Confluence Reduction for Markov Automata[1]

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
Email: {timmer, vdpol, marielle}@cs.utwente.nl
[b] *Software Modelling and Verification, RWTH Aachen University, Germany*
Email: katoen@cs.rwth-aachen.de

Abstract

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.

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

1. Introduction

Markov automata (MAs) [1, 2, 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, 11, 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 confluent 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

[1]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).

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, 11, 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 done 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[2]. 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, 17, 18, 19, 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, 22, 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, 2, 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

[2]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 it is used—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 reduction 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.

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 $\mu = \{s_1 \mapsto p_1, s_2 \mapsto p_2, \ldots, s_n \mapsto p_n\}$ to denote that $\mu(s_1) = p_1$, $\mu(s_2) = p_2$, $\ldots$, $\mu(s_n) = p_n$.*

We use $\mathscr{P}(S)$ to denote the power set *of S, and write* $\mathrm{Distr}(S)$ *for the set of all* discrete probability distributions *over S. We use* $\mathrm{SDistr}(S)$ *for the set of all* substochastic *discrete probability distributions over S, i.e., all functions $\mu\colon S \to [0,1]$ such that $\sum_{s\in S} \mu(s) \leq 1$. Given a function $f\colon S \to T$, we denote by μ_f the* lifting *of μ over f, i.e., $\mu_f(t) = \mu(f^{-1}(t))$, with $f^{-1}(t)$ the inverse image of t under f.*

Given an equivalence relation $R \subseteq S \times S$, we write $[s]_R$ for the equivalence class *of s induced by R, i.e., $[s]_R = \{s' \in S \mid (s,s') \in R\}$. Given two probability distributions $\mu, \mu' \in \mathrm{Distr}(S)$ and an equivalence relation R, we write $\mu \equiv_R \mu'$ to denote that $\mu([s]_R) = \mu'([s]_R)$ for every $s \in S$.*

Markov automata [1, 2, 3] consist of a countable set of states, an initial state, an alphabet of actions, sets of action-labelled (interactive) and rate-labelled (Markovian) transitions, a set of atomic propositions and a state-labelling function. We assume a countable universe of actions *Act* and an internal action $\tau \in Act$.

Definition 2 (Markov automata). *A* Markov automaton *is a tuple $\mathcal{M} = \langle S, s^0, A, \hookrightarrow, \rightsquigarrow, \mathrm{AP}, L\rangle$, where*

- *S is a countable set of* states*, of which $s^0 \in S$ is the* initial state*;*
- *$A \subseteq Act$ is a countable set of* actions*;*
- *$\hookrightarrow \subseteq S \times A \times \mathrm{Distr}(S)$ is a countable* interactive probabilistic transition relation*;*
- *$\rightsquigarrow \subseteq S \times \mathbb{R}^{>0} \times S$ is a countable* Markovian transition relation*;*
- AP *is a countable set of* state labels *(also called* atomic propositions*); and*
- *$L\colon S \to \mathscr{P}(\mathrm{AP})$ is a* state-labelling function*.*

If $(s,a,\mu) \in \hookrightarrow$, we write $s \xhookrightarrow{a} \mu$ and say that action a can be taken from state s, after which the probability to go to each $s' \in S$ is $\mu(s')$. If $(s,\lambda,s') \in \rightsquigarrow$, we write $s \overset{\lambda}{\rightsquigarrow} s'$ and say that s moves to s' with rate λ.

The *(exponential) rate* between two states $s, s' \in S$ is $rate(s,s') = \sum_{(s,\lambda,s')\in\rightsquigarrow} \lambda$, and the *exit rate of s* is $rate(s) = \sum_{s'\in S} rate(s,s')$. We require $rate(s) < \infty$ for every $s \in S$. If $rate(s) > 0$, the *branching probability distribution* of s is denoted by $\mathbb{P}_s$ and defined as $\mathbb{P}_s(s') = \frac{rate(s,s')}{rate(s)}$ for every $s' \in S$. By definition of the exponential distribution, the probability of leaving a state s within t time units is given by $1 - e^{-rate(s)\cdot t}$ (given $rate(s) > 0$), after which the next state is chosen according to $\mathbb{P}_s$.

MAs adhere to the *maximal progress assumption*, postulating that τ-transitions can never be delayed (since they are not subject to any interaction [26]). Hence, a state that has at least one outgoing τ-transition can never take a Markovian transition. This fact is captured below in the definition of extended transitions, which is used to provide a uniform manner for dealing with both interactive and Markovian transitions. Each state has an extended transition per interactive transition, while it has only one extended transition for all its Markovian transitions together (if there are any).

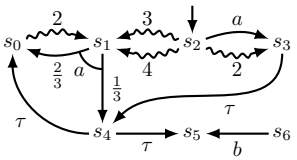
We assume an arbitrary MA $\mathcal{M} = \langle S, s^0, A, \hookrightarrow, \rightsquigarrow, \mathrm{AP}, L\rangle$ in every definition, proposition and theorem.

Definition 3 (Extended action set). *The* extended action set of $\mathcal{M}$ *is $A^\chi = A \cup \{\chi(r) \mid r \in \mathbb{R}_{>0}\}$. Given a state $s \in S$ and an action $\alpha \in A^\chi$, we write $s \xrightarrow{\alpha} \mu$ if*

- *$\alpha \in A$ and $s \xhookrightarrow{\alpha} \mu$; or*
- *$\alpha = \chi(rate(s))$, $rate(s) > 0$, $\mu = \mathbb{P}_s$ and there is no μ' such that $s \xhookrightarrow{\tau} \mu'$.*

A transition $s \xrightarrow{\alpha} \mu$ is called an extended transition*. We write $s \xrightarrow{\alpha} t$ for $s \xrightarrow{\alpha} \mathbb{1}_t$, and $s \to t$ if $s \xrightarrow{\alpha} t$ for some $\alpha \in A^\chi$. We write $s \xrightarrow{\alpha,\mu} s'$ if there is an extended transition $s \xrightarrow{\alpha} \mu$ such that $\mu(s') > 0$. A transition $s \xrightarrow{a} \mu$ is* invisible *if both $a = \tau$ and $L(s) = L(t)$ for every $t \in supp(\mu)$.*

Example 4. Consider the MA shown on the right.



Here, $\mathit{rate}(s_2, s_1) = 3 + 4 = 7$, $\mathit{rate}(s_2) = 7 + 2 = 9$, and $\mathbb{P}_{s_2} = \mu$ such that $\mu(s_1) = \frac{7}{9}$ and $\mu(s_3) = \frac{2}{9}$. There are two extended transitions from s_2: $s_2 \xrightarrow{a} \mathbb{1}_{s_3}$ (also written as $s_2 \xrightarrow{a} s_3$) and $s_2 \xrightarrow{\chi(9)} \mathbb{P}_{s_2}$. □

We define several notions for paths and connectivity. These are based on extended transitions, and thus may contain interactive as well as Markovian steps.

Definition 5 (Paths and traces)**.**

- *A* path *in $\mathcal{M}$ is a finite sequence $\pi^{\text{fin}} = s_0 \xrightarrow{\alpha_1,\mu_1} s_1 \xrightarrow{\alpha_2,\mu_2} s_2 \xrightarrow{\alpha_3,\mu_3} \dots \xrightarrow{\alpha_n,\mu_n} s_n$, possibly with $n = 0$, or an infinite sequence $\pi^{\text{inf}} = s_0 \xrightarrow{\alpha_1,\mu_1} s_1 \xrightarrow{\alpha_2,\mu_2} s_2 \xrightarrow{\alpha_3,\mu_3} \dots$, with $s_i \in S$ for all $0 \le i \le n$ and all $i \ge 0$, respectively, and such that $s_i \xrightarrow{\alpha_{i+1}} \mu_{i+1}$ is an extended transitions in $\mathcal{M}$ with $\mu_{i+1}(s_{i+1}) > 0$, for each i. We refer to π^{fin} as a* path from s_0 to s_n. *We use $\mathit{finpaths}_{\mathcal{M}}$ for the set of all* finite paths *in $\mathcal{M}$ (not necessarily beginning in the initial state), and $\mathit{finpaths}_{\mathcal{M}}(s)$ for all such paths with $s_0 = s$.*
- *A path π is* invisible *(denoted by $\mathit{invis}(\pi)$) if it never alters the state labelling and only consists of internal actions: $L(s_i) = L(s_0)$ and $\alpha_i = \tau$ for all i. Given a sufficiently long path π, we use $\mathit{prefix}(\pi, i)$ to denote the path fragment $s_0 \xrightarrow{\alpha_1,\mu_1} \dots \xrightarrow{\alpha_i,\mu_i} s_i$, and $\mathit{step}(\pi, i)$ for the transition $s_{i-1} \xrightarrow{\alpha_i} \mu_i$. If π is finite, we define $|\pi| = n$ and $\mathit{last}(\pi) = s_n$, otherwise $|\pi| = \infty$ and no final state exists.*

Definition 6 (Connectivity)**.** *Let $s, t \in S$, and consider the relation $\to \subseteq S \times S$ from Definition 3 that relates states $s, t \in S$ if there is a transition $s \xrightarrow{\alpha} \mathbb{1}_t$ for some $\alpha \in A^{\chi}$. We let $\twoheadrightarrow$* (reachability) *be the reflexive and transitive closure of $\to$, and $\twoheadleftarrow\!\!\twoheadrightarrow$* (convertibility) *its reflexive, transitive and symmetric closure. We write $s \twoheadrightarrow\!\!\twoheadleftarrow t$* (joinability) *if $s \twoheadrightarrow u$ and $t \twoheadrightarrow u$ for some state $u \in S$.*

Example 7. The MA in Example 4 has infinitely many paths, for example

$$\pi = s_2 \xrightarrow{\chi(9),\mu_1} s_1 \xrightarrow{a,\mu_2} s_0 \xrightarrow{\chi(2),\mathbb{1}_{s_1}} s_1 \xrightarrow{a,\mu_2} s_4 \xrightarrow{\tau,\mathbb{1}_{s_5}} s_5$$

with $\mu_1(s_1) = \frac{7}{9}$ and $\mu_1(s_3) = \frac{2}{9}$, and $\mu_2(s_0) = \frac{2}{3}$ and $\mu_2(s_4) = \frac{1}{3}$. Here, $\mathit{prefix}(\pi, 2) = s_2 \xrightarrow{\chi(9),\mu_1} s_1 \xrightarrow{a,\mu_2} s_0$, and $\mathit{step}(\pi, 2) = s_1 \xrightarrow{a} \mu_2$. Also, $s_2 \twoheadrightarrow s_5$ (via s_3), as well as $s_3 \twoheadrightarrow\!\!\twoheadleftarrow s_6$ (at s_5) and $s_0 \twoheadleftarrow\!\!\twoheadrightarrow s_5$. However, $s_0 \twoheadrightarrow s_5$ and $s_0 \twoheadrightarrow\!\!\twoheadleftarrow s_5$ do not hold (as s_0 cannot get to s_4 via only transitions with Dirac distributions). □

The relation $\twoheadrightarrow\!\!\twoheadleftarrow$ is symmetric, but not necessarily transitive. Intuitively, $s \twoheadleftarrow\!\!\twoheadrightarrow t$ means that s is connected by extended transitions to t—disregarding their orientation, but all with a Dirac distribution. Clearly, $s \twoheadrightarrow t$ implies $s \twoheadrightarrow\!\!\twoheadleftarrow t$, and $s \twoheadrightarrow\!\!\twoheadleftarrow t$ implies $s \twoheadleftarrow\!\!\twoheadrightarrow t$. These implications do not hold the other way.

To prove confluence reduction correct, we show that it preserves divergence-sensitive branching bisimulation. Basically, this means that there is an equivalence relation R linking states in the original system to states in the reduced system, such that their initial states are related and all related states can mimic each other's transitions and divergences. More precisely, for all $(s, t) \in R$ and every extended transition $s \xrightarrow{\alpha} \mu$, there should be a *branching transition* $t \xRightarrow{\alpha}_R \mu'$ such that $\mu \equiv_R \mu'$. The existence of such a branching transition depends on the existence of a certain *scheduler*. Schedulers resolve nondeterministic choices in MAs by selecting which transitions to take given a history; they may also terminate with some probability.

A state t can do a branching transition $t \xRightarrow{\alpha}_R \mu'$ if either (1) $\alpha = \tau$ and $\mu' = \mathbb{1}_t$, or (2) there is a scheduler that, starting from state s, terminates according to μ', always schedules precisely one α-transition (immediately before terminating), never schedules any other visible transitions and does not leave the equivalence class $[t]_R$ before doing an α-transition.

Example 8. Observe the MA in Figure 1 (left). Due to nondeterminism (that can be resolved probabilistically), there are infinitely many branching transitions from s. We demonstrate the existence of the branching transition $s \xRightarrow{\alpha}_R \mu$, with $\mu = \{s_1 \mapsto \frac{8}{24}, s_2 \mapsto \frac{7}{24}, s_3 \mapsto \frac{1}{24}, s_4 \mapsto \frac{4}{24}, s_5 \mapsto \frac{4}{24}\}$, by the scheduler depicted in Figure 1 (right), assuming $(s, t_i) \in R$ for all t_i.

The scheduling tree illustrates the probabilistic choices by which the nondeterministic choices are resolved. Under this scheduling, the probabilities of ending up in $s_1, \dots, s_5$ can be found by multiplying the probabilities on the paths towards them. Indeed, these correspond to the probabilities prescribed by μ. □

Figure 1: An MA (left), and a tree demonstrating branching transition $s \xRightarrow{\alpha}_R \mu$ (right).

In addition to the mimicking of transitions by branching transitions, we require R-related states to either both be able to perform an infinite invisible path with probability 1 (*diverge)*, or to both not be able to do so. We write $s \approx_{\mathrm{b}}^{\mathrm{div}} t$ if two states s, t are divergence-sensitive branching bisimilar, and $\mathcal{M}_1 \approx_{\mathrm{b}}^{\mathrm{div}} \mathcal{M}_2$ if two MAs are (i.e., if their initial states are so in their disjoint union) [27].

Technicalities. We now formalise the notions of schedulers, branching transitions and (divergence-sensitive) branching bisimulation. These technicalities are needed solely for the proofs of our theorems and propositions, and hence may be skipped by the reader only interested in the results themselves.

The decisions of schedulers may be *randomised*: instead of choosing a single transition, a scheduler may resolve a nondeterministic choice probabilistically. Schedulers can also be *partial*, assigning some probability to not choosing any next transition at all (and hence terminating). They can select from interactive transitions as well as Markovian transitions, as both may be enabled at the same time. This is due to the fact that we consider *open* MAs, in which the timing of visible actions is still to be determined by their context.

Definition 9 (Schedulers)**.** *Let $\to \subseteq S \times A^{\chi} \times \mathrm{Distr}(S)$ be the set of extended transitions of $\mathcal{M}$. Then, a* scheduler *for $\mathcal{M}$ is a function*

$$\mathcal{S} \colon \mathit{finpaths}_{\mathcal{M}} \to \mathrm{Distr}(\{\bot\} \cup \to)$$

such that, for every $\pi \in \mathit{finpaths}_{\mathcal{M}}$, the transitions $s \xrightarrow{\alpha} \mu$ that are scheduled by $\mathcal{S}$ after π are indeed possible, i.e., $\mathcal{S}(\pi)(s, \alpha, \mu) > 0$ implies $s = \mathit{last}(\pi)$. The decision of not choosing any transition is represented by $\bot$.

Note that our schedulers are *time-homogeneous* (also called *time-abstract*): they cannot take into account the amount of time that has already passed during a path. When dealing with time-bounded reachability properties [28], *time-inhomogeneous* schedulers are important, as they may be needed for optimal results. However, in this work we can do without as we will not formally define such properties. Since time-homogeneous schedulers do take into account the rates of an MA, we can use them to define notions of bisimulation that do preserve time-bounded properties (similar to weak bisimulation in [1] being defined in terms of 'time-homogeneous' labelled trees). As discussed in [29], measurability is not an issue for time-homogeneous schedulers. We refer to [30] for a thorough analysis of different types of schedulers. Since Markovian extended transitions only emanate from states without any outgoing τ-transitions, schedulers cannot violate the maximal progress assumption.

We now define finite and maximal paths of an MA under a scheduler. The finite paths under a scheduler are those finite paths of the MA for which each step has been assigned a nonzero probability. The maximal paths are a subset of those; they are the paths after which the scheduler may decide to terminate.

Definition 10 (Finite and maximal paths)**.** *The set of* finite paths *of $\mathcal{M}$ under a scheduler $\mathcal{S}$ is*

$$\mathit{finpaths}_{\mathcal{M}}^{\mathcal{S}} = \{\pi \in \mathit{finpaths}_{\mathcal{M}} \mid \forall 0 \leq i < |\pi| \,.\, \mathcal{S}(\mathit{prefix}(\pi, i))(\mathit{step}(\pi, i+1)) > 0\}$$

Let $\mathit{finpaths}_{\mathcal{M}}^{\mathcal{S}}(s) \subseteq \mathit{finpaths}_{\mathcal{M}}^{\mathcal{S}}$ be the set of all such paths starting in state $s \in S$.

The set of maximal paths of $\mathcal{M}$ under $\mathcal{S}$ *is given by $\mathit{maxpaths}_{\mathcal{M}}^{\mathcal{S}} = \{\pi \in \mathit{finpaths}_{\mathcal{M}}^{\mathcal{S}} \mid \mathcal{S}(\pi)(\bot) > 0\}$. Similarly, $\mathit{maxpaths}_{\mathcal{M}}^{\mathcal{S}}(s)$ is the set of maximal paths of $\mathcal{M}$ under $\mathcal{S}$ starting in s.*

As schedulers resolve all nondeterministic choices, we can compute the probability that, starting from a given state s, the path generated by $\mathcal{S}$ has some finite prefix π. This probability is denoted by $P_{\mathcal{M},s}^{\mathcal{S}}(\pi)$.

Definition 11 (Path probabilities)**.** *Let $\mathcal{S}$ be a scheduler for $\mathcal{M}$, and let $s \in S$ be a state of $\mathcal{M}$. Then, the function $P^{\mathcal{S}}_{\mathcal{M},s}\colon \mathit{finpaths}_{\mathcal{M}}(s) \to [0,1]$ is defined by*

$$P^{\mathcal{S}}_{\mathcal{M},s}(s) = 1 \qquad P^{\mathcal{S}}_{\mathcal{M},s}(\pi \xrightarrow{\alpha,\mu} t) = P^{\mathcal{S}}_{\mathcal{M},s}(\pi) \cdot \mathcal{S}(\pi)(\mathit{last}(\pi), \alpha, \mu) \cdot \mu(t)$$

Based on these probabilities we can compute the probability distribution $F^{\mathcal{S}}_{\mathcal{M}}(s)$ over the states where an MA $\mathcal{M}$ under a scheduler $\mathcal{S}$ terminates, when starting in state s. Note that $F^{\mathcal{S}}_{\mathcal{M}}(s)$ may be substochastic (i.e., the probabilities do not add up to 1), as $\mathcal{S}$ does not necessarily terminate.

Definition 12 (Final state probabilities)**.** *Given a scheduler $\mathcal{S}$ for $\mathcal{M}$, we define $F^{\mathcal{S}}_{\mathcal{M}}\colon S \to \mathrm{SDistr}(S)$ by*

$$F^{\mathcal{S}}_{\mathcal{M}}(s) = \Big\{ s' \mapsto \sum_{\substack{\pi \in \mathit{maxpaths}^{\mathcal{S}}_{\mathcal{M}}(s) \\ \mathit{last}(\pi) = s'}} P^{\mathcal{S}}_{\mathcal{M},s}(\pi) \cdot \mathcal{S}(\pi)(\bot) \mid s' \in S \Big\} \qquad \forall s \in S$$

Extended examples of these definitions can be found in [34].

To introduce branching bisimulation, we first define the *branching transition*. Due to the use of extended transitions as a uniform manner of dealing with both interactive and Markovian transitions, this definition precisely coincides with the definition of branching steps for PAs [12].

Definition 13 (Branching transitions)**.** *Let $s \in S$, and let $R \subseteq S \times S$ be an equivalence relation. Then, $s \overset{\alpha}{\Longrightarrow}_R \mu$ if either (1) $\alpha = \tau$ and $\mu = \mathbb{1}_s$, or (2) a scheduler $\mathcal{S}$ exists such that*

- *$F^{\mathcal{S}}_{\mathcal{M}}(s) = \mu$; and*
- *for every maximal path $s \xrightarrow{\alpha_1,\mu_1} s_1 \xrightarrow{\alpha_2,\mu_2} \dots \xrightarrow{\alpha_n,\mu_n} s_n \in \mathit{maxpaths}^{\mathcal{S}}_{\mathcal{M}}(s)$ it holds that $\alpha_n = \alpha$. Moreover, for every $1 \le i < n$ we have $\alpha_i = \tau$, $(s, s_i) \in R$ and $L(s) = L(s_i)$.*

Example 8 already provided an example of a branching transition.

Based on branching transitions, we define branching bisimulation for MAs as a natural extension of the notion of naive weak bisimulation from [1][3]. Naive weak bisimulation is an intuitive generalisation of weak bisimulation from PAs and IMCs to MAs. For finitely branching systems, naive weak bisimulation is implied by our notion of branching bisimulation, as it is basically obtained by omitting the requirement that $(s, s_i) \in R$ for all $1 \le i < n$, and allowing convex combinations of transitions.

Definition 14 (Branching bisimulation)**.** *An equivalence relation $R \subseteq S \times S$ is a* branching bisimulation *for $\mathcal{M}$ if for every $(s,t) \in R$ and all $\alpha \in A^{\chi}, \mu \in \mathrm{Distr}(S)$, it holds that $L(s) = L(t)$ and*

$$s \xrightarrow{\alpha} \mu \quad \textit{implies} \quad \Big(\exists \mu' \in \mathrm{Distr}(S)\ .\ t \overset{\alpha}{\Longrightarrow}_R \mu' \wedge \mu \equiv_R \mu'\Big)$$

Two states $s, t \in S$ are branching bisimilar *(denoted by $s \approx_{\mathrm{b}} t$) if there exists a branching bisimulation R for $\mathcal{M}$ such that $(s,t) \in R$. Two MAs $\mathcal{M}, \mathcal{M}'$ are branching bisimilar (denoted by $\mathcal{M} \approx_{\mathrm{b}} \mathcal{M}'$) if their initial states are branching bisimilar in their disjoint union.*

Note that, since each branching bisimulation relation R has the property that $(s,t) \in R$ implies $L(s) = L(t)$, the condition "$L(s) = L(s_i)$ for every $1 \le i < n$" in Definition 13 is already implied by $(s, s_i) \in R$, and hence does not explicitly need to be checked for $t \overset{\alpha}{\Longrightarrow}_R \mu'$.

If infinite paths of τ-actions can be scheduled with non-zero probability, then minimal probabilities (e.g., of eventually seeing an a-action) are not preserved by branching bisimulation. Consider for instance the two MAs in Figure 2. Note that, for the one on the left, the a-transition is not necessarily ever taken.

[3]Since our notion of branching bisimulation for MAs is just as naive as naive weak bisimulation for MAs, we could have called it *naive branching bisimulation*. However, since naive weak bisimulation for MAs is actually strongly related to weak bisimulation for PAs and IMCs, we argue that it would have made more sense to omit the 'naive' in the existing notion of naive weak bisimulation for MAs and prefix 'smart' to the existing notion of weak bisimulation for MAs.

Figure 2: Two systems to illustrate divergence.

After all, it is possible to indefinitely and invisibly loop through state s_0: *divergence*. For the MA on the right, the a-transition cannot be avoided (assuming that termination cannot occur in states with outgoing transitions). Still, these MAs are branching bisimilar, and hence branching bisimulation does not leave invariant all properties—in this case, the minimal probability of eventually taking an a-transition.

To solve this problem, *divergence-sensitive* notions of bisimulation have been introduced [27, 31]. They force diverging states to be mapped to diverging states. We adopt this concept for Markovian branching bisimulation. Since Markovian transitions already need to be mimicked, and the same holds for transitions that change the state labelling (since these cannot stay within the same equivalence class), divergence is defined as the traversal of an infinite path π that contains only τ-actions and never changes the state labelling (i.e., a path π such that $\mathit{invis}(\pi)$).

Definition 15 (Divergence-sensitive relations)**.** *An equivalence relation $R \subseteq S \times S$ over the states of $\mathcal{M}$ is* divergence sensitive *if for all $(s, s') \in R$ it holds that*

$$\Big(\exists \mathcal{S} \,.\, \forall \pi \in \mathit{finpaths}^{\mathcal{S}}_{\mathcal{M}}(s) \,.\, \mathit{invis}(\pi) \wedge \mathcal{S}(\pi)(\bot) = 0\Big) \text{ iff } \Big(\exists \mathcal{S}' \,.\, \forall \pi \in \mathit{finpaths}^{\mathcal{S}'}_{\mathcal{M}}(s') \,.\, \mathit{invis}(\pi) \wedge \mathcal{S}'(\pi)(\bot) = 0\Big)$$

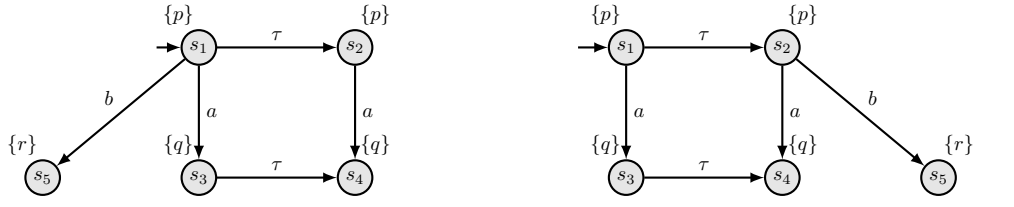
where $\mathcal{S}$ and $\mathcal{S}'$ range over all possible schedulers for $\mathcal{M}$. Two MAs $\mathcal{M}_1, \mathcal{M}_2$ are divergence-sensitive branching bisimilar*, denoted by $\mathcal{M}_1 \approx_{\mathrm{b}}^{\mathrm{div}} \mathcal{M}_2$, if they are related by a divergence-sensitive branching bisimulation.*

Example 16. The two MAs in Figure 2 are branching bisimilar by the equivalence relation that relates s_0 to t_0 and s_1 to t_1. However, whereas from s_0 an infinite invisible path can be scheduled, this cannot be done from t_0; hence, this relation is not divergence sensitive. Indeed, since from s_0 a scheduler can prevent the a-transition from taking place, while from t_0 it cannot, we do not want to consider these MAs equivalent. □

3. Informal introduction of confluence

Before introducing the technical definitions of confluence for MAs, we provide an informal overview of confluence and its usage in state space reduction [11, 12]. Since the additional technical difficulties due to probabilities and Markovian rates may distract from the underlying ideas, we omit them in this introduction.

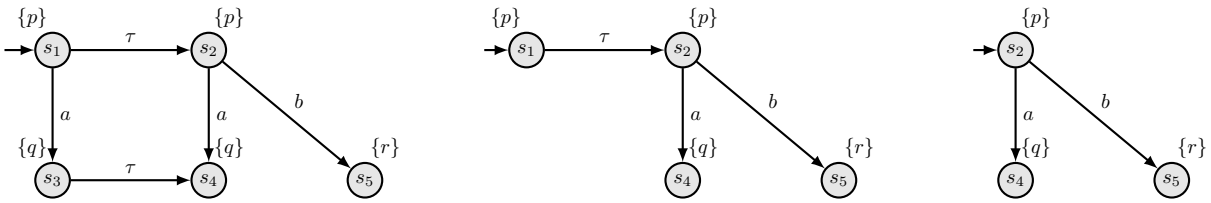
Confluence reduction is based on the idea that some transitions do not influence the observable behaviour of a system—assuming that only visible actions $a \neq \tau$ and changes in the truth values of atomic propositions are observed. Hence, these transitions can be given priority over other transitions. To this end, they at least have to be invisible themselves. Still, some invisible transitions *may* influence the observable behaviour of an MA, even though they are not observable themselves—these cannot be confluent. Figure 3 illustrates this phenomenon.



(a) Observable invisible transition $s_1 \xrightarrow{\tau} s_2$.

(b) Unobservable invisible transition $s_1 \xrightarrow{\tau} s_2$.

Figure 3: Observable versus unobservable invisible transitions.



(a) Reprint of Figure 3(b). (b) Prioritisation. (c) Skipping.

Figure 4: State space reduction based on confluence.

Example 17. While the transition $s_1 \xrightarrow{\tau} s_2$ in Figure 3(a) cannot be observed directly, it does disable the b-transition. Hence, this transition influences the observable behaviour: if it was always taken from s_1 while omitting the other two transitions emanating from this state, then no b-action would ever be observed and the atomic proposition r would never hold. Therefore, states s_1 and s_2 are not branching bisimilar.

The transition $s_1 \xrightarrow{\tau} s_2$ in Figure 3(b) is different: we can always take it from s_1, ignoring $s_1 \xrightarrow{a} s_3$, without losing any observable behaviour: $s_1 \xrightarrow{\tau} s_2$ is confluent. Both the original system and the system reduced in this way may take an a or a b-action and may end up in a state satisfying either q or r. Actually, states s_1 and s_2 are branching bisimilar, and so are the original and the reduced system (Figure 4(b)). □

The example illustrates the most important property of confluent transitions: they connect branching bisimilar states, and hence in principle they can be given priority over their neighbouring transitions without losing any behaviour. To verify that a transition is confluent, it should be invisible *and* still allow all behaviour enabled from its source state to occur from its target state as well. In other words, all other transitions from its source state should be *mimicked* from its target state.

3.1. Checking for mimicking behaviour

To check whether all behaviour from a transition's source state is also enabled from its target state, confluence employs a coinductive approach similar to the common definitions of bisimulation. For an invisible transition $s \xrightarrow{\tau} s'$ to be confluent, the existence of a transition $s \xrightarrow{a} t$ should imply the existence of a transition $s' \xrightarrow{a} t'$ for some t'. Additionally, for all behaviour from s to be present at s', also all behaviour from t should be present at t'. To achieve this, we coinductively require s' to have a confluent transition to t', and then say that the a-transitions and the confluent τ-transitions *commute*. We note that a coinductive approach such as the one just described requires a set of transitions to be defined upfront. Then, we can validate whether or not this set indeed satisfies the conditions for it to be confluent. In practice, we are mostly interested in finding the *largest* set for which this is the case.

Example 18. In Figure 3(a), the set containing both τ-transitions is *not* confluent. After all, for $s_1 \xrightarrow{\tau} s_2$ it is not the case that every action enabled from s_1 is also enabled from s_2. In Figure 3(b), the set containing both τ-transitions *is* confluent. For $s_3 \xrightarrow{\tau} s_4$, the mimicking condition is satisfied trivially, since s_3 has no other outgoing transitions. For $s_1 \xrightarrow{\tau} s_2$ the condition is also satisfied, since $s_1 \xrightarrow{a} s_3$ is mimicked by $s_2 \xrightarrow{a} s_4$. As required, s_3 and s_4 are indeed connected by a confluent transition. □

3.2. State space reduction based on confluence

Confluent transitions can often be given priority, omitting all other transitions emanating from the same state. This may yield many unreachable states, and hence significant state space reductions. Although a system obtained due to prioritisation of confluent transitions is indeed branching bisimilar to the original system (under some assumptions discussed below), it often still contains superfluous states. They have only one outgoing invisible transition, and hence do not contribute to the system's observable behaviour in any way. So, instead of prioritising confluent transitions, we rather completely skip them.

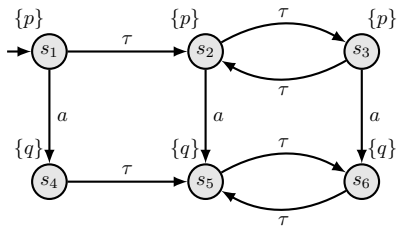
Example 19. Consider again Figure 3(b), repeated in Figure 4(a), where both invisible transitions are confluent. Figure 4(b) demonstrates the reduced state space when giving these transitions priority over their neighbours. Although this reduction is valid, state s_1 has little purpose and can be skipped over. That way, we obtain the system illustrated in Figure 4(c). □

Prioritisation of transitions as well as skipping them only works in the absence of cycles of confluent transitions. To see why, consider the system in Figure 5(a). All invisible transitions are confluent. However, when continuously ignoring all non-confluent transitions (yielding Figure 5(b)), the a-transition is postponed forever and the atomic proposition q will never hold. Clearly, such a reduction does not preserve reachability properties and hence the reduced system is not considered equivalent to the original (indeed, they are not branching bisimilar). This problem is known in the setting of POR as the *ignoring problem* [16, 32], and often dealt with by requiring the reduction to be acyclic. That is, no cycle of states that are all omitting some of their transitions should be present. Indeed, this requirement is violated in Figure 5(b). As a solution, we could also require reductions to be acyclic, forcing at least one state of a cycle to be fully explored.

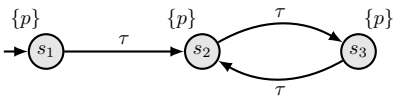
(Note that Valmari [33] recently showed that there is no need for treating the ignoring problem at all for, e.g., safety and fairness-insensitive progress properties, when the transition system is always may-terminating, i.e., when along every execution there is a possibility to reach a terminating state.)

Example 20. Figure 5(c) shows the result of reducing Figure 5(a) based on prioritisation while forcing at least one state on a cycle to be fully explored (here, state s_2). □

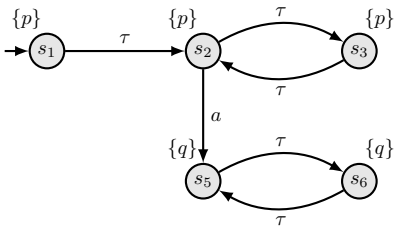
The idea of skipping confluent transitions can be extended to work in the presence of cycles of confluent transitions—that is also the approach we take. In their absence, this approach simply boils down to skipping confluent transitions until reaching a state without any outgoing confluent transitions (state s_2 in Figure 4(c)). In the presence of cycles, we continue until reaching the bottom strongly connected component (BSCC) of the subgraph when considering only confluent transitions. When requiring confluent transitions to always be mimicked by confluent transitions, there is a unique such BSCC reachable from every state (in Figure 5(a), s_1 has BSCC $\{s_2, s_3\}$ when only considering the confluent transitions). In this BSCC, we randomly select one state to be the *representative* for all states that can reach it by skipping confluent transitions. Since confluent transitions never disable behaviour, such a representative state exhibits all behaviour



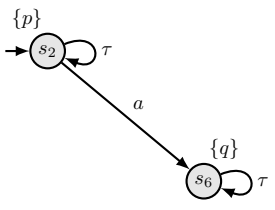
(a) Cyclic confluence.



(b) Erroneous reduction.



(c) Acyclic reduction using prioritisation.



(d) Acyclic reduction using representatives.

Figure 5: Confluence reduction in the presence of cyclic confluent transitions.

of the states that it represents. The representative state is explored fully, and all transitions to states that can reach that representative by confluent transitions are redirected towards the representative.

Example 21. Figure 5(d) illustrates the representatives approach, selecting s_2 as representative of s_1, s_2 and s_3, and s_6 as representative of s_5 and s_6. Note that both the acyclic reduction and the representative approach yield MAs that are branching bisimilar to the original, but the latter allows for more reduction. □

Overview. Summarising, confluence reduction entails:

1. Constructing a subset of the invisible transitions satisfying the confluence restrictions.
2. Choosing a representative state for each state in the original system.
3. Reducing the state space by skipping confluent transitions until reaching a representative state.

We will see that, in practice, the set of confluent transitions is often implicit: we check whether higher-level constructs generate solely confluent transitions. The second and third step are often integrated, choosing representatives on-the-fly for all states that are reached during state space generation.

4. Confluence for Markov automata

4.1. Commutativity of distributions

For non-probabilistic strong confluence, a confluent transition $s \xrightarrow{\tau} t$ and neighbouring transition $s \xrightarrow{a} s'$ have to be accompanied by a transition $t \xrightarrow{a} t'$ and a confluent transition $s' \xrightarrow{\tau} t'$: the transitions $s \xrightarrow{\tau} t$ and $s \xrightarrow{a} s'$ *commute*. No transitions from t' to s' or longer paths of transitions between s' and t' are taken into account. We generalise this idea to the probabilistic setting, where distributions μ, ν have to be connected by confluent transitions. To this end, we consider an equivalence relation $R^{\mathcal{T}}_{\mu,\nu}$ over S based on a set of confluent transitions $\mathcal{T}$ in the MA under consideration, that partitions the state space into equivalence classes requiring the same probability from μ as from ν (i.e., $\mu \equiv_{R^{\mathcal{T}}_{\mu,\nu}} \nu$). Reflecting the non-probabilistic case, we consider only direct transitions from the support of μ to the support of ν[4]; see [12, 34] for more details.

Definition 22. *Given a set of transitions $\mathcal{T}$ and two probability distributions $\mu, \nu \in \mathrm{Distr}(S)$, let $R^{\mathcal{T}}_{\mu,\nu}$ be the smallest equivalence relation over S such that*

$$R^{\mathcal{T}}_{\mu,\nu} \supseteq \{(s,t) \in \mathit{supp}(\mu) \times \mathit{supp}(\nu) \mid (s \xrightarrow{\tau} t) \in \mathcal{T}\}$$

We often omit the subscripts μ, ν and the superscript $\mathcal{T}$ when clear from the context.

The definition is inspired by [24]. It is slightly more powerful than the one in [12] and, in our view, easier to understand. Note that, for $\mu \equiv_{R^{\mathcal{T}}_{\mu,\nu}} \nu$, we require $\mathcal{T}$-transitions from the support of μ to the support of ν. Even though a (symmetric and transitive) equivalence relation is used, transitions from the support of ν to the support of μ do not influence $R^{\mathcal{T}}_{\mu,\nu}$, and neither do confluent paths from μ to ν of length more than one.

Example 23. Consider Figure 6, assume that all τ-transitions are in $\mathcal{T}$ and let $\mu = \{s_1 \mapsto \frac{1}{2}, s_2 \mapsto \frac{1}{2}\}$ and $\nu = \{t_1 \mapsto \frac{1}{3}, t_2 \mapsto \frac{1}{6}, t_3 \mapsto \frac{1}{2}\}$. Then, R gives rise to three equivalence classes: $C_1 = \{s,t\}$, $C_2 = \{s_1, t_3\}$ and $C_3 = \{s_2, t_1, t_2\}$. Now, μ and ν coincide for these classes: $\mu(C_1) = 0 = \nu(C_1)$, $\mu(C_2) = \frac{1}{2} = \nu(C_2)$ and $\mu(C_3) = \frac{1}{2} = \nu(C_3)$. Hence, $\mu \equiv_{R^{\mathcal{T}}_{\mu,\nu}} \nu$.

If the transition between s_2 and t_1 had been directed from t_1 to s_2, that would have resulted in the partitioning $C_1 = \{s,t\}$, $C_2 = \{s_1, t_3\}$, $C_3 = \{s_2, t_2\}$ and $C_4 = \{t_1\}$. Hence, in that case $\mu \not\equiv_{R^{\mathcal{T}}_{\mu,\nu}} \nu$, since $\mu(C_4) = 0 \neq \frac{1}{3} = \nu(C_4)$. □

[4] We could have also chosen to be a bit more liberal, allowing a *path* of $\mathcal{T}$-transitions from s to t. However, the current approach simplifies the definitions and some proofs later on; it also corresponds more directly to the way we detect confluence heuristically in practice.

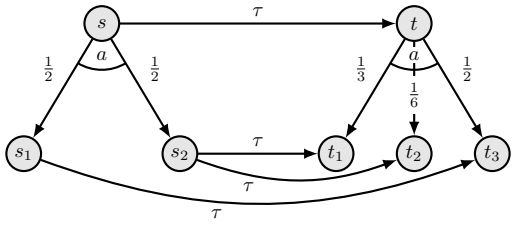


Figure 6: Commutativity in the presence of probabilistic choice.

4.2. Confluence classifications

Earlier approaches just took any subset of the invisible transitions and showed that it was confluent—those confluent sets were not closed under union, though. Now, we impose some more structure, classifying the interactive transitions of an MA into *groups* upfront—allowing overlap and not requiring all interactive transitions to be in at least one group. We will see that this is natural in the context of the process algebra MAPA and can be applied implicitly—as the implementations of earlier approaches on confluence reduction already (unknowingly) did as well.

At this point, the set of interactive transitions as well as the classification are still allowed to be countably infinite. However, for the representation map approach later on, finiteness is required.

Definition 24 (Confluence classification)**.** *A* confluence classification P *for* $\mathcal{M}$ *is a set of sets of interactive transitions* $\{C_1, C_2, \ldots, C_n\} \subseteq \mathscr{P}(\hookrightarrow)$.

Given a set $\mathcal{T} \subseteq P$ of groups, we slightly abuse notation by writing $(s \xrightarrow{a} \mu) \in \mathcal{T}$ to denote that $(s \xrightarrow{a} \mu) \in C$ for some $C \in \mathcal{T}$. Additionally, we use $s \xrightarrow{a}_{C_i} \mu$ to denote that $(s \xrightarrow{a} \mu) \in C_i$ and $s \xrightarrow{a}_{\mathcal{T}} \mu$ to denote that $(s \xrightarrow{a} \mu) \in \mathcal{T}$. Similarly, we subscript reachability, joinability and convertibility arrows (e.g., $s \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} t$) to indicate that they only traverse transitions from a certain group or set of groups of transitions.

4.3. Confluent sets

We define confluence on a confluence classification: we designate *sets of groups* $\mathcal{T} \subseteq P$ to be confluent (now called *Markovian confluent*). Just like in probabilistic branching-time POR [23], only invisible transitions with a Dirac distribution are allowed to be confluent. For a set $\mathcal{T}$ to be Markovian confluent, it is therefore not allowed to contain any visible or probabilistic transitions. Still, prioritising invisible transitions may very well reduce probabilistic transitions too, as we will see in Section 5. The reason for excluding probabilistic τ-transitions from the confluent set is that confluence reduction based on them would not preserve branching bisimulation anymore (see [12] for an example). Hence, at this moment it is unclear which properties would still be preserved.

Confluence requires each transition $s \xrightarrow{a} \mu$ (allowing $a = \tau$) enabled together with a transition $s \xrightarrow{\tau}_{\mathcal{T}} t$ to have a mimicking transition $t \xrightarrow{a} \nu$ such that μ and ν are $\mathcal{R}^{\mathcal{T}}_{\mu,\nu}$-equivalent. Additionally, we require for each group in the classification that transitions from that group are mimicked by transitions from the same group. This turns out to be essential for closure of confluence under union. No restrictions are imposed on transitions that are not in any group, since they cannot be confluent anyway.

All is formalised in the definition below, and illustrated in Figure 7.

Definition 25 (Markovian confluence)**.** *Let* $P \subseteq \mathscr{P}(\hookrightarrow)$ *be a confluence classification for* $\mathcal{M}$*. Then, a set* $\mathcal{T} \subseteq P$ *is* Markovian confluent for P *if (1) it only contains sets of invisible transitions with Dirac distributions, and (2) for all* $s \xrightarrow{\tau}_{\mathcal{T}} t$ *and all transitions* $(s \xrightarrow{a} \mu) \neq (s \xrightarrow{\tau} t)$*:*

$$\begin{array}{lllll} (i) & (s \xrightarrow{a} \mu) \in P & \textit{implies} & \forall C \in P \,.\, s \xrightarrow{a}_{C} \mu & \implies \exists \nu \in \mathrm{Distr}(S) \,.\, t \xrightarrow{a}_{C} \nu \wedge \mu \equiv_{R^{\mathcal{T}}_{\mu,\nu}} \nu \\ (ii) & (s \xrightarrow{a} \mu) \notin P & \textit{implies} & & \exists \nu \in \mathrm{Distr}(S) \,.\, t \xrightarrow{a} \nu \quad \wedge \mu \equiv_{R^{\mathcal{T}}_{\mu,\nu}} \nu \end{array}$$

A transition $s \xrightarrow{\tau} t$ *is* Markovian confluent *if there exists a Markovian confluent set* $\mathcal{T}$ *such that* $s \xrightarrow{\tau}_{\mathcal{T}} t$*. Often, we omit the adjective 'Markovian'.*

(a) $(s \xrightarrow{a} \mu) \in P$. (b) $(s \xrightarrow{a} \mu) \notin P$.

Figure 7: Confluence diagrams for $s \xrightarrow{\tau}_{\mathcal{T}} t$. If the solid steps are present, so should the dashed ones be (such that $\mu \equiv_R \nu$).

Remark 26. Markovian transitions are irrelevant for confluence. After all, states with a τ-transition can never execute a Markovian transition due to the maximal progress assumption. Hence, if $s \xrightarrow{\tau} t$ and $s \xrightarrow{a} \mu$, then by definition of extended transitions $s \xrightarrow{a} \mu$ corresponds to an interactive transition $s \hookrightarrow \mu$.

Note that, due to the confluence classification, confluent transitions are always mimicked by confluent transitions. After all, transitions from a group $C \in P$ are mimicked by transitions from C. So, if C is designated confluent by $\mathcal{T}$, then all these confluent transitions are indeed mimicked by confluent transitions.

Although the confluence classification may appear restrictive, we will see that it is obtained naturally in practice. Transitions are often instantiations of higher-level syntactic constructs, and are therefore easily grouped together. Then, it makes sense to detect the confluence of such a higher-level construct. Also, to show that a certain set of invisible transitions is confluent, we can just take P to consist of one group containing precisely all those transitions. Then, the requirement for P-transitions to be mimicked by the same group coincides with the old requirement that confluent transitions are mimicked by confluent transitions.

Example 27. Figure 8 provides an MA $\mathcal{M}$ with nondeterminism, probability, Markovian rates and state labels. It is our running example to illustrate the various concepts related to confluence. We indicate a confluence classification P for $\mathcal{M}$ by superscripts on the τ-labels of some of the transitions:

$$\begin{aligned} C_1 = \{&(s_0, \tau, \mathbb{1}_{s_1}), (s_2, \tau, \mathbb{1}_{s_3}), (s_3, \tau, \mathbb{1}_{s_4}), (s_5, \tau, \mathbb{1}_{s_6}), (s_8, \tau, \mathbb{1}_{s_9}), (s_9, \tau, \mathbb{1}_{s_{10}}), \\ &(s_{10}, \tau, \mathbb{1}_{s_{11}}), (s_{11}, \tau, \mathbb{1}_{s_8}), (s_{13}, \tau, \mathbb{1}_{s_{14}}), (s_{16}, \tau, \mathbb{1}_{s_{15}}), (s_{15}, \tau, \mathbb{1}_{s_{10}})\} \\ C_2 = \{&(s_3, \tau, \mathbb{1}_{s_5}), (s_4, \tau, \mathbb{1}_{s_6})\} \\ C_3 = \{&(s_6, \tau, \mathbb{1}_{s_{17}})\} \end{aligned}$$

All transitions in P are labelled by τ, have a Dirac distribution and do not change the state labelling. Hence, they may potentially be confluent, if they additionally commute with all neighbouring transitions. Note that no other transitions can be confluent, as they all are visible (i.e., they are either labelled by a visible action or change the state labelling). For $\mathcal{T} = \{C_1\}$, we show that each transition in $\mathcal{T}$ is confluent.

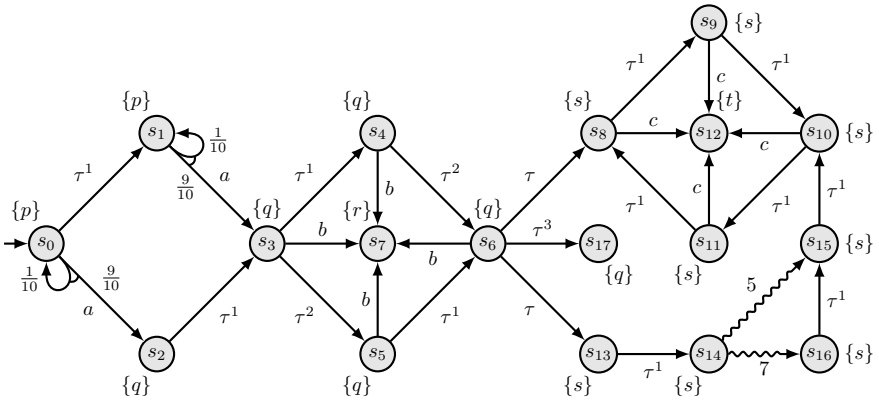


Figure 8: An MA $\mathcal{M}$.

First, consider $s_0 \xrightarrow{\tau}_{\mathcal{T}} s_1$. There is one other transition from s_0, namely $s_0 \xrightarrow{a} \mu$ with $\mu(s_2) = \frac{9}{10}$ and $\mu(s_0) = \frac{1}{10}$. Since $s_0 \xrightarrow{a} \mu \notin P$, we need to show that $\exists \nu \in \mathrm{Distr}(S)\ .\ s_1 \xrightarrow{a} \nu \wedge \mu \equiv_R \nu$. We take $s_1 \xrightarrow{a} \nu$ with $\nu(s_3) = \frac{9}{10}$ and $\nu(s_1) = \frac{1}{10}$. This yields $R = Id \cup \{(s_0, s_1), (s_1, s_0), (s_2, s_3), (s_3, s_2)\}$, with Id the identity relation. Indeed, μ and ν assign the same probability to each equivalence class of R, so $\mu \equiv_R \nu$.

Second, consider $s_2 \xrightarrow{\tau}_{\mathcal{T}} s_3$. Since there are no other transitions from s_2, there is nothing to check.

Finally, consider $s_3 \xrightarrow{\tau}_{\mathcal{T}} s_4$. It has two neighbouring transitions: $s_3 \xrightarrow{b} \mathbb{1}_{s_7}$ and $s_3 \xrightarrow{\tau} \mathbb{1}_{s_5}$. The first one can be mimicked by $s_4 \xrightarrow{b} \mathbb{1}_{s_7}$. Clearly $\mathbb{1}_{s_7} \equiv_R \mathbb{1}_{s_7}$, due to reflexivity. The second can be mimicked by $s_4 \xrightarrow{\tau} \mathbb{1}_{s_6}$. Then, $R = Id \cup \{(s_5, s_6), (s_6, s_5)\}$ and hence indeed $\mathbb{1}_{s_5} \equiv_R \mathbb{1}_{s_6}$. Since $s_3 \xrightarrow{\tau} \mathbb{1}_{s_5} \in C_2$, we additionally need to check that also $s_4 \xrightarrow{\tau} \mathbb{1}_{s_6} \in C_2$. This is indeed the case.

The remaining transitions can be treated similarly. □

4.4. Properties of confluent sets

Since confluent transitions are always mimicked by confluent transitions, confluent paths (i.e., paths following only transitions from a confluent set) are always joinable. This is captured by the following result.

Proposition 28. *Let P be a confluence classification for $\mathcal{M}$, and let $\mathcal{T}$ be a confluent set for P. Then,*

$$s \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} t \qquad \text{if and only if} \qquad s \twoheadleftarrow\!\!\!\twoheadrightarrow_{\mathcal{T}} t$$

Contrary to previous work, we now can show that confluent sets are indeed closed under union. This tells us that it is safe to show confluence of multiple sets of transitions in isolation, and then just take their union as one confluent set. Also, it implies that there exists a unique maximal confluent set.

Theorem 29. *Let P be a confluence classification for $\mathcal{M}$, and let $\mathcal{T}_1, \mathcal{T}_2$ be two Markovian confluent sets for P. Then, $\mathcal{T}_1 \cup \mathcal{T}_2$ is also a Markovian confluent set for P.*

Example 30. Example 27 demonstrated that $\mathcal{T} = \{C_1\}$ is confluent for our running example. In the same way, it can be shown that $\mathcal{T}' = \{C_2\}$ is confluent. Hence, $\mathcal{T}'' = \{C_1, C_2\}$ is also confluent. □

Remark 31. In earlier works [11, 12], confluent sets were not yet closed under union, even though this was assumed and was actually needed for confluence reduction to work. In practical applications the assumption turned out to be valid—in particular, the implementations of confluence reduction for LTSs and PAs were not erroneous. Still, technically, closure under union of confluent sets could not just be assumed. When taking the union of two valid sets of confluent transitions, their requirement that confluent transitions have to be mimicked by confluent transitions was possibly invalidated (as will be discussed in more detail in Section 8.3).

The final result of this section states that confluent transitions connect divergence-sensitive branching bisimilar states. This is a key result: it implies that confluent transitions can be given priority over other transitions without losing behaviour—when being careful not to ignore any behaviour indefinitely.

Theorem 32. *Let $s, s' \in S$ be two states and $\mathcal{T}$ a confluent set for some confluence classification P. Then,*

$$s \twoheadleftarrow\!\!\!\twoheadrightarrow_{\mathcal{T}} s' \text{ implies } s \approx_{\mathrm{b}}^{\mathrm{div}} s'$$

5. State space reduction using confluence

As explained in Section 3.2, we aim at omitting all intermediate states on confluent paths; after all, they are all bisimilar. Confluence even dictates that all visible transitions and divergences enabled from a state s can directly be mimicked from another state t if $s \twoheadrightarrow_{\mathcal{T}} t$. Hence, during state space generation 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. This technique was proposed first in [35], and later used in [11] and [12]. A similar construction was used in [36] for representing sets of states for the so-called *essential state* abstraction for probabilistic transition systems.

Since confluent joinability is transitive (as implied by Proposition 28), it follows immediately that all confluent paths starting in a certain state s always end up in a unique BSCC (as long as the system is finite).

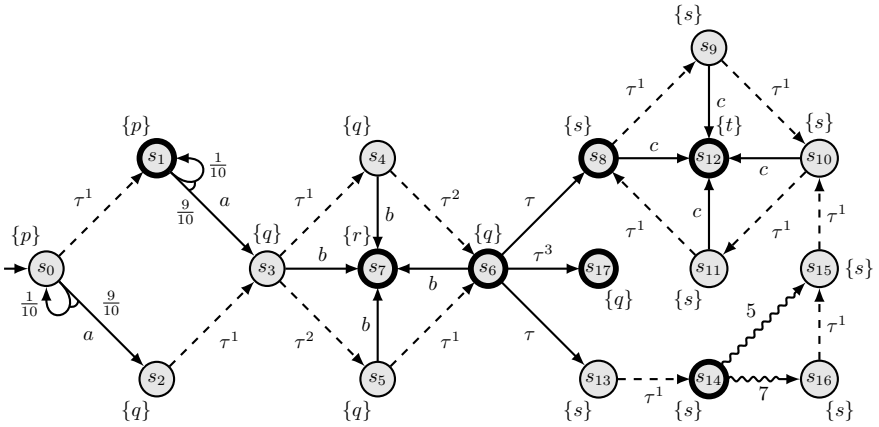


Figure 9: An MA $\mathcal{M}$ with confluent transitions and representatives.

5.1. Representation maps

Formally, we use a *representation map*, assigning a representative state $\phi(s)$ to every $s \in S$. We make sure that $\phi(s)$ exhibits all behaviour of s, by requiring $\phi(s)$ to be in a BSCC reachable from s via $\mathcal{T}$-transitions.

Definition 33 (Representation map)**.** *Let $\mathcal{T}$ be a confluent set for $\mathcal{M}$. Then, a function $\phi_{\mathcal{T}} \colon S \to S$ is a* representation map *for $\mathcal{M}$ under $\mathcal{T}$ if for all $s, s' \in S$*

- $s \twoheadrightarrow_{\mathcal{T}} \phi_{\mathcal{T}}(s)$*; and*
- $s \to_{\mathcal{T}} s'$ *implies* $\phi_{\mathcal{T}}(s) = \phi_{\mathcal{T}}(s')$*.*

The first requirement ensures that every representative is reachable from all states it represents, whereas the second takes care that all $\mathcal{T}$-related states have the same representative. Together, they imply that every representative is contained in a BSCC. Since all $\mathcal{T}$-related states have the same BSCC, as discussed above, it is always possible to find a representation map. We refer to [35] for the algorithm we use to construct it in our implementation—a variation on Tarjan's algorithm for finding strongly connected components [37]. Basically, the representation map is obtained by continuously following confluent transitions until ending up in either a state without any outgoing confluent transitions, or a cycle of confluent transitions.

Example 34. For our running example $\mathcal{M}$ from Example 27, $\mathcal{T} = \{C_1, C_2\}$ is confluent. One of the four possible representation maps ϕ for $\mathcal{M}$ under $\mathcal{T}$ is:

$$\phi(s_0) = \phi(s_1) = s_1 \qquad \phi(s_{12}) = s_{12}$$
$$\phi(s_2) = \phi(s_3) = \phi(s_4) = \phi(s_5) = \phi(s_6) = s_6 \qquad \phi(s_{13}) = \phi(s_{14}) = s_{14}$$
$$\phi(s_7) = s_7 \qquad \phi(s_{17}) = s_{17}$$
$$\phi(s_8) = \phi(s_9) = \phi(s_{10}) = \phi(s_{11}) = \phi(s_{15}) = \phi(s_{16}) = s_8$$

Figure 9 depicts confluent transitions by dashed arrows and representatives by a thick border. Indeed, each state can reach its representative via confluent transitions, and confluently connected states share the same representative. (Due to these requirements, $\phi(s_1) = s_0$ or $\phi(s_0) = s_0$ would be invalid.) □

5.2. Quotienting the state space using a representation map

Since representatives exhibit all behaviour of the states they represent, they can be used for state space reduction. More precisely, it is possible to define the quotient of an MA modulo a representation map. The quotient does not have any $\mathcal{T}$-transitions anymore, except for self-loops on representatives that have outgoing $\mathcal{T}$-transitions in the original system. These self-loops ensure preservation of divergences.

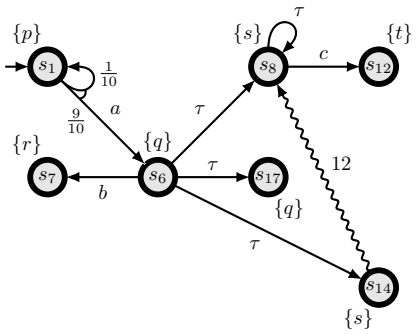


Figure 10: The quotient $\mathcal{M}/\phi$ of $\mathcal{M}$ of the MA $\mathcal{M}$ from Figure 8.

In [11, 12], these self-loops were not added to the quotient, and confluence reduction was not divergence sensitive. For MAs, omitting these self-loops may even erroneously enable Markovian transitions that were disabled in the presence of divergence due to the maximal progress assumption. Our definition does not only make the theory work for MAs, it even yields preservation of divergence-*sensitive* branching bisimulation. In particular, this implies that minimal reachability probabilities are preserved as well.

Definition 35 (Quotient)**.** *Let $\mathcal{T}$ be a confluent set for $\mathcal{M}$, and $\phi\colon S \to S$ a representation map for $\mathcal{M}$ under $\mathcal{T}$. Then, the* quotient of $\mathcal{M}$ modulo ϕ *is given by*

$$\mathcal{M}/\phi = \langle \phi(S), \phi(s^0), A, \hookrightarrow_\phi, \rightsquigarrow_\phi, \mathrm{AP}, L_\phi \rangle$$

such that, for all $s, s' \in S$, $\mu \in \mathrm{Distr}(S)$, $a \in A$, and $\lambda \in \mathbb{R}^{>0}$,

- $\phi(S) = \{\phi(t) \mid t \in S\}$;
- $\phi(s) \overset{a}{\hookrightarrow}_\phi \mu_\phi$ *if and only if* $\phi(s) \overset{a}{\hookrightarrow} \mu$;
- $\phi(s) \overset{\lambda}{\rightsquigarrow}_\phi \phi(s')$ *if and only if* $\lambda = \sum_{\lambda' \in \Lambda(s,s')} \lambda'$; *and*
- $L_\phi(\phi(s)) = L(s)$.

where $\Lambda(s, s')$ is the multiset $\{\!|\lambda' \in \mathbb{R} \mid \exists s^* \in S \,.\, \phi(s) \overset{\lambda'}{\rightsquigarrow} s^* \wedge \phi(s^*) = \phi(s')|\!\}$, and μ_ϕ is the lifting of μ over ϕ as introduced in Definition 1.

Note that each interactive transition from $\phi(s)$ in $\mathcal{M}$ is lifted to $\mathcal{M}/\phi$ by changing all states in the support of its target distribution to their representatives. Additionally, each pair $\phi(s), \phi(s')$ of representative states in $\mathcal{M}/\phi$ has a connecting Markovian transition with rate equal to the total exit rate of $\phi(s)$ in $\mathcal{M}$ to states s^* that have $\phi(s')$ as their representative (unless this sum is 0). It is easy to see that this implies $\phi(s) \xrightarrow{\chi(\lambda)}_\phi \mu_\phi$ if and only if $\phi(s) \xrightarrow{\chi(\lambda)} \mu$.

Example 36. Using the representation map from Example 34, Figure 10 shows the quotient of our running example. The set of states is obtained by applying ϕ to all states of $\mathcal{M}$, resulting in the black states from Figure 9. The initial state of $\mathcal{M}/\phi$ is the representative of s_0, which is s_1.

To understand the construction of $\hookrightarrow$, consider the original transition $s_1 \overset{a}{\hookrightarrow} \nu$ with $\nu(s_3) = \frac{9}{10}$ and $\nu(s_1) = \frac{1}{10}$. It yields the transition $s_1 \overset{a}{\hookrightarrow} \mu_\phi$ in the quotient. Since $\phi(s_3) = s_6$ and $\phi(s_1) = s_1$, it follows that μ_ϕ is the distribution assigning probability $\frac{9}{10}$ to s_6 and $\frac{1}{10}$ to s_1. Note that, in the same way, the confluent transition from s_8 yields a self-loop due to the fact that $\phi(s_9) = s_8$.

To understand the construction of $\rightsquigarrow$, we look at the transition $s_{14} \overset{12}{\rightsquigarrow} s_8$ in the quotient. We derive $\Lambda(s_{14}, s_8) = \{\!|\lambda' \in \mathbb{R} \mid \exists s^* \in S \,.\, s_{14} \overset{\lambda'}{\rightsquigarrow} s^* \wedge \phi(s^*) = s_8|\!\} = \{\!|5, 7|\!\}$. Hence, in $\mathcal{M}$ there is a total exit rate of $5 + 7 = 12$ from s_{14} to states having s_8 as their representative. Therefore, the quotient has a transition $s_{14} \overset{12}{\rightsquigarrow} s_8$. Note that the multiset is necessary due to the possibility of having several transitions with the same rate to states having the same representative.

Note that s_8 has a τ-loop in the quotient, preserving the divergence that was possible by cycling through s_8, s_9, s_{10} and s_{11} in $\mathcal{M}$. This self-loop results from the transition $s_8 \overset{\tau}{\hookrightarrow} \mathbb{1}_{s_9}$, since $\phi(s_8) = \phi(s_9) = s_8$. □

Since $\mathcal{T}$-transitions connect bisimilar states, and representatives exhibit all behaviour of the states they represent, we can prove the following theorem. It is again a key result, showing that our reduction is safe with respect to divergence-sensitive branching bisimulation.

Theorem 37. *For every representation map $\phi\colon S \to S$ for $\mathcal{M}$ under some confluent set $\mathcal{T}$, $\mathcal{M}/\phi \approx_{\mathrm{b}}^{\mathrm{div}} \mathcal{M}$.*

6. Symbolic detection of Markovian confluence

Although the MA-based definition of confluence in Section 4 is useful to show the correctness of our approach, it is often not feasible to check in practice. After all, we want to reduce *on-the-fly* to obtain a smaller state space without first generating the unreduced one. Therefore, we propose a set of heuristics to detect Markovian confluence in the context of the process-algebraic modelling language MAPA [14, 34], similar to the way non-Markovian confluence [11, 12] and classical POR [31] are applied.

MAPA supports all features of MAs: nondeterministic choice, probabilistic choice and Markovian delay. It also has data variables and compositionality features such as parallel composition and action hiding, thus enabling the efficient modelling of large systems. A set of goal states can be selected by means of one or more conditions over the actions and variables, allowing state-based model checking via for instance the tool IMCA [38]. Every well-formed MAPA specification can be *linearised* efficiently to a restricted form of the language: the Markovian Linear Process Equation (MLPE). Hence, it suffices to define our heuristics on this subset of the language. A complete definition of MAPA goes beyond the scope of this paper; see [14, 34].

In essence, an MLPE is a process X with a vector of global variables $\boldsymbol{g}$ of type $\boldsymbol{G}$ and a set of *summands*. The state space consists of an initial state $X(\boldsymbol{g_o})$ and all other states $X(\boldsymbol{g'})$ that can be reached via the set of summands. These summands are symbolic transitions that can be chosen nondeterministically, provided that their guard holds for the current state—similar to a guarded command. Corresponding to the two transition relations of MAs, the MLPE has *interactive summands* (having an action a_i) and *Markovian summands* (having a rate λ_j). We give a global overview here, and refer to [34] for the technical details.

Definition 38 (MLPEs)**.** *An* MLPE *is a MAPA specification of the following format:*

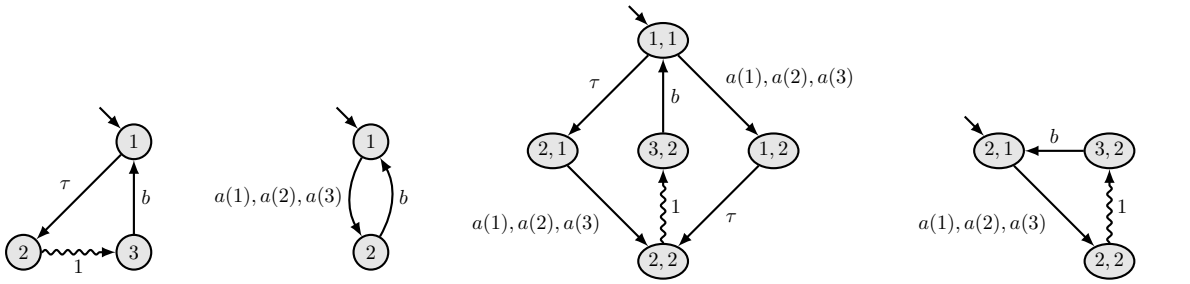
$$X(\boldsymbol{g}:\boldsymbol{G}) = \sum_{i\in I}\sum_{\boldsymbol{d_i}:\boldsymbol{D_i}} c_i \Rightarrow a_i(\boldsymbol{b_i}) \sum_{\boldsymbol{e_i}:\boldsymbol{E_i}} f_i : X(\boldsymbol{n_i}) + \sum_{j\in J}\sum_{\boldsymbol{d_j}:\boldsymbol{D_j}} c_j \Rightarrow (\lambda_j)\cdot X(\boldsymbol{n_j})$$

In an MLPE, $\sum_{\boldsymbol{d_i}:\boldsymbol{D_i}}$ is a nondeterministic choice, $a_i(\boldsymbol{b_i})\sum_{\boldsymbol{e_i}:\boldsymbol{E_i}} f_i$ an action-prefixed probabilistic choice and (λ_j) a Markovian delay. The terms $X(\boldsymbol{n_i})$ and $X(\boldsymbol{n_j})$ indicate process instantiation.

Given a state $X(\boldsymbol{g})$, a (possibly empty) vector of local variables $\boldsymbol{d_i}$ can be chosen nondeterministically from the set $\boldsymbol{D_i}$ (indicated by $\sum_{\boldsymbol{d_i}:\boldsymbol{D_i}}$) such that the Boolean condition $c_i(\boldsymbol{g},\boldsymbol{d_i})$ holds. Then, the action $a_i(\boldsymbol{b_i}(\boldsymbol{g},\boldsymbol{d_i}))$ is taken, a vector $\boldsymbol{e_i}$ of type $\boldsymbol{E_i}$ is chosen probabilistically—each concrete $\boldsymbol{e_i}'$ with probability $f_i(\boldsymbol{g},\boldsymbol{d_i},\boldsymbol{e_i}))$—as indicated by $a_i(\boldsymbol{b_i})\sum_{\boldsymbol{e_i}:\boldsymbol{E_i}} f_i$, and the system continues as $X(\boldsymbol{n_i}(\boldsymbol{g},\boldsymbol{d_i},\boldsymbol{e_i}))$. Note that c_i should be a Boolean expression (possibly depending on $\boldsymbol{g}$ and $\boldsymbol{d_i}$), whereas f_i should be a real-valued expression and $\boldsymbol{n_i}$ should be of type $\boldsymbol{G}$. We use $c_i(\boldsymbol{g},\boldsymbol{d_i})$ to indicate that the actual values of $\boldsymbol{g}$ and $\boldsymbol{d_i}$ are substituted in c_i. Markovian transitions arise similarly by choosing a $\boldsymbol{d_j}$ such that $c_j(\boldsymbol{d_j})$ holds, delaying according to rate $\lambda_j(\boldsymbol{g},\boldsymbol{d_j})$ (indicated by (λ_j)) and continuing as $X(\boldsymbol{n_j}(\boldsymbol{g},\boldsymbol{d_i}))$.

Example 39. We consider a system consisting of two components, shown in Figures 11(a) and 11(b). Component 1 continuously alternates between the actions τ and b, with a Markovian delay with rate one in between. Component 2 alternates between the actions a and b, nondeterministically parameterising the a-action with either one, two or three. The components have to synchronise on the b-action, and hence the composed behaviour is as in Figure 11(c).

Process-algebraically, the components can be written as $X_1 = \tau\cdot(1)\cdot b\cdot X_1$ and $X_2 = \sum_{i:\{1..3\}} a(i)\cdot b\cdot X_2$. The complete system is then the parallel composition $X = X_1 \,||\, X_2$, with synchronisation over shared actions.



(a) Component 1 (b) Component 2 (c) The unreduced state space. (d) The reduced state space.

Figure 11: State space generation using confluence. We write multiple actions at a transition as an abbreviation of multiple transitions between the same two states.

This parallel composition results in the following MLPE (with initial state $X(1,1)$):

$$\begin{aligned} X(pc_1 : \{1,2,3\}, pc_2 : \{1,2\}) = \quad & pc_1 = 1 \Rightarrow \tau \cdot X(2, pc_2) && (1)\\ + \quad & pc_1 = 2 \Rightarrow (1) \cdot X(3, pc_2) && (2)\\ + \textstyle\sum_{d:\{1..3\}} \quad & pc_2 = 1 \Rightarrow a(d) \cdot X(pc_1, 2) && (3)\\ + \quad & pc_1 = 3 \wedge pc_2 = 2 \Rightarrow b \cdot X(1,1) && (4) \end{aligned}$$

Note that two program counters pc_1 and pc_2 are used ($\boldsymbol{g} = (pc_1, pc_2)$), keeping track of the positions of X_1 and X_2. There are three interactive summands and one Markovian, so $|I| = 3$ and $|J| = 1$.

In the first summand, $c_1 = pc_1 = 1$, $a_i = \tau$, $\boldsymbol{b_i} = ()$ and $\boldsymbol{n_i} = (2, pc_2)$[5]. It models that X_1 can do a τ-action whenever $pc_1 = 1$, continuing with $pc_1 = 2$ while leaving pc_2 invariant. Similarly, the second summand models the Markovian delay and the third summand models the nondeterministic action by X_2. The final summand takes care of the synchronisation over b. □

Confluent summands. Each interactive summand of an MLPE yields a set of interactive transitions, whereas each Markovian summand yields a set of Markovian transitions. Instead of detecting *individual* confluent transitions, we detect *confluent summands*. A summand is considered confluent if the set of *all* transitions it may generate is guaranteed to be confluent. Since only interactive transitions can be confluent, only interactive summands can be confluent. In practice, often many of them are, due to the interleaving of behaviour that is local to separate components.

We consider a confluence classification $P = \{C_1, C_2, \ldots, C_k\}$ that, for each interactive summand $i \in I = \{1, \ldots, k\}$, contains a group C_i consisting of all transitions generated by that summand. For each interactive summand i we try to show that $\{C_i\}$ is confluent. By Theorem 29 (closure under union), the set of transitions generated by all confluent summands together is also confluent. Then, this information can be applied during state space generation by ignoring all other summands whenever a confluent summand is enabled (continuing until reaching a representative in a BSCC, as explained in the previous section).

6.1. Characterisation of confluent summands

Given a set C_i of all transitions generated by a summand i, we need to check whether this set satisfies Definition 25. Since we want to reduce *before* generating the full state space, we cannot first construct C_i and then check the conditions. Rather, we overapproximate the confluence requirements by checking symbolically if all transitions that summand i may ever generate satisfy them. We need to check if all these transitions are invisible (so the *action should be* τ and the *state labelling should not be influenced*) and have a *Dirac distribution*, and we need to check whether they all *commute with their neighbouring transitions.*

[5]Technically, a probabilistic choice should be included: $pc_1 = 1 \Rightarrow \tau \sum_{e:\{\perp\}} 1 : X(2, pc_2)$. We omit it, though, since the probabilistic choice is trivial and does not influence the continuation of the process.

We can easily check if all transitions generated by a summand have an invisible action: the summand should just have τ as its action. To check if a summand can only yield Dirac distributions, there should be only one possible next state for each valuation of the global and local parameters that enables its condition:

Definition 40 (Action-invisible and non-probabilistic summands)**.** *An interactive summand i is* action-invisible *if $a_i(\boldsymbol{b_i}) = \tau$. It is* non-probabilistic *if for every $\boldsymbol{v} \in \boldsymbol{G}$ and $\boldsymbol{d'_i} \in \boldsymbol{D_i}$ such that $c_i(\boldsymbol{v}, \boldsymbol{d'_i})$, there exists a unique state $\boldsymbol{v'} \in \boldsymbol{G}$ such that $\boldsymbol{n_i}(\boldsymbol{v}, \boldsymbol{d'_i}, \boldsymbol{e'_i}) = \boldsymbol{v'}$ for every $\boldsymbol{e'_i} \in \boldsymbol{E_i}$*[6].

Example 41. Consider the MLPE from Example 39. Only the first summand is action-invisible, as all others have a Markovian rate or an action different from τ. Also, this summand is non-probabilistic.

As a more involved example of a non-probabilistic summand, consider

$$\sum_{d:\{-1..1\}} (x = 1 \vee x = 2) \wedge d = 1 - x \quad \Rightarrow \quad \tau \sum_{e:\{1+x..d+2x\}} \frac{1}{d+x} : X(d + 2e)$$

Here, if $x = 1$ or $x = 2$, the variable d gets assigned the value -1 or 1, nondeterministically, but restricted by the condition $d = 1 - x$. Hence, this nondeterministic choice is trivial as it leaves only one possible option for d. Then, action τ is taken and followed by a probabilistic choice. The variable e gets a value from the set $\{1 + x, \ldots, d + 2x\}$, where each value has probability $\frac{1}{d+x}$. However, since d can only be equal to $1 - x$, we know that e must always get assigned the value $1 - x + 2x = x + 1$ with probability $\frac{1}{1-x+x} = \frac{1}{1} = 1$, so the process continues as $X(1 - x + 2x + 2) = X(3 + x)$ and there is no actual probabilistic choice. Hence, the summand can be considered non-probabilistic. □

Commuting summands. For an action-invisible non-probabilistic interactive summand i to only generate transitions that commute with all their neighbours, we need to check if every transition $s \xrightarrow{a} \mu$ enabled together with a transition $s \xrightarrow{\tau} \mathbb{1}_t$ generated by i can be mimicked from state t. Additionally, the mimicking transition $t \xrightarrow{a} \nu$ should be from the same group as $s \xrightarrow{a} \mu$. Since each group consists of all transitions generated by one summand, this boils down to the requirement that if $s \xrightarrow{a} \mu$ is generated by summand j, then so should $t \xrightarrow{a} \nu$ be. To check if all transitions generated by a summand i commute in this way with all other transitions, we check for each summand j if *all* transitions that it may generate commute with *all* transitions that summand i may generate. In that case, we say that the two summands *commute*. Again, we look at the specification of the summands and not at the actual transitions they generate.

Two summands i, j commute if they cannot disable each other and do not influence each other's action parameters, probabilities and next states. Then, any transition $s \xrightarrow{a} \mu$ enabled from state s by summand j must still be enabled from state t as a transition $t \xrightarrow{a} \nu$, the τ-transition from summand i is still enabled in all states in the support of μ and the order of the execution of the two transitions does not influence the observable behaviour or the final state. Hence, $\mu \equiv_R \nu$. Since Definition 25 does not require transitions to commute with themselves, a summand trivially commutes with itself if it generates only one transition per state.

To formally define commuting summands, we already assume that one of them is action-invisible and non-probabilistic. Hence, we can write $\boldsymbol{n_i}(\boldsymbol{v}, \boldsymbol{d'_i})$ for its unique next state given $\boldsymbol{v}$ and $\boldsymbol{d'_i}$. For an action-invisible non-probabilistic summand i to commute with a summand j, we have to investigate their behaviour for all state vectors $\boldsymbol{v} \in \boldsymbol{G}$ and local variable vectors $\boldsymbol{d'_i} \in \boldsymbol{D_i}, \boldsymbol{d'_j} \in \boldsymbol{D_j}$ such that both summands are enabled: $c_i(\boldsymbol{v}, \boldsymbol{d'_i}) \wedge c_j(\boldsymbol{v}, \boldsymbol{d'_j})$. The maximal progress assumption dictates that interactive summands and Markovian summands can never be enabled at the same time, so we only need to consider the interactive summands.

Definition 42 (Commuting summands)**.** *An action-invisible non-probabilistic summand i* commutes *with an interactive summand j (possibly $i = j$) if for all $\boldsymbol{v} \in \boldsymbol{G}$, $\boldsymbol{d'_i} \in \boldsymbol{D_i}, \boldsymbol{d'_j} \in \boldsymbol{D_j}$ such that $c_i(\boldsymbol{v}, \boldsymbol{d'_i}) \wedge c_j(\boldsymbol{v}, \boldsymbol{d'_j})$:*

- *Summand i cannot not disable summand j, and vice versa: $c_j(\boldsymbol{n_i}(\boldsymbol{v}, \boldsymbol{d'_i}), \boldsymbol{d'_j}) \wedge c_i(\boldsymbol{n_j}(\boldsymbol{v}, \boldsymbol{d'_j}, \boldsymbol{e'_j}), \boldsymbol{d'_i})$ for every $\boldsymbol{e'_j} \in \boldsymbol{E_j}$ such that $f_j(\boldsymbol{v}, \boldsymbol{d'_j}, \boldsymbol{e'_j}) > 0$.*

[6] We could also weaken this condition slightly to $\forall \boldsymbol{e'_i} \in \boldsymbol{E_i} \,.\, f_i(\boldsymbol{v}, \boldsymbol{d'_i}, \boldsymbol{e'_i}) > 0 \implies \boldsymbol{n_i}(\boldsymbol{v}, \boldsymbol{d'_i}, \boldsymbol{e'_i}) = \boldsymbol{v'}$. However, in that case we need to be more strict later on, requiring the probabilities of a confluent summand to remain invariant. For the current formulation, the probability expression can be changed without influencing the next state.

- *Summand i cannot influence the action parameters of summand j:* $b_j(\boldsymbol{v}, \boldsymbol{d'_j}) = b_j(\boldsymbol{n_i}(\boldsymbol{v}, \boldsymbol{d'_i}), \boldsymbol{d'_j})$. *Note that summand i was already assumed to have no action parameters (as its action is always τ), so these cannot be influenced by j anyway—hence, no converse equality has to be checked.*
- *Summand i cannot influence the probabilities of summand j:* $f_j(\boldsymbol{v}, \boldsymbol{d'_j}, \boldsymbol{e'_j}) = f_j(\boldsymbol{n_i}(\boldsymbol{v}, \boldsymbol{d'_i}), \boldsymbol{d'_j}, \boldsymbol{e'_j})$ *for every* $\boldsymbol{e'_j} \in \boldsymbol{E_j}$. *Note that summand i was already assumed to have a unique next state* $\boldsymbol{v'} \in \boldsymbol{G}$ *for any state vector* $\boldsymbol{v} \in \boldsymbol{G}$ *and local variable vector* $\boldsymbol{d'_i} \in \boldsymbol{D_i}$, *so no converse equality has to be checked. (Under the adapted condition discussed in Footnote 6 on page 18, this would not be the case anymore.)*
- *Execution of summands i and j in either order yield the same next state:* $\boldsymbol{n_j}(\boldsymbol{n_i}(\boldsymbol{v}, \boldsymbol{d'_i}), \boldsymbol{d'_j}, \boldsymbol{e'_j}) = \boldsymbol{n_i}(\boldsymbol{n_j}(\boldsymbol{v}, \boldsymbol{d'_j}, \boldsymbol{e'_j}), \boldsymbol{d'_i})$ *for every* $\boldsymbol{e'_j} \in \boldsymbol{E_j}$ *such that* $f_j(\boldsymbol{v}, \boldsymbol{d'_j}, \boldsymbol{e'_j}) > 0$.

Additionally, i commutes with j if $i = j \wedge \boldsymbol{n_i}(\boldsymbol{v}, \boldsymbol{d'_i}) = \boldsymbol{n_j}(\boldsymbol{v}, \boldsymbol{d'_j})$ *for all* $\boldsymbol{v} \in \boldsymbol{G}$, $\boldsymbol{d'_i} \in \boldsymbol{D_i}$.

Example 43. Consider again the MLPE from Example 39. The first summand commutes with itself, due to the absence of a nondeterministic local choice (the *alternatively* clause in the definition above). It also commutes with the second and the fourth summand, since they can never be enabled at the same time (so the clause $c_i(\boldsymbol{v}, \boldsymbol{d'_i}) \wedge c_j(\boldsymbol{v}, \boldsymbol{d'_j})$ in the definition above never holds). This is immediate from the fact that each possible value of pc_1 enables at most one of these summands.

Finally, the first summand commutes with the third summand, since there is no overlap between the variables changed by summand 1 (pc_1) and the variables used by summand 3 (pc_2) and vice versa. Hence, they cannot disable each other or influence each other's actions or probabilities. Also, their order of execution is irrelevant for the continuation. □

Clearly, if a summand i commutes with all summands including itself, it satisfies the commutativity requirements of confluence. As all formulas for commutativity are quantifier-free and in practice often either trivially false or true, they can generally be solved using an SMT solver for the data types involved. For n summands, n^2 formulas need to be solved; the complexity of this procedure depends on the data types.

Stuttering summands. Finally, for an interactive summand to generate only stuttering transitions, it suffices to check that $c_i(\boldsymbol{v}, \boldsymbol{d'_i}) \implies L(\boldsymbol{v}) = L(\boldsymbol{n_i}(\boldsymbol{v}, \boldsymbol{d'_i}))$ for all $\boldsymbol{v} \in \boldsymbol{G}$ and $\boldsymbol{d'_i} \in \boldsymbol{D_i}$. In MAPA, we defined the state labelling of the MA corresponding to a MAPA specification such that each state is labelled by the set of visible actions it immediately enables[7] [34]. Hence, for a summand to be invisible with respect to the state labelling it should only leave invariant the set of enabled visible actions.[8]

If a summand i is action-invisible and commutes with all summands, we already know that it can never disable another summand—if it disables itself that is fine, since it cannot produce any visible actions. Hence, we only still need to verify whether it can never *enable* a summand having a visible action.

Definition 44 (Stuttering summands)**.** *An action-invisible non-probabilistic summand i that commutes with all interactive summands is* stuttering *if, for each interactive summand j, the condition of j can never be enabled by i if j has a visible action. That is, for all* $\boldsymbol{v} \in \boldsymbol{G}$, $\boldsymbol{d'_i} \in \boldsymbol{D_i}$ *and* $\boldsymbol{d'_j} \in \boldsymbol{D_j}$*:*

$$c_i(\boldsymbol{v}, \boldsymbol{d'_i}) \wedge a_j \neq \tau \wedge \neg c_j(\boldsymbol{v}, \boldsymbol{d'_j}) \implies \neg c_j(\boldsymbol{n_i}(\boldsymbol{v}, \boldsymbol{d'_i}), \boldsymbol{d'_j})$$

Example 45. For the MLPE in Example 39, assume that we are only interested in observing the a-action. Then, it is easy to see that the first summand cannot enable another summand that has a visible action. Since only a is of interest, we just need to consider the third summand. Indeed, the first summand does not change any variables that are used in the condition of the third summand. □

The following proposition follows from the reasoning above.

Proposition 46. *An interactive summand that is action-invisible, non-probabilistic and stuttering, and that also commutes with all interactive summands including itself, generates only confluent transitions.*

[7] This also allows observable conditions over the global variables, by using them as the enabling condition of a special action.
[8] This requirement can be alleviated by hiding all actions that are not used in the properties of interest.

Example 47. In the previous three examples, we saw that the first summand of the MLPE from Example 39 has all properties for it to be confluent. This knowledge can be used for reduction during state space generation. Whenever the confluent summand is enabled in some state it is taken immediately, and the intermediate state is not stored. Figure 11(d) demonstrates the state space obtained in this way. □

6.2. Heuristics for confluent summands

Although the symbolic characterisation introduced above is sound, it may not always be feasible or efficient to apply. Only action invisibility can easily be checked syntactically, so we need heuristics for checking if a summand is non-probabilistic, commuting and stuttering. Although our heuristics are rather simple and intuitive, they are widely applicable (as demonstrated by our case studies).

We refer to [34] for a detailed explanation of these heuristics. Basically, we check for non-probabilistic summands by just looking whether $|\boldsymbol{E_i}| = 1$ and for stuttering summands by looking whether there are no interactive summands with action $a_j \neq \tau$ that can be enabled (based on overlap in variables that are used and changed). We investigate whether two summands commute by checking whether they (1) never are enabled at the same time or (2) act on disjoint sets of variables. This is often the case, due to parallel composition of largely independent components. A summand commutes with itself if $|\boldsymbol{D_i}| = 1$.

Remark 48. To detect confluence, we range over all summands. For each summand, we check if it has label τ and no real probabilistic choice, and produces only stuttering transitions. Additionally, we check if it commutes with all other interactive summands as above. In total, we have to check for each *pair* of summands if their conditions are mutually exclusive, if they coincide or if their variables are disjoint. Hence, the complexity of confluence checking is quadratic in the size of the MLPE. In the worst case, the size of an MLPE is quadratic in the size of the original MAPA specification [34].

7. Case studies

We implemented confluence reduction in SCOOP [39, 34]. This tool takes MAPA specifications as input and is able to perform several reduction techniques, such as dead variable reduction [40] and constant elimination [34]. SCOOP can generate state spaces in multiple formats, among which the one for the IMCA tool for model checking MAs [38, 9]. We have updated the state space generation algorithm as explained in Section 5. The updated algorithm is similar to the one presented in Tables 5 and 6 of [35]; the main differences are that we deal with probabilistic and Markovian transitions and include a τ-labelled selfloop in each representative state having at least one confluent outgoing transition, both according to Definition 35. These selfloop are to ensure the preservation of divergences, as discussed in Section 5.2. Detection of confluent transitions is implemented by ranging over all interactive summands and checking whether they conform to the heuristics introduced in Section 6.2—see [34] for more details and examples on these heuristics.

For three different MAPA specifications, we present the size of their state spaces with and without confluence reduction, as well as the time to generate them with SCOOP and to subsequently analyse them with IMCA. That way, the impact of confluence reduction on both MA generation and analysis becomes clear[9].

It was recently shown [42] that the (quantitative) behavioural equivalence induced by branching bisimulation leaves invariant expected times to reachability and long-run averages. We conjecture that this also holds for time-bounded reachability probabilities, as indeed turned out to be the case for all our models. A logic precisely characterising Markovian branching bisimulation would be interesting future work.

[9] The tool (for download and web-based use [41]) and all models can be found on `http://fmt.cs.utwente.nl/~timmer/scoop`.

Specification	Original state space				Reduced state space				Impact	
	States	Trans.	SCOOP	IMCA	States	Trans.	SCOOP	IMCA	States	Time
`leader-3-7`	25,505	34,257	4.7	102.5	5,564	6,819	5.1	9.3	-78%	-87%
`leader-3-9`	52,465	71,034	9.7	212.0	11,058	13,661	10.4	17.8	-79%	-87%
`leader-3-11`	93,801	127,683	18.0	429.3	19,344	24,043	19.2	31.9	-79%	-89%
`leader-4-2`	8,467	11,600	2.1	74.0	2,204	2,859	2.5	6.8	-74%	-88%
`leader-4-3`	35,468	50,612	9.0	363.8	7,876	10,352	8.7	33.3	-78%	-89%
`leader-4-4`	101,261	148,024	25.8	1,309.8	20,857	28,023	24.3	94.4	-79%	-91%
`polling-2-2-4`	4,811	8,578	0.7	3.7	3,047	6,814	0.7	2.3	-37%	-32%
`polling-2-2-6`	27,651	51,098	12.7	91.0	16,557	40,004	5.4	49.0	-40%	-48%
`polling-2-4-2`	6,667	11,290	0.9	39.9	4,745	9,368	0.9	26.6	-29%	-33%
`polling-2-5-2`	27,659	47,130	4.0	1,571.7	19,721	39,192	4.0	1,054.6	-29%	-33%
`polling-3-2-2`	2,600	4,909	0.4	7.1	1,914	4,223	0.5	4.8	-26%	-29%
`polling-4-6-1`	15,439	29,506	3.1	330.4	4,802	18,869	3.0	109.4	-69%	-66%
`polling-5-4-1`	21,880	43,760	5.1	815.9	6,250	28,130	5.1	318.3	-71%	-61%
`processor-2`	2,508	4,608	0.7	2.8	1,514	3,043	0.8	1.2	-44%	-43%
`processor-3`	10,852	20,872	3.1	66.3	6,509	13,738	3.3	23.0	-45%	-62%
`processor-4`	31,832	62,356	10.8	924.5	19,025	41,018	10.3	365.6	-45%	-60%

Table 1: State space generation and analysis using confluence reduction (on a 2.4 GHz 4 GB Intel Core 2 Duo MacBook). Runtimes in SCOOP and IMCA are in seconds.

Leader election protocol. The first case study is a probabilistic leader election protocol based on Itai-Rodeh (Algorithm $\mathcal{B}$ from [43]), used in [12] as well to demonstrate confluence reduction for probabilistic automata. It uses asynchronous channels and allows for multiple nodes, throwing dice to break the symmetry. We added a rate 1 to a node throwing a die to get an MA model based on the original case study. We computed the minimal probability (with error bound 0.01) of electing the first node as leader within 5 time units. The results are presented in Table 1, where we denote by `leader-i-j` the variant with `i` nodes and `j`-sided dice. The computed probability varies from 0.09 for `leader-4-2` to 0.32 for `leader-3-11`. Confluence saved almost 90% of the total time to generate and analyse the models. The substantial reductions are due to extensive interleaving with little communication.

Omitting the Markovian delay yields a PA, which we can minimise with respect to probabilistic branching bisimulation using CADP [44]. Comparing these results with ours, we find that confluence reduction got rid of approximately 90% of the total number of states that could have been omitted.

Queueing system. The second case study is the queueing system from [14]. It consists of multiple stations with incoming jobs, and one server that polls the stations for work. With some probability, communication fails. There can be different sizes of buffers in the stations, and multiple types of jobs with different service rates. In Table 1, we let `polling-i-j-k` denote the variant with `i` stations, all having buffers of size `j` and `k` types of jobs. Note that, although significant reductions are obtained, the reduction in states precisely corresponds to the reduction in transitions; this implies that only trivially confluent transitions could be reduced (i.e., invisible transitions without any other transitions from the same source state). We computed the minimal and maximal expected time to the situation in which all buffers are full. This turns out to be at least 1.1—for `polling-3-2-2`—and at most 124—for `polling-2-5-2`. Reductions were less substantial, due to the presence of many (visible) probabilistic and Markovian transitions.

Processor architecture. The third case study is a GSPN model of a 2×2 concurrent processor architecture, parameterised in the level k of multitasking, taken from Figure 11.7 in [45]. We constructed a corresponding MAPA model, modelling each place as a global variable and each transition as a summand. As in [45], we computed the throughput of one of the processors, given by the long-run average of having a token in a certain place of the GSPN. Whereas [45] resolved all nondeterminism and found for instance a throughput of 0.903 for $k = 2$, we can retain the nondeterminism and obtain the more informative interval $[0.811, 0.995]$. (When resolving nondeterminism as before, we reproduce the original result 0.903.)

Our results clearly show the significant effect of confluence reduction on the state space sizes and the duration of the heavy numerical computations by IMCA. The generation times by SCOOP are not reduced as much, due to the additional overhead of computing representative states. To keep memory usage in the

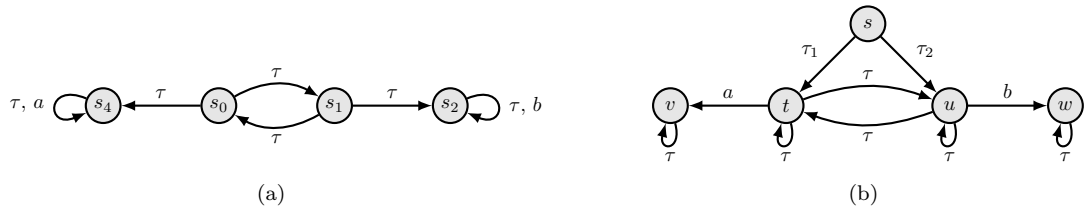


Figure 12: Counterexample for joinable confluent connectivity and explicit confluent mimicking.

order of the reduced state space, the representation map is deliberately not stored and therefore potentially recomputed for some states. More time-efficient storage settings of the representation map, at the cost of more memory usage, are also supported by SCOOP.

Reducing the model to an IMC by changing probabilistic choices to nondeterministic choices, we can minimise with respect to stochastic branching bisimulation using CADP [44]. Comparing these results with ours, we find that confluence reduction managed to find about two thirds of the maximal reduction possible.

8. Failing alternatives

We showed that confluence preserves divergence-sensitive branching bisimilation. In this section we discuss some interesting variations on our definitions, explaining why they do not serve our purpose.

8.1. Joinable confluent connectivity

Given a confluent transition $s \xrightarrow{\tau}_{\mathcal{T}} t$, we require the distributions μ and ν of a transition $s \xrightarrow{a} \mu$ and its mimicking transition $t \xrightarrow{a} \nu$ to assign equal probabilities to each equivalence class of the relation $R \supseteq \{(s,t) \in \mathit{supp}(\mu) \times \mathit{supp}(\nu) \mid (s \xrightarrow{\tau} t) \in \mathcal{T}\}$. Based on the intuition that confluent transitions connect bisimilar states, it may seem possible to relax this definition by using the larger relation

$$R = \{(s,t) \in \mathit{supp}(\mu) \times \mathit{supp}(\nu) \mid s \twoheadleftarrow\!\!\!\rightarrow_{\mathcal{T}} t\}$$

We show that this would be incorrect by demonstrating problems with an even less liberal definition, applying joinability instead of convertibility:

$$R = \{(s,t) \in \mathit{supp}(\mu) \times \mathit{supp}(\nu) \mid s \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} t\}$$

To see what could go wrong, consider the system in Figure 12(a). We take the confluence classification $\mathcal{P} = \{C\}$ with C the set of all τ-transitions in this model, and let $\mathcal{T} = \{C\}$. It is not hard to verify that all conditions for confluence are satisfied. For instance, for $s_0 \xrightarrow{\tau}_{\mathcal{T}} s_1$ we have to show that $s_0 \xrightarrow{\tau} s_4$ can be mimicked from s_1. Indeed, $s_1 \xrightarrow{\tau} s_0$ and $s_0 \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} s_4$.

Based on $\mathcal{T}$, we could for instance omit the transition $s_0 \xrightarrow{\tau} s_4$. However, this is clearly incorrect, as is prevents the observable a-transition from occurring on any path emanating from s_0. The problem is in the fact that while $s \xrightarrow{\tau}_{\mathcal{T}} t_1$ and $s \xrightarrow{\tau}_{\mathcal{T}} t_2$ imply $t_1 \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} t_2$ for all pairs of transitions in the system, not necessarily $s \twoheadrightarrow_{\mathcal{T}} t_1$ and $s \twoheadrightarrow_{\mathcal{T}} t_2$ imply $t_1 \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} t_2$. This problem is well-known in the context of term rewriting: the weak Church-Rosser property (local joinability, also called local confluence in term rewriting terminology) does not imply the Church-Rosser property (global joinability, also called confluence in term rewriting terminology) [46].

This problem can be solved in multiple ways. In [12], when defining weak probabilistic confluence, we explicitly required $\twoheadrightarrow \twoheadleftarrow$ to be an equivalence relation. While this correctly ensures that no behaviour is lost, it is hard to verify in practice. Therefore, we now decided to solve the problem by requiring confluent connectivity between μ and ν by having direct confluent transitions from the support of μ to the support of ν.

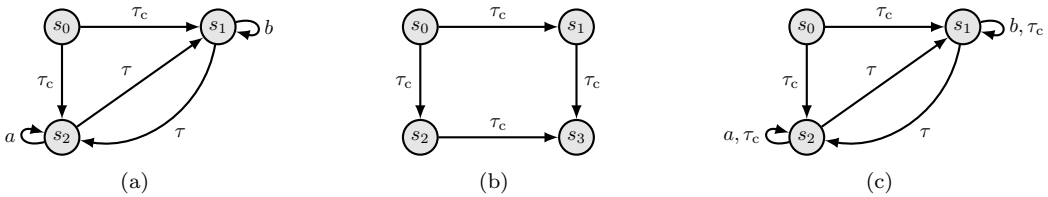


Figure 13: Counterexamples for the existence of a representation map.

8.2. Diamond-shaped mimicking

As discussed in Remark 26, the confluence classification and the corresponding requirement that transitions are mimicked by transitions from the same class together ensure that confluent transitions are mimicked by confluent transitions. This was needed for representation maps to exist. Without the confluence classification this would not be the case anymore, as can be seen from the system in Figure 13(a).

Without confluence classification, we could consider the two outgoing transitions from s_0 confluent. Indeed, the conditions for confluence are satisfied. However, confluent transitions are mimicked by non-confluent transitions, and hence no representation map can exist. After all, the confluent transitions from s_0 to s_1 and s_2 imply that $\phi(s_1) = \phi(s_2)$. However, we also require each state to be able to reach its representative while only traversing confluent transitions. These two requirements now contradict each other.

It may seem that confluent mimicking can also be ensured by requiring commutation to always happen in the shape of a diamond. This would disallow the confluent set taken before for Figure 13(a). Also, for models like the one depicted in Figure 13(b) indeed confluent mimicking holds; for $s_0 \xrightarrow{\tau} s_1$ to be confluent $s_2 \xrightarrow{\tau} s_3$ has to be, and for $s_0 \xrightarrow{\tau} s_2$ to be confluent $s_1 \xrightarrow{\tau} s_3$ has to be. However, this would still not entirely solve the problem. Consider for instance the system in Figure 13(c). Now, we can show that the two transitions from s_0 and the two self-loops together form a confluent set, that does satisfy the property of only commuting in diamonds. As before, also for this confluent set no valid representation map can be found.

To solve this problem, we need to explicitly require confluent transitions to be mimicked by confluent transitions. We chose to do so by grouping transitions by means of a confluence classification, and requiring transitions to be mimicking within their own group.

8.3. Explicit confluent mimicking

As discussed before, the confluence classification is used to ensure that confluent transitions are mimicked by confluent transitions to make sure representation maps exist. It may seem viable to just explicitly require confluent transitions to be mimicked by confluent transitions; actually, that was how previous work dealt with this problem [11]. To see what goes wrong, consider the system in Figure 12(b).

When requiring confluent transitions to be mimicked by confluent transitions (or equivalently explicitly requiring $\twoheadrightarrow \twoheadleftarrow_{\mathcal{T}}$ to be transitive) instead of using a confluent classification, the sets

$$\mathcal{T}_1 = \{(s, \tau_1, t), (t, \tau, t), (u, \tau, u), (v, \tau, v), (w, \tau, w)\}$$
$$\mathcal{T}_2 = \{(s, \tau_2, u), (t, \tau, t), (u, \tau, u), (v, \tau, v), (w, \tau, w)\}$$

would both be perfectly valid confluent sets. Still, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ does not satisfy these requirements. After all, whereas it designates $s \xrightarrow{\tau_1} t$ to be confluent, its mimicking transition $u \xrightarrow{\tau} t$ from u (needed since $s \xrightarrow{\tau_2} u$ is in $\mathcal{T}$) is not confluent. Hence, the old notions were not closed under union[10].

[10]Since [11, 12] constructed a confluent set per summand and then took the union of these sets, the fact that confluent sets are not closed under union may have broken their approach. However, as mentioned before, no problems arise in their implementation due to the fact that they also only check for mimicking transitions by the same summand. Hence, although their claim that confluent sets can just be combined to get larger confluent sets was incorrect in general, their application of this claim did not produce any problem due to the more restricted context in which it was applied.

By using a confluence classification and requiring transitions to be mimicked by the same group, we ascertain that this kind of bad compositionality behaviour does not occur. After all, for $\mathcal{T}_1$ to be a valid confluent set, the confluence classification should be such that $s \xrightarrow{\tau_2} u$ and its mimicking transition $t \xrightarrow{\tau} u$ are in the same group. So, for $s \xrightarrow{\tau_2} u$ to be confluent (as prescribed by $\mathcal{T}_2$), also $t \xrightarrow{\tau} u$ would need to be confluent. The latter is impossible, since the b-transition from u cannot be mimicked from t, and hence $\mathcal{T}_2$ is disallowed.

9. Conclusion

We introduced confluence reduction for MAs: the first efficient reduction technique for this model that abstracts from invisible transitions. We showed that it preserves divergence-sensitive branching bisimulation, and hence yields quantitatively behavioural equivalent models. In addition to working on MAs, our notion of confluence reduction has three additional advantages over previous notions of confluence. First, it preserves divergences, and hence does not affect minimal reachability probabilities. Second, it is closed under union, enabling us to separately detect confluence of different sets of transitions and combine the results. Third, it also works in the presence of state labels. We showed that a representation map approach can be used safely to reduce systems on-the-fly, and discussed how to detect confluence syntactically on the process-algebraic language MAPA. The fact that these approaches only had to be adapted slightly from previous work on confluence reduction for probabilistic automata exemplifies the robustness of our technique.

Our case studies demonstrate that significant reductions can be obtained, reducing state spaces sometimes by an order of magnitude. A decrease in the degree of nondeterminism, as is to be expected from confluence reduction, often yields even better reductions in analysis time than in state space size.

We conjecture that our technique can also be applied to different settings, e.g., the PRISM model checker [47]. As PRISM's analysis generally exploits MTBDDs, an application of confluence during model checking may be difficult. However, the detection of confluence on the PRISM modelling language can be done in quite the same way as described in this paper. This information can be used to optimise specifications syntactically, e.g., by strengthening guards of non-confluent commands so as to disable them in case at least one confluent command is enabled. Since the representation maps cannot be applied in this context, it would require acyclicity of the confluent commands to avoid the ignoring problem. Alternatively, useful future work could be to add confluence reduction to the explicit-state model checking engine of PRISM.

Several variations of confluence exist for LTSs and PAs [35, 11, 48, 12]. They are categorised from ultra weak to strong—the strength denoting their ability to distinguish systems, similar to the terminology for bisimulation. The stronger notions are easier to detect, but yield less powerful reductions. For process algebras such as MAPA it is most practical to apply strong confluence, as its detection on the algebraic level is still feasible. Therefore, this paper generalised strong confluence to the Markovian realm.

An interesting direction for future work could be to look into preservation of bisimulation on distributions [49, 50]; possibly, those kind of bisimulation relations would be preserved by a weaker notion of confluence reduction that also allows probabilistic transitions to be confluent. Other future work is to consider compositional confluence reduction [48] for MAs.

References

[1] C. Eisentraut, H. Hermanns, L. Zhang, On probabilistic automata in continuous time, in: Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science (LICS), IEEE, 2010, pp. 342–351.

[2] C. Eisentraut, H. Hermanns, L. Zhang, Concurrency and composition in a stochastic world, in: Proceedings of the 21st International Conference on Concurrency Theory (CONCUR), volume 6269 of *LNCS*, Springer, 2010, pp. 21–39.

[3] Y. Deng, M. Hennessy, On the semantics of Markov automata, Information and Computation 222 (2013) 139–168.

[4] C. Eisentraut, H. Hermanns, J.-P. Katoen, L. Zhang, A semantics for every GSPN, in: Proceedings of the 34th International Conference on Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN), volume 7927 of *LNCS*, Springer, 2013, pp. 90–109.

[5] H. Boudali, P. Crouzen, M. I. A. Stoelinga, A rigorous, compositional, and extensible framework for dynamic fault tree analysis, IEEE Transactions on Dependable and Secure Computing 7 (2010) 128–143.

[6] H. Boudali, P. Crouzen, B. R. Haverkort, M. Kuntz, M. I. A. Stoelinga, Architectural dependability evaluation with Arcade, in: Proceedings of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2008, pp. 512–521.
[7] M. Bozzano, A. Cimatti, J.-P. Katoen, V. Y. Nguyen, T. Noll, M. Roveri, Safety, dependability and performance analysis of extended AADL models, The Computer Journal 54 (2011) 754–775.
[8] J.-P. Katoen, H. Wu, Exponentially timed SADF: Compositional semantics, reduction, and analysis, in: Proceedings of the 14th International Conference on Embedded Software (EMSOFT), ACM, 2014, pp. 1–10.
[9] D. Guck, H. Hatefi, H. Hermanns, J.-P. Katoen, M. Timmer, Analysis of timed and long-run objectives for Markov automata, Logical Methods in Computer Science 10 (2014) 1–29.
[10] J. Groote, M. Sellink, Confluence for process verification, Theoretical Computer Science 170 (1996) 47–81.
[11] S. C. C. Blom, J. C. van de Pol, State space reduction by proving confluence, in: Proceedings of the 14th International Conference on Computer Aided Verification (CAV), volume 2404 of *LNCS*, Springer, 2002, pp. 596–609.
[12] M. Timmer, M. I. A. Stoelinga, J. C. van de Pol, Confluence reduction for probabilistic systems, in: Proceedings of the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), volume 6605 of *LNCS*, Springer, 2011, pp. 311–325.
[13] R. J. van Glabbeek, W. P. Weijland, Branching time and abstraction in bisimulation semantics, Journal of the ACM 43 (1996) 555–600.
[14] M. Timmer, J.-P. Katoen, J. C. van de Pol, M. I. A. Stoelinga, Efficient modelling and generation of Markov automata, in: Proceedings of the 23rd International Conference on Concurrency Theory (CONCUR), volume 7454 of *LNCS*, Springer, 2012, pp. 364–379.
[15] H. Hansen, M. Timmer, A comparison of confluence and ample sets in probabilistic and non-probabilistic branching time, Theoretical Computer Science 538 (2014) 103–123.
[16] A. Valmari, Stubborn sets for reduced state space generation, in: Proceedings of the 10th International Conference on Applications and Theory of Petri Nets (ICATPN), volume 483 of *LNCS*, Springer, 1990, pp. 491–515.
[17] A. Valmari, On-the-fly verification with stubborn sets, in: Proceedings of the 5th International Conference on Computer Aided Verification (CAV), volume 697 of *LNCS*, Springer, 1993, pp. 397–408.
[18] D. Peled, All from one, one for all: on model checking using representatives, in: Proceedings of the 5th International Conference on Computer Aided Verification (CAV), volume 697 of *LNCS*, Springer, 1993, pp. 409–423.
[19] P. Godefroid, D. Pirottin, Refining dependencies improves partial-order verification methods, in: Proceedings of the 5th International Conference on Computer Aided Verification (CAV), volume 697 of *LNCS*, Springer, 1993, pp. 438–449.
[20] P. Godefroid, Partial-order Methods for the Verification of Concurrent Systems: an Approach to the State-explosion Problem, volume 1032 of *LNCS*, Springer, 1996.
[21] P. R. D'Argenio, P. Niebert, Partial order reduction on concurrent probabilistic programs, in: Proceedings of the 1st International Conference on Quantitative Evaluation of Systems (QEST), IEEE, 2004, pp. 240–249.
[22] C. Baier, M. Größer, F. Ciesinski, Partial order reduction for probabilistic systems, in: Proceedings of the 1st International Conference on Quantitative Evaluation of Systems (QEST), IEEE, 2004, pp. 230–239.
[23] C. Baier, P. R. D'Argenio, M. Größer, Partial order reduction for probabilistic branching time, in: Proceedings of the 3rd Workshop on Quantitative Aspects of Programming Languages (QAPL), volume 153(2) of *Electronic Notes in Theoretical Computer Science*, Elsevier, 2006, pp. 97–116.
[24] A. Hartmanns, M. Timmer, On-the-fly confluence detection for statistical model checking, in: Proceedings of the 5th International NASA Formal Methods Symposium (NFM), volume 7871 of *LNCS*, Springer, 2013, pp. 337–351.
[25] M. Timmer, M. I. A. Stoelinga, J. C. van de Pol, Confluence reduction for Markov automata, in: Proceedings of the 11th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS), volume 8053 of *LNCS*, Springer, 2013, pp. 243–257.
[26] H. Hermanns, J.-P. Katoen, The how and why of interactive Markov chains, in: Proceedings of the 8th International Symposium on Formal Methods for Components and Objects (FMCO), volume 6286 of *LNCS*, Springer, 2009, pp. 311–337.
[27] J. F. Groote, F. W. Vaandrager, An efficient algorithm for branching bisimulation and stuttering equivalence, in: Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP), volume 443 of *LNCS*, Springer, 1990, pp. 626–638.
[28] H. Hatefi, H. Hermanns, Model checking algorithms for Markov automata, in: Proceedings of the 12th International Workshop on Automated Verification of Critical Systems (AVOCS), volume 53 of *Electronic Communications of the EASST*.
[29] L. Zhang, M. R. Neuhäußer, Model checking interactive Markov chains, in: Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), volume 6015 of *LNCS*, Springer, 2010, pp. 53–68.
[30] M. R. Neuhäußer, M. I. A. Stoelinga, J.-P. Katoen, Delayed nondeterminism in continuous-time Markov decision processes, in: Proceedings of the 12th International Conference on Foundations of Software Science and Computational Structures (FOSSACS), volume 5504 of *LNCS*, Springer, 2009, pp. 364–379.
[31] C. Baier, J.-P. Katoen, Principles of Model Checking, MIT Press, 2008.
[32] S. Evangelista, C. Pajault, Solving the ignoring problem for partial order reduction, International Journal on Software Tools for Technology Transfer 12 (2010) 155–170.
[33] A. Valmari, Stop it, and be stubborn!, CoRR abs/1504.02587 (2015).
[34] M. Timmer, Efficient Modelling, Generation and Analysis of Markov Automata, Ph.D. thesis, University of Twente, 2013.
[35] S. C. C. Blom, Partial τ-confluence for efficient state space generation, Technical Report SEN-R0123, Centrum voor Wiskunde en Informatica, 2001.

[36] P. R. D'Argenio, B. Jeannet, H. E. Jensen, K. G. Larsen, Reduction and refinement strategies for probabilistic analysis, in: Proceedings of the 2nd Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM-PROBMIV), volume 2399 of *LNCS*, Springer, 2002, pp. 57–76.
[37] R. E. Tarjan, Depth-first search and linear graph algorithms, SIAM Journal on Computing 1 (1972) 146–160.
[38] D. Guck, H. Hatefi, H. Hermanns, J.-P. Katoen, M. Timmer, Modelling, reduction and analysis of Markov automata, in: Proceedings of the 10th International Conference on Quantitative Evaluation of Systems (QEST), volume 8054 of *LNCS*, Springer, 2013, pp. 55–71.
[39] M. Timmer, SCOOP: A tool for symbolic optimisations of probabilistic processes, in: Proceedings of the 8th International Conference on Quantitative Evaluation of Systems (QEST), IEEE, 2011, pp. 149–150.
[40] J. C. van de Pol, M. Timmer, State space reduction of linear processes using control flow reconstruction, in: Proceedings of the 7th International Symposium on Automated Technology for Verification and Analysis (ATVA), volume 5799 of *LNCS*, Springer, 2009, pp. 54–68.
[41] A. Belinfante, A. Rensink, Publishing Your Prototype Tool on the Web: PUPTOL, a Framework, Technical Report TR-CTIT-13-15, Centre for Telematics and Information Technology, University of Twente, 2013.
[42] S. Sazonov, Property Preservation under Bisimulations on Markov Automata, Master's thesis, RWTH Aachen University, 2014.
[43] W. Fokkink, J. Pang, Simplifying Itai-Rodeh leader election for anonymous rings, in: Proceedings of the 4th International Workshop on Automated Verification of Critical Systems (AVOCS), volume 128(6) of *Electronic Notes in Theoretical Computer Science*, Elsevier, 2005, pp. 53–68.
[44] H. Garavel, F. Lang, R. Mateescu, W. Serwe, CADP 2011: a toolbox for the construction and analysis of distributed processes, International Journal on Software Tools for Technology Transfer 15 (2013) 89–107.
[45] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, Modelling with Generalized Stochastic Petri Nets, John Wiley & Sons, 1994.
[46] F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.
[47] M. Z. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: Verification of probabilistic real-time systems, in: Proceedings of the 23rd International Conference on Computer Aided Verification (CAV), volume 6806 of *LNCS*, Springer, 2011, pp. 585–591.
[48] G. J. Pace, F. Lang, R. Mateescu, Calculating τ-confluence compositionally, in: Proceedings of the 15th International Conference on Computer Aided Verification (CAV), volume 2725 of *LNCS*, Springer, 2003, pp. 446–459.
[49] L. Doyen, T. A. Henzinger, J.-F. Raskin, Equivalence of labeled Markov chains, International Journal of Foundations of Computer Science 19 (2008) 549–563.
[50] H. Hermanns, J. Krcál, J. Kretínský, Probabilistic bisimulation: Naturally on distributions, in: Proceedings of the 25th International Conference on Concurrency Theory (CONCUR), volume 8704 of *LNCS*, Springer, 2014, pp. 249–265.
[51] M. I. A. Stoelinga, Alea jacta est: Verification of Probabilistic, Real-time and Parametric Systems, Ph.D. thesis, University of Nijmegen, 2002.

Appendix A. Proofs

Whenever a confluent set $\mathcal{T}$ is given, we abuse notation by writing *confluent transition* to denote a transition in this set $\mathcal{T}$. Note that, in general, there may also be confluent transitions that are not in $\mathcal{T}$.

Proposition 28. *Let P be a confluence classification for $\mathcal{M}$, and let $\mathcal{T}$ be a confluent set for P. Then,*

$$s \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} t \qquad \textit{if and only if} \qquad s \twoheadleftarrow\!\!\!\twoheadrightarrow_{\mathcal{T}} t$$

Proof. We separately prove both directions of the equivalence.

($\Longrightarrow$) Let $s \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} t$. Then, by definition there is a state u such that $s \twoheadrightarrow_{\mathcal{T}} u$ and $t \twoheadrightarrow_{\mathcal{T}} u$. This immediately implies that $s \twoheadleftarrow\!\!\!\twoheadrightarrow_{\mathcal{T}} t$.

($\Longleftarrow$) Let $s \twoheadleftarrow\!\!\!\twoheadrightarrow_{\mathcal{T}} t$. This means that there is a path from s to t such as

$$s_0 \leftarrow s_1 \rightarrow s_2 \rightarrow s_3 \leftarrow s_4 \leftarrow s_5 \rightarrow s_6,$$

where $s_0 = s$, $s_6 = t$ and each of the transitions is in $\mathcal{T}$. Note that $s_i \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} s_{i+1}$ for all s_i, s_{i+1} on this path. After all, if $s_i \rightarrow s_{i+i}$ then they can join at s_{i+1}, otherwise they can join at s_i. Hence, to show that $s \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} t$, it suffices to show that $\twoheadrightarrow \twoheadleftarrow_{\mathcal{T}}$ is transitive.

Let $s' \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} s$ and $s \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} s''$. We show that $s' \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} s''$. Let t' be a state such that $s \twoheadrightarrow_{\mathcal{T}} t'$ and $s' \twoheadrightarrow_{\mathcal{T}} t'$, and likewise, let t'' be a similar state for s and s''. If we can show that there is some state t such that $t' \twoheadrightarrow_{\mathcal{T}} t$ and $t'' \twoheadrightarrow_{\mathcal{T}} t$, we have the result. Let a minimal confluent path from s to t' be given by

$s_0 \to_{\mathcal{T}} s_1 \to_{\mathcal{T}} \cdots \to_{\mathcal{T}} s_n$, with $s_0 = s$ and $s_n = t'$. By induction on the length of this path, we show that for each state s_i on it, there is some state t such that $s_i \twoheadrightarrow_{\mathcal{T}} t$ and $t'' \twoheadrightarrow_{\mathcal{T}} t$. Since t' is also on the path, this completes the argument.

Base case. There clearly is a state t such that $s_0 \twoheadrightarrow_{\mathcal{T}} t$ and $t'' \twoheadrightarrow_{\mathcal{T}} t$, namely t'' itself. After all, $s_0 = s$ and $s \twoheadrightarrow_{\mathcal{T}} t''$, and $\twoheadrightarrow_{\mathcal{T}}$ is reflexive.

Inductive case. Let there be a state t_k such that $s_k \twoheadrightarrow_{\mathcal{T}} t_k$ and $t'' \twoheadrightarrow_{\mathcal{T}} t_k$. We show that there exists a state t_{k+1} such that $s_{k+1} \twoheadrightarrow_{\mathcal{T}} t_{k+1}$ and $t'' \twoheadrightarrow_{\mathcal{T}} t_{k+1}$. Let $s_k \xrightarrow{\tau} u$ be the first transition on the $\mathcal{T}$-path from s_k to t_k. Let $s_k \xrightarrow{\tau} s_{k+1}$ be the $\mathcal{T}$-transition between s_k and s_{k+1}. Since it is in $\mathcal{T}$, there must be at least one group $C \in P \cap \mathcal{T}$ such that $s_k \xrightarrow{\tau}_C s_{k+1}$. By definition of confluence, since $(s_k \xrightarrow{\tau} u) \in \mathcal{T}$ and $s_k \xrightarrow{\tau}_C s_{k+1}$ for some $C \in P$, either (1) $s_{k+1} = u$ (the transitions coincide), or (2) there is a transition $u \xrightarrow{\tau}_C u'$ such that $\mathbb{1}_{s_{k+1}} \equiv_R \mathbb{1}_{u'}$, with R the equivalence relation given in Definition 22.

In case (1), we directly find $s_{k+1} \twoheadrightarrow_{\mathcal{T}} t_k$. Hence, we can just take $t_{k+1} = t_k$. In case (2), either $s_{k+1} = u'$ or $s_{k+1} \xrightarrow{\tau}_{\mathcal{T}} u'$. In both cases, if $u = t_k$, we can take $t_{k+1} = u'$ and indeed $s_{k+1} \twoheadrightarrow_{\mathcal{T}} t_{k+1}$ and $t'' \twoheadrightarrow_{\mathcal{T}} t_{k+1}$. Otherwise, we can use the same reasoning to show that there is a state t_{k+1} such that $u' \twoheadrightarrow_{\mathcal{T}} t_{k+1}$ and $t'' \twoheadrightarrow_{\mathcal{T}} t_{k+1}$, based on $u \twoheadrightarrow_{\mathcal{T}} t_k$, $t'' \twoheadrightarrow_{\mathcal{T}} t_k$ and $u \xrightarrow{\tau}_{\mathcal{T}} u'$. Since the path from u to t_k is one transition shorter than the path from s_k to t_k, this argument terminates. □

Theorem 29. *Let P be a confluence classification for $\mathcal{M}$, and let $\mathcal{T}_1, \mathcal{T}_2$ be two Markovian confluent sets for P. Then, $\mathcal{T}_1 \cup \mathcal{T}_2$ is also a Markovian confluent set for P.*

Proof. Let $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$. Clearly, $\mathcal{T}$ still only contain invisible transitions with Dirac distributions, since $\mathcal{T}_1$ and $\mathcal{T}_2$ do. Consider a transition $(s \xrightarrow{\tau}_{\mathcal{T}} t)$, and another transition $s \xrightarrow{a} \mu$. We need to show that

$$\begin{array}{llll} (i) & (s \xrightarrow{a} \mu) \in P & \textit{implies} & \forall C \in P \,.\, s \xrightarrow{a}_C \mu \implies \exists \nu \in \mathrm{Distr}(S) \,.\, t \xrightarrow{a}_C \nu \wedge \mu \equiv_R \nu \\ (ii) & (s \xrightarrow{a} \mu) \notin P & \textit{implies} & \exists \nu \in \mathrm{Distr}(S) \,.\, t \xrightarrow{a} \nu \quad \wedge \mu \equiv_R \nu \end{array}$$

where R is the smallest equivalence relation such that $R \supseteq \{(s,t) \in \mathit{supp}(\mu) \times \mathit{supp}(\nu) \mid (s \xrightarrow{\tau} t) \in \mathcal{T}\}$. Without loss of generality, assume that $s \xrightarrow{\tau}_{\mathcal{T}_1} t$. Hence, by definition of Markovian confluence, we find that

$$\begin{array}{llll} (i) & (s \xrightarrow{a} \mu) \in P & \textit{implies} & \forall C \in P \,.\, s \xrightarrow{a}_C \mu \implies \exists \nu \in \mathrm{Distr}(S) \,.\, t \xrightarrow{a}_C \nu \wedge \mu \equiv_{R_1} \nu \\ (ii) & (s \xrightarrow{a} \mu) \notin P & \textit{implies} & \exists \nu \in \mathrm{Distr}(S) \,.\, t \xrightarrow{a} \nu \quad \wedge \mu \equiv_{R_1} \nu \end{array}$$

where R_1 is the smallest equivalence relation such that $R_1 \supseteq \{(s,t) \in \mathit{supp}(\mu) \times \mathit{supp}(\nu) \mid (s \xrightarrow{\tau} t) \in \mathcal{T}_1\}$. Note that $R \supseteq R_1$ since $\mathcal{T} \supseteq \mathcal{T}_1$. Therefore, $\mu \equiv_{R_1} \nu$ implies $\mu \equiv_R \nu$ (using Proposition 5.2.1.5 from [51]). The result now immediately follows. □

The next lemma, relating the behaviour of a state to the behaviour of states that it can reach via a confluent path, will be of use in the proof of Theorem 32.

Lemma 49. *Let $s, s' \in S$ be two states, $a \in A$, and $\mu \in \mathrm{Distr}(S)$. Moreover, let $P \subseteq \mathscr{P}(\hookrightarrow)$ be a confluence classification for $\mathcal{M}$ and let $\mathcal{T}$ be a Markovian confluent set for P. Then,*

$$s \twoheadrightarrow_{\mathcal{T}} s' \wedge s \xrightarrow{a} \mu \implies (a = \tau \wedge \mu \equiv_R \mathbb{1}_{s'}) \vee \big(\exists \nu \in \mathrm{Distr}(S) \,.\, s' \xrightarrow{a} \nu \wedge \mu \equiv_R \nu\big)$$

where $R = \{(u,v) \mid u \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} v\}$.

Proof. First of all, we note that R is indeed an equivalence relation, as shown in Proposition 28.

Let $s, s' \in S$ be such that $s \twoheadrightarrow_{\mathcal{T}} s'$, and assume a transition $s \xrightarrow{a} \mu$. Let $R = \{(u,v) \mid u \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} v\}$. We show that either $a = \tau \wedge \mu \equiv_R \mathbb{1}_{s'}$ or that there exists a transition $s' \xrightarrow{a} \nu$ such that $\mu \equiv_R \nu$, by induction on the length of the confluent path from s to s'. Let $s_0 \xrightarrow{\tau}_{\mathcal{T}} s_1 \xrightarrow{\tau}_{\mathcal{T}} \ldots \xrightarrow{\tau}_{\mathcal{T}} s_{n-1} \xrightarrow{\tau}_{\mathcal{T}} s_n$, with $s_0 = s$ and $s_n = s'$, denote this path. Then, we show that

$$(a = \tau \wedge \mu \equiv_R \mathbb{1}_{s'}) \vee \big(\exists \nu \in \mathrm{Distr}(S) \,.\, s_i \xrightarrow{a} \nu \wedge \mu \equiv_R \nu\big)$$

holds for every state s_i on this path. For the base case s this is immediate, since $s \xrightarrow{a} \mu$ and $\equiv_R$ is reflexive.

As induction hypothesis, assume that the formula holds for some state s_i ($0 \leq i < n$). We show that it still holds for state s_{i+1}. If the above formula was true for s_i due to the clause $a = \tau \wedge \mu \equiv_R \mathbb{1}_{s'}$, then this still holds for s_{i+1}. So, assume that $s_i \xrightarrow{a} \nu$ such that $\mu \equiv_R \nu$.

Since $s_i \xrightarrow{\tau}_{\mathcal{T}} s_{i+1}$ and $s_i \xrightarrow{a} \nu$, by definition of confluence either (1) $a = \tau$ and $\nu = \mathbb{1}_{s_{i+1}}$, or (2) there is a transition $s_{i+1} \xrightarrow{a} \nu'$ such that $\nu \equiv_{R'} \nu'$, where R' is the smallest equivalence relation such that

$$R' \supseteq \{(s,t) \in \mathit{supp}(\nu) \times \mathit{supp}(\nu') \mid (s \xrightarrow{\tau} t) \in \mathcal{T}\}$$

(1) In the first case, $\nu = \mathbb{1}_{s_{i+1}}$ implies that $\nu \equiv_R \mathbb{1}_{s'}$ as there is a $\mathcal{T}$-path from s_{i+1} to s' and hence $(s_{i+1}, s') \in R$. Since we assumed that $\mu \equiv_R \nu$, and the relation $\equiv_R$ is transitive, this yields $\mu \equiv_R \mathbb{1}_{s'}$. Together with $a = \tau$, this completes the proof.

(2) In the second case, note that $R \supseteq R'$. After all, $R = \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} = \twoheadleftarrow\!\!\twoheadrightarrow_{\mathcal{T}}$ (by Proposition 28), and obviously $(s,t) \in R'$ implies that $s \twoheadleftarrow\!\!\twoheadrightarrow_{\mathcal{T}} t$. Hence, $\nu \equiv_{R'} \nu'$ implies $\nu \equiv_R \nu'$ (using Proposition 5.2.1.5 from [51]). Since we assumed that $\mu \equiv_R \nu$, by transitivity of $\equiv_R$ we obtain $\mu \equiv_R \nu'$. Hence, there is a transition $s_{i+1} \xrightarrow{a} \nu'$ such that $\mu \equiv_R \nu'$, which completes the proof. □

Theorem 32. *Let $s, s' \in S$ be two states and $\mathcal{T}$ a confluent set for some confluence classification P. Then,*

$$s \twoheadleftarrow\!\!\twoheadrightarrow_{\mathcal{T}} s' \text{ implies } s \approx_{\mathrm{b}}^{\mathrm{div}} s'$$

Proof. We show that $s \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} s'$ implies $s \approx_{\mathrm{b}}^{\mathrm{div}} s'$. By Proposition 28, this is equivalent to the theorem. So, assume that $s \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} s'$. To show that $s \approx_{\mathrm{b}}^{\mathrm{div}} s'$, consider the relation

$$R = \{(u,v) \mid u \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} v\}$$

Clearly $(s, s') \in R$, and from Proposition 28 and the obvious fact that $\twoheadleftarrow\!\!\twoheadrightarrow_{\mathcal{T}}$ is an equivalence relation, it follows that R is an equivalence relation as well.

It remains to show that R is a divergence-sensitive branching bisimulation. First, $(p,q) \in R$ indeed implies $L(p) = L(q)$ by the requirements that confluent transitions are invisible (Definition 25). Second, let $(p,q) \in R$, i.e., $p \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} q$. We need to show that for every extended transition $p \xrightarrow{a} \mu$ there is a transition $q \xRightarrow{a}_R \mu'$ such that $\mu \equiv_R \mu'$.

So, assume such a transition $p \xrightarrow{a} \mu$. Let r be a state such that $p \twoheadrightarrow_{\mathcal{T}} r$ and $q \twoheadrightarrow_{\mathcal{T}} r$. By Lemma 49, either (1) $a = \tau \wedge \mu \equiv_R \mathbb{1}_r$ or (2) there is a distribution $\nu \in \mathrm{Distr}(S)$ such that $r \xrightarrow{a} \nu \wedge \mu \equiv_R \nu$.

(1) In the first case, note that $q \twoheadrightarrow_{\mathcal{T}} r$ immediately implies that $q \xRightarrow{\tau}_R \mathbb{1}_r$. After all, we can schedule the (invisible) confluent transitions from q to r and then terminate. Indeed, all intermediate states are clearly related by R. Together with the assumption that $\mu \equiv_R \mathbb{1}_r$, this completes the argument.

(2) In the second case, note that $q \twoheadrightarrow_{\mathcal{T}} r$ and $r \xrightarrow{a} \nu$ together immediately imply that $q \xRightarrow{a}_R \nu$. After all, we can schedule the (invisible) confluent transitions from q to r, perform the transition $r \xrightarrow{a} \nu$ and then terminate. Indeed, all intermediate states before the a-transition are clearly related by R. Together with the assumption that $\mu \equiv_R \nu$, this completes the argument.

It remains to show that R is divergence sensitive. So, let $(s, s') \in R$ (and hence $s \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} s'$) and assume that there is a scheduler $\mathcal{S}$ such that $\forall \pi \in \mathit{finpaths}^{\mathcal{S}}_{\mathcal{M}}(s)\ .\ \mathit{invis}(\pi) \wedge \mathcal{S}(\pi)(\bot) = 0$. It is well known that we can assume that such diverging schedulers are memoryless and deterministic.

We show that there also is a diverging scheduler from s'. First, note that since $s \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} s'$, there is a state t such that $s \twoheadrightarrow_{\mathcal{T}} t$ and $s' \twoheadrightarrow_{\mathcal{T}} t$. We show that there is a diverging scheduler from t; then, the result follows as from s' we can schedule to first follow the confluent (and hence invisible) transitions to t and then continue with the diverging scheduler from t.

Let $s_0 \xrightarrow{\tau}_{\mathcal{T}} s_1 \xrightarrow{\tau}_{\mathcal{T}} s_2 \xrightarrow{\tau}_{\mathcal{T}} \ldots \xrightarrow{\tau}_{\mathcal{T}} s_n$ be the confluent path from s to t; hence, $s_0 = s$ and $s_n = t$. It might be the case that some states on this path also occur on the tree associated with $\mathcal{S}$; hence, for those states a diverging scheduler already exists. Let s_i be the last state on the path from s_0 to s_n that occurs on

the tree of $\mathcal{S}$. We show that s_n also has a diverging scheduler by induction on the length of the path from s_i to s_n; note that the base case is immediate.

Assume that s_j (with $i \leq j < n$) has a diverging scheduler $\mathcal{S}'$. We show that s_{j+1} has one too. If s_{j+1} occurs on the tree associated with $\mathcal{S}'$ this is immediate, so from now on assume that it does not. From s_j there now is a confluent transition $s_j \xrightarrow{\tau}_{\mathcal{T}} s_{j+1}$ and an invisible (not necessarily confluent) transition $s_j \xrightarrow{\tau} \mu$ (chosen by $\mathcal{S}'$ as first step of the diverging path form s_j). By definition of confluence, either these transitions coincide or there is a transition $s_{j+1} \xrightarrow{\tau} \nu$ such that $\mu \equiv_{R'} \nu$, with R' the smallest equivalence relation such that $R \supseteq \{(s,t) \in \mathit{supp}(\mu) \times \mathit{supp}(\nu) \mid (s \xrightarrow{\tau} t) \in \mathcal{T}\}$. The first option is impossible, since we assumed that s_{j+1} is not on the tree associated with $\mathcal{S}'$. Therefore, there is a transition $s_{j+1} \xrightarrow{\tau} \nu$ such that $\mu \equiv_{R'} \nu$.

Since $s_j \xrightarrow{\tau} \mu$ is invisible, $L(u) = L(s_j)$ for all states $u \in \mathit{supp}(\mu)$. Additionally, by definition of R', $\mu \equiv_{R'} \nu$ implies that each state $v \in \mathit{supp}(\nu)$ is either (1) in $\mathit{supp}(\mu)$ as well or (2) has an incoming confluent transition $u \xrightarrow{\tau}_{\mathcal{T}} v$ with $u \in \mathit{supp}(\mu)$. Hence, since confluent transitions are invisible, also $L(v) = L(s_j)$ for all states $v \in \mathit{supp}(\nu)$. Finally, since $s_j \xrightarrow{\tau} s_{j+1}$ is confluent, also $L(s_j) = L(s_{j+1})$. Together, these facts imply that $L(v) = L(s_{j+1})$ for all states $v \in \mathit{supp}(\nu)$, and hence that $s_{j+1} \xrightarrow{\tau} \nu$ is invisible. We schedule this transition from s_{j+1} in order to diverge. So, we still need to show that it is possible to diverge from all states $q \in \mathit{supp}(\nu)$.

By definition of R', $\mu \equiv_{R'} \nu$ implies that each state $v \in \mathit{supp}(\nu)$ is either (1) in $\mathit{supp}(\mu)$ as well or (2) has an incoming confluent transition $u \xrightarrow{\tau}_{\mathcal{T}} v$ with $u \in \mathit{supp}(\mu)$. In the first case, we can diverge from v using $\mathcal{S}'$. In the second case, we have reached the situation of a state v with an incoming confluent transition from a state u that has a diverging scheduler. Now, the above reasoning can be applied again, taking $s_j = u$ and $s_{j+1} = v$. Either at some point overlap with the scheduler of u occurs, or this argument is repeated indefinitely; in both cases, divergence is obtained. □

Theorem 37. *For every representation map $\phi\colon S \to S$ for $\mathcal{M}$ under some confluent set $\mathcal{T}$, $\mathcal{M}/\phi \approx_{\mathrm{b}}^{\mathrm{div}} \mathcal{M}$.*

Proof. We denote the extended transition relation of $\mathcal{M}$ by $\to$, and the one of $\mathcal{M}/\phi$ by $\to_\phi$. We take the disjoint union $\mathcal{M}'$ of $\mathcal{M}$ and $\mathcal{M}/\phi$, to provide a bisimulation relation over this state space that contains their initial states. We denote the transition relation of $\mathcal{M}'$ by $\to'$. Note that, based on whether $s \in \mathcal{M}$ or $s \in \mathcal{M}/\phi$, a transition $s \xrightarrow{a}' \mu$ corresponds to either $s \xrightarrow{a} \mu$ or $s \xrightarrow{a}_\phi \mu$.

To distinguish between for instance a state $\phi(s) \in \mathcal{M}$ and the corresponding state $\phi(s) \in \mathcal{M}/\phi$, we denote all states $s, \phi(s)$ from $\mathcal{M}$ just by $s, \phi(s)$, and all states $s, \phi(s)$ from $\mathcal{M}/\phi$ by $\hat{s}, \hat{\phi}(s)$.

Let R be the smallest equivalence relation containing the set $\{(s, \hat{\phi}(s)) \mid s \in S\}$, i.e., R relates all states from $\mathcal{M}$ that have the same representative to each other and to this mutual representative from $\mathcal{M}/\phi$. Clearly, $(s^0, \hat{\phi}(s^0)) \in R$. Also, $(p,q) \in R$ implies $L(p) = L(q)$ by definition of the representation map (prescribing all states that have the same representative to be connected by confluent transitions), the fact that confluent transitions are invisible and the construction of the quotient (leaving the state labelling invariant).

Note that given this equivalence relation R, for every probability distribution μ we have $\mu \equiv_R \mu_\phi$ (no matter whether μ_ϕ is in $\mathcal{M}$ or in $\mathcal{M}/\phi$). After all, the lifting over ϕ just changes the states in the support of μ to their representatives; as R relates precisely such states, clearly $\mu \equiv_R \mu_\phi$. This observation is used several times in the proof below.

Now, let $(s, s') \in R$ and assume that there is an extended transition $s \xrightarrow{a}' \mu$. We show that also $s' \xRightarrow{a}'_R \mu'$ such that $\mu \equiv_R \mu'$. Note that there are four possible cases to consider with respect to the origin of s and s', indicated by the presence or absence of hats:

- Case 1: $(\hat{s}, \hat{s}')$. Since every equivalence class of R contains precisely one representative from $\mathcal{M}/\phi$, we find that $\hat{s} = \hat{s}'$. Hence, the result follows directly by the scheduler that takes the transition $s \xrightarrow{a}' \mu$ and then terminates.
- Case 2: (s, s'). If both states are in $\mathcal{M}$, then the quotient is not involved and $\phi(s) = \phi(s')$. By definition of the representation map, we find $s \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} s'$. Using Theorem 32, this immediately implies that $s' \xRightarrow{a}_{R'} \mu'$ such that $\mu \equiv_{R'} \mu'$ for $R' = \{(u,v) \mid u \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} v\}$. Since all states connected by $\mathcal{T}$-transitions are required to have the same representative, we have $R \supseteq R'$. Hence, also $s' \xRightarrow{a}_R \mu'$,

as this is then less restrictive. Moreover, $\mu \equiv_R \mu'$ by Proposition 5.2.1.5 from [51]. Finally, note that $s' \stackrel{a}{\Longrightarrow}_R \mu'$ implies $s' \stackrel{a}{\Longrightarrow}'_R \mu'$.

- Case 3: $(\hat{s}, s')$. Since $\hat{s}$ is in $\mathcal{M}/\phi$ and s' is not, by definition of R we find that $\hat{s} = \hat{\phi}(s')$. Hence, by assumption $\hat{\phi}(s') \stackrel{a}{\to}_\phi \mu$, and thus by definition of the extended arrow either (1) $a \in A$ and $\hat{\phi}(s') \stackrel{a}{\hookrightarrow}_\phi \mu$, or (2) $a = \chi(\lambda)$ for $\lambda = \mathit{rate}(\hat{\phi}(s'))$, $\lambda > 0$, $\mu = \mathbb{P}_{\hat{\phi}(s')}$ and there is no μ' such that $\hat{\phi}(s') \stackrel{\tau}{\hookrightarrow}_\phi \mu'$. We make a case distinction based on this.

  (1) Let $a \in A$ and $\hat{\phi}(s') \stackrel{a}{\hookrightarrow}_\phi \mu$. By definition of the quotient, this implies that there is a transition $\phi(s') \stackrel{a}{\hookrightarrow} \mu'$ in $\mathcal{M}$ such that $\mu = \mu'_\phi$. By definition of the representation map, there is a $\mathcal{T}$-path (which is invisible and deterministic) from s' to $\phi(s')$ in $\mathcal{M}$. Hence, $s' \stackrel{a}{\Longrightarrow}_R \mu'$ (and therefore also $s' \stackrel{a}{\Longrightarrow}'_R \mu'$) by the scheduler from s' that first goes to $\phi(s')$ and then executes the $\phi(s') \stackrel{a}{\hookrightarrow} \mu'$ transition. Note that the transition is indeed branching, as all steps in between have the same representative and thus are related by R.

  It remains to show that $\mu \equiv_R \mu'$. We already saw that $\mu = \mu'_\phi$; hence, the result follows from the observation that $\mu \equiv_R \mu_\phi$ for every μ.

  (2) Let $a = \chi(\lambda)$ for $\lambda = \mathit{rate}(\hat{\phi}(s'))$, $\lambda > 0$, $\mu = \mathbb{P}_{\hat{\phi}(s')}$ and there is no μ' such that $\hat{\phi}(s') \stackrel{\tau}{\hookrightarrow}_\phi \mu'$. Note that this means that from $\hat{\phi}(s')$ there is a total exit rate of λ, spreading out according to μ. Hence, given an arbitrary state $\hat{u}$ in $\mathcal{M}/\phi$, we have $\mu(\hat{u}) = \frac{\mathit{rate}(\hat{\phi}(s'), \hat{u})}{\lambda}$.

  By definition of the quotient there is at most one Markovian transition between any pair of states in $\mathcal{M}/\phi$, so for every $\hat{u} \in \mathit{supp}(\mu)$, there is precisely one Markovian transition $\hat{\phi}(s') \stackrel{\lambda'}{\rightsquigarrow}_\phi \hat{u}$ with $\lambda' = \mu(\hat{u}) \cdot \lambda$. By definition of the quotient we then also find that λ' is the sum of all outgoing Markovian transitions in $\mathcal{M}$ from $\phi(s')$ to states t such that $\phi(t) = u$. Since each state in $\mathcal{M}$ has precisely one representative and $\hat{\phi}(s')$ has a Markovian transition to all representatives of states reached from $\phi(s')$ by Markovian transitions, it follows that the total exit rate of $\phi(s')$ is also λ.

  Additionally, there is no outgoing τ-transition from $\phi(s')$, since by definition of the quotient this would have resulted in a τ-transition from $\hat{\phi}(s')$, which we assumed is not present. Hence, there is an extended transition $\phi(s') \stackrel{\chi(\lambda)}{\longrightarrow} \mu'$ in $\mathcal{M}$. As the total exit rates of $\phi(s')$ and $\hat{\phi}(s')$ are equal, and the sum of all outgoing Markovian transitions from $\phi(s')$ to states t such that $\phi(t) = u$ equals the rate from $\hat{\phi}(s')$ to $\hat{u}$, we find that $\mu \equiv_R \mu'$ since R equates states to their representative and to other states with the same representative.

  By definition of the representation map, there is a $\mathcal{T}$-path (which is invisible and deterministic) from s' to $\phi(s')$ in $\mathcal{M}$. Hence, $\phi(s') \stackrel{\chi(\lambda)}{\longrightarrow} \mu'$ implies that $s' \stackrel{\chi(\lambda)}{\Longrightarrow}_R \mu'$ and therefore also $s' \stackrel{\chi(\lambda)}{\Longrightarrow}'_R \mu'$. As $\chi(\lambda) = a$ and we already saw that $\mu \equiv_R \mu'$, this completes this part of the proof.

- Case 4: $(s, \hat{s}')$. Since $\hat{s}'$ is in $\mathcal{M}/\phi$ and s is not, by definition of R we find that $\hat{s}' = \hat{\phi}(s)$. By definition of the representation map, there is a $\mathcal{T}$-path from s to $\phi(s)$ in $\mathcal{M}$. Hence, since $s \stackrel{a}{\to} \mu$, by Lemma 49 we have either (1) $a = \tau \wedge \mu \equiv_{R'} \mathbb{1}_{\phi(s)}$, or (2) there exists a transition $\phi(s) \stackrel{a}{\to} \nu$ such that $\mu \equiv_{R'} \nu$, for $R' = \{(u, v) \mid u \twoheadrightarrow \twoheadleftarrow_{\mathcal{T}} v\}$. Again, as in case 2 we can safely substitute R' by R.

  (1) We need to show that $\hat{\phi}(s) \stackrel{\tau}{\Longrightarrow}'_R \mu'$ such that $\mathbb{1}_{\phi(s)} \equiv_R \mu'$. By definition of branching steps, we trivially have $\hat{\phi}(s) \stackrel{\tau}{\Longrightarrow}'_R \mathbb{1}_{\hat{\phi}(s)}$. Note that indeed $\mathbb{1}_{\phi(s)} \equiv_R \mathbb{1}_{\hat{\phi}(s)}$, since $(\phi(s), \hat{\phi}(s)) \in R$.

  (2) If $\phi(s) \stackrel{a}{\to} \nu$, by definition of the extended arrow either (2.a) $a \in A$ and $\phi(s) \stackrel{a}{\hookrightarrow} \mu$, or (2.b) $a = \chi(\lambda)$ for $\lambda = \mathit{rate}(\phi(s))$, $\lambda > 0$, $\mu = \mathbb{P}_{\phi(s)}$ and there is no μ' such that $\phi(s) \stackrel{\tau}{\hookrightarrow} \mu'$.

  In case of (2.a), by definition of the quotient we find that $\hat{\phi}(s) \stackrel{a}{\hookrightarrow}_\phi \nu_\phi$. Hence, also $\hat{\phi}(s) \stackrel{a}{\Longrightarrow}'_R \nu_\phi$. As observed above, $\nu_\phi \equiv_R \nu$. Also, since $\mu \equiv_R \nu$ by assumption, transitivity of $\equiv_R$ yields $\mu \equiv_R \nu_\phi$.

  In case of (2.b), $\phi(s)$ has a total exit rate of λ and this is spread out according to μ. That is, for each state t, there is a rate of $\lambda \cdot \mu(t)$ from $\phi(s)$ to t. Let $C = [t]_R$ for some state t, and let λ' be the total rate from $\phi(s)$ to C. By definition of the quotient, this implies that there is a rate of λ' from $\hat{\phi}(s)$ to $\hat{\phi}(t)$ in $\mathcal{M}/\phi$ as well. Since the only element of C reachable from $\hat{\phi}(s)$ is $\hat{\phi}(t)$, this implies

that there is a rate from $\hat{\phi}(s)$ to C of λ'. Hence, for an arbitrary equivalence class C we find identical rates from $\phi(s)$ to C and from $\hat{\phi}(s)$ to C. This immediately implies that the exit rates of $\phi(s)$ and $\hat{\phi}(s)$ coincide, and that $\mathbb{P}_{\phi(s)} \equiv_R \mathbb{P}_{\hat{\phi}(s)}$. By definition of extended transitions now $\hat{\phi}(s) \xrightarrow{\chi(\lambda)}_{\phi} \mathbb{P}_{\hat{\phi}(s)}$, and hence $\hat{\phi}(s) \stackrel{a}{\Longrightarrow}'_R \mathbb{P}_{\hat{\phi}(s)}$. Since $\mu = \mathbb{P}_{\phi(s)}$ and $\mathbb{P}_{\phi(s)} \equiv_R \mathbb{P}_{\hat{\phi}(s)}$, this completes this part of the proof.

It remains to show that R is divergence sensitive. So, let $(s, s') \in R$. Again, we make a case distinction based on the origin of s and s'. Like before, if both states are in $\mathcal{M}/\phi$ then they coincide, and hence the result immediately follows. Also, if both states are in $\mathcal{M}$, then divergence of s' is implied by divergence of s. After all, having the same representative they must be connected by confluent transitions, and hence Theorem 32 and the fact that the quotient is not involved give the result.

So, we only need to show (1) whether divergence in a state s in $\mathcal{M}$ implies divergence in its representative $\hat{\phi}(s)$ in $\mathcal{M}/\phi$, and (2) whether divergence in a state $\hat{t} \in \mathcal{M}/\phi$ implies divergence in all states s in $\mathcal{M}$ such that $\phi(s) = t$.

(1) Assume that there is a diverging scheduler for some state s in $\mathcal{M}$. We need to show that there also is a diverging scheduler for $\hat{\phi}(s)$ in $\mathcal{M}/\phi$. First of all note that, by Theorem 32, divergence of s implies divergence of $\phi(s)$ in $\mathcal{M}$. Hence, we can assume that there is a scheduler $\mathcal{S}$ such that

$$\forall \pi \in \mathit{finpaths}^{\mathcal{S}}_{\mathcal{M}}(\phi(s)) \;.\; \mathit{invis}(\pi) \wedge \mathcal{S}(\pi)(\bot) = 0$$

It is well known that we can assume that this diverging scheduler is memoryless and deterministic.

By the existence of $\mathcal{S}$, there must be some transition $\phi(s) \xhookrightarrow{\tau} \mu$ such that every state $t \in \mathit{supp}(\mu)$ is also diverging and has the same labelling as $\phi(s)$. By the definition of the quotient, then there also is a transition $\hat{\phi}(s) \xhookrightarrow{\tau}_{\phi} \mu_{\phi}$ in $\mathcal{M}/\phi$. Since μ_{ϕ} is obtained from μ by taking the representatives of all states in its support, $\hat{\phi}(s) \xhookrightarrow{\tau}_{\phi} \mu_{\phi}$ is also invisible by definition of the representation map, the fact that confluent transitions are invisible and the fact that the quotient leaves the state labelling invariant.

We can now construct a diverging scheduler for $\hat{\phi}(s)$ that starts with this transition. Then, it invisibly ends up in either one of a set of states that are all representatives of diverging states. From all those states, the above argument can be repeated to take the next invisible transition. As this process can be extended indefinitely, indeed $\hat{\phi}(s)$ is diverging too.

(2) Assume that there is a scheduler $\mathcal{S}$ such that

$$\forall \pi \in \mathit{finpaths}^{\mathcal{S}}_{\mathcal{M}/\phi}(\hat{s}) \;.\; \mathit{invis}(\pi) \wedge \mathcal{S}(\pi)(\bot) = 0$$

for some state $\hat{s}$ in $\mathcal{M}/\phi$. It is well known that we can assume that this diverging scheduler is memoryless and deterministic.

We need to show that there also is a diverging scheduler for every state s' in $\mathcal{M}$ such that $\phi(s') = s$. First of all note that, by Theorem 32, divergence of $\phi(s')$ implies divergence of s' in $\mathcal{M}$. Hence, it suffices to show divergence of $\phi(s')$ based on divergence of $\hat{\phi}(s')$ $(= \hat{s})$.

By the existence of $\mathcal{S}$, there must be an invisible transition $\hat{\phi}(s') \xhookrightarrow{\tau}_{\phi} \mu$ such that every state $\hat{t} \in \mathit{supp}(\mu)$ is also diverging. By the definition of the quotient, then there also is a transition $\phi(s') \xhookrightarrow{\tau} \nu$ in $\mathcal{M}$ such that $\nu_{\phi} = \mu$. Again, it can easily be seen that $\phi(s') \xhookrightarrow{\tau} \nu$ is also invisible. Hence, we can now construct a diverging scheduler for $\phi(s')$ that starts with this transition. Then, it invisibly ends up in either one of a set of states that all have a diverging representative. From all those states, the above argument can be repeated to take the next invisible transition. As this process can be extended indefinitely, indeed $\phi(s')$ (and hence s') is diverging too. □