAN model, which is calculated using an extension [15] of the saturation based approach [3]. The satisfaction sets calculation (SAT in Fig. 1) follows a breadth-first search algorithm. During this process, the RSS is labelled with all subformulas of the input formula.

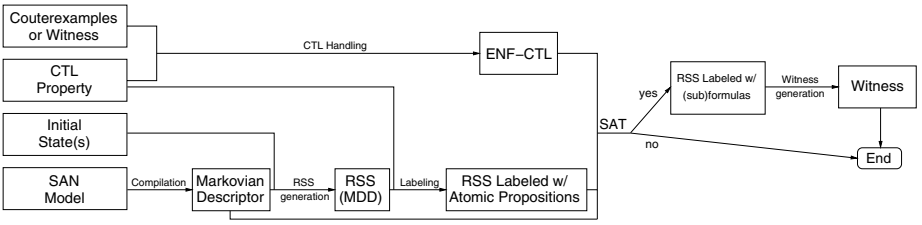

Fig. 1. The tool architecture

Whenever a counterexample is desired, the tool negates the input formula to generate a witness. The witness generator supports ENF-CTL operators and generates trace structured witnesses. To enrich witness information, whenever a branching is avoided the respective state of the trace is annotated with the subformula that holds from that state.

A parallel approach was proposed that replicates the entire RSS and assigns specific partitions of the state space to be computed by different nodes. Each node may locally compute successor states even these cross partition borders, without requiring communication. Communication is only required for fix-point calculation, which is executed as rounds of synchronization between nodes.

3 Experiments

We report CTL model checking results[1] on both sequential and parallel implementations of the model checker through set of experiments with two different models: the dining philosophers (DP) problem [15] and a model for an *ad hoc*

[1] As mentioned, our tool does not perform stochastic verification and thus numerical analysis is not carried out.