Confluence reduction for probabilistic systems*

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

ABSTRACT. In this presentation we introduce a novel technique for state space reduction of probabilistic specifications, based on a newly developed notion of confluence for probabilistic automata. We proved that this reduction preserves branching probabilistic bisimulation and can be applied on-the-fly. To support the technique, we introduce a method for detecting confluent transitions in the context of a probabilistic process algebra with data, facilitated by an earlier defined linear format. We present a case study, demonstrating that significant reductions can be obtained.

1 Introduction

Model checking of probabilistic systems is getting more and more attention, but there still is a large gap between the number of techniques supporting traditional model checking and those supporting probabilistic model checking. Especially methods aimed at reducing state spaces are greatly needed to battle the omnipresent state space explosion.

In this work, we generalise the notion of confluence [8] from labelled transition systems (LTSs) to probabilistic automata (PAs) [9]. Basically, an (invisible) τ-step is confluent (and then often denoted by τ_c) if it commutes with all other transitions. As a consequence, we are able to generalise a reduction technique based on confluence to PAs, preserving branching probabilistic bisimulation [10]. Our methodology follows the approach for LTSs from [4]. This approach leads to an on-the-fly (potentially exponential) state space reduction, and consists of the following steps:

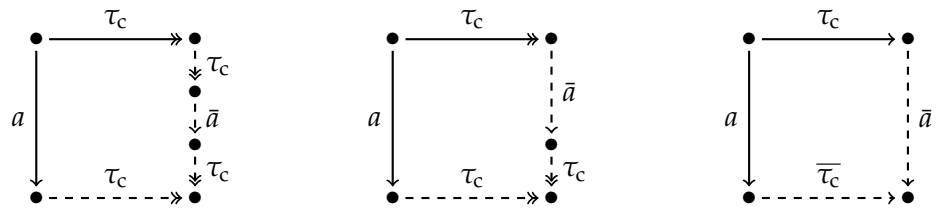
1. specifying a system as a parallel composition of processes with data;
2. transforming the specification to a linear format (linearisation);
3. checking which symbolic τ-transitions are confluent using first-order logic formulas;
4. giving confluent τ-transitions priority during LTS generation.

The next section will give an informal overview of our reduction method. For a complete explanation, including all the formal definitions and proofs and a case study, see [11].

2 Approach

Intuitively, τ-transitions are *confluent* if they do not influence the behaviour of a system. Stated differently, a confluent transition $s \xrightarrow{\tau} s'$ should imply that s is in some way equivalent to s' (in this work we take branching probabilistic bisimulation). These transitions therefore pave the way for state space reductions (for instance by giving confluent transitions priority).

*This research has been partially funded by NWO under grant 612.063.817 (SYRUP) and grant Dn 63-257 (ROCKS), and by the European Union under FP7-ICT-2007-1 grant 214755 (QUASIMODO).



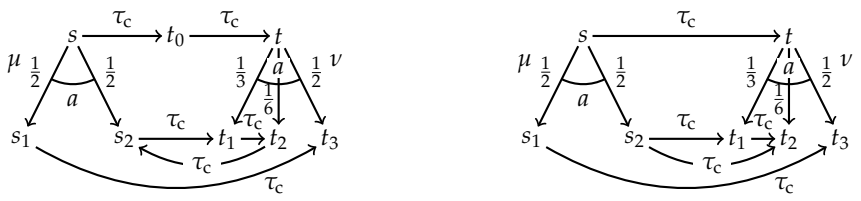
(a) Weak confluence. (b) Confluence. (c) Strong confluence.

Figure 1: Three non-probabilistic variants of confluence.

For (non-probabilistic) labelled transition systems several notions of confluence already exist, defining when a τ-transition is considered confluent or not [3]. Basically, they all require that if an action a is enabled from some state s that also enables a confluent τ-transition, then (1) a will still be enabled after taking that τ-transition (possibly requiring some additional confluent τ-transitions first), and (2) we can always end up in the same state traversing only confluent τ-steps, no matter whether we started by the a- or the τ-transition.

The series of probabilistic confluence notions we introduce are inspired by the non-probabilistic confluence notions from [3]. The most relevant notions for our work are *weak confluence*, *confluence* and *strong confluence*, depicted in Figure 1. Here, we use an arrow with label $\bar{a}$ to denote a step that is optional in case $a = \tau$ (i.e., in that case its source and target states might be the same state). The weaker the confluence notion, the more reduction potentially can be achieved. However, the weak notions are harder to detect.

For probabilistic systems the situation is more difficult, as transitions are associated with a probability distribution instead of a single target state. To still enable reductions based on confluence, only τ-transitions with a (non-probabilistically) fixed target state might be considered confluent. Figure 2 provides an example to depict the notions of weak probabilistic confluence and strong probabilistic confluence that we developed. The basic idea for both is still the same: the target states of the a-transition on the left should be connected by τ_c-transitions to the target states of the a-transition on its right. More precisely, when giving all states that can reach each other via τ_c-transitions the same colour, μ and ν should assign the same probability to the set of states having a colour. For strong probabilistic confluence we require these τ_c-transitions to be in the same direction as the



(a) Weak probabilistic confluence. (b) Strong probabilistic confluence.

Figure 2: Probabilistic generalisations of weak and strong confluence.

transition on top, whereas for weak probabilistic confluence we also allow transitions in the other direction. For strong probabilistic bisimulation, moreover, we require the a-transition to still be directly enabled after the confluent step.

We established some basic facts about the relations between these probabilistic confluence notions, and proved that (weakly) probabilistically confluent τ-transitions always connect branching probabilistically bisimilar states. Based on this fact, we propose a confluence reduction for PAs. The reduced automaton is guaranteed to be branching probabilistically bisimilar to the original one, so it preserves virtually all interesting temporal properties (as expressed in for instance WPCTL). The idea is to give confluent transitions priority, except on one state of each terminal strongly connected component, to avoid a problem similar to the "postponing" problem in partial-order reduction for systems containing τ-loops.

To use our methods for probabilistic verification in practice, generating the unreduced probabilistic automaton should be avoided. Therefore, probabilistic confluence must be lifted to the level of (symbolic) specifications. Here we exploit our previously established probabilistic linear format [7]. We have shown then (for a probabilistic process algebra with data) that a system of parallel components with data can be transformed into a specification in linear format. In this presentation, we show how the confluence of symbolic τ-transitions can be expressed as formulas in first-order logic over this format. As a consequence, confluent τ-steps can be detected at the symbolic level. Subsequently, the reduced PA can be generated on-the-fly.

Related work. Our work has some similarities to partial-order reduction for probabilistic systems. In [2] and [5] partial-order reduction techniques were presented that preserve quantitative LTL$\setminus$X. This was refined in [1] to probabilistic computation tree logic, a branching logic. These papers could not yet report on experimental results. Due to our connection between confluence at the automaton level and the symbolic linear format, we can now perform actual experiments, and achieve on-the-fly reductions on probabilistic systems.

Recently, a revision of partial-order reduction for distributed schedulers was introduced and implemented in PRISM [6]. Confluence reduction differs from this approach on several accounts. First of all, the definition of confluence is very different from the notion of independence used in POR. Because of this, no language-specific independence heuristics are needed anymore; the proof obligations to be fed to a theorem prover directly follow from the formalisms. Moreover, this results in easier definitions and shorter proofs.

3 Conclusions

We developed new notions of confluence for probabilistic automata. Several facts were established about the relations between these notions, and we proved that they identify branching probabilistically bisimilar states. Based on this, probabilistic confluence can be used for state space reduction, even for systems containing τ-loops. We developed a method for probabilistic confluence to be detected in the context of a probabilistic process algebra with data by proving formulas in first-order logic. This way, we enable on-the-fly reductions when generating the state space corresponding to a process-algebraic speci-

fication. A case study on leader election, as described in [11] and to be included in the presentation, illustrates the power of our method. Here, the number of states and transitions both reduced by approximately a factor of 3.

References

[1] C. Baier, P.R. D'Argenio, and M. Größer. Partial order reduction for probabilistic branching time. *Electronic Notes in Theoretical Computer Science*, 153(2):97–116, 2006.

[2] C. Baier, M. Größer, and F. Ciesinski. Partial order reduction for probabilistic systems. In *Proc. of the 1st Int. Conf. on Quantitative Evaluation of Systems (QEST)*, pages 230–239. IEEE Computer Society, 2004.

[3] S.C.C. Blom. Partial τ-confluence for efficient state space generation. Technical Report SEN-R0123, CWI, Amsterdam, 2001.

[4] S.C.C. Blom and J.C. van de Pol. State space reduction by proving confluence. In *Proc. of the 14th Int. Conf. on Computer Aided Verification (CAV)*, volume 2404 of *LNCS*, pages 596–609. Springer, 2002.

[5] P.R. D'Argenio and P. Niebert. Partial order reduction on concurrent probabilistic programs. In *Proc. of the 1st Int. Conf. on Quantitative Evaluation of Systems (QEST)*, pages 240–249. IEEE Computer Society, 2004.

[6] S. Giro, P.R. D'Argenio, and L. María Ferrer Fioriti. Partial order reduction for probabilistic systems: A revision for distributed schedulers. In *Proc. of the 20th Int. Conf. on Concurrency Theory (CONCUR)*, volume 5710 of *LNCS*, pages 338–353. Springer, 2009.

[7] J.-P. Katoen, J.C. van de Pol, M.I.A. Stoelinga, and M. Timmer. A linear process-algebraic format for probabilistic systems with data. In *Proc. of the 10th Int. Conf. on Application of Concurrency to System Design (ACSD)*, LNCS. Springer, To appear. 2010.

[8] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[9] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.

[10] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computation*, 2(2):250–273, 1995.

[11] M. Timmer, J.C. van de Pol, and M.I.A. Stoelinga. Confluence reduction for probabilistic systems. Technical report, University of Twente, 2010. Available at `http://wwwhome.cs.utwente.nl/~timmer/papers/YRCONCUR.pdf`.