Science of Computer Programming 128 (2016) 22–50

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

PTRebeca: Modeling and analysis of distributed and asynchronous systems

Ali Jafari [a,*], Ehsan Khamespanah [a,b], Marjan Sirjani [a,c], Holger Hermanns [d], Matteo Cimini [e]

[a] Reykjavik University, School of Computer Science and CRESS, Iceland
[b] University of Tehran, School of ECE, Islamic Republic of Iran
[c] Mälardalen University, Embedded Systems, Sweden
[d] University of Saarland, School of Computer Science, Germany
[e] Indiana University, Bloomington, Center for Research in Extreme Scale Technologies, United States

ARTICLE INFO

Article history:
Received 16 May 2015
Received in revised form 12 March 2016
Accepted 14 March 2016
Available online 31 March 2016

Keywords:
Probabilistic Timed Automata
Timed Markov Decision Process
IMCA model checker
Probabilistic Timed Rebeca
Model checking
Performance analysis

ABSTRACT

Distributed systems exhibit probabilistic and non-deterministic behaviors and may have time constraints. Probabilistic Timed Rebeca (PTRebeca) is introduced as a timed and probabilistic actor-based language for modeling distributed real-time systems with asynchronous message passing. The semantics of PTRebeca is a Timed Markov Decision Process. In this paper, we provide SOS rules for PTRebeca, introduce a new tool-set and describe the corresponding mappings. The tool-set automatically generates a Markov Automaton from a PTRebeca model in the form of the input language of the Interactive Markov Chain Analyzer (IMCA). The IMCA can be used as a back-end model checker for performance analysis of PTRebeca models against expected reachability and probabilistic reachability properties. Comparing to the existing tool-set, proposed in the conference paper, we now have the ability of analyzing significantly larger models, and we also can add different rewards to the model. We show the applicability of our approach and efficiency of our tool by analyzing a Network on Chip architecture as a real-world case study.

1. Introduction

Our modern society more and more relies on software systems that are distributed and consist of concurrently executing components which communicate asynchronously over networks. Modeling and analyzing these complex systems is a non-trivial and intricate task. There is thus a need for modeling languages that match well with computational models of such systems, and are supported by tools for analyzing performance and dependability aspects of these systems.

A well-established paradigm for modeling the functional behavior of distributed and asynchronous systems is the actor model. Actor model is introduced by Hewitt as an agent-based language for programming distributed systems [1], and is later developed by Agha [2–4] into a concurrent object-based model. Actors are distributed, autonomous objects that interact via asynchronous message passing. Building on an event-driven and message-based foundation, actors provide scalability and are easy-to-grasp concurrency models. With the growth of cloud computing, web services, networks of embedded computers, and multicore architectures, programming using the actor model has become increasingly relevant.

* Corresponding author. Tel.: +354 776 6603, +98 936 723 6840.
E-mail address: ali11@ru.is (A. Jafari).

http://dx.doi.org/10.1016/j.scico.2016.03.004
0167-6423/

Popular actor programming languages and frameworks include Erlang [5] and the Scala/Akka family [6]. Many projects in industry, e.g. at Google (like DART) and Microsoft (like Asynchronous Agents Library), have explored the actor model. Large applications such as Twitter's message queuing, image processing in MS Visual Studio 2010, as well as the Vendetta game engine [7] have been designed on the basis of this model.

Rebeca [8,9] is an actor-based modeling language designed to enable formal verification of actor models. It hence bridges the gap between formal methods and software engineering. Using Rebeca we can deploy a model-driven development approach with a formal basis. Rebeca is supported by formal verification tools and techniques which are based on the formal semantics of the language [10]. An extension of Rebeca [11] has been proposed to provide the ability of modeling and verification of distributed systems with real-time constraints. In this context, Floating Time Transition System (FTTS) is introduced to significantly reduce the state space generated when model checking Timed Rebeca (TRebeca) models [12]. Checks for absence of deadlock freedom and schedulability analysis of TRebeca models can be performed using FTTS.

Since its introduction, TRebeca has been used in different areas. Examples include the analysis of different routing algorithms and scheduling policies in NoC (Network on Chip) designs [13,14], as well as schedulability analysis of distributed real-time sensor network applications [15], more specifically a real-time continuous sensing application for structural health monitoring in [16]. In analyzing the above mentioned applications, we observed the need for modeling probabilistic behavior. In an earlier work, pRebeca has beem proposed as an extension of Rebeca to model probabilistic systems [17]. However, pRebeca does not support the timing features.

In [18], we proposed Probabilistic Timed Rebeca (PTRebeca) which benefits from and integrates modeling features of TRebeca and pRebeca, combining their respective syntax. This aims at enhancing our modeling ability in order to cover more properties, so as to support performance evaluation of probabilistic real-time actors. To keep the consistency, we designed the syntax of PTRebeca as a combination of TRebeca and pRebeca. Still, as to be expected, existing formal semantics and supporting tools are not directly applicable to PTRebeca. Consequently, Timed Markov Decision Processes (TMDP) are used as the semantics of PTRebeca, to support timing, probabilistic, and non-deterministic features. TMDP can be regarded as the discrete-time semantics of probabilistic timed automata (PTA) [19], or as variation of interactive probabilistic chains [20]. For performance evaluation of PTRebeca models we employ probabilistic model checking, for both functional verification and performance evaluation. The benefits of combining performance evaluation with functional verification is elaborated upon in [21].

This paper is an extended version of the paper presented at AVoCS conference [18]. In this paper, we provide Structural Operational Semantics (SOS rules) for the PTRebeca language in the style of Plotkin [22]. The tool developed in [18] uses PRISM [23] as a back-end model checker while in this paper we instead use the IMCA (Interactive Markov Chain Analyzer) tool [24]. In our conference paper [18], we mapped a PTRebeca model to a single, flat Markov Decision Process (MDP) module. This approach is consistent with the semantics of PTRebeca. But as the entire PTRebeca model is mapped into a single MDP module, the module becomes prohibitively large. As a consequence, the analysis time is very high. To overcome this problem, we used the explicit engine of PRISM which works with an intermediate transition matrix representation. This allows us to analyze larger models, but PRISM does not provide full support for this format. Therefore, we were only able to use it for the analysis of probabilistic reachability properties, but not for the expected reachability ones.

An alternative way, detailed in the present paper, maps each component (reactive object) in a PTRebeca model to a Probabilistic Timed Automaton (PTA). Then the parallel composition of PTA (of all components) represents the behavior of the PTRebeca model. We call this approach the parallel composition approach and we will show this approach in Section 4.3. In Section 4.3, we will also compare the parallel composition approach with TMDP semantics. We will demonstrate that the state space generated via the TMDP semantics (proposed in the conference paper) is much smaller than the state space generated from the parallel composition approach. So, although this way we can use the full power of PRISM the state space explosion problem occurs very quickly.

To deal with the restriction of the previous approaches, as explained above, we turned to the IMCA (Interactive Markov Chain Analyzer) model checker [24], and we used it as the back-end model checker for the analysis of PTRebeca models. IMCA accepts Markov Automata (MA) [25] and Interactive Markov Chain (IMC) [26] models. An MA-transition is either labelled with an action (probabilistic transition), or with a positive real number representing the rate of a negative exponential distribution (Markovian transition). An action (probabilistic) transition leads to a discrete probability distribution over states. MA can thus model action transitions as in labelled transition systems, probabilistic branching, as well as delays that are governed by exponential distributions [27].

In order to use IMCA as a back-end model checker, we need to convert the TMDP of an underlying PTRebeca model to its corresponding MA. There are two types of transitions in a TMDP: action (probabilistic) transition, leading to a discrete probability distribution over states, and delay transition, carrying a positive integer value. Probabilistic transitions in TMDP are mapped directly to probabilistic transitions in MA. For the transition rate in the MA, the inverse of the integer value of a delay transition in TMDP is considered as the rate of the corresponding transition in the MA. This conversion is proved to be correct for checking expectation properties, and not for time-bounded reachability property. Therefore, we can use our previously developed tools to generate the TMDP of our models automatically. The obtained TMDP is converted to its Markov Automaton which is then the input to IMCA. Using this approach, we are able to evaluate the performance of our models against probabilistic reachability, expected reward reachability, and expected time reachability properties. In Section 5, we mathematically prove that the values of expected time reachability in TMDP and its corresponding MA are identical.