A Near-Linear-Time Algorithm for Weak Bisimilarity on Markov Chains

David N. Jansen
State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing, China
dnjansen@ios.ac.cn

Jan Friso Groote
Department of Mathematics and Computer Science,
Eindhoven University of Technology, The Netherlands
J.F.Groote@tue.nl

Ferry Timmers
Department of Mathematics and Computer Science,
Eindhoven University of Technology, The Netherlands
F.Timmers@tue.nl

Pengfei Yang
State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China
yangpf@ios.ac.cn

Abstract

This article improves the time bound for calculating the weak/branching bisimulation minimisation quotient on state-labelled discrete-time Markov chains from $O(mn)$ to an expected-time $O(m \log^4 n)$, where n is the number of states and m the number of transitions. For these results we assume that the set of state labels AP is small ($|AP| \in O(m/n \log^4 n)$). It follows the ideas of Groote et al. (ACM ToCL 2017) in combination with an efficient algorithm to handle decremental strongly connected components (Bernstein et al., STOC 2019).

2012 ACM Subject Classification Theory of computation → Random walks and Markov chains; Theory of computation → Formal languages and automata theory; Theory of computation → Probabilistic computation; Software and its engineering → Formal software verification

Keywords and phrases Behavioural Equivalence, weak Bisimulation, Markov Chain

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2020.8

Funding *David N. Jansen*: This research is partly done during a visit to Eindhoven University of Technology, The Netherlands. This author is supported by the National Natural Science Foundation of China, Grants No. 61761136011 and 61532019.
Jan Friso Groote: This research is partly done during a visit to the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, P.R. China.

1 Introduction

Bisimilarity formalises when two behaviours are equal, where behaviours are given by automata or variants of directed graphs, such as labelled transition systems, Kripke structures or Markov chains. In these fields bisimilarity is also known as the zig-zag relation or lumping. Bisimilarity is an equivalence relation that preserves all core properties of behaviour. Moreover, calculating the bisimilarity quotient of a behaviour can lead to substantially smaller graphs. This is particularly useful when analysing the behaviour either by visual inspection or using other tools.

Further reduction of the behaviour is possible by considering some or all of the edges as silent steps, steps that cannot be observed directly. Milner referred to such steps as τ-actions [18]. Bisimulation equivalences that take such internal steps into account are weak bisimilarity [18], branching bisimilarity [9], stuttering equivalence [5], and weak/branching bisimulation on fully probabilistic systems [1].

In this article we are interested in weak behavioural equivalences for discrete-time Markov chains, similar to those introduced by [1]. There it was shown that branching and weak bisimilarity are equal notions on fully probabilistic systems, and an $O(mn)$ algorithm was given to calculate the weak/branching bisimilarity quotient, where m is the number of transitions and n is the number of states. In this paper we substantially improve upon this by providing an expected-time $O(m \log^4 n)$ algorithm, which is nearly linear in the number of transitions m, to calculate the weak/branching bisimilarity quotient on Markov chains.

The algorithm is an intricate combination of a number of rather different ideas stemming from various algorithms.

- The first idea is to use the principle "Process the smaller half" of Hopcroft [13] in the setting of probabilistic processes as in [7, 12, 20]. This means that whenever a state is revisited in an algorithm, its context is at least reduced by half compared to the previous visit. For n states this then means that each state is processed at most $O(\log n)$ times. This leads to $O(m \log n)$ algorithms to calculate (strong) bisimilarity on graphs. In our algorithm this is reflected in the use of two partitions of states, one containing constellations and a finer (or equal) partition containing blocks. The blocks are the context of a state that are reduced by a factor 2 for each visit.
- As weak and branching bisimilarity coincide on Markov chains, we can use the ideas from the algorithms for branching bisimulation minimisation. The first idea stems from [11]: detecting whether blocks need to be split can be done as efficiently as in strong bisimulation by only looking at *bottom states,* states without outgoing silent steps. The second idea comes from [10, 14]: the actual splitting of blocks can be done in time proportional to the smaller resulting subblock, guaranteeing that each state is visited at most $O(\log n)$ times in a split, according to the principle "Process the smaller half". This is achieved by simultaneously extending the markings and non-markings of bottom states to all other states in a block, and stop when the first of these two processes finishes.

  We introduce the notion of *$\mathcal{C}$-silent states,* which are states of which all outgoing transitions lead to the constellation containing the state itself. The role of the bottom states can now be played by non-$\mathcal{C}$-silent states. A block is only splittable if it has non-$\mathcal{C}$-silent states marked with different probabilities to go to a splitter block. These probabilities are then extended to the $\mathcal{C}$-silent states in the block in time proportional to the sizes of the smallest resulting subblocks.
- For bottom states and the simultaneous extension of the (non-)markings, it is essential that strongly connected components (SCCs) of inert steps can be contracted to a single state in the behavioural graph. The algorithms in [11, 10, 14] preprocess the graph accordingly. Unfortunately, it is not possible to contract inert SCCs in Markov chains, as this does not preserve weak bisimilarity. This is caused by the fact that often, the probability to leave an SCC through a specific edge is different for every state in the SCC, which means that states within SCCs are not necessarily weakly bisimilar.

  To extend the markings and non-markings, we need to know what the $\mathcal{C}$-silent SCCs are, i.e., the SCCs restricted to the $\mathcal{C}$-silent states in each block. This would not be a problem if the SCCs were static throughout each run of the algorithm, as they could be determined in time $O(m)$ as a preprocessing step. But whenever a constellation is split, the SCCs

change, as more states become non-$\mathcal{C}$-silent. It is not an option to recalculate the SCCs each time a state becomes non-$\mathcal{C}$-silent and is moved, as done in [21] for orthogonal bisimulation, as this would lead to a runtime of $O(mn)$.

Fortunately, we can use a recent result showing that SCCs can be maintained under the removal of edges within an expected $O(m \log^4 n)$ time [4]. Whenever splitting a constellation, states become non-$\mathcal{C}$-silent, and in particular their transitions are removed from the SCCs. The algorithm of [4] maintains the SCCs using the above complexity, allowing at any moment during the run of our algorithm to determine in constant time which states are in the same SCC, and that is exactly what we require.

The result is the first algorithm with an expected near-linear time complexity and a linear space complexity for the reduction of weak/branching bisimilarity of Markov chains.

Note that contrary to the previous less efficient algorithms we can only guarantee an *expected* runtime. The reason for this is deeply embedded in the algorithm of [4] to maintain decremental SCCs. The SCCs are constructed using a generalization of so-called ES-trees [8]. When SCCs fall apart, the ES-trees have to be recalculated, except for those SCCs which contain the roots of the old ES-trees. If such roots are chosen uniformly at random, there is a higher probability that the roots are in the larger ES-trees, and the work to recalculate them falls within an expected "Process the smaller half" regime. This is the only place where the algorithm is randomized. For the remainder it is completely deterministic.

The structure of this article is as follows. In Section 2 the required preliminaries are explained. In Section 3 the algorithm is outlined. The details, correctness and complexity are presented in Section 4, followed by a conclusion.

2 Preliminaries

We consider finite discrete-time Markov chains in the line of [1, 3]. In order to distinguish states, we allow for a state labelling with atomic propositions from a set AP.

▶ **Definition 1.** *A* discrete-time Markov chain *(DTMC) is a quadruple* $\mathcal{M} = (S, AP, \mathbf{P}, L)$ *where:*

- S *is a finite set of states. We write* n *for the number of states.*
- AP *is a finite set of atomic propositions.*
- $\mathbf{P} : S \times S \to [0, 1]$ *is a probability matrix satisfying* $\sum_{t \in S} \mathbf{P}(s, t) = 1$ *for all* $s \in S$*. We write* m *for the number of non-zero entries in* $\mathbf{P}$*.*
- $L : S \to 2^{AP}$ *is a labelling function, which assigns to each state* $s \in S$ *the set* $L(s)$ *of atomic propositions that are valid in* s*.*

In a DTMC $\mathcal{M} = (S, AP, \mathbf{P}, L)$, the transition probability function $\mathbf{P}(s, t)$ intuitively gives the probability of a state s going to t in a single step. For $s \in S$ and $A \subseteq S$, we define $\mathbf{P}(s, A) := \sum_{t \in A} \mathbf{P}(s, t)$ to be the probability of a state s entering A in a single step. If $\mathbf{P}(s, t) > 0$, we sometimes write $s \to t$ if the numerical value of the probability is irrelevant. We define the sets of incoming transitions $in(A) = \{s \to t \mid s \notin A \wedge t \in A\}$ and outgoing transitions $out(A) = \{s \to t \mid s \in A \wedge t \notin A\}$ for $A \subseteq S$. We assume the set AP to be small. For the complexity results we concretely require that $|AP| \in O(m/n \log^4 n)$.

As a side note we observe that it is easy to accommodate subprobabilistic DTMCs (i.e., to allow $\mathbf{P}$ to be a subprobability matrix satisfying $0 \leq \mathbf{P}(s, S) \leq 1$). In that case, one adds a pseudostate $\perp \notin S$ and defines $\mathbf{P}(s, \perp) = 1 - \mathbf{P}(s, S)$ and $\mathbf{P}(\perp, \perp) = 1$. Also, $\perp$ is separated from normal states by an atomic proposition: $L(\perp) = \{\mathsf{pseudo}\}$ for $\mathsf{pseudo} \notin AP$. Then, $(S \cup \{\perp\}, AP \cup \{\mathsf{pseudo}\}, \mathbf{P}, L)$ is a fully probabilistic DTMC.

A *path* in a DTMC is an infinite sequence of states $\pi = (s_0, s_1, s_2, \ldots)$ with $s_{i-1} \to s_i$ for $i > 0$. A *cylinder set* $Cyl(s_0, s_1, \ldots, s_n)$ is the set of paths that start with the sequence $(s_0, s_1, \ldots, s_n)$. Given an initial probability distribution μ on states, each cylinder set is assigned a probability: $Pr_\mu(Cyl(s_0, s_1, \ldots, s_n)) = \mu(s_0) \prod_{i=1}^{n} \mathbf{P}(s_{i-1}, s_i)$. The probability space of a DTMC can be defined as the unique extension of this content Pr_μ to the σ-algebra generated by the cylinder sets; details can be found in [16, Chapters 2, 4]. For a state s we write the Dirac distribution on s as $\delta(s)$, i.e., $\delta(s)(s) = 1$ and $\delta(s)(t) = 0$ for $t \neq s$. We denote the probability to take a path to a state in $C \subseteq S$ through states in $B \subseteq S$:

$$Pr(s, B, C) := \sum_{s_0,\ldots,s_{n-1} \in B \setminus C,\, s_n \in C} Pr_{\delta(s)}\left(Cyl(s_0, s_1, \ldots, s_n)\right).$$

Note that if $s \in C$, we have $Pr(s, B, C) = 1$ for any B; but if $s \notin B \cup C$, then $Pr(s, B, C) = 0$.

The presented algorithm is based on refining partitions of finite sets of states. For any set S, a *partition* of S is a set $\mathcal{B} = \{B_i \subseteq S \mid i \in I\}$ satisfying $\emptyset \notin \mathcal{B}$, $B_i \cap B_j = \emptyset$ whenever $i \neq j$, and $\bigcup \mathcal{B} = S$. We call each B_i a *block.*

For two partitions $\mathcal{B}_1 \neq \mathcal{B}_2$ of S, we say $\mathcal{B}_1$ is *finer* than $\mathcal{B}_2$, or that $\mathcal{B}_2$ is *coarser* than $\mathcal{B}_1$, iff for every block $B_1 \in \mathcal{B}_1$, there is a block $B_2 \in \mathcal{B}_2$, such that $B_1 \subseteq B_2$.

Given a partition $\mathcal{B}$, we denote the block containing state s by $[s]_\mathcal{B}$. If a set of states $B' \subseteq B$ for some block B in $\mathcal{B}$, we also write $[B']_\mathcal{B}$ for B being the block in which B' is contained. Every partition $\mathcal{B}$ induces an equivalence relation, also denoted $\mathcal{B}$, defined by $s \,\mathcal{B}\, t$ iff $s \in [t]_\mathcal{B}$. Conversely, every equivalence relation R on S has a unique corresponding partition $\{\{t \mid t\ R\ s\} \mid s \in S\}$. This partition is denoted as S/R. We denote by $[s]_R$ the R-equivalence class of $s \in S$, i.e., $[s]_R = \{t \mid t\ R\ s\}$.

Given a DTMC $\mathcal{M} = (S, AP, \mathbf{P}, L)$, we define AP-equivalence to be the relation that distinguishes states based on their labels: $s \equiv_{AP} t$ iff $L(s) = L(t)$. Its equivalence class for s is denoted $[s]_{AP}$.

▶ **Definition 2.** *Let $\mathcal{M} = (S, AP, \mathbf{P}, L)$ be a DTMC and $R \subseteq S \times S$ an equivalence relation. A state s is* R-silent *iff $\mathbf{P}(s, [s]_R) = 1$. A transition $s \to t$ is* R-inert *iff $s\ R\ t$.*

If the equivalence relation R is given by a partition $\mathcal{B}$ of states, we also speak about a $\mathcal{B}$-silent state. For non-R-silent states, we define the conditional probability to enter some set of states in a single step under the condition to leave the R-equivalence class:

$$\mathbf{P}(s, B \mid \text{non-}R\text{-inert}) := \frac{\mathbf{P}(s, B)}{1 - \mathbf{P}(s, [s]_R)}.$$

We are now ready to introduce the notions of weak and branching bisimilarity.

▶ **Definition 3.** *Let $\mathcal{M} = (S, AP, \mathbf{P}, L)$ be a DTMC and $R \subseteq \equiv_{AP}$ an equivalence relation on S (which respects the atomic propositions of states). We say that R is*

a weak bisimulation *iff $s\ R\ t$ implies, for all R-equivalence classes $C \in S/R$ with $s, t \notin C$, that $Pr(s, [s]_{AP}, C) = Pr(t, [t]_{AP}, C)$.*

a branching bisimulation *iff $s\ R\ t$ implies, for all R-equivalence classes $C \in S/R$ with $s, t \notin C$, that $Pr(s, [s]_R, C) = Pr(t, [t]_R, C)$.*

a conditional-probability bisimulation *iff $s\ R\ t$ implies*

1. *If s and t are both non-R-silent, for all R-equivalence classes $C \in S/R$ with $s, t \notin C$, we have $\mathbf{P}(s, C \mid \text{non-}R\text{-inert}) = \mathbf{P}(t, C \mid \text{non-}R\text{-inert})$; and*
2. *s has a path to a state outside $[s]_R$ iff t has a path to a state outside $[t]_R$.*

The states s and t are weakly bisimilar, *denoted $s \approx_w t$, iff a weak bisimulation R exists such that $s\ R\ t$. Similarly, s and t are* branching bisimilar, *denoted $s \approx_b t$, iff a branching bisimulation R exists such that $s\ R\ t$. Finally, s and t are* conditional-probability-bisimilar, *denoted $s \approx_c t$, iff a conditional-probability bisimulation R exists such that $s\ R\ t$.*

All three relations $\approx_w$, $\approx_b$ and $\approx_c$ are equivalence relations. The essential difference between branching and weak bisimulation is that in branching bisimulation a step from state s must be mimicked by a number of steps through the equivalence class of s, whereas in weak bisimulation a step can be mimicked by steps through states labelled with the same propositions. For nondeterministic transition systems branching bisimilarity implies weak bisimilarity but not vice versa. Furthermore, deciding branching bisimilarity is more efficient. Remarkably, in the context of Markov chains, the notions are equal:

▶ **Proposition 4.** *Let $\mathcal{M} = (S, AP, \mathbf{P}, L)$ be a DTMC. States in S are weakly bisimilar iff they are branching bisimilar iff they are conditional-probability-bisimilar.*

Proof. [1, 2] prove this result for fully probabilistic systems (with action labels instead of atomic propositions). See the Appendix for an adaptation of the proof to DTMCs. ◀

Due to this proposition, we only write "weak bisimulation" in the remainder of this paper. Because conditional-probability bisimulation can be tested by looking at single-step probabilities only, we use its conditions to calculate the weak bisimilarity quotient without having to calculate a transitive closure.

If we allowed DTMCs with an infinite state space, conditional-probability bisimilarity would not imply weak/branching bisimilarity. See [15] for an example and a possible strengthening of Condition 2 in the definition of conditional-probability bisimulation.

3 Main ideas of the algorithm

Now we state our problem formally:

> Given a DTMC $\mathcal{M} = (S, AP, \mathbf{P}, L)$, we need to compute the weak bisimilarity relation, or equivalently, to give a partition $\mathcal{B}$ of the state space S consisting of the weak bisimilarity equivalence classes.

This section explains the main ideas to solve this problem efficiently.

Partition refinement. Typically, bisimilarity is computed by partition refinement. In our case this starts from an initial partition where states are in the same block iff they have the same atomic propositions and satisfy Condition 2 of conditional-probability bisimulation – note that refinements of a partition preserve these conditions. Then the algorithm checks Condition 1 of conditional-probability bisimulation for every block. If it finds a pair of states in one block with different transition probabilities to another block, it splits the former block, bringing the validity of Condition 1 closer. The latter block is called a *splitter*.

Avoid superfluous refinements. After a splitter $Sp \in \mathcal{B}$ has been used to split all blocks with transitions to Sp, every further refined partition is stable w.r.t. Sp. However, the algorithm of [1] does not register this information and may check whether Sp is a splitter repeatedly. We optimize by registering former splitters. In addition to the current partition $\mathcal{B}$, we store a coarser (or equal) partition $\mathcal{C}$ to record the splitters already used in previous iterations. We call blocks in $\mathcal{C}$ *constellations* and print them in boldface $\mathbf{C} \in \mathcal{C}$. The relation between blocks $\in \mathcal{B}$ and constellations $\in \mathcal{C}$ is described by the main invariant:

▶ **Main Invariant 5.** *If s and t are non-$\mathcal{C}$-silent states in the same block, i.e., $[s]_{\mathcal{B}} = [t]_{\mathcal{B}}$, and $\mathbf{C} \in \mathcal{C}$ is a constellation that does not contain s and t, then*

$$\mathbf{P}(s, \mathbf{C} \mid \text{non-}\mathcal{C}\text{-inert}) = \mathbf{P}(t, \mathbf{C} \mid \text{non-}\mathcal{C}\text{-inert}).$$

This invariant expresses that states in the same block have the same conditional probability to enter every other constellation in a single step. This means that blocks cannot be split by constellations. They can however be split by other blocks.

▶ **Example 6.** The formulation of the main invariant is inspired by conditional-probability bisimulation. The DTMC fragment in Figure 1 shows that a formulation inspired by weak or branching bisimulation does not appear to work.

States s and t have the same conditional probability to enter $\mathbf{C}$ in a single step: $\mathbf{P}(s, \mathbf{C} \mid \text{non-}\mathcal{C}\text{-inert}) = \frac{0.5}{1-0.2-0.3} = 1 = \frac{1}{1-0} = \mathbf{P}(t, \mathbf{C} \mid \text{non-}\mathcal{C}\text{-inert})$. However, because s also has a transition to state u, which is in a different block of the same constellation and has no $\mathcal{C}$-inert path to $\mathbf{C}$, we have $Pr(s, [s]_{\mathcal{C}}, \mathbf{C}) = 0.5 + 0.2 \cdot 1 = 0.7 \neq 1 = Pr(t, [t]_{\mathcal{C}}, \mathbf{C})$, and also $\mathbf{P}(s, \mathbf{C} \mid \text{non-}\mathcal{B}\text{-inert}) = \frac{0.5}{1-0.2} = 0.625 \neq 1 = \frac{1}{1-0} = \mathbf{P}(t, \mathbf{C} \mid \text{non-}\mathcal{B}\text{-inert})$. If $\mathcal{B} = \mathcal{C}$, then no states like u exist, and Main Invariant 5 (together with Condition 2 of conditional-probability bisimulation, which was already ensured by the initial partition) implies weak bisimulation.

Refining constellations in $\mathcal{C}$. As mentioned in the example, we should try to reach the situation that $\mathcal{B} = \mathcal{C}$. Therefore, as long as these partitions are different, we choose a small splitter block $Sp \in \mathcal{B} \setminus \mathcal{C}$. We move Sp to its own constellation, and reestablish Main Invariant 5 by splitting blocks with transitions to Sp. By choosing a small splitter, we ensure that each state takes part in constellation processing only logarithmically often – according to the principle "Process the smaller half".

To register the fact that we have used some Sp as a splitter, we refine its $\mathcal{C}$-equivalence class $[Sp]_{\mathcal{C}}$ into Sp and the rest $[Sp]_{\mathcal{C}} \setminus Sp$. Note that establishing Main Invariant 5 w.r.t. Sp also automatically establishes the invariant w.r.t. $[Sp]_{\mathcal{C}} \setminus Sp$ for most states: As we know $\mathbf{P}(s, [Sp]_{\mathcal{C}} \mid \text{non-}\mathcal{C}\text{-inert}) = \mathbf{P}(t, [Sp]_{\mathcal{C}} \mid \text{non-}\mathcal{C}\text{-inert})$ for states s, t in the same block $\not\subseteq [Sp]_{\mathcal{C}}$, and the algorithm ensures that $\mathbf{P}(s, Sp \mid \text{non-}\mathcal{C}\text{-inert}) = \mathbf{P}(t, Sp \mid \text{non-}\mathcal{C}\text{-inert})$, we get for free that $\mathbf{P}(s, [Sp]_{\mathcal{C}} \setminus Sp \mid \text{non-}\mathcal{C}\text{-inert}) = \mathbf{P}(t, [Sp]_{\mathcal{C}} \setminus Sp \mid \text{non-}\mathcal{C}\text{-inert})$. Only for states in Sp itself we have to additionally check whether they are stable under $[Sp]_{\mathcal{C}} \setminus Sp$.

Refining blocks in $\mathcal{B}$. For every block B with transitions to the selected splitter Sp, we first split its non-$\mathcal{C}$-silent states into subblocks whose conditional probabilities to enter the splitter in a single step are equal. Similarly, in algorithms for branching bisimilarity [11, 10], one splits the bottom states into two subblocks first. But in our case the non-$\mathcal{C}$-silent states may fall apart into more than two subblocks, depending on the (conditional) probability to enter the splitter in a single step.

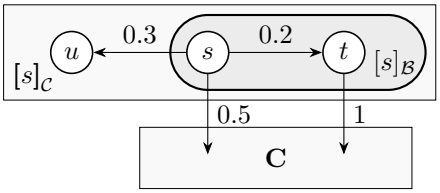


Figure 1 Even if $\mathbf{P}(s, \mathbf{C} \mid \text{non-}\mathcal{C}\text{-inert}) = \mathbf{P}(t, \mathbf{C} \mid \text{non-}\mathcal{C}\text{-inert})$, we still may have $Pr(s, [s]_{\mathcal{C}}, \mathbf{C}) \neq Pr(t, [t]_{\mathcal{C}}, \mathbf{C})$.

The next step is to extend this splitting to the $\mathcal{C}$-silent states in B. Assume for the moment that there are no strongly connected components of $\mathcal{C}$-silent states. This means that the $\mathcal{B}$-inert transitions in B form a dag. The $\mathcal{C}$-silent states that only have paths to a single subblock of non-$\mathcal{C}$-silent states join this subblock. This condition is checked by traversing the transition relation backwards, to find states whose outgoing $\mathcal{B}$-inert transitions all lead to the same subblock. Only after we have investigated all outgoing $\mathcal{B}$-inert transitions of a state, we can extend the splitting to its predecessors. Those $\mathcal{C}$-silent states that have $\mathcal{B}$-inert paths to multiple subblocks move to a special subblock, which we call the *residue*. These paths to other subblocks may have different probabilities for different states, but as the residue will be split further later in the algorithm, this is not a problem.

However, the subblocks have to be extended in a way compatible with the principle "Process the smaller half". This forbids to spend time on a subblock with more than half the states of B. Because we do not know which subblock will end up to be the largest, all subblocks are extended simultaneously, and when a subblock (which can be the residue) becomes too large, we abort extending this subblock. Note that at most one block can be so large. This process of simultaneous backward extension of the subblocks ends if all but one block have been completed. All non-visited states necessarily belong to this non-completed block. The time used for the backward extension can now be attributed completely to the smaller subblocks whose sizes are guaranteed to be at most half that of B.

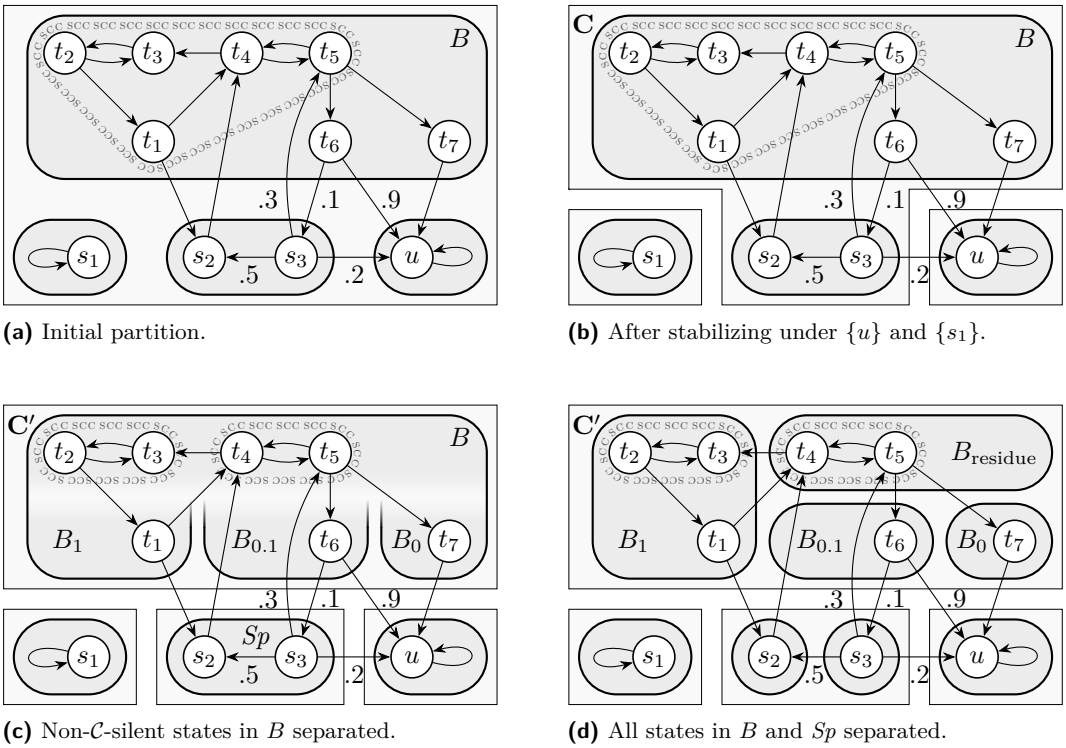
Strongly connected components. Unfortunately, if there are non-trivial SCCs in the $\mathcal{B}$-inert transitions, the backward extension does not work. The reason is that in order to decide whether a state belongs to a subblock or to the residue, all its outgoing $\mathcal{B}$-inert transitions must have been investigated first. In a dag this is guaranteed, but with non-trivial SCCs, this is impossible. Still, all states in such a non-trivial SCC have paths to the same subblocks. It is therefore possible to view this SCC as a single state, and apply the dag-based backward extension of subblocks as described above to the complete SCC.

For this to work we have to keep track of SCCs in the $\mathcal{B}$-inert transitions within the $\mathcal{C}$-silent states of every block. When a constellation is split, some states may become non-$\mathcal{C}$-silent, and we have to dynamically recompute the sub-SCCs within the states that remain $\mathcal{C}$-silent. It is not possible to use a classical linear algorithm for this purpose, as the strongly connected components have to be recomputed for all new blocks, including a potential block that is larger than half of the size of B. Therefore, the "Process the smaller half" strategy cannot be applied, leaving us with an $O(mn)$ algorithm.

Fortunately, recently an efficient algorithm has been presented that can maintain the strongly connected components within a lower time bound [4]. This algorithm initializes a data structure for SCCs in worst-case time complexity $O(m \log^4 n)$ and recomputes all sub-SCCs in total *expected* time complexity $O(m \log^4 n)$.

▶ **Example 7.** Figure 2 illustrates the above ideas for an example DTMC. The state labellings with atomic propositions are not explicitly indicated, but different letters in the state name indicate different labellings. For some transitions the exact probability value is not shown, but it is nonzero, and all outgoing probabilities sum up to 1.

Subfigure 2a shows the initial partition: states with different labelling are in different blocks (shown as rounded, darker grey shapes). Additionally, to satisfy Condition 2 of conditional-probability bisimulation, we have separated s_1 from states s_2 and s_3 as only the latter two have paths to a state outside $\{s_1, s_2, s_3\}$. All states are in one constellation (shown as light grey rectangle).



(a) Initial partition.

(b) After stabilizing under $\{u\}$ and $\{s_1\}$.

(c) Non-$\mathcal{C}$-silent states in B separated.

(d) All states in B and Sp separated.

Figure 2 The first few steps of partition refinement.

Subfigure 2b shows the situation after blocks $\{u\}$ and $\{s_1\}$ have been used as splitters. This is recorded by putting them into their own constellations. However, these two splitters did not lead to actual refinements. State s_2 only has a transition to another block in the same constellation, and when this block becomes a splitter, it will separate s_2 from s_3.

Then, the only remaining small splitter is block $Sp = \{s_2, s_3\}$ in constellation $\mathbf{C}$. Block B, the block with transitions to Sp, needs to be refined. Subfigure 2c shows the situation after the non-$\mathcal{C}$-silent states t_1, t_6 and t_7 have been separated: every state is in a block corresponding to its conditional probability to enter Sp in a single step. Also, state t_1 was $\mathcal{C}$-silent but is so no longer, so it had to be removed from its SCC, and the algorithm quickly finds two sub-SCCs in the remaining $\mathcal{C}$-silent states. Note that the transition $t_1 \to t_4$ is ignored when calculating the conditional probability $\mathbf{P}(t_1, Sp \mid \text{non-}\mathcal{C}\text{-inert})$.

The refinement has to be extended to the $\mathcal{C}$-silent SCCs in the block. The situation after this has finished is shown in subfigure 2d: SCC $\{t_2, t_3\}$ is completely added to B_1 because its only outgoing transition goes to a state in B_1. Here, one can see that it is necessary to move the SCC to B_1 as a whole. If we would have tried to move state t_2 individually to B_1, we would have to wait until t_3 is in B_1 first, which in turn depends on t_2 already being in B_1. The SCC $\{t_4, t_5\}$ is moved to the residue because t_5 can enter both $B_{0.1}$ and B_0 in a single step. Note that also Sp has been split into two because it was unstable under $\mathbf{C}' = [Sp]_{\mathcal{C}} \setminus Sp$. State s_2 has (conditional) probability 1 to enter $\mathbf{C}'$ in a single step, while s_3 has conditional probability $.3/(.3 + .2) = 0.6$.

In further refinements, where the new blocks are used as splitters, the algorithm will find that the states in SCC $\{t_2, t_3\}$ are equivalent, but SCC $\{t_4, t_5\}$ will be split up because the states have different probabilities to reach other blocks like B_0.

Algorithm 1 Efficient algorithm for weak bisimulation of Markov chains.

1.1: $\mathcal{B}$:= coarsest partition of S that is at least as fine as $S/\equiv_{AP}$ and satisfies Condition 2 of conditional-probability bisimulation
1.2: Add all $B \in \mathcal{B}$ (except one maximal-size one) to the set of potential splitters
1.3: Label all transitions as $\mathcal{C}$-inert, $\mathcal{C} := \{S\}$
1.4: Construct all SCCs based on the $\mathcal{B}$-inert transitions
1.5: **for all** potential splitters $Sp \in \mathcal{B}$ **do**
1.6: **for all** incoming transitions $t \to s \in in(Sp)$ **do**
1.7: **if** $t \to s$ is labelled as $\mathcal{C}$-inert **then** {i.e. $t \in [Sp]_{\mathcal{C}} \setminus Sp$}
1.8: **if** t is $\mathcal{C}$-silent **then** remove t from $SCC(t)$, $P_{\text{non-}\mathcal{C}\text{-inert}}(t) := 0$, $P_{\to\text{splitter}}(t) := 0$
1.9: $P_{\text{non-}\mathcal{C}\text{-inert}}(t) := P_{\text{non-}\mathcal{C}\text{-inert}}(t) + \mathbf{P}(t, s)$
1.10: Label $t \to s$ as non-$\mathcal{C}$-inert
1.11: **end if**
1.12: **if** $P_{\to\text{splitter}}(t) = 0$ **then** mark state t (as a predecessor)
1.13: $P_{\to\text{splitter}}(t) := P_{\to\text{splitter}}(t) + \mathbf{P}(t, s)$
1.14: **end for**
1.15: **for all** outgoing transitions $s \to t \in out(Sp)$ labelled as $\mathcal{C}$-inert **do** {i.e. $t \in [Sp]_{\mathcal{C}} \setminus Sp$}
1.16: **if** s is $\mathcal{C}$-silent **then** remove s from $SCC(s)$, $P_{\text{non-}\mathcal{C}\text{-inert}}(s) := 0$, $P_{\to\text{splitter}}(s) := 0$
1.17: $P_{\text{non-}\mathcal{C}\text{-inert}}(s) := P_{\text{non-}\mathcal{C}\text{-inert}}(s) + \mathbf{P}(s, t)$
1.18: Label $s \to t$ as non-$\mathcal{C}$-inert
1.19: **if** $P_{\to\text{splitter}}(s) = 0$ **then** mark state s (as a predecessor)
1.20: $P_{\to\text{splitter}}(s) := P_{\to\text{splitter}}(s) + \mathbf{P}(s, t)$
1.21: **end for**
1.22: Remove Sp from the set of potential splitters, $\mathcal{C}' := (\mathcal{C} \setminus \{[Sp]_{\mathcal{C}}\}) \cup \{[Sp]_{\mathcal{C}} \setminus Sp, Sp\}$
1.23: **for all** all blocks B containing marked states **do**
1.24: Refine B according to $\frac{P_{\to\text{splitter}}(\,\cdot\,)}{P_{\text{non-}\mathcal{C}\text{-inert}}(\,\cdot\,)}$ (Algorithm 2)
1.25: **end for**
1.26: $\mathcal{C} := \mathcal{C}'$
1.27: **end for**
1.28: **return** $\mathcal{B}$

4 Detailed algorithm for weak bisimilarity on Markov chains

In this section, we walk through the pseudocode to explain how the ideas of the previous section can be fleshed out.

We store all states such that we can easily traverse all incoming transitions. There is a partition $\mathcal{B}$ of states. We store a set of potential splitter blocks, and we label every transition to indicate whether it is $\mathcal{C}$-inert. As this is all we need to know of the constellation, we do not have to maintain $\mathcal{C}$ explicitly. We treat $\mathcal{C}$ as a ghost variable, as it is only used to prove correctness. Code referring to the ghost variable is printed in grey.

Non-$\mathcal{C}$-silent states are handled individually, while $\mathcal{C}$-silent states are handled as part of an SCC. When some state becomes non-$\mathcal{C}$-silent, the algorithm moves it to the individually-handled states and recalculates the maximal sub-SCCs in the remaining $\mathcal{C}$-silent states.

To make the code slightly more efficient, every non-$\mathcal{C}$-silent state also has an associated probability $P_{\text{non-}\mathcal{C}\text{-inert}}(s)$, which is the probability to take a non-$\mathcal{C}$-inert transition. This is the denominator in the conditional probability $\mathbf{P}(s, \cdot \mid \text{non-}\mathcal{C}\text{-inert})$.

Algorithm 2 Refine block B (Line 1.24).

2.1: Move the non-$\mathcal{C}$-silent states of B to new subblocks $B_p = \{s \in B \mid \frac{P_{\to\text{splitter}}(s)}{P_{\text{non-}\mathcal{C}\text{-inert}}(s)} = p\}$
2.2: Unmark all states and set $P_{\to\text{splitter}}(\,\cdot\,) := 0$ in the subblocks B_p
2.3: **if** for some B_p, we have $|B_p| > \frac{1}{2}$(original size of B) **then** Mark this B_p as aborted
2.4: Create an empty block B_{residue} for the residue
2.5: **while** there are at least two non-completed subblocks (including B_{residue}) **do**
2.6: **for all** non-completed and non-aborted normal subblocks B_p **do**
2.7: Take one step for B_p in Algorithm 3
2.8: **end for**
2.9: **if** B_{residue} is not aborted **then** take one step for B_{residue} in Algorithm 4
2.10: **end while**
2.11: Let B_{aborted} be the only non-completed subblock (or residue).
2.12: **if** $B_{\text{residue}} = \emptyset$ **then**
2.13: **if** $B_{\text{residue}} = B_{\text{aborted}}$ **then** let B_{aborted} := a maximal-size normal B_p
2.14: Delete B_{residue}
2.15: **end if**
2.16: Move all states in B_{aborted} back to B and delete B_{aborted}

Algorithm 1: Main program. We initialize all data structures in Lines 1.1–1.4. To construct the initial partition (Line 1.1), one starts with the partition $S/\equiv_{AP}$ and marks all states s that have a path to a state outside their block $[s]_{AP}$. Then, every block is separated into the marked and unmarked states if necessary. The resulting partition satisfies Condition 2 of conditional-probability bisimulation. – Line 1.4 does not need to check whether states are $\mathcal{C}$-silent, as there are no non-$\mathcal{C}$-inert transitions at this moment.

Each iteration of the main loop refines one constellation of $\mathcal{C}$. This is done by selecting a small splitter Sp and the following two steps.

First, it calculates the probability to enter the splitter Sp in a single step for each state in the loop at Lines 1.6–1.14. It also calculates, for the states in Sp, the probability to enter $[Sp]_{\mathcal{C}} \setminus Sp$ in a single step at Lines 1.15–1.21 because Sp is the one block that is not automatically stabilized under $[Sp]_{\mathcal{C}} \setminus Sp$ by stabilizing under Sp. During these calculations, the algorithm may find that some states in $[Sp]_{\mathcal{C}}$ are no longer $\mathcal{C}$-silent at Lines 1.8 and 1.16. In that case, it (efficiently) recalculates the SCCs that formerly contained these states.

Second, from Line 1.23 it splits all blocks that have some transition to their splitter by calling Algorithm 2.

Algorithm 2: Refine block B. At Line 2.1, we split the non-$\mathcal{C}$-silent states. This can be done by sorting them according to their conditional probability and group those with the same probability in one subblock.

To extend the separation to $\mathcal{C}$-silent states, we start a coroutine (see Algorithm 3) for each initial subblock. Additionally, we also start one coroutine (see Algorithm 4) for the so-called residue, i.e. the part of the block that cannot be put into one subblock because it has paths to several non-$\mathcal{C}$-silent states with different probabilities $P_{\to\text{splitter}}(s)/P_{\text{non-}\mathcal{C}\text{-inert}}(s)$. In the main loop in Lines 2.5–2.10, we let each coroutine do one step (execute one loop iteration, handle one transition) in turn until all of them except one have completed their search. Then, the remaining states must belong to the incomplete subblock. If there is a subblock containing the majority of the states, this will be the incomplete subblock; otherwise, the incomplete subblock will often be large, but it does not matter much. This guarantees that the processing time is proportional to the cumulative size of those subblocks that are at most half the original block.

Algorithm 3 Extend a normal subblock B_p (Line 2.7).

```
3.1: for all unmarked states s ∈ B_p do
3.2:   for all incoming transitions t → s ∈ in(SCC(s)) with C-silent t do
3.3:     if SCC(t) ⊆ B then
3.4:       Move SCC(t) from B to the marked states of B_p
3.5:       untested(SCC(t)) := |{outgoing B-inert transitions of SCC(t)}|
3.6:     end if
3.7:     if SCC(t) ⊆ B_p then {SCC(t) must have been marked}
3.8:       untested(SCC(t)) := untested(SCC(t)) − 1
3.9:       if untested(SCC(t)) = 0 then
3.10:        Unmark SCC(t)
3.11:        if |{unmarked states ∈ B_p}| > ½(original size of B) then
3.12:          Mark B_p as aborted
3.13:          Abort this copy of Algorithm 3
3.14:        end if
3.15:      end if
3.16:    else if SCC(t) ⊆ B_q for some q ≠ p then {SCC(t) must have been marked}
3.17:      Unmark SCC(t) and move it from B_q to B_residue
3.18:      if |B_residue| > ½(original size of B) then Mark B_residue as aborted
3.19:    end if
3.20:  end for
3.21: end for
3.22: Unmark all marked SCCs in B_p and move them to B_residue
3.23: if |B_residue| > ½(original size of B) then Mark B_residue as aborted
3.24: Insert B_p into the set of potential splitters
3.25: Mark B_p as completed
```

After B has been divided completely, the identity of B is reused to become a large subblock (so that most pointers from states to their block need not be changed). Only if the residue is empty, it may have become aborted accidentally, so we correct for that.

Algorithm 3: Extend a normal subblock B_p. This algorithm extends the subblock B_p by all $\mathcal{C}$-silent states in B that only have transitions to B_p-states. The basic idea is as follows. As soon as we find a transition from such a state t to an unmarked B_p-state s, we provisionally add it to B_p as a marked state at Line 3.4 – here, marked states indicate that if t is not in the residue, then it is in B_p. Note that each such transition from t to s is only investigated once. We also initialize a counter $\mathit{untested}(t)$ to the number of outgoing $\mathcal{B}$-inert transitions of t that are not known to go to B_p. When we have determined that no more of such transitions exist, we unmark t at Line 3.10 to indicate that it is definitely in B_p. If, however, we find that state t has already been moved to another $B_q \neq B_p$, it has transitions to several subblocks, so it has to move to the residue at Line 3.17.

The above is slightly complicated by the fact that we may have nontrivial SCCs in B. Therefore, we do not handle $\mathcal{C}$-silent states individually, but always their SCC as a whole. Instead of moving a single state t, we move the whole $\mathit{SCC}(t)$ to B_p or the residue. And instead of considering the incoming transitions of a single $\mathcal{C}$-silent state s, we simultaneously consider all incoming transitions of $\mathit{SCC}(s)$ at Line 3.2. In case s is not $\mathcal{C}$-silent, we regard $\mathit{in}(\mathit{SCC}(s))$ to be $\mathit{in}(s)$ at this line.

Algorithm 4 Extend the residue B_{residue} (Line 2.9).

```
4.1: while there are at least two non-completed subblocks B_p, B_q do
4.2:   for all states s ∈ B_residue that are not yet completely handled do
4.3:     for all incoming transitions t → s ∈ in(SCC(s)) with C-silent t do
4.4:       if SCC(t) ⊆ B_p for some p or SCC(t) ⊆ B then
4.5:         Unmark SCC(t) and move it to B_residue
4.6:         if |B_residue| > ½(original size of B) then
4.7:           Mark the residue as aborted
4.8:           Abort Algorithm 4
4.9:         end if
4.10:      end if
4.11:    end for
4.12:  end for
4.13: end while
4.14: Insert B_residue into the set of potential splitters
4.15: Mark the residue as completed
```

When B_p gets too large at Line 3.11 its handling is aborted, simply meaning that it is not handled further in the algorithm. Note that at most one subblock can be too large, so every subblock except possibly one is completely investigated. At the end of Algorithm 2 the aborted block is cleaned up. When no more states can be added, the states that are still marked must have a transition to B_p and either to some other $B_q \neq B_p$ or to the residue. So, they also move to the residue at Line 3.22.

Algorithm 4: Extend the residue. The handling of the residue is, in a sense, simpler than B_p: every predecessor t of a state in the residue with t originally in B also belongs to the residue. Therefore, we greedily add all states in $SCC(t)$ to the residue.

However, when all states currently in the residue have been handled and still two or more normal subblocks B_p, B_q are incomplete, it may happen that new states will be added to the residue at Lines 3.17 or 3.22 in the coroutines for B_p or B_q. Therefore Algorithm 4 has to wait until (at most) one normal subblock is incomplete (Line 4.1). When only one subblock is incomplete, no other subblock has marked states left that could still move to the residue.

4.1 Correctness

The correctness can be expressed by the following lemmas.

▶ **Lemma 8** (Loop invariants in Algorithm 1)**.**

1. *Whenever Line 1.5 is reached, the following holds:*
   - *$\mathcal{B}$ satisfies Condition 2 of the definition of conditional-probability bisimulation, and is at least as coarse as weak bisimilarity and at least as fine as $S/\equiv_{AP}$ and $\mathcal{C}$.*
   - *Main Invariant 5.*
   - *For every constellation $\mathbf{C} \in \mathcal{C}$, exactly one block $B \subseteq \mathbf{C}$ is not in the set of potential splitters.*
   - *For non-$\mathcal{C}$-silent state $s \in S$, we have $P_{\text{non-}\mathcal{C}\text{-inert}}(s) = 1 - \mathbf{P}(s, [s]_{\mathcal{C}})$ and $P_{\to\text{splitter}}(s) = 0$.*
   - *Every transition is labelled as $\mathcal{C}$-inert iff it is (cf. Line 1.10).*

2. *After executing Line 1.22, it holds:*
   - *For every constellation $\mathbf{C} \in \mathcal{C}'$, exactly one block $B \subseteq \mathbf{C}$ is not in the set of potential splitters.*
   - *For every state $s \in S$,* (†)
     - $P_{\text{non-}\mathcal{C}\text{-inert}}(s) = 1 - \mathbf{P}(s, [s]_{\mathcal{C}'})$.
     - *If $s \notin Sp$, then* $P_{\to\text{splitter}}(s) = \mathbf{P}(s, Sp)$.
     - *If $s \in Sp$, then* $P_{\to\text{splitter}}(s) = \mathbf{P}(s, [Sp]_{\mathcal{C}} \setminus Sp)$.
   - *Every transition is labelled as $\mathcal{C}$-inert iff it is $\mathcal{C}'$-inert.*

In the proof of Lemma 8, we refer to Algorithm 2. Therefore, we first state its correctness:

▶ **Lemma 9** (Correctness of Algorithm 2). *If all states in B satisfy the conditions* (†) *of Lemma 8, then Algorithm 2 splits B into the coarsest subblocks that satisfy*

- *Main Invariant 5 w.r.t. $\mathcal{C}'$, i.e. if s and t are non-$\mathcal{C}'$-silent states in B before Algorithm 2, then they are in the same subblock B_p after Algorithm 2 iff for all constellations $\mathbf{C} \in \mathcal{C}'$ that do not contain s or t, $\mathbf{P}(s, \mathbf{C} \mid \text{non-}\mathcal{C}'\text{-inert}) = \mathbf{P}(t, \mathbf{C} \mid \text{non-}\mathcal{C}'\text{-inert})$.*
- *$\mathcal{C}'$-silent states in B_p have a B_p-inert path to some state outside B_p, but not to a state in another B_q.*
- *All states $\in B_{\text{residue}}$ are $\mathcal{C}'$-silent and have B_{residue}-inert paths to at least two subblocks of the form B_p.*
- *For every state $s \in B$ before Algorithm 2, we have $P_{\to\text{splitter}}(s) = 0$ after Algorithm 2.*

Proof of Lemma 9. Line 2.1 ensures the condition on non-$\mathcal{C}'$-silent states holds; the remainder of Algorithms 2–4 only moves $\mathcal{C}'$-silent states around.

During the execution of Algorithm 3, we have for every $\mathcal{C}'$-silent SCC $\subseteq B$: If all its outgoing $\mathcal{B}$-inert transitions have been tested and go to B_p, then the SCC also is in the unmarked part of B_p. If some (but not all) of its outgoing $\mathcal{B}$-inert transitions have been tested, then the SCC is either in the marked part of B_p or in B_{residue}. (It is in B_{residue} only if it satisfies the conditions in the next paragraph.) Also, every state in B_p has a B_p-inert path to some non-$\mathcal{C}'$-silent state in B_p. From this we can conclude that at Line 3.22, the unmarked $\mathcal{C}'$-silent states of B_p have B_p-inert paths to some non-$\mathcal{C}'$-silent state in B_p (and therefore some state outside B_p), but they have no B_p-inert path to different B_q. The marked $\mathcal{C}'$-silent states of B_p can go in a single step to some state in B_p and also some other state, that either is in some $B_q \neq B_p$ or in B_{residue}. In both cases, adding the marked states to B_{residue} maintains the condition on B_{residue} of the lemma.

During the execution of Algorithm 4, we have for every SCC $\subseteq B$: The SCC is in the residue if it has tested transitions to at least two subblocks of the form B_p (because it was a marked SCC of one copy of Algorithm 3, and another copy executed Line 3.17), or if it has a tested transition to a completed subblock B_p and a transition that leads elsewhere (because it was a marked SCC of B_p at Line 3.22), or if it has a transition to some other state in B_{residue} that was visited in Line 4.5. In all such cases, it consists of $\mathcal{C}'$-silent states which have B_{residue}-inert paths to at least two subblocks of the form B_p. When Algorithm 4 leaves the loop at Lines 4.1–4.13, at most one copy of Algorithm 3 is still running. This copy will not add anything to the residue because all states with transitions to completed copies of Algorithm 3 have already been visited and moved to the appropriate B_p or B_{residue}. ◀

Proof of Lemma 8. When Line 1.5 is reached for the first time, $\mathcal{B}$ and $\mathcal{C}$ have just been initialized to values that obviously satisfy Item 1 of Lemma 8.

Let's now look at the situation after Line 1.22. Lines 1.6–1.22 do not change $\mathcal{B}$ or $\mathcal{C}$, so Item 1 remains valid, except that Sp is not marked as a splitter any more. – The only difference between $\mathcal{C}$ and $\mathcal{C}'$ is that Sp is now in its own constellation. As Sp was in the set of potential splitters, exactly one other block $\subseteq [Sp]_{\mathcal{C}}$ was not in this set, and we now have

that this is the one block $\subseteq [Sp]_{\mathcal{C}} \setminus Sp$ not in the set of potential splitters. Therefore, the first part of Item 2 of Lemma 8 holds. Condition (†) is ensured for every state by the loops in Lines 1.6–1.21. Note that $\mathcal{C}$-inert transitions that enter or leave Sp must be transitions from and to $[Sp]_{\mathcal{C}} \setminus Sp$, so these transitions are not $\mathcal{C}'$-inert.

After Algorithm 2 has been called for all blocks with marked states (i.e. all blocks that have transitions to a splitter) at Lines 1.23–1.25, we have that Item 1 of Lemma 8 is satisfied for $\mathcal{C}'$. In particular, no split by Algorithm 2 separated weakly bisimilar states. Line 1.26 then ensures Item 1 for $\mathcal{C}$. ◀

▶ **Theorem 10.** *Algorithm 1 returns the partition of S into weak bisimilarity equivalence classes.*

Proof. This is an immediate consequence of Item 1 of Lemma 8, in particular Main Invariant 5 and the fact that $\mathcal{B} = \mathcal{C}$ if the set of potential splitters is empty. ◀

4.2 Complexity

The time complexity can be described by the following theorem. Observe that $n \leq m$.

▶ **Theorem 11.** *All operations in the main loop of Algorithm 1 fall under one of the following cases.*

1. *State s is handled as part of a small splitter at most $\lfloor \log_2 n \rfloor$ times. Whenever this happens, $O((|in(s)| + 1) \log n + |out(s)|)$ time is spent.*
2. *State s becomes part of a small subblock at most $\lfloor \log_2 n \rfloor$ times. Whenever this happens, $O(|in(s)| + 1)$ time is spent.*
3. *All operations on decremental SCC handling together run in expected time $O(m \log^4 n)$.*

Summing up, the overall time complexity is in expected time $O(m \log^4 n)$.

Data structures. To prove Theorem 11, we propose that the algorithm stores the following information.

Per block: a set of unmarked non-$\mathcal{C}$-silent states, a set of marked non-$\mathcal{C}$-silent states, a set of unmarked $\mathcal{C}$-silent SCCs, a set of marked $\mathcal{C}$-silent SCCs, and a way to iterate over its incoming and outgoing non-$\mathcal{B}$-inert transitions (possibly through its states and SCCs). The set of potential splitters can be stored as a set or list of pointers to blocks.
In Algorithm 2, some subblocks are marked as completed, and possibly one block is marked as aborted; one can store the non-completed and the completed subblocks in two (circular) list of pointers to blocks, and the aborted block as a (possibly NULL) pointer to a block.

Per state: a flag to indicate whether it is $\mathcal{C}$-silent.

Per non-$\mathcal{C}$-silent state: its block, $P_{\to\text{splitter}}$ and $P_{\text{non-}\mathcal{C}\text{-inert}}$, and a way to iterate over its incoming $\mathcal{B}$-inert transitions.

Per $\mathcal{C}$-silent state: the SCC it belongs to.

Per $\mathcal{C}$-silent SCC: its block, the *untested* counter, the number of states, the number of outgoing $\mathcal{B}$-inert transitions, and a way to iterate over its incoming $\mathcal{B}$-inert transitions. The last three items can be collected and stored while (re)computing the ES-trees in the SCC algorithm without increasing the time complexity.

Per transition: a label to indicate whether it is $\mathcal{C}$-inert.

Proof of Theorem 11. We walk through the algorithms and show under which clause each individual step can be subsumed. *Algorithm 1* mostly falls under Clause 1: the incoming and outgoing transitions of Sp are visited in Lines 1.6–1.22, and every transition is handled in

constant time, except for Lines 1.8 and 1.16. In these lines predecessor states are removed from SCCs. Since a state is initially in at most one SCC, and it can effectively only be removed once, these lines fall under Clause 3. In Lines 1.23–1.25, Algorithm 2 is called.

The first part of *Algorithm 2* runs in time proportional to $\log n$ times the marked states of B. Note that marked states either have a transition to Sp or are in Sp. Therefore, the calls to Algorithm 2 caused by one splitter together lead to at most $|in(Sp)| + |Sp|$ marked states. Hence, those parts of Algorithm 2 that use marked states fall under Clause 1.

At Line 2.1, we separate the non-$\mathcal{C}$-silent states. This boils down to sorting the marked states of B according to the conditional probability $P_{\to\text{splitter}}(s)/P_{\text{non-}\mathcal{C}\text{-inert}}(s)$, which can be done in time $O(|Marked(B)| \log n)$. Note that using techniques as in [7, 20, 12] the factor $\log n$ can be saved, but this will not reduce the overall complexity of our algorithm. The subblock B_p with $p = 0$ requires special care, as it receives the unmarked non-$\mathcal{C}$-silent states of $B \neq Sp$. If $p > 0$, states moved to B_p have transitions to the small splitter Sp, and we are guaranteed to move only a small number of states to B_p. This does not hold for B_0; there may be too many non-$\mathcal{C}'$-silent states that have no transition to the splitter. In our proposed data structure, B_0 can be initialized from the unmarked non-$\mathcal{C}$-silent states, which are stored in a separate subset of B.

Unmarking states and related operations at Lines 2.2–2.3 take $O(|Marked(B)|)$.

The loop at Lines 2.5–2.10 falls under Clause 2. We interpret "Take one step" as "execute one loop iteration". In *Algorithms 3 and 4*, this takes constant time. The work in these algorithms is proportional to the number of states and incoming transitions of the respective subblock. The only part that is not trivial is Line 3.22, which can be executed in time proportional to the incoming transitions of the (clearly small) subblock B_p, as every marked SCC in B_p has a transition to the unmarked states of B_p.

In most cases, executing one loop iteration means that one state or transition is handled, or alternatively, that Algorithm 4 confirms that nothing needs to be done right now. In this way, we ensure that at most one more step is handled in the largest subblock than in the *second* largest subblock. The overhead caused by this is at most proportional to the work done for the second largest subblock (including the residue) and can be attributed to this.

In Clauses 1 and 2 it is indicated that each state is handled at most $\lfloor \log_2 n \rfloor$ times. This is due to the fact that each state when it is processed again is in a block of at most half the size of the previous block it belonged to, according the the principle "Process the smaller half". When we sum the complexity up over all states in Clauses 1 and 2 we obtain the complexity $O((m + n) \log^2 n)$. Together with the complexity of Clause 3, we obtain the overall expected time complexity $O(m \log^4 n)$, as $n \leq m$. ◀

In Theorem 11 we did not include the time complexity of the initialisation of the algorithm. However, if we assume that AP is small enough to find the initial partition fast, namely $|AP| \in O(m/n \log^4 n)$, then we can find an initial partition within the desired overall time complexity. In Line 1.4 the SCCs are constructed, which also fits within the time assigned to Clause 3 of Theorem 11.

5 Conclusion and Outlook

The combination of the ideas from the algorithms from [10, 14] on the one hand and [4] on the other hand allow to come up with a near-linear expected-time algorithm for weak bisimilarity on Markov chains. There are many equivalences that have the same structure, namely direct steps between equivalence classes as in branching bisimilarity and irreducible strongly connected components of inert steps in these equivalence classes. Examples are orthogonal

bisimilarity [21], governed stuttering bisimilarity [6], various branching bisimilarities on non-deterministic probabilistic systems [3, 19] and equivalences using other forms of weighted transitions [17]. We expect that all these equivalences can be provided with a near-linear expected-time algorithm using the techniques provided in this paper.

References

1 Christel Baier and Holger Hermanns. Weak bisimulation for fully probabilistic processes. In Orna Grumberg, editor, *Computer aided verification: CAV*, volume 1254 of *LNCS*, pages 119–130. Springer, Berlin, 1997. doi:10.1007/3-540-63166-6_14.

2 Christel Baier and Holger Hermanns. Weak bisimulation for fully probabilistic processes. Technical Report TR-CTIT-12, Center for Telematics and Information Technology, Enschede, The Netherlands, 1999. URL: https://research.utwente.nl/files/26732523/00000015.pdf.

3 Christel Baier, Joost-Pieter Katoen, Holger Hermanns, and Verena Wolf. Comparative branching-time semantics for Markov chains. *Information and computation*, 200(2):149–214, 2005. doi:10.1016/j.ic.2005.03.001.

4 Aaron Bernstein, Maximilian Probst, and Christian Wulff-Nilsen. Decremental strongly-connected components and single-source reachability in near-linear time. In Moses Charikar and Edith Cohen, editors, *STOC'19: Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*, pages 365–376. ACM, New York, 2019. doi:10.1145/3313276.3316335.

5 M. C. Browne, E. M. Clarke, and O. Grümberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical computer science*, 59(1–2):115–131, 1988. doi:10.1016/0304-3975(88)90098-9.

6 Sjoerd Cranen, Jeroen J.A. Keiren, and Tim A.C. Willemse. A cure for stuttering parity games. In Abhik Roychoudhury and Meenakshi D'Souza, editors, *Theoretical aspects of computing – ICTAC 2012*, volume 7521 of *LNCS*, pages 198–212. Springer, Heidelberg, 2012. doi:10.1007/978-3-642-32943-2_16.

7 Salem Derisavi, Holger Hermanns, and William H. Sanders. Optimal state-space lumping in Markov chains. *Information processing letters*, 87(6):309–315, 2003. doi:10.1016/S0020-0190(03)00343-0.

8 Shimon Even and Yossi Shiloach. An on-line edge-deletion problem. *Journal of the ACM*, 28(1):1–4, 1981. doi:10.1145/322234.322235.

9 Rob J. van Glabbeek and Peter W. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996. doi:10.1145/233551.233556.

10 Jan Friso Groote, David N. Jansen, Jeroen J. A. Keiren, and Anton Wijs. An $O(m \log n)$ algorithm for computing stuttering equivalence and branching bisimulation. *ACM transactions on computational logic*, 18(2):Article 13, 2017. doi:10.1145/3060140.

11 Jan Friso Groote and Frits Vaandrager. An efficient algorithm for branching bisimulation and stuttering equivalence. In M. S. Paterson, editor, *Automata, languages and programming [ICALP]*, volume 443 of *LNCS*, pages 626–638. Springer, Berlin, 1990. doi:10.1007/BFb0032063.

12 Jan Friso Groote, Jao Rivera Verduzco, and Erik P. de Vink. An efficient algorithm to determine probabilistic bisimulation. *Algorithms*, 11(9):131, 2018. doi:10.3390/a11090131.

13 John Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Zvi Kohavi and Azaria Paz, editors, *Theory of machines and computations*, pages 189–196. Academic Press, New York, 1971. doi:10.1016/B978-0-12-417750-5.50022-1.

14 David N. Jansen, Jan Friso Groote, Jeroen J. A. Keiren, and Anton Wijs. An $O(m \log n)$ algorithm for branching bisimilarity on labelled transition systems. In Armin Biere and David Parker, editors, *Tools and algorithms for the construction and analysis of systems: TACAS*, volume 12079 of *LNCS*, pages 3–20. Springer, Cham, 2020. doi:10.1007/978-3-030-45237-7_1.

15 David N. Jansen, Lei Song, and Lijun Zhang. Revisiting weak simulation for substochastic Markov chains. In Kaustubh Joshi, Markus Siegle, Mariëlle Stoelinga, and Pedro D'Argenio, editors, *Quantitative evaluation of systems: QEST*, volume 8054 of *LNCS*, pages 209–224. Springer, Heidelberg, 2013. doi:10.1007/978-3-642-40196-1_18.

16 John G. Kemeny, J. Laurie Snell, and Anthony W. Knapp. *Denumerable Markov chains*. Van Nostrand, Princeton, NJ, 1966.

17 Marino Miculan and Marco Peressotti. Deciding weak weighted bisimulation. In Dario Della Monica, Aniello Murano, Sasha Rubin, and Luigi Sauro, editors, *ICTCS 2017 and CILC 2017: Italian conference on theoretical computer science and Italian conference on computational logic*, volume 1949 of *CEUR Workshop Proceedings*, pages 126–137. CEUR-WS.org, 2017. URL: `http://ceur-ws.org/Vol-1949/ICTCSpaper11.pdf`.

18 Robin Milner. *A calculus of communicating systems*, volume 92 of *LNCS*. Springer, Berlin, 1980. `doi:10.1007/3-540-10235-3`.

19 Andrea Turrini and Holger Hermanns. Polynomial time decision algorithms for probabilistic automata. *Information and computation*, 244:134–171, 2015. `doi:10.1016/j.ic.2015.07.004`.

20 Antti Valmari and Giuliana Franceschinis. Simple $O(m \log n)$ time Markov chain lumping. In Javier Esparza and Rupak Majumdar, editors, *Tools and algorithms for the construction and analysis of systems: TACAS*, volume 6015 of *LNCS*, pages 38–52. Springer, Berlin, 2010. `doi:10.1007/978-3-642-12002-2_4`.

21 Thuy Duong Vu. Deciding orthogonal bisimulation. *Formal aspects of computing*, 19(4):475–485, 2007. `doi:10.1007/s00165-007-0023-x`.

A Proof of Proposition 4

This appendix compares the three notions of bisimilarity. First let's recall the definition of the three bisimulation relations:

▶ **Definition 3.** *Let $\mathcal{M} = (S, AP, \mathbf{P}, L)$ be a DTMC and $R \subseteq \equiv_{AP}$ an equivalence relation on S (which respects the atomic propositions of states). We say that R is*

a weak bisimulation *iff $s\ R\ t$ implies, for all R-equivalence classes $C \in S/R$ with $s, t \notin C$, that $Pr(s, [s]_{AP}, C) = Pr(t, [t]_{AP}, C)$.*

a branching bisimulation *iff $s\ R\ t$ implies, for all R-equivalence classes $C \in S/R$ with $s, t \notin C$, that $Pr(s, [s]_R, C) = Pr(t, [t]_R, C)$.*

a conditional-probability bisimulation *iff $s\ R\ t$ implies*

1. *If s and t are both non-R-silent, for all R-equivalence classes $C \in S/R$ with $s, t \notin C$, we have $\mathbf{P}(s, C \mid \text{non-}R\text{-inert}) = \mathbf{P}(t, C \mid \text{non-}R\text{-inert})$; and*
2. *s has a path to a state outside $[s]_R$ iff t has a path to a state outside $[t]_R$.*

The states s and t are weakly bisimilar, *denoted $s \approx_{\mathrm{w}} t$, iff a weak bisimulation R exists such that $s\ R\ t$. Similarly, s and t are* branching bisimilar, *denoted $s \approx_{\mathrm{b}} t$, iff a branching bisimulation R exists such that $s\ R\ t$. Finally, s and t are* conditional-probability-bisimilar, *denoted $s \approx_{\mathrm{c}} t$, iff a conditional-probability bisimulation R exists such that $s\ R\ t$.*

▶ **Proposition 4.** *Let $\mathcal{M} = (S, AP, \mathbf{P}, L)$ be a DTMC. States in S are weakly bisimilar iff they are branching bisimilar iff they are conditional-probability-bisimilar.*

We prove this proposition by the three lemmas below. The proofs are adapted from [2], where Markov chains are defined with action labels instead of atomic propositions.

▶ **Lemma 12.** *Let R be a conditional-probability bisimulation. Then R is a branching bisimulation.*

Proof. Consider a state $s \in S$. If all states in $[s]_R$ are R-silent, they are all trivially branching bisimilar to s, as $Pr(s, [s]_R, C) = 0$ for all $C \in S/R \backslash \{[s]_R\}$. Otherwise, $Pr(s, [s]_R, S \backslash [s]_R) = 1$ because S is finite. Fix some non-R-silent $s' \in [s]_R$. Note that for any $C \in S/R \setminus \{[s]_R\}$, we have that $\mathbf{P}(s', C \mid \text{non-}R\text{-inert})$ does not depend on the concrete choice of s'. Then,

$$
\begin{aligned}
Pr(s,[s]_R,C) &= \sum_{s_1,\ldots,s_n\in[s]_R, s_{n+1}\in C} Pr_{\delta(s)}(Cyl(s,s_1,\ldots,s_n,s_{n+1})) \\
&= \sum_{s_1,\ldots,s_n\in[s]_R} Pr_{\delta(s)}(Cyl(s,s_1,\ldots,s_n))\mathbf{P}(s_n,C) \\
&= \sum_{s_1,\ldots,s_n\in[s]_R} Pr_{\delta(s)}(Cyl(s,s_1,\ldots,s_n))\mathbf{P}(s_n,C \mid \text{non-}R\text{-inert})(1-\mathbf{P}(s_n,[s]_R)) \\
&= \mathbf{P}(s',C \mid \text{non-}R\text{-inert}) \sum_{s_1,\ldots,s_n\in[s]_R} Pr_{\delta(s)}(Cyl(s,s_1,\ldots,s_n))\mathbf{P}(s_n,S\setminus[s]_R) \\
&= \mathbf{P}(s',C \mid \text{non-}R\text{-inert}) \sum_{s_1,\ldots,s_n\in[s]_R, s_{n+1}\notin[s]_R} Pr_{\delta(s)}(Cyl(s,s_1,\ldots,s_n,s_{n+1})) \\
&= \mathbf{P}(s',C \mid \text{non-}R\text{-inert})Pr(s,[s]_R,S\setminus[s]_R) \\
&= \mathbf{P}(s',C \mid \text{non-}R\text{-inert})
\end{aligned}
$$

By the same method $Pr(t,[s]_R,C)=\mathbf{P}(s',C \mid \text{non-}R\text{-inert})$ for any $t\in[s]_R$, so $Pr(s,[s]_R,C)=Pr(t,[t]_R,C)$ whenever $s\ R\ t$. ◀

▶ **Lemma 13.** *Let R be a branching bisimulation. Then R is a weak bisimulation.*

Proof. We first define the auxiliary notion of a block-cylinder set: $Cyl(B_1,B_2,\ldots,B_n)$ is the set of paths that start in B_1, after visiting a number of states in B_1 move on to B_2, and visit all further blocks in the order mentioned. After entering B_n the paths may continue without any restrictions. A block-cylinder set is a countable union of (basic) cylinder sets. Hence, it is in the σ-algebra generated by the basic cylinder sets allowing to write $Pr_{\delta(s)}(Cyl([s]_R,B_1,B_2,\ldots,B_n,C))$. We can therefore define:

$$Pr(s,[s]_R,B_1,B_2,\ldots,B_n,C) := Pr_{\delta(s)}(Cyl([s]_R,B_1,B_2,\ldots,B_n,C)).$$

Further, we have, for all states $s\in S\setminus C$,

$$Pr(s,[s]_{AP},C) = \sum_{\substack{B_1,B_2,\ldots,B_n\in S/R \\ B_1,B_2,\ldots,B_n\subseteq[s]_{AP}\setminus C \\ [s]_R\neq B_1\neq B_2\neq\cdots\neq B_n}} Pr(s,[s]_R,B_1,B_2,\ldots,B_n,C).$$

Now assume given two branching bisimilar states $s\ R\ t$. Note that $[s]_{AP}=[t]_{AP}$. It is easy to show that $Pr(s,[s]_R,B_1,B_2,\ldots,B_n,C)=Pr(t,[t]_R,B_1,B_2,\ldots,B_n,C)$ for blocks $B_i\in S/R$ for $i=1,\ldots,n$, and therefore $Pr(s,[s]_{AP},C)=Pr(t,[t]_{AP},C)$. ◀

▶ **Lemma 14.** *Weak bisimilarity, $\approx_{\mathrm{w}}$, is a conditional-probability bisimulation.*

Proof. We adapt the proof of [1] (for action-labelled fully probabilistic systems) to our state-labelled DTMCs. Condition 2 of conditional-probability bisimulation is easy to check, so we concentrate on Condition 1.

Let $B_1,B_2,\ldots,B_k$ be an enumeration of the $\approx_{\mathrm{w}}$-equivalence classes with at least one non-$\approx_{\mathrm{w}}$-silent state, and let $B_0,B_{-1},\ldots,B_{k_{\mathrm{sil}}}$ be an enumeration of the other $\approx_{\mathrm{w}}$-equivalence classes, in which all states are $\approx_{\mathrm{w}}$-silent. Let s be a non-$\approx_{\mathrm{w}}$-silent state. Assume w.l.o.g. that $[s]_{\approx_{\mathrm{w}}}=B_k$. Let $C\neq[s]_{\approx_{\mathrm{w}}}$ be an $\approx_{\mathrm{w}}$-equivalence class. Then

$$Pr(s,[s]_{AP},C) = \sum_{i=k_{\mathrm{sil}}}^{k} \mathbf{P}(s,B_i)Pr(B_i,[s]_{AP},C).$$

As $\approx_{\mathrm{w}}$ is a weak bisimulation, $Pr(B_i, [s]_{AP}, C)$ does not depend on the concrete initial state in B_i if $B_i \neq C$, and $Pr(C, [s]_{AP}, C) = 1$. Therefore the above sum is well-defined. Note that $Pr(B_i, [s]_{AP}, C) = 0$ if $C \neq B_i \not\subseteq [s]_{AP}$. Then

$$\begin{aligned}\frac{Pr(s, [s]_{AP}, C)}{1 - \mathbf{P}(s, [s]_{\approx_{\mathrm{w}}})} &= \sum_{i=k_{\mathrm{sil}}}^{k} \frac{\mathbf{P}(s, B_i)}{1 - \mathbf{P}(s, [s]_{\approx_{\mathrm{w}}})} Pr(B_i, [s]_{AP}, C) \\ &= \sum_{i=k_{\mathrm{sil}}}^{k-1} \frac{\mathbf{P}(s, B_i)}{1 - \mathbf{P}(s, [s]_{\approx_{\mathrm{w}}})} Pr(B_i, [s]_{AP}, C) \quad + \frac{\mathbf{P}(s, B_k)}{1 - \mathbf{P}(s, [s]_{\approx_{\mathrm{w}}})} Pr(B_k, [s]_{AP}, C) \\ &= \sum_{i=k_{\mathrm{sil}}}^{k-1} \frac{\mathbf{P}(s, B_i)}{1 - \mathbf{P}(s, [s]_{\approx_{\mathrm{w}}})} Pr(B_i, [s]_{AP}, C) \quad + \frac{\mathbf{P}(s, [s]_{\approx_{\mathrm{w}}})}{1 - \mathbf{P}(s, [s]_{\approx_{\mathrm{w}}})} Pr(s, [s]_{AP}, C).\end{aligned}$$

We now solve the above equation for $Pr(s, [s]_{AP}, C)$ and get:

$$Pr(s, [s]_{AP}, C) = \sum_{i=k_{\mathrm{sil}}}^{k-1} \frac{\mathbf{P}(s, B_i)}{1 - \mathbf{P}(s, [s]_{\approx_{\mathrm{w}}})} Pr(B_i, [s]_{AP}, C).$$

This shows that the conditional probabilities $x_i = \mathbf{P}(s, B_i \mid \text{non-}\approx_{\mathrm{w}}\text{-inert}) = \frac{\mathbf{P}(s,B_i)}{1-\mathbf{P}(s,[s]_{\approx_{\mathrm{w}}})}$ for $i \in \{k_{\mathrm{sil}}, \ldots, k-1\}$ solve the following linear equation system:

$$\sum_{i=k_{\mathrm{sil}}}^{k-1} x_i Pr(B_i, [s]_{AP}, C) = Pr([s]_{\approx_{\mathrm{w}}}, [s]_{AP}, C) \quad \text{for all } C \in \{B_{k_{\mathrm{sil}}}, \ldots, B_{k-1}\} \tag{1}$$

We claim that Equation System (1) has at most one solution. Therefore, for each non-$\approx_{\mathrm{w}}$-silent state $t \in [s]_{\approx_{\mathrm{w}}}$, we have $\mathbf{P}(t, B_i \mid \text{non-}\approx_{\mathrm{w}}\text{-inert}) = \mathbf{P}(s, B_i \mid \text{non-}\approx_{\mathrm{w}}\text{-inert})$, which proves this lemma.

We now prove the claim that Equation System (1) has at most one solution. We first prove that the matrix $\mathbf{A} = (Pr(B_j, [s]_{AP}, B_i))_{i,j \in \{1,\ldots,k\}}$ is regular. For every B_i (with $i \geq 1$), fix some state $s_i \in B_i$ that is non-$\approx_{\mathrm{w}}$-silent. Let

$$\mathbf{C} = (c_{ij})_{i,j \in \{1,\ldots,k\}} \quad \text{where } c_{ij} = \begin{cases} \mathbf{P}(s_j, B_i) & \text{if } s_j \in [s]_{AP} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{E} = \begin{pmatrix} 1 - e_1 & & 0 \\ & \ddots & \\ 0 & & 1 - e_k \end{pmatrix} \quad \text{where } e_i = \begin{cases} \sum_{h=1}^{k} \mathbf{P}(s_i, B_h) Pr(B_h, [s]_{AP}, B_i) & \text{if } s_i \in [s]_{AP} \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathbf{D} = (d_{ij})_{i,j \in \{1,\ldots,k\}} = \mathbf{A} \cdot \mathbf{C} + \mathbf{E}$. We show that $\mathbf{D} = \mathbf{A}$. For the diagonal elements d_{ii} with $s_i \in [s]_{AP}$, we have:

$$d_{ii} = \sum_{h=1}^{k} Pr(B_h, [s]_{AP}, B_i) \mathbf{P}(s_i, B_h) + 1 - e_i = 1 = Pr(B_i, [s]_{AP}, B_i)$$

and for the off-diagonal elements d_{ij} with $s_j \in [s]_{AP}$, we have:

$$Pr(B_j, [s]_{AP}, B_i) = Pr(s_j, [s]_{AP}, B_i) = \sum_{h=1}^{k} Pr(B_h, [s]_{AP}, B_i) \mathbf{P}(s_j, B_h) = d_{ij}$$

Elements d_{ij} with $s_j \notin [s]_{AP}$ are trivial.

This proves that $\mathbf{A} \cdot \mathbf{C} + \mathbf{E} = \mathbf{A}$, so also $\mathbf{E} = \mathbf{A} \cdot (\mathbf{I} - \mathbf{C})$. Next we show that all $e_i < 1$. This is trivial if $s_i \notin [s]_{AP}$, so we now assume that $B_i \subseteq [s]_{AP}$.

- Assume that there is some $B_{h_0} \neq B_i$ with $Pr(B_{h_0}, [s]_{AP}, B_i) \neq 0$. Note that if some state s' satisfies $Pr(s', [s]_{AP}, B_i) = 1$ then $s' \in B_i$ (here we use that we started with $\approx_{\mathrm{w}}$ and not just any weak bisimulation); therefore, $Pr(B_{h_0}, [s]_{AP}, B_i) < 1$. Then

$$e_i \leq \sum_{\substack{h=1 \\ h \neq h_0}}^{k} \mathbf{P}(s_i, B_h) + \mathbf{P}(s_i, B_{h_0}) Pr(B_{h_0}, [s]_{AP}, B_i) < \mathbf{P}(s_i, S) \leq 1.$$

- If for all $B_i \neq B_j$ we have $Pr(B_j, [s]_{AP}, B_i) = 0$, then

$$e_i = \mathbf{P}(s_i, B_i) Pr(B_i, [s]_{AP}, B_i) = \mathbf{P}(s_i, B_i) < 1.$$

Therefore, $\mathbf{E}$ is regular, and hence $\mathbf{A}$ is regular because $\mathbf{A}^{-1} = (\mathbf{I} - \mathbf{C}) \cdot \mathbf{E}^{-1}$.

Now we prove that the extended matrix $\mathbf{A}^0 = (Pr(B_j, [s]_{AP}, B_i))_{i,j \in \{k_{\mathrm{sil}}, \ldots, k\}}$ is regular. Note that if $i \neq j \leq 0$, then $Pr(B_j, [s]_{AP}, B_i) = 0$. So, we can write $\mathbf{A}^0$ as follows

$$\mathbf{A}^0 = \begin{pmatrix} \mathbf{I} & * \\ 0 & \mathbf{A} \end{pmatrix}.$$

Hence, $\mathbf{A}^0$ is regular as well. The matrix of Equation System (1) is the matrix $\mathbf{A}^0$ without the last column and row, i.e., $(Pr(B_j, [s]_{AP}, B_i))_{i,j \in \{k_{\mathrm{sil}}, \ldots, k-1\}}$. Let $\mathbf{A'}$ be $\mathbf{A}^0$ without the last row, and consider the equation system $\mathbf{A'x} = \mathbf{b}$, where $\mathbf{b}_i = Pr([s]_{\approx_{\mathrm{w}}}, [s]_{AP}, B_i)$ for $i = k_{\mathrm{sil}}, \ldots, k-1$. If some $\mathbf{x}$ solves this equation system, then $(\mathbf{x}_i)_{i=k_{\mathrm{sil}}, \ldots, k-1}$ solves (1) iff $\mathbf{x}_k = 0$. We argue that there is at most one such $\mathbf{x}$.

As $\mathbf{A}^0$ is regular, the solution space of $\mathbf{A'x} = \mathbf{b}$ is (at most) one-dimensional and can therefore be described by $\{\mathbf{a} + t \cdot \mathbf{c} \mid t \in \mathbb{R}\}$ for some vectors $\mathbf{a}, \mathbf{c} \in \mathbb{R}^{k - k_{\mathrm{sil}} + 1}$.

To come to a contradiction, we assume $\mathbf{a}_k = \mathbf{c}_k = 0$ and $\mathbf{c} \neq \mathbf{0}$; in that case, there would be more than one solution of (1). We know $\mathbf{A'}(\mathbf{a} + t\mathbf{c}) = \mathbf{b}$ for all $t \in \mathbb{R}$, so

$$\sum_{h=k_{\mathrm{sil}}}^{k-1} Pr(B_h, [s]_{AP}, B_i)(\mathbf{a}_h + t\mathbf{c}_h) = Pr([s]_{\approx_{\mathrm{w}}}, [s]_{AP}, B_i) \quad \text{for } i = k_{\mathrm{sil}}, \ldots, k-1.$$

Because this holds for all $t \in \mathbb{R}$, we must have that

$$\sum_{h=k_{\mathrm{sil}}}^{k-1} Pr(B_h, [s]_{AP}, B_i)\mathbf{c}_h = 0 \quad \text{for } i = k_{\mathrm{sil}}, \ldots, k-1.$$

Because $\mathbf{A}^0$ is regular, we cannot have $\mathbf{A}^0\mathbf{c} = \mathbf{0}$, therefore

$$c := \sum_{h=1}^{k-1} Pr(B_h, [s]_{AP}, B_k)\mathbf{c}_h \neq 0.$$

By these equations, the vector $(\frac{1}{c}\mathbf{c}_i)_{i=1,\ldots,k}$ is equal to the last row of $\mathbf{A}^{-1}$. We have $0 = \frac{1}{c}\mathbf{c}_k = (\mathbf{A}^{-1})_{kk} = (\mathbf{I} - \mathbf{C})_{kk}/(1 - e_k) = (1 - \mathbf{P}(s_k, B_k))/(1 - e_k) \neq 0$. Contradiction! ◀