Confluence Reduction for Markov Automata*

Mark Timmer Jaco van de Pol Mariëlle Stoelinga
Formal Methods and Tools, Faculty of EEMCS
University of Twente, The Netherlands
{timmer, vdpol, marielle}@cs.utwente.nl

Markov automata are a novel formalism for specifying systems exhibiting nondeterminism, probabilistic choices and Markovian rates. Recently, the process algebra MAPA was introduced to efficiently model such systems. As always, the state space explosion threatens the analysability of the models generated by such specifications. We therefore introduce confluence reduction for Markov automata, a powerful reduction technique to keep these models small. We define the notion of confluence directly on Markov automata, and discuss how to syntactically detect confluence on the MAPA language as well. 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

Over the past two decades, model checking algorithms were generalised to handle more and more expressive models. This now allows us to verify probabilistic as well as hard and soft real-time systems, modelled by timed automata, Markov decision processes, probabilistic automata, continuous-time Markov chains, interactive Markov chains, and Markov automata. Except for timed automata—which incorporate real-time deadlines—all other models are subsumed by the Markov automaton (MA) [7, 6, 5]. MAs can therefore be used as a semantic model for a wide range of formalisms, such as generalised stochastic Petri nets (GSPNs) [9], dynamic fault trees [3], Arcade [2] and the domain-specific language AADL [4].

Before the introduction of MAs, the above models could not be analysed to their full extent. For instance, the semantics of a (potentially nondeterministic) GSPN were given as a fully probabilistic CTMC. To this end, weights had to be assigned to resolve the nondeterminism between immediate transitions. As argued in [8], it is often much more natural to omit most of these weights, retaining rates and probability as well as nondeterminism, and thus obtaining an MA. For example, consider the GSPN in Figure 1(a), taken from [6]. Immediate transitions are indicated in black, Markovian transitions in white, and we assume a partial weight assignment. The underlying MA is given in Figure 1(b). We added a selfloop labelled *target* to indicate a possible state of interest (having one token in p_3 and p_4), and for convenience labelled the interactive transitions of the MA by the immediate transition of the GSPN they resulted from (except for the probabilistic transition, which is the result of t_3 and t_4 together).

Recently, the data-rich process-algebraic language MAPA was introduced to efficiently specify MAs in a compositional manner [10]. As always, though, the state space explosion threatens the feasibility of model checking, especially in the presence of data and interleaving. Therefore, reduction techniques for MAs are vital to keep the state spaces of these models manageable. In this presentation we introduce such a technique, generalising *confluence reduction* to MAs. It is a powerful state space reduction technique based on commutativity of transitions, removing spurious nondeterminism often arising from the parallel

*This research has been partially funded by NWO under grants 612.063.817 (SYRUP), 12238 (ArRangeer) and Dn 63-257 (ROCKS), and EU under 318490 (SENSATION).

Submitted to:
QAPL 2013

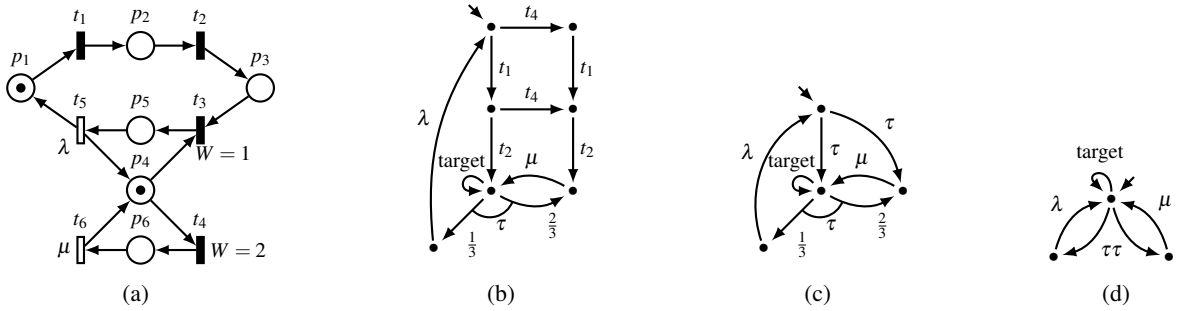


Figure 1: A GSPN and the corresponding unreduced and reduced state spaces.

composition of largely independent components. To the best of our knowledge, it is the first technique of this kind for MAs. We give heuristics to apply confluence reduction directly on specifications in the MAPA language, reducing them on-the-fly while preserving divergence-sensitive branching bisimulation.

To illustrate confluence reduction, reconsider the MA in Figure 1(b) and consider all t_i-transitions to be invisible (and hence labelled with τ). We are able to detect automatically that several τ-transitions in this system are superfluous; they can thus safely be omitted, without losing any behaviour. Figure 1(c) shows the reduced state space, generated on-the-fly using confluence reduction. If all weights are omitted from the specification, an even smaller reduced state space is obtained (Figure 1(d)), while the only change in the unreduced state space is the substitution of the probabilistic choice by a nondeterministic choice.

2 Approach

We generalise strong probabilistic confluence [11] from probabilistic automata to the Markovian realm. Weaker variants are conceivable, but in a process-algebraic context—where confluence is detected heuristically over a syntactic description of a system—it is most practical to apply strong confluence.

Although MAs in addition to a PA's interactive transitions may also contain Markovian transitions, these are irrelevant for confluence. After all, states having a τ-transition can never execute a Markovian transition, due to the maximal progress assumption. Hence, such transitions need not be mimicked, and the original notions of confluence for PAs might seem to still work for MAs. This is not entirely true, though, since the old definition was not yet divergence sensitive and hence might loose divergences; when applying the old concepts to MAs, this could erroneously enable Markovian transitions that were disabled in the presence of divergence due to the maximal progress assumption. Additionally, the old definitions had a subtle flaw: earlier work relied on the assumption that confluent sets are closed under unions [1, 11]. In practical applications this indeed was a valid assumption, but for the theoretical notions of confluence this was not yet the case. This closure property is key, though, to the way we detect confluence on MAPA specifications. Hence, we now improve on the definition to resolve both issues. We also took the opportunity to improve on the way that equivalence of distributions is defined, making it slightly more powerful and, in our view, easier to understand.

So, in comparison to earlier approaches, our definition of confluence is different in three ways:

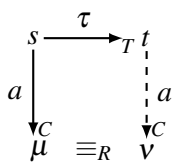
- It is now able to handle MAs.
- It now preserves divergences and hence minimal reachability probabilities.
- It is now closed under unions.

We give an informal overview of the way confluence is defined and how it can be applied to reduce MAs. We first discuss theoretical conditions for a set of transitions to be confluent, then briefly discuss how to reduce a state space based on this information, and finally mention how we instantiate this framework in the context of the MAPA process algebra.

Confluent sets of transitions. To obtain closure under unions, we basically assume the interactive transitions of an MA to be *grouped*. Then, confluence is defined on these groups: either all transitions of a group are confluent, or none of them are. For a group of interactive transitions to be confluent, it should satisfy a few conditions:

1. All transitions in the group are invisible (i.e., are labelled by τ).
2. All transitions in the group have a Dirac distribution to decide their next state.
3. All behaviour possible prior to any transition in the group is still be possible after it.

To illustrate the third requirement, consider the confluence diagram on the right. It is a simplified visualisation of the conditions for a group T of interactive transitions to indeed be confluent. Basically, it states that for any transition $s \xrightarrow{\tau} t$ in the set T, it should hold that t can *mimic* all other transitions $s \xrightarrow{a} \mu$ enabled from s. More precisely, if $s \xrightarrow{a} \mu$, then there should be a transition $t \xrightarrow{a} \nu$ such that μ and ν are related. Without going into technicalities, this boils down to all states in the support of μ and ν to be connected by transitions from T. Finally, the grouping comes into play by requiring that the transition $s \xrightarrow{a} \mu$ and its mimicking transition $t \xrightarrow{a} \nu$ are in the same group.



Although the newly added restriction of grouping may appear restrictive, in practice it is obtained naturally. Transitions are often instantiations of higher-level constructs, and are therefore easily grouped together. Then, it makes sense to detect the confluence of such a higher-level construct (as we indeed do and will demonstrate). Alternatively, all confluent transitions could just be put in one big group.

State space reduction using confluence. We proved that transitions satisfying the confluence requirements above connect divergence-sensitive branching bisimilar states. We can thus reduce state spaces by prioritising them, i.e., omitting all other transitions from a state that also enables a confluent transition. Better still, we aim at omitting all intermediate states on a confluent path altogether; after all, they are all bisimilar anyway. Confluence even dictates that all visible transitions and divergences enabled from a state s can directly be mimicked from any state reachable from s by confluent transitions. Hence, we can just keep following a confluent path and only retain the last state. To avoid getting stuck in an infinite confluent loop, we detect entering a bottom strongly connected component (BSCC) of confluent transitions and choose a unique *representative* from this BSCC for all states that can reach it.

Confluence detection of MAPA specifications. We use heuristics to detect Markovian confluence in the context of the process-algebraic modelling language MAPA. This way, we can reduce *on-the-fly* to obtain a smaller state space without first generating the unreduced one. Each MAPA specification can be transformed to an MLPPE: a structured specification partitioned into a set of *summands* (which can be seen as symbolic transitions). We group all transitions based on the summand they originate from, and check for confluence of each summand by seeing if its action is τ, its distribution is Dirac and if it commutes with all transitions possibly generated by every summand. We over-approximate this commutativity by checking whether, when two summands are enabled, they do not disable each other and do not influence each other's actions, probabilities and next states. After all, that implies that each transition can be mimicked by a transition from the same summand (and hence also that it is indeed mimicked by the same group).

Case studies Case studies with our tool SCOOP on several instances of three different models (a processor grid, a leader election protocol and a queueing system) show state space reductions from 26% to 83%. We also linked SCOOP to the IMCA model checker to illustrate the significant impact of these reductions on the expected time, time-bounded reachability and long-run average computations. Due to confluence reduction, for some models the entire process from MAPA specification to results is now more than fifteen times as fast.

3 Conclusions

We introduced confluence reduction for MAs: the first reduction technique for this model that abstracts from invisible transitions. It preserves branching bisimulation, and hence yields quantitatively behavioural equivalent models. In addition to working on MAs, our novel notion of confluence reduction has two additional advantages over previous notions. First, it preserves divergences, and hence does not alter minimal reachability probabilities. Second, it is closed under unions, enabling us to separately detect confluence of different sets of transitions and combine the results. The representatives approach can still be used safely to reduce systems on-the-fly, and we discussed how to detect confluence syntactically on the process-algebraic language MAPA. Case studies show the significant impact of our approach on both state space generation and analysis time.

References

[1] S. C. C. Blom & J. C. van de Pol (2002): *State Space Reduction by Proving Confluence*. In: *CAV*, *LNCS* 2404, pp. 596–609.

[2] H. Boudali, P. Crouzen, B. R. Haverkort, M. Kuntz & M. I. A. Stoelinga (2008): *Architectural dependability evaluation with Arcade*. In: *DSN*, pp. 512–521.

[3] H. Boudali, P. Crouzen & M. I. A. Stoelinga (2010): *A Rigorous, Compositional, and Extensible Framework for Dynamic Fault Tree Analysis*. *IEEE Transactions on Dependable and Secure Compututing* 7(2), pp. 128–143.

[4] M. Bozzano, A. Cimatti, J.-P. Katoen, V. Y. Nguyen, T. Noll & M. Roveri (2011): *Safety, Dependability and Performance Analysis of Extended AADL Models*. *The Computer Journal* 54(5), pp. 754–775.

[5] Y. Deng & M. Hennessy (2011): *On the Semantics of Markov Automata*. In: *ICALP*, *LNCS* 6756, pp. 307–318.

[6] C. Eisentraut, H. Hermanns & L. Zhang (2010): *Concurrency and Composition in a Stochastic World*. In: *CONCUR*, *LNCS* 6269, pp. 21–39.

[7] C. Eisentraut, H. Hermanns & L. Zhang (2010): *On Probabilistic Automata in Continuous Time*. In: *LICS*, pp. 342–351.

[8] Joost-Pieter Katoen (2012): *GSPNs Revisited: Simple Semantics and New Analysis Algorithms*. In: *ACSD*, pp. 6–11.

[9] M. A. Marsan, G. Conte & G. Balbo (1984): *A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems*. *ACM Transactions on Computer Systems* 2(2), pp. 93–122.

[10] M. Timmer, J.-P. Katoen, J. C. van de Pol & M. I. A. Stoelinga (2012): *Efficient Modelling and Generation of Markov Automata*. In: *CONCUR*, *LNCS* 7454, pp. 364–379.

[11] M. Timmer, M. I. A. Stoelinga & J. C. van de Pol (2011): *Confluence Reduction for Probabilistic Systems*. In: *TACAS*, *LNCS* 6605, pp. 311–325.