AVACS – Automatic Verification and Analysis of Complex Systems

REPORTS

of SFB/TR 14 AVACS

Editors: Board of SFB/TR 14 AVACS

Towards Symbolic Stochastic Aggregation

by

Ralf Wimmer Holger Hermanns Marc Herbstritt

Bernd Becker

AVACS Technical Report No. 16

August 2007

ISSN: 1860-9821

Publisher: Sonderforschungsbereich/Transregio 14 AVACS
(Automatic Verification and Analysis of Complex Systems)
Editors: Bernd Becker, Werner Damm, Martin Fränzle, Ernst-Rüdiger Olderog,
Andreas Podelski, Reinhard Wilhelm
ATRs (AVACS Technical Reports) are freely downloadable from `www.avacs.org`

Author(s) contact: Ralf Wimmer (`wimmer@informatik.uni-freiburg.de`).

Towards Symbolic Stochastic Aggregation*

Ralf Wimmer[1], Holger Hermanns[2], Marc Herbstritt[1], and Bernd Becker[1]

[1] Albert-Ludwigs-University, Freiburg im Breisgau, Germany
{wimmer|herbstri|becker}@informatik.uni-freiburg.de
[2] Saarland University, Saarbrücken, Germany
hermanns@cs.uni-sb.de

Abstract. Bisimulations are one of the classical means to fight the state explosion problem. Especially in combination with symbolic methods, like BDD-based data representation and algorithms that exploit the compact BDD representation, they enable the minimization of very large state spaces without losing relevant properties.
In this report, we show how a symbolic algorithm, that was originally developed for non-stochastic systems, can be extended to compute strong and branching bisimulations on interactive Markov chains (IMCs). An IMC is a very general model that combines the stochastic behavior of traditional Markov chains with action-labeled transition systems. To the best of our knowledge, our suggested algorithm is the first symbolic algorithm for both stochastic *strong* bisimulation and stochastic *branching* bisimulation on IMCs.

1 Introduction

Two well-known modelling formalisms in system analysis are labeled transition systems (LTSs) and continuous-time Markov chains (CTMCs). The former represents the interaction of a system with its environment, the latter its stochastic behavior. A natural combination of the two are interactive Markov chains (IMCs), introduced by Hermanns in [1]. They are also a generalization of continuous-time Markov decision processes (CTMDPs) in the sense that an IMC is a CTMDP when interactive transitions and Markov transitions can only be taken alternately. A large class of systems can therefore be modelled with IMCs.

As it is the case for all state-based models, state space explosion is also an issue of IMCs. Realistic systems often consist of billions of states and are therefore hardly manageable by explicit state algorithms. One method to reduce the state space without losing relevant properties is bisimulation minimization. This is a well-known method for LTSs and Markov chains—in the latter case also known as lumping [2, 3].

* This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS). See www.avacs.org for more information.

While bisimulations on Markov chains are naturally restricted to strong bisimulation [1], there is a great number of different variants for labeled transition systems with so-called unobservable actions (marked with the special label "τ"). They represent internal behavior of the system that is not observable from the outside. The different notions of bisimulation then differ in what is meant by "unobservable": Is the execution of τ-steps observable, but the corresponding number of steps is unknown? Can the user only indirectly see τ-steps when a transition is taken to an inequivalent state? Or is it completely impossible to observe τ-steps?

Because IMCs may also exhibit unobservable transitions one can define different kinds of bisimulations on IMCs. Hermanns introduced stochastic strong and stochastic weak bisimulation for IMCs in [1]. Another important variant, namely stochastic branching bisimulation, first appeared in [4].

For labeled transition systems symbolic algorithms (i.e., in our case algorithms using BDDs for the representation of the systems and the computation of the bisimulations) are very successful in handling huge state spaces. But to the best of our knowledge there exists no *symbolic* algorithm that computes bisimulations on IMCs.

In this report we will present such a symbolic algorithm for the computation of stochastic strong and branching bisimulation, we will prove its correctness, and we will describe in detail concepts for its symbolic implementation.

1.1 Related Work

Bisimulations on labeled transition systems have quite a long history. Strong bisimulation goes back to the early 1980s. It was introduced independently by Milner in 1980 [5] and by Park in 1981 [6]. Branching bisimulation was defined a few years later by van Glabbeek and Weijland [7].

For both types of bisimulation there are explicit as well as symbolic algorithms available: for strong bisimulation, Paige and Tarjan developed an explicit algorithm with optimal complexity [8], symbolic algorithms for non-stochastic strong bisimulation are presented for example in [9–11].

An explicit algorithm for branching bisimulation was developed by Groote and Vandrager [12]. The symbolic algorithm by Bouali and de Simone [9] can be extended to branching and weak bisimulation easily. Signature-based computation was first used by Blom and Orzan [13–15] in a distributed environment to compute strong and branching bisimulation, thereby representing all the data explicitly. In [16] we have shown how this algorithm can be modified to use a symbolic representation of the data such that it can handle much larger systems. It was generalized in [17] such that the signature-based algorithm is able to compute not only strong and branching bisimulation but eight of the most relevant types of bisimulation within a uniform framework that itself is easily extendible to further types. Further optimizations of the algorithm are presented in [18].

For Markov chains there is also a number of publications about bisimulation computation (or lumping): Markov chain lumping first appeared in [19]. Explicit

lumping algorithms are for example described in [3, 20, 21]. The first symbolic algorithm for Markov chain lumping was developed by Derisavi in [22]. A variant of this algorithm, that is highly influenced by our work on signature-based bisimulation computation for LTSs [17], will appear in [23].

Stochastic strong and weak bisimulation for IMCs were proposed by Hermanns in [1], along with explicit algorithms for their computation. They are implemented in the minimization tool BcgMin [24] that is part of the Cadp tool box [25, 26]. Stochastic branching bisimulation was first used in [4]. It is also implemented in BcgMin using an extension of the explicit algorithm of Groote and Vaandrager [12]. We are not aware of any symbolic algorithm that computes stochastic strong and branching bisimulations on IMCs.

This report is structured as follows: First, in Section 2 we introduce definitions and notations that will be used throughout the report. In Section 3, we will present our novel signature-based algorithm for stochastic strong and branching bisimulation on IMCs. Section 3 also contains the correctness proof of the algorithm. In the following section, we describe in detail concepts for a symbolic implementation that exploits the compact BDD representation of the IMC. We conclude the report in Section 5 and additionally point out tasks for future work.

2 Background

Interactive Markov chains (IMCs), for which we will compute bisimulation relations in the following, are a generalization of labeled transition systems (LTSs) and continuous-time Markov chains (CTMCs). They unite interactive and stochastic transitions. In contrast to Markov decision processes, the interactive and stochastic transitions do not need to be taken alternately, but may be executed in any order.

Definition 1. *An **Interactive Markov Chain** (IMC) M is a quadruple $M = (S, A, T, R)$, where*

- *S is a nonempty set of states,*
- *A is a finite set of actions, including the so-called unobservable action τ, i.e.$\tau \in$,*
- *$T \subseteq S \times A \times S$ is a set of interactive transitions, and*
- *$R : S \times S \to \mathbb{R}^{\geq 0}$ is a matrix of Markov transition rates.*

An IMC $M = (S, A, T, R)$ is isomorphic to a CTMC if $T = \emptyset$ and to an LTS if $R(s, s') = 0$ for all $s, s' \in S$. Analogously to LTSs, IMCs can exhibit so-called unobservable (or internal) transitions. As usual we assume that these transitions are labeled with τ.

We define a (nonnegative) real-valued function $\gamma_M : S \times 2^S \to \mathbb{R}^{\geq 0}$, that calculates the cumulative rate to reach a set of states C from a single state s:

$$\gamma_M(s, C) = \sum_{s' \in C} R(s, s').$$

Like in continuous-time Markov chains, the Markov transitions are governed by an exponential distribution: The probability that a Markov transition is enabled within t time units is given by $1 - e^{-R(s,s')\cdot t}$. If a state has several out-going Markov transitions, a race condition exists. The first transition that is enabled is taken. The overall probability to take the Markov transition from s to s' within t time units is then

$$\frac{R(s,s')}{\gamma_M(s,S)} \cdot \left(1 - e^{-\gamma_M(s,S)\cdot t}\right)$$

if no interactive transition is taken.

For the sake of simplicity of notation we introduce some abbreviations for an IMC $M = (S, A, T, R)$ and states $s, t, x \in S$:

- $s \xrightarrow{a} t$ for $(s, a, t) \in T$,
- $s \overset{\tau}{\not\to}$ for $\nexists s' \in S : s \xrightarrow{\tau} s'$
- $s \xrightarrow{a^*} t$ for the reflexive transitive closure of $T(\cdot, a, \cdot)$, i. e., the smallest set T^{a^*} with $(s, a, s) \in T^{a^*}$, $T(\cdot, a, \cdot) \subseteq T^{a^*}$ and $(s, a, x) \in T^{a^*} \wedge (x, a, t) \in T^{a^*} \Rightarrow (s, a, t) \in T^{a^*}$.
- $s \xrightarrow{a^+} t$ for the non-reflexive transitive closure of $T(\cdot, a, \cdot)$, i. e. $s \xrightarrow{a^+} t$ iff $s' \in S$ exists with $s \xrightarrow{a} s' \xrightarrow{a^*} t$.

On the one hand, the probability that a Markovian transition is enabled instantaneously is $1 - e^{-\lambda \cdot 0} = 0$. On the other hand, an internal transition (labeled with τ) may happen instantaneously because nothing may prevent or delay it. This observation justifies to assume that an IMC that may perform an internal action is not allowed to let time pass. This assumption is normally called the *maximal progress assumption*. For more details on maximal progress the reader is referred to [1].

Definition 2. *A state $s \in S$ is said to be **maximal-progress-cut** if $s \xrightarrow{\tau}$ implies $\gamma_M(s, S) = 0$. An IMC $M = (S, A, T, R)$ is maximal-progress-cut, if each state $s \in S$ is maximal-progress-cut.*

A consequence of the maximal progress assumption is that we may remove the Markovian transitions from those states that have an out-going τ-transition. For an IMC M, we denote by cut(M) the IMC obtained by cutting Markov transitions from all states s satisfying $s \xrightarrow{\tau}$. cut(M) is obviously maximal-progress-cut.

In the same way as on labeled transition systems and Markov chains, one can define equivalence relations on the state space of an IMC. Since IMCs can contain transitions labeled with τ, there are also different kinds of bisimulation for IMCs. Among the most important ones are stochastic strong bisimulation and stochastic branching bisimulation.

Later, we will represent the equivalence relations using partitions. Hence, we now define the notion of a partition more formally:

Definition 3. *Let S be a set. A **partition** P of S is a subset $P \subseteq 2^S$ such that*

$$\bigcup_{C \in P} C = S \qquad \text{and} \qquad \forall C, C' \in P : (C = C' \vee C \cap C' = \emptyset).$$

*The elements of P are called classes or blocks. A partition P is **coarser** than a partition Q (denoted $P \sqsubseteq Q$) if $\forall C \in P \exists C' \in Q : C \subseteq C'$.*

For an IMC $M = (S, A, T, R)$ and a partition P of S, we will use following notations:

- $s \equiv_P t$ if there is a block $C \in P$ such that $s \in C$ and $t \in C$. Obviously $\equiv_P$ is an equivalence relation.
- $s \xrightarrow[P]{a} t$ if $s \xrightarrow{a} t$ and $s \equiv_P t$,
- $s \xrightarrow[P]{a^*} t$ if $s \xrightarrow{a^*} t$ and $s \equiv_P t$. Such a transition sequence whose first and last state are contained in the same block is called **inert**.
- $P(s) := \{t \in S \mid s \equiv_P t\}$ denotes the block of P that contains s.

2.1 Stochastic Strong Bisimulation

Definition 4. *An equivalence relation $B \subseteq S \times S$ is a **stochastic strong bisimulation** iff $s \equiv_B t$ implies for all $a \in A$ and all equivalence classes C of B*

1. *$s \xrightarrow{a} s'$ implies $t \xrightarrow{a} t'$ for a $t' \in S$ with $s' \equiv_B t'$.*
2. *$s \xrightarrow{\tau} \nrightarrow$ implies $\gamma_M(s, C) = \gamma_M(t, C)$.*

Two states s and t are strongly bisimilar ($s \equiv^s t$) if they are contained in some stochastic strong bisimulation. Obviously, $M \equiv^s \text{cut}(M)$.[3]

If $R(s, s') = 0$ for all $s, s' \in T$, stochastic strong bisimulation coincides with non-stochastic strong bisimulation, and if $T = \emptyset$ it coincides with lumping on Markov chains.

Lemma 1. *For a maximal-progress-cut IMC $M = (S, A, T, R)$, $B \subseteq S \times S$ is a stochastic strong bisimulation iff $s \equiv_B t$ implies for all $a \in A$ and all equivalence classes C of B*

1. *$s \xrightarrow{a} s'$ implies $t \xrightarrow{a} t'$ for some $t' \in S$ with $s' \equiv_B t'$.*
2. *$\gamma_M(s, C) = \gamma_M(t, C)$.*

Proof. This lemma follows immediately from the definitions of stochastic strong bisimulation and maximal-progress-cut. □

This characterization of stochastic strong bisimulation will be used later for the computation.

[3] This should be understood in the following way: Construct the disjoint union of the two IMCs resulting in a new IMC M'. M and cut(M) are then strongly bisimilar if there is a stochastic strong bisimulation on M' which relates all corresponding state pairs of M and cut(M).

2.2 Stochastic Branching Bisimulation

Definition 5. *An equivalence relation $B \subseteq S \times S$ is a **stochastic branching bisimulation** iff $s \equiv_B t$ implies for all $a \in A$ and all equivalence classes C of B*

1. $s \xrightarrow{a} s'$ *implies*
   (a) $a = \tau$ *and* $s \equiv_B s'$, *or*
   (b) *there exist* $t', t'' \in S$ *with* $t \xrightarrow[B]{\tau^*} t' \xrightarrow{a} t''$ *and* $s' \equiv_B t''$.
2. $s \overset{\tau}{\nrightarrow}$ *implies*
   (c) $\exists t' \in S : t \xrightarrow[B]{\tau^*} t' \overset{\tau}{\nrightarrow} \wedge \gamma_M(s, C) = \gamma_M(t', C)$.

Two states $s, t \in S$ are stochastic branching bisimilar ($s \equiv^b t$) if they are equivalent w. r. t. some stochastic branching bisimulation. It is easy to see that every stochastic strong bisimulation is also a stochastic branching bisimulation. This implies that M and cut(M) are stochastic branching bisimilar.

We will now give more insight into stochastic branching bisimulation and prove two lemmas which give a more appropriate characterization for computing the bisimulation relation. They go back to an unpublished note [27] by Holger Hermanns.

Lemma 2. *For a maximal-progress-cut IMC $M = (S, A, T, R)$, $B \subseteq S \times S$ is a stochastic branching bisimulation iff $s \equiv_B t$ implies for all $a \in A$ and all equivalence classes C of B*

1. $s \xrightarrow{a} s'$ *implies*
   (a) $a = \tau$ *and* $s \equiv_B s'$, *or*
   (b) *there are* $t', t'' \in S$ *with* $t \xrightarrow[B]{\tau^*} t' \xrightarrow{a} t''$ *and* $s' \equiv_B t''$.
2. $\gamma_M(s, C) > 0$ *implies*
   (c) $\gamma_M(s, C) = \gamma_M(t', C)$ *for some* $t' \in S$ *such that* $t \xrightarrow[B]{\tau^*} t' \overset{\tau}{\nrightarrow}$.
3. $s \overset{\tau}{\nrightarrow}$ *implies that there is a* $t' \in S$ *such that* $t \xrightarrow{\tau^*} t' \overset{\tau}{\nrightarrow}$

Proof. Let B be a stochastic branching bisimulation according to definition 5. To see that it also satisfies the requirements of lemma 2, we briefly discuss why the second condition of lemma 2 is satisfied. The first and third clause are obvious implications of the definition of stochastic branching bisimulation.

For $s \equiv_B t$ let $\gamma_M(s, C) > 0$. Since the IMC is maximal-progress-cut, this implies $s \overset{\tau}{\nrightarrow}$; hence we can use the second clause of definition 5 to conclude the proof of this case.

Now let B be an equivalence relation satisfying lemma 2, and assume $s \equiv_B t$. The requirements of the first condition of definition 5 are obviously fulfilled. For the second condition, assume $s \overset{\tau}{\nrightarrow}$. In this situation, we distinguish the cases $\gamma_M(s, C) > 0$ and $\gamma_M(s, C) = 0$. In the former case, the second clause of lemma 2 provides us with a state t' that satisfies all the requirements of definition 5. In particular, $\gamma_M(t', C) > 0$, hence we get $t' \overset{\tau}{\nrightarrow}$ since the IMC is maximal-progress-cut. In case $\gamma_M(s, C) = 0$ we cannot use the second clause of lemma 2. The third

clause however provides us with a state t'. We now consider the case $s \equiv_B t'$, the converse is treated afterwards. If $\gamma_M(t', C) = 0$ the proof of this case is complete. If instead $\gamma_M(t', C) > 0$, we have by symmetry of B, that $t' \equiv_B s$. Recalling $t' \overset{\tau}{\not\to}$ and $s \overset{\tau}{\not\to}$, the second clause of lemma 2 implies $\gamma_M(t', C) = \gamma_M(s, C)$, because $s \overset{\tau^*}{\longrightarrow} s$ is the only option. This yields a contradiction, completing the proof for the case $s \equiv_B t'$. It remains the case that $s \equiv_B t'$ does not hold. In this case, we know for sure that $t \overset{\tau^*}{\longrightarrow} t'$ has at least length 1. We use induction on the length of this sequence. We only consider the base of the induction here, that is $t \overset{\tau}{\to} t'$. Now, using symmetry of B, we apply clause 1(b) of lemma 2 to $t \equiv_B s$. Since $t \overset{\tau}{\to} t'$, also $s \overset{\tau^+}{\longrightarrow} s'$ must hold for some s', contradicting the assumption $s \overset{\tau}{\not\to}$. □

In the setting of IMCs, *divergence* corresponds to cycles of τ-transitions (they may induce time-locks). The third clause of the lemma above makes it evident that stochastic branching bisimulation is divergence sensitive, in the sense that two states are only equivalent if

- either of them can escape divergence by means of a τ-step, or
- neither of them can escape divergence by means of a τ-step.

This fact can be made explicit by strengthening the third clause as follows:

Lemma 3. *For a maximal-progress-cut IMC $M = (S, A, T, R)$, B is a stochastic branching bisimulation iff $s \equiv_B t$ implies for all $a \in A$ and all equivalence classes C of B*

1. *$s \overset{a}{\to} s'$ implies*
   (a) *$a = \tau$ and $s \equiv_B s'$, or*
   (b) *there are $t', t'' \in S$ with $t \overset{\tau^*}{\underset{B}{\to}} t' \overset{a}{\to} t''$ and $s' \equiv_B t''$.*
2. *$\gamma_M(s, C) > 0$ implies*
   (c) *$\gamma_M(s, C) = \gamma_M(t', C)$ for some t' such that $t \overset{\tau^*}{\underset{B}{\to}} t'$*
3. *$\exists s' : s \overset{\tau^*}{\longrightarrow} s' \overset{\tau}{\not\to}$ iff $\exists t' : t \overset{\tau^*}{\longrightarrow} t' \overset{\tau}{\not\to}$.*

Proof. The implication from lemma 3 to lemma 2 is obvious. For the converse, let B be an equivalence relation satisfying lemma 2, and assume $s \equiv_B t$. Due to the symmetry of B we only consider the implication from $\exists s' : s \overset{\tau^*}{\longrightarrow} s' \overset{\tau}{\not\to}$ to $\exists t' : t \overset{\tau^*}{\longrightarrow} t' \overset{\tau}{\not\to}$. To show the latter assuming the earlier, we proceed by induction on the length of the sequence $s \overset{\tau^*}{\longrightarrow} t' \overset{\tau}{\not\to}$. If this length is 0, we can use the third clause of lemma 2 to conclude $t \overset{\tau^*}{\longrightarrow} t' \overset{\tau}{\not\to}$. Assume we know this for length n, i. e.

$$s'' \equiv_B t'' \text{ and } s'' \underbrace{\overset{\tau}{\to} \cdots \overset{\tau}{\to}}_{n} s' \overset{\tau}{\not\to} \text{ imply } t'' \overset{\tau}{\to} \cdots \overset{\tau}{\to} t' \overset{\tau}{\not\to}$$

We need to derive from the above induction hypothesis that

$$s \equiv_B t \text{ and } s \overset{\tau}{\to} s'' \underbrace{\overset{\tau}{\to} \cdots \overset{\tau}{\to}}_{n} s' \overset{\tau}{\not\to} \text{ imply } t \overset{\tau}{\to} \cdots \overset{\tau}{\to} t' \overset{\tau}{\not\to}$$

holds. If $s \equiv_B s''$ we use symmetry and transitivity of B to derive $s'' \equiv_B t$ and conclude the proof via the induction hypothesis. Otherwise we apply clause 1(b) of lemma 2 to conclude that $t \xrightarrow{\tau^+} t''$ and $s'' \equiv_B t''$. The latter allows us to use the induction hypothesis to eventually arrive at $t \xrightarrow{\tau^+} t'' \xrightarrow{\tau^*} t' \overset{\tau}{\nrightarrow}$, completing the proof. □

In the following we will use the characterization of stochastic branching bisimulation given in lemma 3.

One of the most important applications of stochastic strong and branching bisimulation (or bisimulations in general) is—besides equivalence checking w. r. t. observable behavior—the minimization of systems. The so-called quotient system is constructed by collapsing the equivalence classes into new states and adapting the transitions:

Definition 6. *Let $M = (S, A, T, R)$ be a maximal-progress-cut IMC and P a stochastic strong or branching bisimulation. The **quotient** M/P is an IMC $M' = (S', A', T', R')$ with*

- $S' = \{P(s) \mid s \in S\}$*, the set of equivalence classes,*
- $A' = A$*,*
- $(C, a, C') \in T'$ *iff* $\begin{cases} \exists s \in C, s' \in C' : (s, a, s') \in T & \text{for strong bisim.} \\ \exists s \in C, s' \in C' : (s, a, s') \in T \wedge & \\ \quad \neg\big(a = \tau \wedge C = C' \wedge & \text{for branching bisim.} \\ \quad (\exists t \in C\, \exists t' \in S : t \xrightarrow{\tau^*} t' \overset{\tau}{\nrightarrow})\big) & \end{cases}$
- $R(C, C') = \max\limits_{s \in C} \gamma_M(s, C')$.

The transition relations for both sorts of bisimulation are well-defined: all strongly bisimilar states have interactive transitions with the same label to equivalent states. In case of branching bisimulation, all states can execute the same observable transitions leading to the same equivalence classes, possibly after some inert τ-steps. For stochastic branching bisimulation, if the states of an equivalence class can escape divergence, the inert τ-self-loops are not needed in the quotient and therefore removed.

The reason for taking the maximum of the rates for constructing the quotient is only branching bisimulation: some of the equivalent states may have outgoing τ-transitions and thus rate 0. But if there are states in the same class without τ-transitions, their rate $r \geq 0$ has to be used for the quotient. This rate however is the same for all equivalent states without τ-steps. For the rate matrix of the quotient w. r. t. stochastic strong bisimulation, the rates of an arbitrary state contained in the equivalence class could be used instead of the maximum.

3 Bisimulation Computation

We will now show how the following problem can be solved:

Given an initial partition $P^{(0)}$, compute the coarsest stochastic strong / branching bisimulation that is a refinement of $P^{(0)}$.

The standard approach is to start with the initial partition and refine it iteratively until a bisimulation is obtained. We will also follow this approach.

The rough idea of our algorithm is the following: compute the signatures of all states—a signature characterizes a state regarding the actions that can be executed in this state—and group the states according to their signatures. This yields a refined partition. Iterate this refinement step until a fixpoint is reached. The result is the coarsest stochastic strong / branching bisimulation.

Details about the signatures and the refinement are given in the following.

3.1 Stochastic Strong Bisimulation

As shown in [23], we cannot compute the signatures of all states first and then refine the whole partition in one step if we want to be able to take an arbitrary initial partition $P^{(0)}$ into account. Instead, we will group the blocks of the current partition according to $P^{(0)}$. This will be called a hyper-partition.

Definition 7. *Let $P^{(0)}$ and P be partitions with $P \sqsubseteq P^{(0)}$. Then the **hyper-partition** of P induced by $P^{(0)}$ is given by*

$$\hat{P} := \left\{ \{C \in P \mid C \subseteq C^{(0)}\} \,\middle|\, C^{(0)} \in P^{(0)} \right\}.$$

*The elements of $\hat{P}$ are called **hyper-blocks**.*

In the following we will always assume that a fixed initial partition $P^{(0)}$ is given. For a partition $P \sqsubseteq P^{(0)}$ we will denote the corresponding hyper-partition by $\hat{P}$. Obviously the following facts hold for partitions $P, Q \sqsubseteq P^{(0)}$:

$$P = \cup\hat{P} := \bigcup_{M \in \hat{P}} M \tag{1}$$

$$P = Q \quad \Leftrightarrow \quad \hat{P} = \hat{Q} \tag{2}$$

We will now define two signatures and a refinement operator, which are used for the computation of stochastic strong bisimulations. While in the non-stochastic setting, one signature suffices to compute strong bisimulation [17], we have to use two for IMCs: the non-stochastic signature sig^s_{ns} is the same as used for labeled transition systems; it describes the interactive transitions executable in a state s. The stochastic signature sig^s_s was used e. g. in [23] for the lumping relation; it assigns the cumulative transition rates into the blocks of the current partition to the states. They are formally defined as follows.

Definition 8. *Let P be a partition. Then, the non-stochastic signature sig^s_{ns} and the stochastic signature sig^s_s for stochastic strong bisimulation are defined as*

$$\mathrm{sig}^s_{ns}(P, s) = \{(a, C) \in A \times P \mid \exists s' \in C : s \xrightarrow{a} s'\} \tag{3}$$

$$\mathrm{sig}^s_s(P, s) = \{(r, C) \in \mathbb{R}^{\geq 0} \times P \mid r = \gamma_M(s, C)\} \tag{4}$$

Given a hyper-partition $\hat{P}$, the refinement operator sigref^s groups the states of the hyper-blocks according to their signatures such that afterwards states with different signatures are placed in different blocks. Thereby no state may leave its hyper-block.

Definition 9. *Let $\hat{P}$ be a hyper-partition. Then, the refinement operator sigref^s is defined as*

$$\begin{aligned}\text{sigref}^s(\hat{P}, M) = \Big\{\{t \in \cup M \mid \text{sig}^s_{ns}(\cup\hat{P}, s) = \text{sig}^s_{ns}(\cup\hat{P}, t) \\ \wedge\, \text{sig}^s_s(\cup\hat{P}, s) = \text{sig}^s_s(\cup\hat{P}, t)\} \,\Big|\, s \in \cup M\Big\} \end{aligned} \tag{5}$$

$$\text{sigref}^s(\hat{P}) = \{\text{sigref}^s(\hat{P}, M) \mid M \in \hat{P}\} \tag{6}$$

This operator is applied iteratively until the hyper-partition does not change anymore. The resulting algorithm works as follows:

Algorithm 1: computeStochasticStrongBisimulation($P^{(0)}$)
Begin
$\hat{P}^{(0)} \leftarrow \left\{\{C\} \,\middle|\, C \in P^{(0)}\right\}$ (1)
$i \leftarrow 0$ (2)
Repeat (3)
$\hat{P}^{(i+1)} \leftarrow \text{sigref}^s(\hat{P}^{(i)})$ (4)
$i \leftarrow i + 1$ (5)
Until $\hat{P}^{(i)} = \hat{P}^{(i-1)}$ (6)
Return $\cup\hat{P}^{(i)}$ (7)
End

We will now prove the correctness of the algorithm in three steps: first we show that it terminates after a finite number of iterations. Then we prove that the result is a stochastic strong bisimulation, before we strengthen this result by proving that the partition returned by the algorithm is the coarsest stochastic strong bisimulation which refines the given initial partition.

Lemma 4. *Algorithm 1 terminates after a finite number of iterations.*

Proof. To keep the notation simple, we will write P for $\cup\hat{P}$.

We will prove that $P^{(i)} \sqsubseteq P^{(i-1)}$ holds for all $i > 0$. Since there is only a finite number of partitions of a finite set and the sequence is monotonically decreasing, this implies that the sequence of partitions converges after n iterations for some $n \geq 0$, i. e. $P^{(n)} = P^{(n-1)}$. According to equation (2) we then have $\hat{P}^{(n)} = \hat{P}^{(n-1)}$, and the algorithm terminates.

We will prove the claim by induction on i. The induction base ($P^{(1)} \sqsubseteq P^{(0)}$) is trivial because every hyper-partition that is compatible with $P^{(0)}$ is a refinement of $\hat{P}^{(0)}$. Let us assume that $i > 1$ and that the claim is true for all $j < i$ and $s \equiv_{P^{(i)}} t$. We have to show that this implies $s \equiv_{P^{(i-1)}} t$.

The states s and t are contained in the same hyper-block of $\hat{P}^{(i)}$. Since states always stay in the same hyper-block, s and t are also contained in the same

hyper-block of $\hat{P}^{(i-1)}$. We make the assumption that $s \not\equiv_{P^{(i-1)}} t$. This means that $\mathrm{sig}^s_{ns}(P^{(i-2)}, s) \neq \mathrm{sig}^s_{ns}(P^{(i-2)}, t)$ or $\mathrm{sig}^s_{s}(P^{(i-2)}, s) \neq \mathrm{sig}^s_{s}(P^{(i-2)}, t)$

We first consider the former case. W. l. o. g. there is $(a, C) \in \mathrm{sig}^s_{ns}(P^{(i-2)}, s)$ such that $(a, C) \notin \mathrm{sig}^s_{ns}(P^{(i-2)}, t)$. Hence, there is $s' \in C$ with $s \xrightarrow{a} s'$ and for all $t' \in C$ there is no transition $t \xrightarrow{a} t'$. This again implies $(a, P^{(i-1)}(s')) \in \mathrm{sig}^s_{ns}(P^{(i-1)}, s)$ and $(a, P^{(i-1)}(t')) \notin \mathrm{sig}^s_{ns}(P^{(i-1)}, t)$ for all $t' \in C$. It follows that $\mathrm{sig}^s_{ns}(P^{(i-1)}, s) \neq \mathrm{sig}^s_{ns}(P^{(i-1)}, t)$ and therefore $s \not\equiv_{P^{(i)}} t$. This contradicts our assumption and concludes the proof for the first case.

In the latter case, assume $(r, C) \in \mathrm{sig}^s_{s}(P^{(i-2)}, s)$ and $(r, C) \notin \mathrm{sig}^s_{s}(P^{(i-2)}, t)$, i. e. $\gamma_M(s, C) \neq \gamma_M(t, C)$. Since $P^{(i-1)}$ is a refinement of $P^{(i-2)}$ by induction hypothesis, C is the union of pairwise disjoint blocks $C_1, \ldots, C_k$ of $P^{(i-1)}$, i. e., $\gamma_M(u, C) = \sum_{j=1}^{k} \gamma_M(u, C_j)$ for all states $u \in S$. So there must be a block C_j with $\gamma_M(s, C_j) \neq \gamma_M(t, C_j)$. Then $(\gamma_M(s, C_j), C_j) \in \mathrm{sig}^s_{s}(P^{(i-1)}, s)$ and $(\gamma_M(s, C_j), C_j) \notin \mathrm{sig}^s_{s}(P^{(i-1)}, t)$. This implies $s \not\equiv_{P^{(i)}} t$, again contradicting our assumption. □

Lemma 5. *If $\hat{P}$ is a hyper-partition with $\hat{P} = \mathrm{sigref}^s(\hat{P})$, then $\cup\hat{P}$ is a stochastic strong bisimulation.*

Proof. For simplicity of notation we again use P as an abbreviation for $\cup\hat{P}$. Assume $\hat{P}$ is a fixpoint of sigref^s. Furthermore, let $s \equiv_P t$. If the transition $s \xrightarrow{a} s'$ exists, $(a, P(s')) \in \mathrm{sig}^s_{ns}(P, s)$. Because $\hat{P}$ is a fixpoint, all equivalent states have the same signature, i. e., $(a, P(s')) \in \mathrm{sig}^s_{ns}(P, t)$. Then there is a transition $t \xrightarrow{a} t'$ with $s' \equiv_P t'$.

Let C be a block of P and $r = \gamma_M(s, C)$. If $s \xrightarrow{\tau}$ there is nothing to show. Otherwise $(r, C) \in \mathrm{sig}^s_{s}(P, s)$. Because s and t have the same signatures, $(r, C) \in \mathrm{sig}^s_{s}(P, t)$. This means $r = \gamma_M(t, C)$. We can conclude that P is a stochastic strong bisimulation. □

Lemma 6. *Let $M = (S, A, T, R)$ be a maximal-progress-cut IMC, $\mathfrak{B}^s$ a stochastic strong bisimulation on M, and $\hat{P}$ a hyper-partition with $\mathfrak{B}^s \sqsubseteq \cup\hat{P}$. Then $\mathfrak{B}^s \sqsubseteq \cup\,\mathrm{sigref}^s(\hat{P})$ holds.*

Proof. As usual, we write P for $\cup\hat{P}$. Let $s \equiv_{\mathfrak{B}^s} t$ and hence $s \equiv_P t$. We have to show that this implies $\mathrm{sig}^s_{ns}(P, s) = \mathrm{sig}^s_{ns}(P, t)$ and $\mathrm{sig}^s_{s}(P, s) = \mathrm{sig}^s_{s}(P, t)$.

Let $(a, C) \in \mathrm{sig}^s_{ns}(P, s)$. By definition of the non-stochastic signature, this means that there is $s' \in C$ such that $s \xrightarrow{a} s'$. Since $s \equiv_{\mathfrak{B}^s} t$ and $\mathfrak{B}^s$ is a stochastic strong bisimulation, we can conclude that there is $t' \in S$ with $t \xrightarrow{a} t'$ and $s' \equiv_{\mathfrak{B}^s} t'$. Because of $\mathfrak{B}^s \sqsubseteq P$, we have $t' \in C$ and $(a, C) \in \mathrm{sig}^s_{ns}(P, t)$. We can conclude that $\mathrm{sig}^s_{ns}(P, s) \subseteq \mathrm{sig}^s_{ns}(P, t)$. The equality of the non-stochastic signatures follows then for symmetry reasons.

Let $(r, C) \in \mathrm{sig}^s_{s}(P, s)$. Then $r = \gamma_M(s, C)$. Because $\mathfrak{B}^s \sqsubseteq P$ holds, C is the union of pairwise disjoint blocks $C_1, \ldots, C_k$ of $\mathfrak{B}^s$. Since M is a stochastic strong bisimulation and M maximal-progress-cut (otherwise the following would only hold if $s \xrightarrow{\tau}$), the condition $\gamma_M(s, C_j) = \gamma_M(t, C_j)$ is satisfied for $1 \leq j \leq k$ and

hence

$$r = \gamma_M(s, C) = \sum_{j=1}^{k} \gamma_M(s, C_j) = \sum_{j=1}^{k} \gamma_M(t, C_j) = \gamma_M(t, C).$$

Therefore we have $(r, C) \in \mathrm{sig}_s^s(P, t)$. Since there is exactly one entry in $\mathrm{sig}_s^s(P, t)$ per block, the equality of the stochastic signatures follows. □

These three lemmas together prove the following theorem:

Theorem 1. *For any maximal-progress-cut IMC $M = (S, A, T, R)$ and any initial partition $P^{(0)}$ of S, algorithm 1 computes the coarsest stochastic strong bisimulation that refines $P^{(0)}$.*

3.2 Stochastic Branching Bisimulation

The idea for the computation of the coarsest stochastic branching bisimulation is the same as for stochastic strong bisimulation: We characterize the interactive and the stochastic behavior by two signatures. They differ from the signatures of stochastic strong bisimulation in the way they take τ-transitions into account: Before executing the observable transition and before taking a Markov transition to a certain equivalence class, an inert sequence of τ-steps of arbitrary length may be executed.

Definition 10. *Let P be a partition. Then, the non-stochastic signature* sig_{ns}^b *and the stochastic signature* sig_s^b *for stochastic branching bisimulation are defined as*

$$\mathrm{sig}_{ns}^b(P, s) = \Big\{(a, C) \in A \times P \;\Big|\; \exists s' \in S, s'' \in C : s \xrightarrow[P]{\tau^*} s' \xrightarrow{a} s'' \wedge (a \neq \tau \vee (s, s'') \notin P)\Big\} \tag{7}$$

$$\mathrm{sig}_s^b(P, s) = \Big\{(r, C) \in P \times \mathbb{R}^{\geq 0} \;\Big|\; C \in P \wedge r = \max\{\gamma_M(s', C) \mid s \xrightarrow[P]{\tau^*} s' \not\xrightarrow{\tau}\}\Big\} \tag{8}$$

For efficiency considerations we use a slightly different refinement operator for stochastic branching bisimulation: Since the signatures depend on the pairs of states that are connected by an inert τ-sequence it would be necessary to compute the pairs (s, t) of states with $s \equiv_P t$. For reasons which will become clear in section 4 about the symbolic implementation, the computation of this relation is extremely inefficient using our partition representation. It can be avoided by refining the current partition block by block:

Definition 11. *Let P be a partition. Then, the refinement operator* sigref^b *is defined as*

$$\mathrm{sigref}^b(P, C) = \Big\{\{t \in C \mid \mathrm{sig}_{ns}^b(P, s) = \mathrm{sig}_{ns}^b(P, t) \wedge \mathrm{sig}_s^b(P, s) = \mathrm{sig}_s^b(P, t)\} \;\Big|\; s \in C\Big\} \tag{9}$$

$$\mathrm{sigref}^b(P) = \bigcup_{C \in P} \mathrm{sigref}(P, C) \tag{10}$$

Another difference to stochastic strong bisimulation is given by the divergence sensitivity of stochastic branching bisimulation (see clause 3 of lemma 3). Divergence can be taken into account by using an appropriate initial partition: we split the blocks of the given initial partition $P^{(0)}$ into the states that can escape divergence and those that cannot. This leads to a refined initial partition, which is defined as follows:

$$\begin{aligned}\tilde{P}^{(0)} = &\left\{\{s \in B \mid \exists s' \in S : s \xrightarrow{\tau^*} s' \not\xrightarrow{\tau}\} \;\middle|\; B \in P^{(0)}\right\} \\ &\cup \left\{\{s \in B \mid \nexists s' \in S : s \xrightarrow{\tau^*} s' \not\xrightarrow{\tau}\} \;\middle|\; B \in P^{(0)}\right\}\end{aligned} \tag{11}$$

The result is the following algorithm:

Algorithm 2: computeStochasticBranchingBisimulation($P^{(0)}$)
Begin
Compute $\tilde{P}^{(0)}$ as given in formula (11) (1)
$i \leftarrow 0$ (2)
Repeat (3)
$\tilde{P}^{(i+1)} \leftarrow \text{sigref}^b\left(\tilde{P}^{(i)}\right)$ (4)
$i \leftarrow i + 1$ (5)
Until $\tilde{P}^{(i)} = \tilde{P}^{(i-1)}$ (6)
Return $\tilde{P}^{(i)}$ (7)
End

The correctness of the algorithm can be shown in a similar way as for the algorithm for stochastic strong bisimulation:

Lemma 7. *Let $P^{(0)}, P^{(1)}, P^{(2)}, \ldots$ be a sequence of partitions with $P^{(i+1)} = \text{sigref}^b(P^{(i)})$. Then there exists $n \geq 0$ with $P^{(n+1)} = P^{(n)}$.*

Proof. The sequence of partitions is bounded below by $\{\{s\} \mid s \in S\}$. Furthermore it is monotonic decreasing and hence convergent. Because there are only finitely many partitions of a finite set, the limit is reached after finitely many refinement steps. □

Lemma 8. *Let*

$$P_{\text{div}} = \left\{\{s \in S \mid \exists s' \in S : s \xrightarrow{\tau^*} s' \not\xrightarrow{\tau}\}, \{s \in S \mid \nexists s' \in S : s \xrightarrow{\tau^*} s' \not\xrightarrow{\tau}\}\right\}.$$

If P is a partition with $P \sqsubseteq P_{\text{div}}$ and $\text{sigref}_b(P) = P$, then P is a stochastic branching bisimulation.

Proof. Let P be a partition satisfying the prerequisites of lemma 8. Because of $P \sqsubseteq P_{\text{div}}$, the third clause of lemma 3 is satisfied.

Let us assume, that $s \equiv_P t$ and that the transition $s \xrightarrow{a} s'$ exists. If $a = \tau$ and $s \equiv_P s'$ there is nothing to show. Otherwise $(a, P(s')) \in \text{sig}^b_{ns}(P, s)$. Because P is a fixpoint, $\text{sig}^b_{ns}(P, s) = \text{sig}^b_{ns}(P, t)$. This implies $(a, P(s')) \in \text{sig}^b_{ns}(P, t)$. By definition

of the signature there exist $t', t'' \in S$ with $t \xrightarrow[P]{\tau^*} t' \xrightarrow{a} t''$ and $t'' \equiv_P s'$. This shows the first clause of lemma 3.

Now, let $(r, C) \in \mathrm{sig}^b_s(P, s)$ and $s \equiv_P t$. If s has an out-going τ-transition there is nothing to show. So we can assume that s has no out-going τ-transitions. Then the set $\{\gamma_M(s', C) \mid s \xrightarrow[P]{\tau^*} s' \overset{\tau}{\nrightarrow}\}$ contains exactly one element, namely $\gamma_M(s, C)$, i. e., $r = \gamma_M(s, C)$. Because the signatures of s and t are identical (otherwise P would not be a fixpoint), we have that $r = \max\{\gamma_M(t', C) \mid t \xrightarrow[P]{\tau^*} t' \overset{\tau}{\nrightarrow}\}$ and in particular that there is $t' \in S$ with $t \xrightarrow[P]{\tau^*} t' \overset{\tau}{\nrightarrow}$ and $r = \gamma_M(t, C)$. Now, it follows that also the second clause of lemma 3 is fulfilled and P is a stochastic branching bisimulation. □

Lemma 9. *Let $M = (S, A, T, R)$ be a maximal-progress-cut IMC, $\mathfrak{B}^b$ a stochastic branching bisimulation on M, and P a partition with $\mathfrak{B}^b \sqsubseteq P$. Then $\mathfrak{B}^b \sqsubseteq \mathrm{sigref}^b(P)$ holds.*

Proof. Let $s_0 \equiv_{\mathfrak{B}^b} t_0$. We have to show that the signatures of s_0 and t_0 are identical. The proof for the non-stochastic signature is exactly the same as in [13]. So we only give the proof for the stochastic signature here.

Let $(r, C) \in \mathrm{sig}^b(P, s_0)$, i. e., $r = \max\{\gamma_M(s', C) \mid s_0 \xrightarrow[P]{\tau^*} s' \overset{\tau}{\nrightarrow}\}$. We have to prove, that $r = \max\{\gamma_M(t', C) \mid t_0 \xrightarrow[P]{\tau^*} t' \overset{\tau}{\nrightarrow}\}$. Due to symmetry reasons it suffices to show that $r \leq \max\{\gamma_M(t', C) \mid t_0 \xrightarrow[P]{\tau^*} t' \overset{\tau}{\nrightarrow}\}$.

Let us assume first that $r > 0$. Then there exist $s_1, \ldots, s_n \in S$ with $s_i \xrightarrow{\tau} s_{i+1}$ for $0 \leq i < n$ and $s_0 \equiv_P s_n$ as well as $r = \gamma_M(s_n, C)$. We now distinguish two cases:

- $s_0 \equiv_{\mathfrak{B}^b} s_n$. Since $\mathfrak{B}^b$ is an equivalence relation $(s_n, t_0) \in \mathfrak{B}^b$ holds. It is furthermore a stochastic branching bisimulation on the maximal-progress-cut IMC M. This means, according to lemma 3, since $r > 0$, that there is $t' \in S$ with $t_0 \xrightarrow[\mathfrak{B}^b]{\tau^*} t' \overset{\tau}{\nrightarrow}$ and $\gamma_M(t', C) = r$. We have hereby implicitly used the fact that C is the pairwise disjoint union of blocks of $\mathfrak{B}^b$.
- $s_0 \not\equiv_{\mathfrak{B}^b} s_n$. We will construct a path starting in t_0 inductively. Let us assume that t_i is already defined for $i < n$ such that $s_i \equiv_{\mathfrak{B}^b} t_i$. If $s_i \equiv_{\mathfrak{B}^b} s_{i+1}$ holds, set $t_{i+1} := t_i$. Otherwise, there are (since $\mathfrak{B}^b$ is a stochastic branching bisimulation) t'_i and t_{i+1} with $t_i \xrightarrow[\mathfrak{B}^b]{\tau^*} t'_i \xrightarrow{\tau} t_{i+1}$ and $s_{i+1} \equiv_{\mathfrak{B}^b} t_{i+1}$. Taken this together, we have $t_0 \xrightarrow{\tau^*} t_n$ and $s_n \equiv_{\mathfrak{B}^b} t_n$. By definition of stochastic branching bisimulation there is t'_n with $t_n \xrightarrow[\mathfrak{B}^b]{\tau^*} t'_n \overset{\tau}{\nrightarrow}$ and $\gamma_M(t'_n, C) = r$. Because $\mathfrak{B}^b$ is a refinement of P it holds that $t_0 \xrightarrow[P]{\tau^*} t'_n$ and hence also $r \in \{\gamma_M(t', C) \mid t \xrightarrow[P]{\tau^*} t' \overset{\tau}{\nrightarrow}\}$, i. e., $r \leq \max\{\gamma_M(t', C) \mid t \xrightarrow[P]{\tau^*} t' \overset{\tau}{\nrightarrow}\}$

We still have to show that $r = 0$ implies $\max\{\gamma_M(t', C) \mid t_0 \xrightarrow[P]{\tau^*} t' \not\xrightarrow{\tau}\} = 0$. We prove this by contradiction. Therefore, we make the assumption that there is t' with $t_0 \xrightarrow[P]{\tau^*} t' \not\xrightarrow{\tau}$ and $\gamma_M(t', C) > 0$. If $t_0 \equiv_{\mathfrak{B}^b} t'$, $\mathfrak{B}^b$ being a stochastic branching bisimulation implies that there is s' with $s_0 \xrightarrow[\mathfrak{B}^b]{\tau^*} s' \not\xrightarrow{\tau}$ and $\gamma_M(s', C) = \gamma_M(t', C) > 0$. This is a contradiction to $r = 0$. Otherwise, if $t_0 \not\equiv_{\mathfrak{B}^b} t'$ we can construct a path $s_0 \xrightarrow{\tau^*} s'$ as above with $s' \equiv_{\mathfrak{B}^b} t'$. Because $\mathfrak{B}^b$ is a stochastic branching bisimulation we can conclude the existence of s'' with $s \xrightarrow[\mathfrak{B}^b]{\tau^*} s'' \not\xrightarrow{\tau}$ and $\gamma_M(s'', C) = \gamma_M(t', C) > 0$, which is a contradiction to $r = 0$. □

These three lemmas together proof the following theorem:

Theorem 2. *For maximal-progress-cut IMCs, algorithm 2 computes the coarsest stochastic branching bisimulation that refines the given initial partition.*

What we described above is not the only possibility for computing the bisimulation relations. We discuss some alternatives in the following.

Remark 1. We could use the following stochastic signature for branching bisimulation instead:

$$\widehat{\mathrm{sig}}^b_s(P, S) = \left\{(C, \{r \mid r = \gamma_M(s', C) \wedge s \xrightarrow[P]{\tau^*} s' \not\xrightarrow{\tau}\}) \mid C \in P\right\}$$

If we would use this signature for the refinement instead of sig^b_s, the result would also be the coarsest stochastic branching bisimulation that refines the initial partition. The reason not to do so is due to implementation: The use of the modified signature would require to store sets of numbers in the leafs of MTBDDs which is not possible in any state-of-the-art implementation of MTBDDs, e.g. [28].

Remark 2. Using the signatures for branching bisimulation together with algorithm 1 and the divergence-sensitive initial partition would also yield the same result as algorithm 2.

Also the other variant would work: using the signatures of stochastic strong bisimulation together with algorithm 2 (without the refinement of the initial partition to take divergence into account) would yield the coarsest stochastic strong bisimulation that refines the initial partition.

As already said, the reason for using two different algorithms is efficiency. We will explain in the next section why it is very important to avoid the computation of the relation $s \equiv_P t$ symbolically when using our BDD-based representation. For strong bisimulation, this relation is not needed. So we use the more efficient algorithm 1 in that case. The signatures of branching bisimulation depend on inert τ-transitions. To avoid the expensive computation of the equivalence relation above we compute the signature only of one block at a time as indicated in algorithm 2.

4 Concepts for Symbolic Implementation

In this section we will present concepts for the symbolic implementation of the algorithms introduced in the previous section. We will go into details regarding the symbolic representation of IMCs, partitions and the signatures. After that we will first show how to refine the given initial partition such that the divergence of stochastic branching bisimulation is taken into account. Then we explain how the stochastic and non-stochastic signatures of strong and branching bisimulation can be computed symbolically. Afterwards we show how to refine the partition using the signatures and how to compute the quotient system once the bisimulation has been computed. Finally, we show how to carry over some optimizations that were proved useful in the non-stochastic domain [18].

4.1 Symbolic Representation using OBDDs and MTBDDs

Binary decision diagrams (BDDs) will play a central role in our implementation of the bisimulation algorithm. They can be used to represent boolean and pseudo-boolean functions $f : \{0,1\}^n \to \{0,1\}$ (OBDDs) and $f : \{0,1\}^n \to \mathbb{R}$ (MTBDDs), respectively, in a compact way. Algorithms that exploit the structure of (MT)BDDs can often handle much larger problems than explicit algorithms.

In the following we will assume that the reader is familiar with BDDs. Detailed information about them can be found for example in [29].

Representation of an IMC To represent the set $S \subseteq \{0,1\}^n$ of states, one can use the OBDD of its characteristic function, i. e. an OBDD $\mathcal{S}$ with $\mathcal{S}(s) = 1$ iff $s \in S$. The same kind of representation can be used for the interactive transitions: we use an OBDD $\mathcal{T}(s,a,t)$ such that $\mathcal{T}(s,a,t) = 1$ iff $s \xrightarrow{a} t$.

For the representation of the Markov transition relation we make use of an MTBDD $\mathcal{R}$ whose leaves are labeled with the distinct rates contained in the matrix. The path given by a state pair (s,t) ends in the leaf labeled with $R(s,t)$.

Partition Representation To represent partitions symbolically there are several possibilities:

The simplest one is to use one OBDD per block that describe the states contained in the corresponding block. This has the drawback that a large number of OBDDs has to be maintained if the number of blocks becomes large.

A more memory-efficient representation was suggested by Derisavi [22]: Derisavi does not store one OBDD per block, but uses a kind of logarithmic encoding, thereby needing only a logarithmic number of OBDDs. Accessing the blocks then needs an amortized constant number of OBDD-operations if the order in which the blocks are accessed is chosen carefully. Nevertheless, the access to the blocks needs a significant amount of time.

A third possibility, which is used by Bouali and de Simone in [9], is to represent the corresponding equivalence relation, i. e., to use an OBDD $\mathcal{E}(s,t)$ with $\mathcal{E}(s,t) = 1$ iff $s \equiv_P t$. This representation has the disadvantage that it is

often quite memory-consuming and that the extraction of the quotient system is inefficient.

We assume that the initial partition is given using the first representation with one OBDD per block. Since the initial partition normally consists only of few blocks, this does not cause any memory problems. For the actual execution of the refinement algorithm we will use a fourth possibility: we assign a unique number to each block, encoding this number using a new vector k of binary variables. Then, a partition $P = \{C_0, \ldots, C_{m-1}\}$ is represented by an OBDD $\mathcal{P}(s,k)$ such that $\mathcal{P}(s,k) = 1$ iff $s \in C_k$. The advantages of this representation are (1) that the signatures and the refinement can be computed very efficiently as we will see in the following and (2) that the extraction of the quotient system can be done easily. As long as the number of blocks does not grow too much, it is also memory-efficient.

Signature Representation We use the concept of the block numbers also for the representation of the signatures: The non-stochastic signatures are represented by an OBDD $\sigma_{ns}^{s/b}(s,a,k)$ such that

$$\sigma_{ns}^{s/b}(s,a,k) = 1 \quad \text{iff} \quad (a, C_k) \in \text{sig}_{ns}^{s/b}(P,s)$$

For the stochastic-signatures we cannot use OBDDs since they are restricted to the values in $\{0,1\}$, but we have to represent arbitrary real numbers. The obvious generalization are MTBDDs whose leaves may be labeled with arbitrary numbers. Hence, an MTBDD $\sigma_{s}^{s/b}(s,k)$ is used with

$$\sigma_{s}^{s/b}(s,k) = r \quad \text{iff} \quad (r, C_k) \in \text{sig}_{s}^{s/b}(P,s).$$

4.2 Computation of the Initial Partition for Stochastic Branching Bisimulation

As we have seen in section 3.2, we have to compute a refined initial partition for stochastic branching bisimulation which takes divergence into account. This means that we have to split all blocks of the given initial partition into those states that can escape divergence and those that cannot (see line 11 of algorithm 2).

The set of all states which can escape divergence is computed with the following algorithm:

Algorithm 3: DivergenceEscapingStates(IMC M)
Begin
$\mathcal{T}_\tau(s,t) \leftarrow \mathit{Cofactor}(\mathcal{T}(s,a,t),\ a = \tau)$ (1)
$\mathcal{D}(s) \leftarrow \mathcal{S}(s) \wedge \neg\exists t : \mathcal{T}_\tau(s,t)$ (2)
$\mathcal{I}(s,t) \leftarrow \mathit{Closure}(\mathcal{T}_\tau(s,t))$ (3)
Return $\exists t : \mathcal{I}(s,t) \wedge \mathcal{D}(t)$ (4)
End

It proceeds as follows: First, the pairs of states that are connected by a τ-transition are extracted (line 1). Then we can determine all states without out-going τ-step in line 2. Now we have to compute all states that can reach such a dead-end state by an arbitrarily long τ-sequence. For this we need the reflexive transitive closure of the τ-transitions (line 3). The states from which τ-sequences lead to a state without out-going τ-transition give the result (line 4).

If $\mathcal{S}_{\text{esc}}$ is the result of algorithm 3, i. e., the set of divergence escaping states, we get the divergence-sensitive initial partition as

$$\tilde{P}^{(0)} \leftarrow \Big(\{\mathcal{B} \wedge \mathcal{S}_{\text{esc}} \,|\, \mathcal{B} \in P^{(0)}\} \cup \{\mathcal{B} \wedge \neg\mathcal{S}_{\text{esc}} \,|\, \mathcal{B} \in P^{(0)}\}\Big) \setminus \{\emptyset\}.$$

We will then convert this refined initial partition to our block number-based representation to perform the actual bisimulation computation.

4.3 Computation of the Signatures

The symbolic computation of the non-stochastic signatures can be performed as described in [17]. So we will only give a brief explanation and concentrate on the stochastic signatures.

Non-stochastic Signature For the computation of the non-stochastic signature we provide several basic BDD operations which make the computation a straightforward task:

- Extraction of the τ-transitions from $\mathcal{T}$:
  $\mathit{Cofactor}(\mathcal{T}(s,a,k), a = \tau)$
- Pairs of states (s,t) that are in the same block:
  $\exists k : \mathcal{P}(s,k) \wedge \mathcal{P}(t,k)$
  It turns out that the computation of this expression is extremely expensive. The main reason is the variable order we have to use for the refinement operator (see below): the s- and t-variables are placed at the top of the BDD and the k-variables at the bottom. While traversing the BDD recursively the algorithm has to store all combinations of assignments of the s- and t-variables until it reaches the bottom of the BDD where it is able to decide whether they are contained in the same block. Furthermore the resulting BDD can be significantly larger than the original one.
  The solution is—as already applied in the previous section in algorithm 2—to carry out the refinement block by block. If $\mathcal{B}(s)$ describes the states of the block for which the signatures have to be computed, the pairs of inert states can be computed using the expression
  $\mathcal{B}(s) \wedge \mathcal{B}(t)$.
- Computation of the reflexive transitive closure (RTC) of a relation $R(s,t)$: There are several symbolic algorithms for the computation of the RTC (e. g. [30, 31]). We apply the iterative squaring method of [30].
- Computation of the non-τ or non-inert transitions:
  $\mathcal{T}(s,a,t) \wedge \neg\Big(\mathcal{B}(s) \wedge \mathcal{B}(t) \wedge a = \tau\Big)$

– Concatenation of two relations, given by OBDDs $\mathcal{R}_1(s,t)$ and $\mathcal{R}_2(s,t)$:
  $\exists x : \mathcal{R}_1(s,x) \wedge \mathcal{R}_2(x,t)$
– Substitution of t in $\mathcal{R}(s,t)$ by its block number:
  $\exists t : \mathcal{R}(s,t) \wedge \mathcal{P}(t,k)$

Algorithm 4 shows how these basic operations can be combined to compute the non-stochastic signature of branching bisimulation. The signature of strong bisimulation can be computed in a similar way.

Algorithm 4: ComputeSignature$^{b}_{ns}$(IMC M, partition $\mathcal{P}$, block $\mathcal{B}$)
Begin
$\mathcal{T}_\tau(s,t) \leftarrow \mathit{Cofactor}(\mathcal{T}(s,a,t),\ a = \tau)$ (1)
$\mathcal{I}_\tau(s,t) \leftarrow \mathit{Closure}(\mathcal{T}_\tau(s,t)) \wedge \mathcal{B}(s) \wedge \mathcal{B}(t)$ (2)
$\mathit{pre}(s,a,t) \leftarrow \exists x : \Big(\mathcal{I}_\tau(s,x) \wedge \mathcal{T}(x,a,t)\Big) \wedge \neg\Big(a = \tau \wedge \mathcal{B}(s) \wedge \mathcal{B}(t)\Big)$ (3)
Return $\exists t : \mathit{pre}(s,a,t) \wedge \mathcal{P}(t,k)$ (4)
End

At first, all pairs of states that are connected by a τ-transition and their RTC, such that source and target state are contained in the same block, are computed. In line 3 the closure of the τ-transitions and the observable transitions are concatenated. Finally, in line 4 the target state of the transition sequence is replaced by its block number.

Stochastic Signatures The next step is the computation of the stochastic strong and branching signature. Since we have to cope with real numbers we need extensions of the existential and universal quantification operators from the boolean domain:

Definition 12. *Let $\circ$ be a commutative and associative binary operator on MTBDDs. Then the quantification of an MTBDD $\mathcal{F}$ w. r. t. variable x and operation $\circ$ is defined as*

$$Q_\circ x.\mathcal{F} := \mathcal{F}_{x=0} \circ \mathcal{F}_{x=1}.$$

On OBDDs, existential and universal quantification are special cases of the quantification operator defined above:

$$\exists x.\mathcal{F} \equiv Q_\vee.\mathcal{F}$$
$$\forall x.\mathcal{F} \equiv Q_\wedge.\mathcal{F}$$

The quantification operator can be implemented on MTBDDs in a similar way as the traditional quantifiers on OBDDs [30].

Stochastic Strong Bisimulation Given an MTBDD $\mathcal{R}(s,t)$ for the transition rates, an (MT)BDD $\mathcal{P}(s,k)$ for the current partition, and an (MT)BDD $\mathcal{B}(s)$ for the block for which the stochastic signature shall be computed, we can proceed for stochastic strong bisimulation as described in algorithm 5:

Algorithm 5: ComputeSignature^s_s(IMC M, partition $\mathcal{P}$, state set $\mathcal{S}'$)
Begin
$\Gamma(s,t,k) \leftarrow \mathcal{S}'(s) \cdot \mathcal{R}(s,t) \cdot \mathcal{P}(t,k)$ (1)
$\sigma^s_s(s,k) \leftarrow \mathbf{Q}_+ t.\Gamma(s,t,k)$ (2)
Return $\sigma^s_s(s,k)$ (3)
End

The first step is to create an MTBDD $\Gamma(s,t,k)$ which describes the rates associated with the transition from s to t which is in the block with number k. Then we quantify t, thereby summing up all cofactors w. r. t. t. This yields the stochastic signature of strong bisimulation.

Stochastic Branching Bisimulation The stochastic signature of branching bisimulation can be computed in a similar way as the signature of strong bisimulation:

Algorithm 6: ComputeSignature^b_s(IMC M, partition $\mathcal{P}$, block $\mathcal{B}$)
Begin
$\mathcal{T}_\tau(s,t) \leftarrow Cofactor(\mathcal{T}(s,a,t), a = \tau)$ (1)
$\mathcal{I}_{\tau^*}(s,t) \leftarrow Closure(\mathcal{T}_\tau(s,t)) \wedge \mathcal{B}(s) \wedge \mathcal{B}(t)$ (2)
$\mathcal{I}^{\tau}_{\tau^*\!\!\not\rightarrow}(s,t) \leftarrow \mathcal{I}_\tau(s,t) \wedge \neg\exists x : \mathcal{T}_\tau(t,x)$ (3)
$\sigma^b_s(s,k) \leftarrow \mathbf{Q}_{\max} t.\Big(\mathbf{Q}_+ x.\Big(\mathcal{I}^{\tau}_{\tau^*\!\!\not\rightarrow}(s,t) \cdot \mathcal{R}(t,x) \cdot \mathcal{P}(x,k)\Big)\Big)$ (4)
Return $\sigma^b_s(s,k)$ (5)
End

We first extract the τ-transitions (line 1), compute its closure and restrict it to inert τ-sequences (line 2). In line 3 we restrict these sequences to those which end in a state with no out-going τ-transition. The actual signature computation is done in line 4: The expression $\mathcal{I}^{\tau}_{\tau^*\!\!\not\rightarrow}(s,t) \cdot \mathcal{R}(t,x) \cdot \mathcal{P}(x,k)$ yields an MTBDD $\Gamma(s,t,x,k)$ such that $\Gamma(s,t,x,k) = R(t,x)$ with $s \xrightarrow[P]{\tau^*} t \overset{\tau}{\not\rightarrow}$ and $x \in C_k$. We first quantify the x variables summing over the cofactors; the result is an MTBDD that describes $\gamma_M(t, C_k)$. Then we quantify the t variables, computing the maximum of the cofactors. This yields the stochastic signature of branching bisimulation.

4.4 Stochastic Refinement

The missing operation for the computation of the coarsest stochastic strong or branching bisimulation is the refinement operation. For this we assume we have computed the two signatures as described above.

If we restrict the allowed variable orders such that the state variables s are placed at the top and the label and block number variables a and k, respectively, at the bottom we can exploit the following observation:

Let s be the encoding of a state. If we follow the paths given by s in the signature-BDDs, we reach a pair of node v_{ns}, v_s. The sub-BDD at node v_{ns} represents the non-stochastic signature of s, while the sub-BDD at node v_s represents the stochastic signature. Furthermore, all states with the same signatures as s

lead to the same pair of nodes. The idea is then to traverse both signature DDs simultaneously. If in both DDs a node representing a signature is reached that has never been visited before, the result is a new block number. On top of the block numbers the paths which led to the signatures are reconstructed. This is illustrated in Algorithm 7.

Algorithm 7: StochasticRefine(Signature OBDD σ_{ns}, Signature MTBDD σ_s)
Begin

```
(1)  If (σ_ns, σ_s) ∈ ComputedTable Then
(2)      Return ComputedTable[(σ_ns, σ_s)]
(3)  End If
(4)  x ← topVar(σ_ns, σ_s)
(5)  If x is a state variable Then
(6)      low ← StochasticRefine(σ_ns|x=0, σ_s|x=0)
(7)      high ← StochasticRefine(σ_ns|x=1, σ_s|x=1)
(8)      result ← newOBDDnode(x, low, high)
(9)  Else
(10)     result ← newBlockNumberOBDD()
(11)     If σ_ns = 0 and σ_s = 0 Then leaf0number ← result
(12) End If
(13) ComputedTable[(σ_ns, σ_s)] ← result
(14) Return result
```

End

The algorithm proceeds in the following way: first, it checks whether the same pair of nodes was reached before. If this is the case the result is already contained in the ComputedTable and can be returned immediately (lines 1–3).

Otherwise, the top-most variable x of σ_{ns} and σ_s is determined (line 4). We check if x is a state variable. If this is the case we have not reached a signature node in both DDs yet. Therefore we have to traverse the DDs further by computing the cofactors w. r. t. x and calling the algorithm recursively. The result is then constructed by creating a new BDD node with the results of the recursive calls as children (lines 5–8).

If x is not a state variable the algorithm has reached two nodes which represent the non-stochastic and the stochastic signature, respectively, of the state corresponding to the path the algorithm has traversed in the BDDs. Furthermore, due to the usage of the ComputedTable we assure that this combination of non-stochastic and stochastic signature has not be seen before by the algorithm. Hence, the result is a new block number that is stored in the ComputedTable (line 13) before returning the result.

There is only one pitfall caused by state encodings that do not correspond to a real state of the system. These invalid state encodings come from the symbolic representation of the state space: the number of representable states is always $2^{\#\text{state bits}}$, so there are invalid states if the original state space is not a power of 2 or a redundant encoding is used. The signatures of all invalid states are empty, i. e. the corresponding paths in the DDs lead to the leaf 0. But there can also be

valid states with empty signatures—hence invalid states and states with empty signatures cannot be distinguished easily by the algorithm. We remember the number of the blocks that was assigned to the state encodings with empty signature instead (line 11). We can check after the refinement step if this block contains valid states. If not, then we delete this block from the refined partition.

4.5 Quotient Extraction

When we have reached the fixpoint of the bisimulation computation, the final step is the computation of the quotient system, which will be described now. It is based on using the block numbers as new state encodings.

Algorithm 8: QuotientExtraction(IMC M, Bisimulation $\mathcal{P}$)
Begin

$\mathcal{S}_{\text{new}}(s) \leftarrow [k \to s](\exists s : \mathcal{P}(s,k))$ (1)
$\mathcal{T}_{\text{new}}(s,a,t) \leftarrow [k \to t](\exists t : \mathcal{T}(s,a,t) \wedge \mathcal{P}(t,k))$ (2)
$\mathcal{T}_{\text{new}}(s,a,t) \leftarrow [k \to s](\exists s : \mathcal{T}_{\text{new}}(s,a,t) \wedge \mathcal{P}(s,k))$ (3)
If $\mathcal{P}$ is a branching bisimulation **Then** (4)
  $C_{\text{esc}}(s) \leftarrow [k \to s](\exists s : \mathcal{P}(s,k) \wedge \mathcal{S}_{\text{esc}}(s))$ (5)
  $\mathcal{T}_{\text{new}}(s,a,t) \leftarrow \mathcal{T}_{\text{new}}(s,a,t) \wedge \neg(a = \tau \wedge s = t \wedge C_{\text{esc}}(s))$ (6)
End If (7)
$\mathcal{R}_{\text{new}}(s,t) \leftarrow [t \to k]\Big(Q_{+}t.\mathcal{R}(s,t) \cdot \mathcal{P}(t,k)\Big)$ (8)
$\mathcal{R}_{\text{new}}(s,t) \leftarrow [s \to k]\Big((Q_{\max}s.\mathcal{R}_{\text{new}}(s,t) \cdot \mathcal{P}(s,k)\Big)$ (9)
Return $(\mathcal{S}_{\text{new}}, \mathcal{A}, \mathcal{T}_{\text{new}}, \mathcal{R}_{\text{new}})$ (10)
End

The notation $[k \to s](\mathcal{P})$ denotes the renaming of the k-variables to s-variables in the BDD $\mathcal{P}$. The remaining part of the algorithm strictly follows definition 6 of the bisimulation quotient.

4.6 Optimizations

We have developed three optimizations [18] for the non-stochastic case which can be applied if the blocks are refined one by one. This is the case for stochastic branching bisimulation. For this reason we will now show how the optimizations can be transfered to the stochastic context.

Block Forwarding After we have refined a block we replace it in the current partition by the result of the refinement. The refinement of the subsequent blocks is then based on a finer partition. This reduces the number of iterations and the number of blocks that have to be refined.

Split-driven Refinement Observations in the non-stochastic context have shown that—except for the first few iterations—only a very small fraction of the blocks is actually split. The vast majority remains unchanged. We suppose that this will also be the case in the stochastic setting.

The idea is the following: if a block C is split this influences only the stability of blocks that are connected with C by an observable transition and—depending on the type of the bisimulation—by a sequence of τ-steps. This is formalized in the backward signature:

Definition 13. *Let $M = (S, A, T, R)$ be a maximal-progress-cut IMC, $s \in S$, P a partition of S, and C a block of P. The strong/branching **backward signature** is given by*

$$\begin{aligned}\mathrm{bwsig}_s^{s/b}(P, C) = \{C' \in P \mid \exists s \in C\, \exists s' \in C : (\exists a \in A : (a, C) \in \mathrm{sig}_{ns}^{s/b}(P, s) \\ \vee \exists r > 0 : (r, C) \in \mathrm{sig}_s^{s/b}(P, s))\}.\end{aligned}$$

The backward signature of a block contains exactly those blocks that are contained in one of the signatures. These blocks can become unstable if C is split. This implies that we have to refine only those blocks which are contained in the backward signature of a block that has been split in the previous iteration.

If we want to compute the backward signature we encounter the same problem as for the normal signatures: We need to compute the inert τ-sequences. We cannot avoid the computation by refining only one block at a time because we walk backwards. But instead we can use an *over-approximation* of the backward signature. This will not influence the correctness of the algorithm but only cause the (unnecessary) refinement of some stable blocks. We will use the following over-approximation for stochastic branching bisimulation:

$$\begin{aligned}\mathrm{bwsig}_s^{*,b}(P, C) =& \{C' \in P \mid \exists s \in C\, \exists s' \in C\, \exists a \in A : s' \xrightarrow{a} s\} \\ &\cup \{C' \in P \mid \exists s' \in C'\, \exists s \in C : R(s', s) > 0\}.\end{aligned}$$

Algorithm 9 summarizes the changes to the algorithm for stochastic branching bisimulation. Hereby contains P the current partition and U the set of potentially unstable blocks.

Algorithm 9: StochasticBranchingBisimulation_opt(IMC M, partition $P^{(0)}$)
Begin

```
Compute the divergence sensitive initial partition P̃^(0)                (1)
P ← P̃^(0)                                                               (2)
U ← P                                                                   (3)
U_new ← ∅                                                               (4)
While U ≠ ∅ Do                                                          (5)
    For Each block C ∈ U Do                                             (6)
        P ← (P \ {C}) ∪ sigref_s^b(P, C)                                (7)
        U_new ← U_new \ {C}                                             (8)
        If C was split Then                                             (9)
            U_new ← U_new ∪ bwsig_s^{*,b}(C)                            (10)
        End If                                                          (11)
    End For                                                             (12)
```

$U \leftarrow U_{\text{new}}$ (13)
End While (14)
Return P (15)
End

Heuristics for the Block Order Experiments in the non-stochastic context have shown that the effectiveness of the block forwarding technique strongly depends on the order in which the potentially unstable blocks are refined in line 6 of algorithm 9. The idea is to provide a block order such that blocks having a great impact on the stability of others are split first. Then, more blocks may be split in one iteration. Before the actual refinement, we therefore sort the potentially unstable blocks w. r. t. one of the following block ordering heuristics:

- *SortByBWSig*: in decreasing order w. r. t. $\left|\bigcup \text{bwsig}^{*,b}(P, C)\right|$
- *SortByBlockSize*: in decreasing order w. r. t. $|C|$.

Please note that both heuristics can be computed efficiently using the BDD-function *satisfy_count* [32] that is linear in the size of the BDD.

5 Conclusion and Future Work

In this technical report we have described—to the best of our knowledge—the first fully symbolic algorithms for the computation of the coarsest stochastic strong bisimulation and stochastic branching bisimulation on an IMC that refines a given initial partition. The algorithms rely on a signature-based refinement of the initial partition until a fixpoint is reached. We have proven the correctness of the algorithms.

Furthermore, we have presented concepts for an efficient implementation of the algorithms using (MT)BDDs and we have shown how optimizations from the non-stochastic domain can be transferred to bisimulations on IMCs.

The next step will be the implementation of the algorithms and an experimental evaluation on practical benchmarks. We will also extend the signature refinement algorithm to stochastic weak bisimulation, whereby we expect that providing an appropriate signature will solve most of the problem.

References

1. Hermanns, H.: Interactive Markov Chains – The Quest for Quantified Quality. Vol. 2428 of Lecture Notes in Computer Science. Springer-Verlag (2002)
2. Buchholz, P.: Exact and ordinary lumpability in finite Markov chains. Journal of Applied Probability **31** (1994) 59–74
3. Buchholz, P.: Efficient computation of equivalent and reduced representations for stochastic automata. International Journal of Computer System Science & Engineering **15**(2) (2000) 93–103

4. Böde, E., Herbstritt, M., Hermanns, H., Johr, S., Peikenkamp, T., Pulungan, R., Wimmer, R., Becker, B.: Compositional perfomability evaluation for statemate. In: 3rd International Conference on Quantitative Evaluation of Systems (QEST), Riverside, CA, USA, IEEE Computer Society Press (2006)
5. Milner, R.: A Calculus of Communicating Systems. Vol. 92 of Lecture Notes in Computer Science. Springer-Verlag (1980)
6. Park, D.: Concurrency and automata on infinite sequences. In: GI Conference on Theoretical Computer Science. Vol. 104 of Lecture Notes in Computer Science., Springer-Verlag (1981)
7. van Glabbeek, R.J., Weijland, W.P.: Branching time and abstraction in bisimulation semantics. Journal of the ACM **43**(3) (1996) 555–600
8. Paige, R., Tarjan, R.E.: Three partition refinement algorithms. SIAM Journal on Computing **16**(6) (1987) 973–989
9. Bouali, A., de Simone, R.: Symbolic bisimulation minimisation. In von Bochmann, G., Probst, D.K., eds.: Proceedings of the 4th International Conference on Computer Aided Verification (CAV). Vol. 663 of Lecture Notes in Computer Science., Montreal, Canada, Springer Verlag (1992) 96–108
10. Klarlund, N.: An $n \log n$ algorithm for online BDD refinement. In Grumberg, O., ed.: 9th International Conference on Computer Aided Verification (CAV). Vol. 1254 of Lecture Notes in Computer Science., Haifa, Israel, Springer-Verlag (1997) 107–118
11. Dovier, A., Gentilini, R., Piazza, C., Policriti, A.: Rank-based symbolic bisimulation (and model checking). In Guerra, R.J., de Queiroz, B., eds.: Proceedings of the 9th Workshop on Logic, Language, Information, and Computation (WoLLIC). Vol. 67 of Electronic Notes in Theoretical Computer Science., Elsevier Science Publishers B. V. (2002)
12. Groote, J.F., Vaandrager, F.W.: An efficient algorithm for branching bisimulation and stuttering equivalence. In Paterson, M.S., ed.: Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP). Vol. 443 of Lecture Notes in Computer Science., Warwick, England, Springer-Verlag (1990) 626–638
13. Blom, S., Orzan, S.: Distributed branching bisimulation reduction of state spaces. In: 2nd International Workshop on Parallel and Distributed Model Checking (PDMC). Vol. 89(1) of Electronic Notes in Theoretical Computer Science., Elsevier (2003)
14. Blom, S., Orzan, S.: A distributed algorithm for strong bisimulation reduction of state spaces. International Journal on Software Tools for Technology Transfer (STTT) **7**(1) (2005) 74–86
15. Blom, S., Orzan, S.: Distributed state space minimization. International Journal on Software Tools for Technology Transfer (STTT) **7**(3) (2005) 280–291
16. Wimmer, R., Herbstritt, M., Becker, B.: Minimization of large state spaces using symbolic branching bisimulation. In Reorda, M.S., Novák, O., Staube, B., Kubátová, H., Kotásek, Z., Kubalík, P., Ubar, R., Bucek, J., eds.: Proceedings of the 9th IEEE Workshop on Design & Diagnostics of Electronic Circuits & Systems (DDECS), Prague, Czech Republic, IEEE Computer Society Press (2006) 9–14
17. Wimmer, R., Herbstritt, M., Hermanns, H., Strampp, K., Becker, B.: Sigref – a symbolic bisimulation tool box. In Graf, S., Zhang, W., eds.: Proceedings of the 4th International Symposium on Automated Technology for Verification and Analysis (ATVA). Vol. 4218 of Lecture Notes in Computer Science., Beijing, China, Springer-Verlag (2006) 477–492
18. Wimmer, R., Herbstritt, M., Becker, B.: Optimization techniques for BDD-based bisimulation minimization. In Zhou, H., Macii, E., eds.: Proceedings of the 17th ACM Great Lakes Symposium on VLSI, Stresa, Italy, ACM Press (2007) 405–410

19. Kemeney, J.G., Snell, J.L.: Finite Markov Chains. D. Van Nostrand Company, Inc. (1960)
20. Derisavi, S.: Solution of Large Markov Models using Lumping Techniques and Symbolic Data Structures. PhD thesis, University of Illinois, Urbana-Champaign (2005)
21. Derisavi, S., Hermanns, H., Sanders, W.H.: Optimal state-space lumping in Markov chains. Information Processing Letters **87**(6) (2003) 309–315
22. Derisavi, S.: A symbolic algorithm for optimal Markov chain lumping. In Grumberg, O., Huth, M., eds.: 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science, Braga, Portugal, Springer-Verlag (2007)
23. Derisavi, S.: Signature-based symbolic algorithm for optimal Markov chain lumping. In: Proceedings of the 4th International Conference on Quantitative Evaluation of Systems (QEST), IEEE Computer Society Press (2007) (to appear).
24. Garavel, H., Hermanns, H.: On combining functional verification and performance evaluation using CADP. In Eriksson, L.H., Lindsay, P.A., eds.: Proceedings of the International Symposium of Formal Methods in Europe (FME). Vol. 2391 of Lecture Notes in Computer Science., Copenhagen, Denmark, Springer-Verlag (2002)
25. Fernandez, J.C., Garavel, H., Kerbrat, A., Matescu, R., Mounier, L., Sighireanu, M.: CADP: A protocol verification toolbox. In Alur, R., Henzinger, T.A., eds.: Proceedings of the 8th International Conference on Computer Aided Verification (CAV). Number 1102 in Lecture Notes in Computer Science, New Brunswick, NJ, USA, Springer-Verlag (1996) 437–440
26. Garavel, H., Land, F., Matescu, R.: An overview of CADP 2001. European Association for Software Science and Technology (EASST) Newsletter **4** (2002) 13–24
27. Hermanns, H.: Some notes on the stochastic equivalences implemented in bcg_min (2005) (unpublished note).
28. Somenzi, F.: CUDD: CU Decision Diagram Package Release 2.4.1, University of Colorado at Boulder. (2006)
29. Wegener, I.: Branching Programs and Binary Decision Diagrams – Theory and Applications. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA (2000)
30. Burch, J.R., Clarke, E.M., Long, D.E., McMillan, K.L., Dill, D.L.: Symbolic model checking for sequential circuit verification. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **13**(4) (1994) 401–424
31. Matsunaga, Y., McGeer, P.C., Brayton, R.K.: On computing the transitive closure of a state transition relation. In: Proceedings of the 30th Design Automation Conference, Dallas, Texas, USA, ACM Press (1993) 260–265
32. Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. IEEE Transactions on Computers **35**(8) (1986) 677–691