Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Confluence reduction for Markov automata[☆]

Mark Timmer[a,*], Joost-Pieter Katoen[a,b], Jaco van de Pol[a], Mariëlle Stoelinga[a]

[a] *Formal Methods and Tools, Faculty of EEMCS, University of Twente, The Netherlands*
[b] *Software Modelling and Verification, RWTH Aachen University, Germany*

A R T I C L E I N F O

Article history:
Received 7 July 2015
Accepted 12 January 2016
Available online xxxx

Keywords:
Markov automata
Confluence
State space reduction
Process algebra
Divergence-sensitive branching bisimulation
Partial order reduction

A B S T R A C T

Markov automata are a novel formalism for specifying systems exhibiting nondeterminism, probabilistic choices and Markovian rates. As expected, the state space explosion threatens the analysability of these models. We therefore introduce confluence reduction for Markov automata, a powerful reduction technique to keep them small by omitting internal transitions. We define the notion of confluence directly on Markov automata, and discuss additionally how to syntactically detect confluence on the process-algebraic language MAPA that was introduced recently. That way, Markov automata generated by MAPA specifications can be reduced on-the-fly while preserving divergence-sensitive branching bisimulation. Three case studies demonstrate the significance of our approach, with reductions in analysis time up to an order of magnitude.

1. Introduction

Markov automata (MAs) [1–3] are an expressive model incorporating concepts such as random delays, probabilistic branching, as well as nondeterminism. They are compositional and subsume Segala's probabilistic automata (PAs), Markov decision processes (MDPs), continuous-time Markov chains (CTMCs), interactive Markov chains (IMCs) and continuous-time Markov decision processes (CTMDPs). Their large expressiveness turns the MA into an adequate semantic model for high-level modelling formalisms of various application domains. So far, MAs have been used to provide an interpretation to (possibly confused) generalised stochastic Petri nets (GSPNs) [4], a widely used modelling formalism in performance engineering. They have acted as compositional semantics of dynamic fault trees [5], a key model in reliability engineering, exploited for component-based system architecture languages [6,7], and for scenario-aware data-flow computation [8].

The classical analysis of sub-models such as MDPs and CTMCs is typically state-based, and so are the more recently developed quantitative analysis techniques for MAs [9]. As a result, the analysis of MAs suffers from the problem of state space explosion—the curse of dimensionality. A major source of this problem is the occurrence of concurrent, independent transitions. Hence, this paper introduces an *on-the-fly* reduction technique for MAs that—akin to partial-order reduction—is based on detecting commutative transitions that typically arise from the parallel composition of largely independent components. This is done by generalising the technique of *confluence reduction* [10–12] to MAs. The crux of this approach is to detect so-called confluent sets of invisible transitions (i.e., internal stutter transitions). While generating the state space, these con-

[☆] This research has been partially funded by NWO under grant 612.063.817 (SYRUP), by STW under grant 12238 (ArRangeer), and the EU under grant 318490 (SENSATION).

[*] Corresponding author.
E-mail addresses: timmer@cs.utwente.nl (M. Timmer), katoen@cs.rwth-aachen.de (J.-P. Katoen), vdpol@cs.utwente.nl (J. van de Pol), marielle@cs.utwente.nl (M. Stoelinga).

http://dx.doi.org/10.1016/j.tcs.2016.01.017
0304-3975/© 2016 Elsevier B.V. All rights reserved.

fluent sets are given priority over their neighbouring transitions. This yields a reduced MA that is divergence-sensitive (probabilistic) branching bisimilar [13] to the original one. Besides being an on-the-fly reduction technique, an advantage of confluence reduction is that it can be done *symbolically*, i.e., it can be applied directly on a high-level description of the model.

Lifting the notion of confluence [10–12] to MAs is subject to various subtleties. This paper discusses these subtleties thoroughly and carefully justifies the definition of confluence for MAs. Although the presence of random delays does not necessarily impact the notion of confluence compared to earlier variants for probabilistic systems, it does complicate the correctness proofs and necessitates the preservation of divergences when reducing based on confluence.

The central concept in this paper is the definition of *confluent sets* of transitions. It is shown that confluent sets are closed under union, as opposed to earlier work on confluence reduction [11,12], and that confluent transitions connect divergence-sensitive branching bisimilar states. To obtain a reduced MA efficiently, we present a mapping of states to their representatives. We show that confluence can be detected symbolically by treating in detail how confluence detection can be performed on specifications in the data-rich process-algebraic language MAPA [14]. This results in a technique to generate reduced MAs on-the-fly in a symbolic fashion, i.e., while generating the state space from a MAPA specification. We discuss heuristics so as to carry out confluence reduction efficiently. Case studies applying these techniques demonstrate state space reductions with factors up to five, decreasing analysis time sometimes with more than 90%. The obtained symbolic, on-the-fly state space reductions reduce up to 90% of the states that could potentially have been reduced using direct branching-bisimulation minimisation.

Related work Although this paper is inspired by earlier approaches on confluence reduction for process algebras [11,12], there are important differences. First, our notion considers state labels in addition to observable actions, thus lifting confluence reduction to a larger class of systems. Secondly, we consider divergence-sensitivity, i.e., infinite internal behaviour, hence preserving minimal reachability probabilities (inspired by [15]). Third, we correct a subtle flaw in [11,12] by introducing a classification of the interactive transitions. In this way, confluent sets are closed under union.[1] This property is key to the way we detect confluence on MAPA specifications. Finally, we allow random delays and hence prove correctness of confluence reduction for a larger class of systems.

Confluence reduction is akin to partial-order reduction (POR) [16–20]. These techniques are based on ideas similar to confluence, choosing a subset of the outgoing transitions per state (often called ample, stubborn or persistent sets) to reduce the state space while preserving a certain notion of bisimulation or trace equivalence. The ample set approach has been successfully extended to MDPs [21–23]. For Segala's PAs, it has been demonstrated that confluence reduction is more powerful in theory as well as practice when restricting to the preservation of branching-time properties [15,24]. To the best of our knowledge, POR has not been adapted to MAs, or to continuous-time Markov models in general.

The theory of MAs has been equipped with notions of strong and weak bisimulations [1–3]. These notions are congruences with respect to parallel composition. Whereas strong bisimulation can be checked in polynomial time, it is an open question whether this also holds for weak bisimulation. As we show in this paper, checking confluence symbolically can be done in polynomial time in the size of the process algebraic description. In addition, confluence reduction is an on-the-fly reduction technique, whereas bisimulation typically is based on a partition-refinement scheme that requires the state space prior to the minimisation.

Organisation of the paper We introduce Markov automata in Section 2, followed by an informal introduction to the concept of confluence reduction in Section 3. Section 4 introduces our notion of confluence for MAs, proves closure under union, and shows that confluent transitions connect divergence-sensitive branching bisimilar states. Section 5 presents our state space reduction technique based on confluence and representation maps. Section 6 provides a characterisation for detecting confluence on MAPA specifications. This is applied to several case studies in Section 7. Finally, Section 8 motivates our design choices by discussing the disadvantages of possible (naive) variations, and Section 9 concludes.

This paper extends [25] by more extensive explanations, state labels for MAs, proofs (in the appendix) and a discussion of alternative confluence notions that may seem reasonable, but turn out to be inadequate.

2. Preliminaries

Definition 1 *(Basics).* A *probability distribution* over a countable set S is a function $\mu \colon S \to [0,1]$ such that $\sum_{s \in S} \mu(s) = 1$. For $S' \subseteq S$, let $\mu(S') = \sum_{s \in S'} \mu(s)$. We define *supp*$(\mu) = \{s \in S \mid \mu(s) > 0\}$ to be the *support* of μ, and write $\mathbb{1}_s$ for the *Dirac distribution* for s, determined by $\mathbb{1}_s(s) = 1$. Sometimes, we use the notation $\mu = \{s_1 \mapsto p_1, s_2 \mapsto p_2, \dots, s_n \mapsto p_n\}$ to denote that $\mu(s_1) = p_1$, $\mu(s_2) = p_2$, ..., $\mu(s_n) = p_n$.

We use $\mathcal{P}(S)$ to denote the *power set* of S, and write $\text{Distr}(S)$ for the set of all *discrete probability distributions* over S. We use $\text{SDistr}(S)$ for the set of all *substochastic* discrete probability distributions over S, i.e., all functions $\mu \colon S \to [0,1]$

[1] Our approach resembles [11,12] to a substantial degree, albeit that [11,12] consider a less expressive model and does not preserve divergences. Whereas the theoretical set-up in [11,12] does not guarantee closure under union—although that is assumed—the corresponding implementations did work correctly. We show in this paper that an additional technical restriction (the confluence classification) is needed to remedy the theoretical flaw. This restriction happens to be satisfied in the old implementations (in the same way as in ours). As the confluence reductions in [11,12] are restricted variants of our notion, they could be fixed by introducing a confluence classification precisely in the same way as we do here.