Abstract Model Repair for Probabilistic Systems

G. Chatzieleftheriou[a], P. Katsaros[a,*]

[a]*Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece*

Abstract

Given a Discrete Time Markov Chain M and a probabilistic temporal logic formula φ, where M violates φ, the problem of *Model Repair* is to obtain a new model M', such that M' satisfies φ. Additionally, the changes made to M in order to obtain M' should be minimum with respect to all such M'. The state explosion problem makes the repair of large probabilistic systems almost infeasible. In this paper, we use the abstraction of Discrete Time Markov Chains in order to speed-up the process of model repair for temporal logic reachability properties. We present a framework based on abstraction and refinement, which reduces the state space of the probabilistic system to repair at the price of obtaining an approximate solution. A metric space is defined over the set of DTMCs, in order to measure the differences between the initial and the repaired models. For the repair, we introduce an algorithm and we discuss its important properties, such as soundness and complexity. As a proof of concept, we provide experimental results for probabilistic systems with diverse structures of state spaces, including the well-known Craps game, the IPv4 Zeroconf protocol, a message authentication protocol and the gambler's ruin model.

Keywords: Model Repair, Abstraction, Probabilistic Models

1. Introduction

Given a model M and a property φ, the problem of *model checking* is to find if the property is satisfied by the model [1]. Today, a number of mature model

*Corresponding author

Email addresses: gchatzie@csd.auth.gr (G. Chatzieleftheriou), katsaros@csd.auth.gr (P. Katsaros) checking algorithms exist, for probabilistic and non probabilistic models. If the property is not satisfied, some algorithms return a cause for the refutation of the property known as counterexample.

The problem of *model repair* is an extension of the model checking problem for the case where the property is refuted. More specifically, the aim of model repair is to find the minimal changes to the model, such that the property φ, which has been violated in the original model, will be satisfied. The model repair problem has been examined in the probabilistic setting for the first time in [2].

The *state space explosion* problem is inherent in model checking and makes its application infeasible to large models. This problem is also present in existing probabilistic model repair techniques, which aim to directly change the model under repair. For example, in [2] the authors transform the repair problem to a non-linear optimization problem using parametric model checking, and the time needed for computing a repaired model increases rapidly with respect to the size of the state space.

The main method for fighting the state space explosion in model checking is the use of *abstraction* techniques [3, 4, 5]. Such an approach has also been used in the non-probabilistic setting, for the repair of models with large state spaces [6, 7]. Inspired from the use of abstraction in model checking and non-probabilistic model repair, we present here a framework based on abstraction and refinement for the repair of probabilistic models. More specifically, we focus on Discrete Time Markov Chains (DTMCs) and their repair with respect to temporal logic reachability properties.

The main contributions of our paper are:

- We introduce a framework for the repair of a DTMC with respect to a (not-nested) Probabilistic Computation Tree Logic (PCTL) reachability formula, using an Abstract Discrete Time Markov Chain (ADTMC) for the given DTMC and the 3-valued semantics of PCTL over ADTMCs. Based on a strong preservation theorem, if a PCTL property is refuted or satisfied in the ADTMC (abstract model), then the same also holds for the concrete DTMC [8].

- A metric space is defined over the DTMCs with the same state labeling, in order to measure the distance of repaired DTMCs from the original DTMC.

- We introduce a Probabilistic Abstract Model Repair (PAMR) algorithm that transforms the DTMC repair problem to a non-linear minimization problem for the state space of the abstract model, instead of the concrete one. If a solution is found, the repaired DTMC is returned, which corresponds to an approximate (not the optimal) solution; otherwise, the algorithm is iteratively applied to refined ADTMCs until a solution is found. The refinement can be potentially adapted by the analyst, for implementing alternative repair strategies.

- We analyze the PAMR computational gains and more specifically the achieved reduction in the expensive non-linear optimization and linear equation solving problems, which are involved respectively in the concrete model repair and model checking techniques.

- As a proof of concept, we provide experimental results for the DTMCs of extended versions of the Craps game, the IPv4 Zeroconf protocol, a message authentication protocol and the gambler's ruin model.

The paper is organized as follows. In Section 2, the notion of DTMC is introduced which is the formalism for the concrete model in our framework. Section 3 discusses how an ADTMC can serve as an abstraction of a DTMC and how a reachability PCTL formula can be verified in an ADTMC. In Section 4, the model repair problem for probabilistic systems is formulated together with a metric space for DTMCs. We present the abstract model repair process for probabilistic systems in Section 5 together with the basic model repair operations. The PAMR algorithm is described in Section 6. The algorithm's steps are illustrated using an application in Section 7, where we also elaborate on the method's efficiency gains, its cost in terms of the solution's optimality and its flexibility perspectives. In Section 8, we present the experimental results for extended models with progressively larger state spaces of the Craps game, the IPv4 Zeroconf protocol, a message authentication protocol and the gambler's ruin model. The related work is reviewed in Section 9 and we conclude with Section 10, where we also discuss the future work.

2. Reachability PCTL properties over DTMCs

Let AP be a set of *atomic propositions* and the set Lit of *literals* given as:

$$Lit = AP \cup \{\neg p : p \in AP\}$$

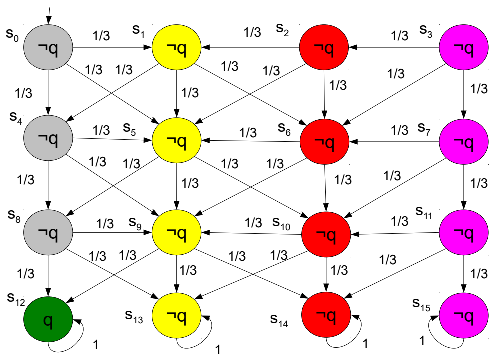


Figure 1: A Discrete Time Markov Chain.

Definition 1. A (labeled) *Discrete Time Markov Chain* (DTMC) is a 4-tuple $M = (S, s_{init}, P, L)$, where:

1. S is a finite set of states;
2. $s_{init} \in S$ is the initial state;
3. $P : S \times S \rightarrow [0, 1]$ is a transition probability function with $\sum_{s' \in S} P(s, s')$ $= 1$ for all $s \in S$;
4. $L : S \rightarrow 2^{Lit}$ is a state *labeling function* such that $\forall s \in S$, $\forall p \in AP$, $p \in L(s) \Leftrightarrow \neg p \notin L(s)$.

A DTMC is a transition system with labeled states and probabilities assigned to its transitions.

Example. (ROBOT) We use a robot system as a running example. The DTMC for the robot system is shown in Fig. 1. The robot moves among states with different color and tries to reach a state, where the color will be *green*. The robot is "fair", in the sense that for each state the transition probabilities to the directly accessible states are equal $\left(\frac{1}{3}\right)$. The set of atomic propositions for the robot DTMC is $AP = \{q\}$ with $q \equiv (color = green)$. $\square$

Definition 2. A *path* π in a DTMC $M = (S, s_{init}, P, L)$ is an (infinite) sequence $\pi = [s_0, s_1.s_2, ...]$; we denote by $\pi[i]$ the (i+1)-th state of π. A *path fragment* $\rho = [s_0, s_1.s_2, ..., s_n]$ is a finite prefix of a path.

Definition 3. Let $Path_s^M$ be the set of all paths starting from state s. Let Ω be the set $Path^M = \bigcup_{s \in S} Path_s^M$ and $Cyl(s_0, ..., s_k)$ be the (*cylinder*) *set* *of states $s_0, ..., s_k$, i.e. the set of all paths in $Path^M$ with prefix $s_0, ..., s_k$. For any such Cyl, a probability measure x is defined such that:*

$$x(Cyl(s_0, ..., s_k)) = \prod_{i=0}^{k-1} P(s_i, s_{i+1})$$

The cylinder sets include paths in DTMCs, and their probability measure is used to evaluate the probability of reachability properties, which can be expressed in Probabilistic Computation Tree Logic (PCTL).

Definition 4. The syntax of the reachability fragment of PCTL over a set of atomic propositions AP is given by the following grammar:

$$\varphi ::= P_{\bowtie p}[F\psi] \mid P_{\bowtie p}[F^{\leq k}\psi]$$

$$\psi ::= true|q|\neg\psi|\psi \vee \psi|\psi \wedge \psi$$

where $q \in AP$, $p \in [0,1]$, $k \in \mathbb{N}$ and, $\bowtie \in \{<, \leq, \geq, >\}$

Def. 4 is focused on reachability properties of PCTL without nested probabilistic quantifiers. The model checking problem for reachability properties is reduced to the solution of a system of equations, as it is shown in Def. 5 for unbounded properties and in Def. 6 for bounded properties .

Definition 5. Let $M = (S, s_{init}, P, L)$ be a DTMC. The probability measure x_s of a state $s \in S$ to satisfy the property $F\psi$ is defined as:

$$x_s = \begin{cases} 1 & \text{if } (M,s) \models \psi \\ 0 & \text{if there is no path from } s \text{ to a state satisfying } \psi \\ \sum_{t \in S} P(s,t) \cdot x_t & \text{otherwise} \end{cases}$$

Definition 6. Let $M = (S, s_{init}, P, L)$ be a DTMC. The probability measure x_s^k of a state $s \in S$ to satisfy the property $F^{\leq k}\psi$ with $k \in \mathbb{N}$ is defined as:

$$x_s^k = \begin{cases} 1 & \text{if } (M,s) \models \psi \\ 0 & \text{if there is no path from } s \text{ to a state satisfying } \psi \\ & \text{or there is a path from } s \text{ to a state satisfying } \psi \\ & \text{and } k = 0 \\ \sum_{\mathrm{t} \in \mathrm{S}} \mathrm{P(s,t)} \cdot \mathrm{x_t^{k-1}} & \text{otherwise} \end{cases}$$

3. Abstract DTMCs

Definition 7. Given a transition probability function $P : S \times S \rightarrow [0,1]$ defined over a finite set of states S, the *extended* transition probability function $\hat{P} : S \times 2^S \rightarrow [0,1]$ is defined, such that $\forall s \in S, S' \subseteq S$, $\hat{P}(s, S') = \sum_{s' \in S'} P(s, s')$.

Definition 8 ([8]). An *Abstract Discrete Time Markov Chain* (ADTMC) is a 5-tuple $\hat{M} = (\hat{S}, \hat{s}_{init}, P^{\ell}, P^{u}, \hat{L})$, where:

1. $\hat{S}$ is a finite set of *states*;
2. $\hat{s}_{init} \in \hat{S}$ is an initial state;
3. $P^{\ell} : \hat{S} \times \hat{S} \rightarrow [0,1]$ and $P^{u} : \hat{S} \times \hat{S} \rightarrow [0,1]$ are two transition probability functions, such that for all $\hat{s} \in \hat{S}$: $\hat{P}^{\ell}(\hat{s}, \hat{S}) \leq 1 \leq \hat{P}^{u}(\hat{s}, \hat{S})$ and $P^{\ell}(\hat{s}, \hat{s}') \leq P^{u}(\hat{s}, \hat{s}')$ for all $\hat{s}' \in \hat{S}$;
4. $\hat{L} : \hat{S} \rightarrow 2^{Lit}$ is a state-labeling, such that $\forall \hat{s} \in \hat{S}$, $\forall p \in AP$, $\hat{s}$ is labeled by at most one of p and $\neg p$.

In an ADTMC, the exact transition probability between any two states is not defined explicitly, but instead an upper and a lower bound for this probability is given. We therefore call the P^{ℓ} and P^{u} as *lower* and *upper* transition probability functions. Consequently, the verification of a PCTL formula φ over an ADTMC may yield an indefinite answer ($\perp$). The semantics of a reachability PCTL formula φ at a state $\hat{s}$ of an ADTMC $\hat{M}$ is given in Def. 9 (for a bounded reachability property $P_{\bowtie p}[F^{\leq k}\psi]$ the semantics differs in that only the finite paths with length k are taken into account).

Definition 9 ([8]). Let $\hat{M} = (\hat{S}, \hat{s}_{init}, P^{\ell}, P^{u}, \hat{L})$ be a ADTMC. The semantics of a reachability PCTL formula $\varphi = P_{\bowtie p}[F\psi]$ at a state $\hat{s}$ of $\hat{M}$, denoted as $[\![(\hat{M}, \hat{s}) \models^3 \varphi]\!]$, is defined as follows:

- If $\varphi = P_{\leq p}[F\psi]$
  - $[\![(\hat{M}, \hat{s}) \models^3 \varphi]\!] = \mathit{true}$, if $x^{u}(\{\pi \in Path_{\hat{s}}^{\hat{M}} \mid \exists i \geq 0. \pi[i] \models \psi\}) \leq p$.
  - $[\![(\hat{M}, \hat{s}) \models^3 \varphi]\!] = \mathit{false}$, if $x^{\ell}(\{\pi \in Path_{\hat{s}}^{\hat{M}} \mid \exists i \geq 0. \pi[i] \models \psi\}) > p$.
  - $[\![(\hat{M}, \hat{s}) \models^3 \varphi]\!] = \perp$, otherwise.
- If $\varphi = P_{\geq p}[F\psi]$

- $[\![(\hat{M}, \hat{s}) \models^3 \varphi]\!] = \mathit{true}$, if $x^\ell(\{\pi \in \mathit{Path}_{\hat{s}}^{\hat{M}} \mid \exists i \geq 0.\pi[i] \models \psi\}) \geq p$.
- $[\![(\hat{M}, \hat{s}) \models^3 \varphi]\!] = \mathit{false}$, if $x^u(\{\pi \in \mathit{Path}_{\hat{s}}^{\hat{M}} \mid \exists i \geq 0.\pi[i] \models \psi\}) < p$.
- $[\![(\hat{M}, \hat{s}) \models^3 \varphi]\!] = \perp$, otherwise.

where x^ℓ, x^u are the probability measures for the *lower* and *upper* transition probability functions. □

From the 3-valued PCTL semantics over ADTMCs, it follows that the truth of $P_{\geq p}[F\psi]$ is checked based on the paths with lower bound probabilities, as opposed to $P_{\leq p}[F\psi]$, for which the paths with the upper bound probabilities are used. For checking the refutation of $P_{\geq p}[F\psi]$, the upper bound probabilities are used, whereas for the refutation of $P_{\leq p}[F\psi]$ the check is based on the lower bound probabilities.

In Def. 10, given a DTMC M and a pair of total functions α (abstraction) and γ (concretization), an ADTMC $\alpha(M)$ is defined with state space the set $\hat{S}$, derived by abstracting the concrete state space S of M. The ADTMC $\alpha(M)$ serves as the abstract model and the DTMC M as the concrete model in our framework.

Definition 10 ([8]). Given a DTMC $M = (S, s_{init}, P, L)$ and a pair of total functions $(\alpha : S \to \hat{S}, \gamma : \hat{S} \to 2^S)$ such that

$$\forall s \in S, \forall \hat{s} \in \hat{S} : \alpha(s) = \hat{s} \Leftrightarrow s \in \gamma(\hat{s})$$

the ADTMC $\alpha(M) = (\hat{S}, \hat{s}_{init}, P^\ell, P^u, \hat{L})$ is defined as follows:

1. $\hat{s}_{init} = \alpha(s_{init})$;
2. for all $\hat{s}$, $\mathit{lit} \in \hat{L}(\hat{s})$ only if for all $s \in \gamma(\hat{s})$, $\mathit{lit} \in L(s)$;
3. for all $\hat{s}_1, \hat{s}_2$, $P^\ell(\hat{s}_1, \hat{s}_2) = \inf_{s \in \gamma(\hat{s}_1)} \hat{P}(s, \gamma(\hat{s}_2))$;
4. for all $\hat{s}_1, \hat{s}_2$, $P^u(\hat{s}_1, \hat{s}_2) = \min(1, \sup_{s \in \gamma(\hat{s}_1)} \hat{P}(s, \gamma(\hat{s}_2))$.

A state $\hat{s}$ of $\alpha(M)$ is an abstract state of some $s \in S$, if and only if $\hat{s} = \alpha(s)$ (equivalently $s \in \gamma(\alpha(s))$). The abstract state $\hat{s}_{init}$ of the concrete state s_{init} of the DTMC is initial in $\alpha(M)$. Proposition $q \in AP$ is true (resp. false) in $\hat{s} = \alpha(s)$, if and only if q is true (resp. false) in all $s \in \gamma(\hat{s})$. Otherwise, q is unknown ($\perp$) at $\hat{s}$. $P^\ell(\hat{s}_1, \hat{s}_2)$ for two abstract $\hat{s}_1$, $\hat{s}_2$ is defined as the infinum of the transition probabilities from each $s_i \in \gamma(\hat{s}_1)$ to all $s_j \in \gamma(\hat{s}_2)$. On the other hand, $P^u(\hat{s}_1, \hat{s}_2)$ for $\hat{s}_1$, $\hat{s}_2$ is defined as the minimum of 1

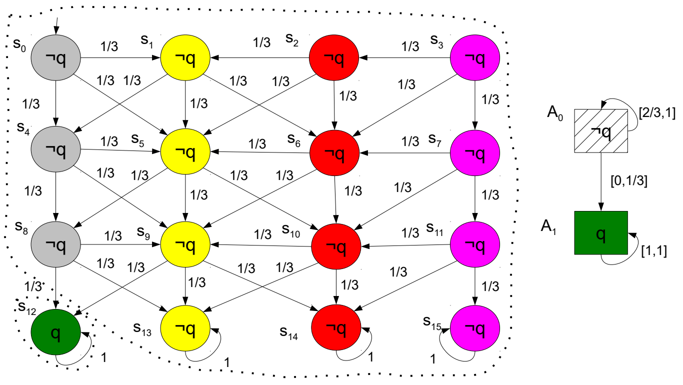


Figure 2: The abstraction of a Discrete Time Markov Chain.

and the supremum of the transition probabilities from each $s_i \in \gamma(\hat{s}_1)$ to all $s_j \in \gamma(\hat{s}_2)$.

Example. (ROBOT) In Fig. 2, an abstract DTMC of the robot system is presented where the green state (labeled with q) and the non-green states (labeled with $\neg q$) are abstracted using one abstract state for each case. The lower bound of the outgoing transition from the non-green abstract state A_0 to A_1 is 0, because there are no concrete non-green states with outgoing transition of probability greater than 0 to the concrete green state. The upper bound for the same transition is $\left(\frac{1}{3}\right)$, because the greatest transition probability from a concrete non-green state to the concrete green state is $\left(\frac{1}{3}\right)$. All other lower and upper bound probabilities in Fig. 2 are calculated in the same way. □

Theorem 1 ([8]). *Let $M = (S, s_{init}, P, L)$ a DTMC and $\alpha(M) = (\hat{S}, \hat{s}_{init}, P^{\ell}, P^{u}, \hat{L})$ the ADTMC as in Def. 10. Then for every PCTL formula φ and every $s \in S, \hat{s} \in \hat{S}$ with $\hat{s} = \alpha(s)$ we have:*

$$[\![(\alpha(M), \hat{s}) \models \varphi]\!] \neq \bot \Rightarrow [\![(M, s) \models \varphi]\!] = [\![(\alpha(M), \hat{s}) \models \varphi]\!]$$

From Theorem 1, it follows that if a PCTL formula is satisfied (or violated) in the ADTMC, then it is also satisfied (or violated) in the DTMC. If the answer of model checking a PCTL formula over the ADTMC is unknown, then there is no definite conclusion for the same formula over the DTMC. To be concise, in the rest of the paper we will denote $[\![(\alpha(M), \hat{s}) \models \varphi]\!] = \mathit{false}$ by just writing $(\alpha(M), \hat{s}) \not\models \varphi$.

4. The Model Repair problem for probabilistic systems

In this section, we define a metric space for measuring the distance between DTMCs with the same state labeling, and then the model repair problem for DTMCs [2].

Definition 11. For any two vectors $A = (a_1, ..., a_n)$ and $B = (b_1, ..., b_n)$ with length n, the Manhattan distance d_m is defined as follows:

$$d_m(A, B) = \sum_{i=1}^{n} |a_i - b_i|$$

. □

Definition 12. For any two $M = (S, s_{init}, P, L)$ and $M' = (S, s_{init}, P', L)$ in the set D_M of all DTMCs with the same state space S and the same labeling function L, the *distance function* d over D_M is defined as follows:

$$d(M, M') = d_m(A, B)$$

with $A = (P(s_0, s_0), P(s_0, s_1), \ldots, P(s_i, s_j) \ldots P(s_n, s_n))$
and $B = (P'(s_0, s_0), P'(s_0, s_1), \ldots, P'(s_i, s_j) \ldots P'(s_n, s_n))$ for all $s_i, s_j \in S$.

For any two DTMCs with the same state space S and the same labeling function L, the function d measures the differences between the corresponding transition probabilities.

Proposition 2. *The ordered pair (D_M, d) is a metric space.*

PROOF. This is inferred from the fact that the Manhattan distance between any two vectors is a distance metric. □

Definition 13. Given a DTMC M and a PCTL formula φ where $M \not\models \varphi$, the model repair problem is to find a DTMC M', such that $M' \models \varphi$ and $d(M, M')$ is minimum with respect to all such M'.

The objective of the model repair problem in the context of probabilistic systems is to modify the given DTMC, in order to satisfy a PCTL property φ, which is not satisfied. The repair solution should have the smallest possible distance from the initial model compared to all other models satisfying φ. In the following sections we show how the model repair problem is transformed into a minimization problem for the changes of the transition probabilities in the DTMC.

5. The PAMR framework

The PAMR framework uses 3-valued abstract probabilistic models (ADTMCs) for the repair of DTMCs with respect to given probabilistic reachability properties. The goal of our framework is to provide a method that looks for the repair solution with the smallest possible distance from a given DTMC. This method should be feasible and efficient for models with large state spaces.

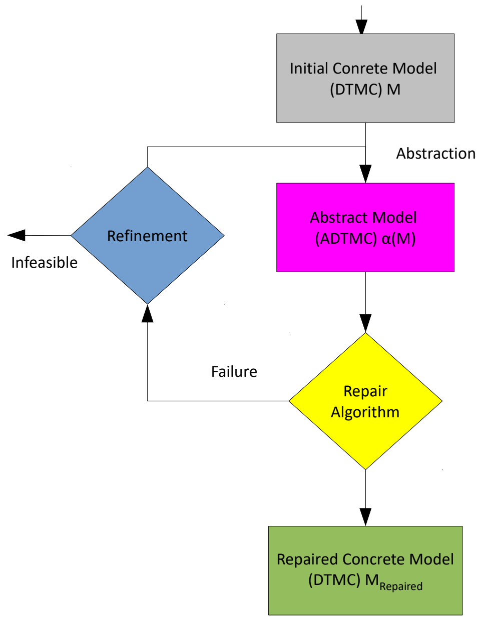


Figure 3: Probabilistic Abstract Model Repair Framework.

The process consists of the following steps, which are shown in Fig. 3.

Step 1. For a DTMC M, a state s and a property $\varphi = P_{\leq p}[F\psi]$ or $\varphi = P_{\geq p}[F\psi]$ (or their bounded variants), such that $(M, s) \not\models \varphi$, an ADTMC $\hat{M} = \alpha(M)$ is acquired as in Def. 10.

Step 2. The PAMR algorithm is called with inputs M, $\hat{M}$, $\hat{s} = \alpha(s)$ and the property φ.

Case 1. If the result is FAILURE, i.e. a repair solution is not found, then a *refinement* step takes place for $\hat{M}$, and:

Case 1.1. If an ADTMC $\hat{M}_{Refined}$ is found, then the control is transfered to Step 2 with $\hat{M}_{Refined}$ as input.

Case 1.2. If it is not feasible to retrieve a new refined ADTMC, the repair process terminates with a FAILURE.

Case 2. A repaired DTMC $M_{Repaired}$ is found.

Example. (ROBOT) The initial abstraction for the DTMC of the robot system is shown in Fig. 2. In this case, a predicate abstraction is used where the concrete state s_{12}, which is the only one where q is true, is abstracted to one state, while all the other states which are not labeled with q are abstracted to a different state. Thus, a two-state ADTMC is constructed which is the first abstract model arising from the concrete DTMC M. □

5.1. Refinement

The refinement step plays a fundamental role in the PAMR process, since it is the means to obtain a repair solution, if possible, when the PAMR algorithm terminates with a FAILURE result in a previous step. Moreover, even if a repair solution has been already found, it may be possible to apply a refinement step in order to find a more fine-grained repair solution.

In the related bibliography, there are various proposals for the refinement of models for probabilistic systems [9, 10, 11, 12, 13, 14]. The ultimate goal of such a technique is to get an *optimally* refined model with respect to some criteria, which differ in each work. We adopt a simple, yet effective refinement technique, which fulfills the following objective: the refinement procedure monotonically converges to the concrete DTMC with respect to the size of the model's state space. To this end, at each refinement step we split the abstract states invalidating the atomic propositions of the reachability property.

Nevertheless, the PAMR process can be adapted through the adoption of alternative refinement methods according to different criteria (the so-called *repair strategies* [2]).

Example. (ROBOT) From the ADTMC in Fig. 2 for the robot system, a refined model can be derived by splitting the abstract state not satisfying q in two states, thus getting a three-state ADTMC. The refined ADTMC and the corresponding partitioning of the DTMC's state space are shown in Fig. 4.

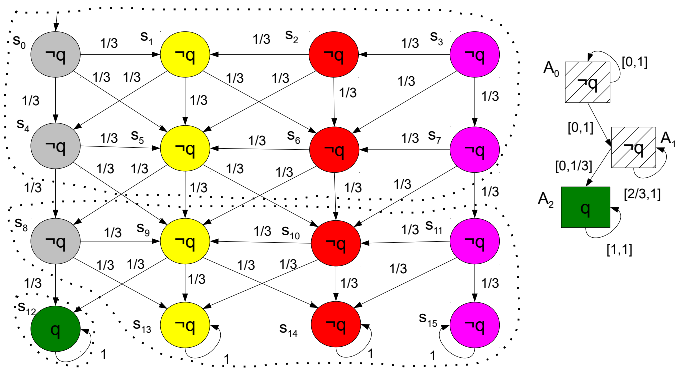


Figure 4: DTMC and ADTMC after the first refinement step.

The ADTMC of Fig. 4 can be further refined by splitting both abstract states that do not satisfy q in two states. This second refinement and the corresponding partitioning of the DTMC's state space are shown in Fig. 5. The result obtained from the third refinement step is shown in Fig. 6, from which a subsequent refinement yields the concrete DTMC.

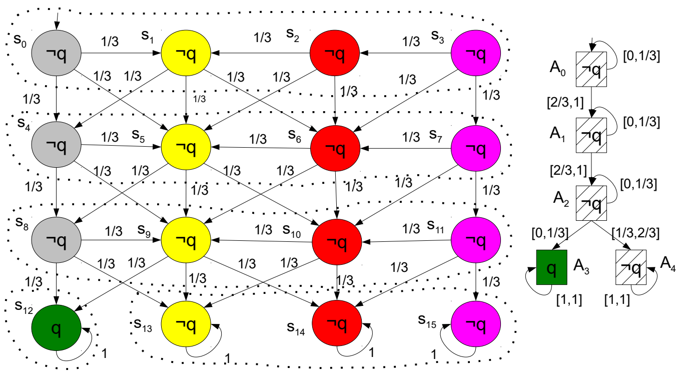


Figure 5: DTMC and ADTMC after the second refinement step.

The impact of the refinement method on the PAMR process is better illustrated if we consider the ADTMC obtained from Fig. 2 using an alternative refinement. The result is shown in Fig. 7, which differs from the ADTMC in Fig. 4 and will eventually lead to a different repair solution. □

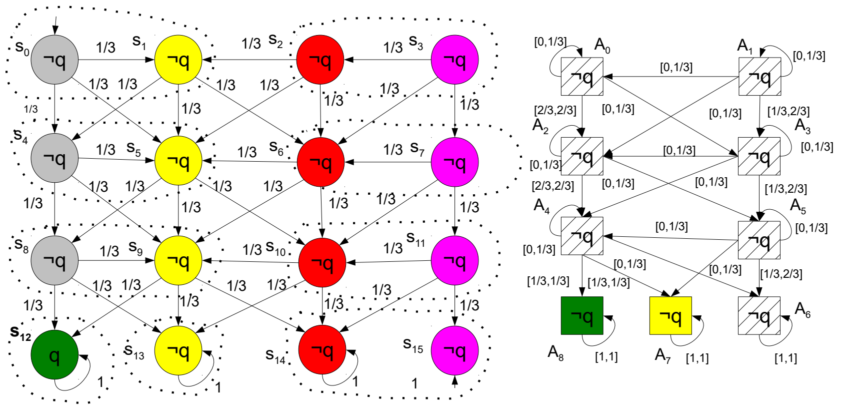


Figure 6: DTMC and ADTMC after the third refinement step.

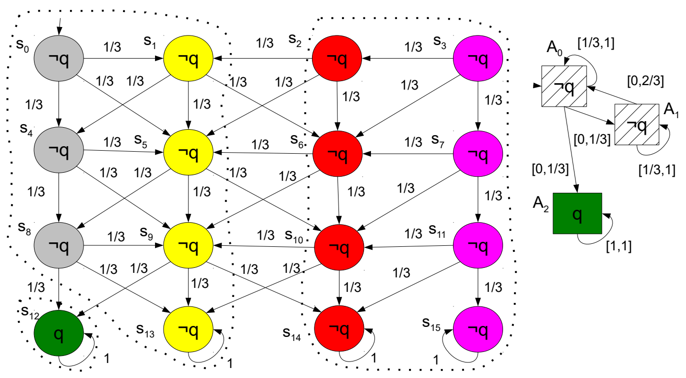


Figure 7: DTMC and ADTMC for a different refinement from Fig. 4.

5.2. Repair operations for ADTMCs

In the PAMR process, the model repair problem is reduced to a problem addressed through a smaller state space, i.e. that of the abstract ADTMC. Every change to the transition probabilities of the ADTMC by the PAMR algorithm is mapped to the concrete DTMC through the pair of functions (α, γ) in Def. 10.

Repairing the DTMC with respect to the reachability property $\varphi = P_{\geq p}[F\psi]$ means, according to the 3-valued semantics of PCTL in Def. 9, that specific lower bound transition probabilities of the ADTMC should be increased (similarly in the case of $P_{\geq p}[F^{\leq k}\psi]$). When the DTMC is repaired with respect to $\varphi = P_{\leq p}[F\psi]$, specific upper bound transition probabilities should be decreased. We hereby introduce two distinct repair operations applicable to ADTMCs, the IncreaseLowerBound and DecreaseUpperBound.

Definition 14 (IncreaseLowerBound). Let $M = (S, s_{init}, P, L)$ a DTMC and the ADTMC derived from M as in Def. 10 be $\alpha(M) = (\hat{S}, \hat{s}_{init}, P^{\ell}, P^{u}, \hat{L})$. For a given $v_0 \in \mathbb{R}_+$ and a pair $\hat{s}_i, \hat{s}_j \in \hat{S}$, let $prob_{min} = P^{\ell}(\hat{s}_i, \hat{s}_j) + v_0$. Then, $\forall s_m \in S \setminus \gamma(\hat{s}_i), s_n \in S, P'(s_m, s_n) = P(s_m, s_n)$, whereas for all $s_i \in \gamma(\hat{s}_i), s_j \in \gamma(\hat{s}_j), s_k \in S \setminus \gamma(\hat{s}_j)$:

$$P'(s_i, s_j) = \begin{cases} P(s_i, s_j) + \mathit{diff}/card_j & \text{if } \mathit{diff} = prob_{min} - prob_{out} > 0 \\ P(s_i, s_j) & \text{otherwise} \end{cases}$$

$$P'(s_i, s_k) = \begin{cases} P(s_i, s_k) - \mathit{diff}/card_k & \text{if } \mathit{diff} = prob_{min} - prob_{out} > 0 \\ P(s_i, s_k) & \text{otherwise} \end{cases}$$

with $prob_{out} = \sum_{s_i \in \gamma(\hat{s}_i), s_j \in \gamma(\hat{s}_j)} P(s_i, s_j)$, $card_j = |\{(s_i, s_j)\}|$ and $card_k = |\{(s_i, s_k)\}|$.

If P' fulfills the stochastic conditions, thus $P'(s_i, s_j), P'(s_i, s_k) \in [0, 1]$ for all s_i, s_j, s_k, then the DTMC $M' = (S, s_{init}, P', L) = \mathit{IncreaseLowerBound}$ $(M, \alpha(M), (\hat{s}_i, \hat{s}_j), v_0)$ is defined.

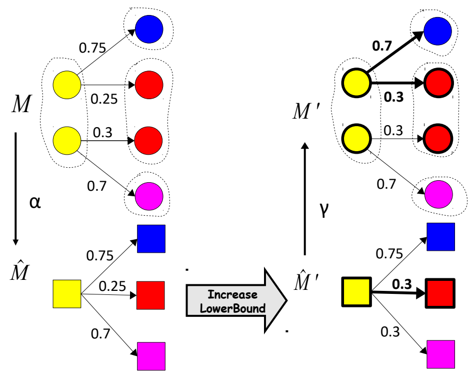


Figure 8: Increasing the lower bound of an ADTMC and the resulting concrete DTMC. (The lower bound probabilities of the ADTMC are only shown.)

The IncreaseLowerBound operation, which is illustrated in Fig. 8, increases the lower bound probability of a transition $(\hat{s}_i, \hat{s}_j)$ of the ADTMC (yellow to red transition of $\hat{M}$) by some given value $v_0 \in \mathbb{R}_+$ through modifying the DTMC M; this is feasible for the given v_0, if and only if:

$$\hat{P}(s_i, S_j) + v_0 \leq 1, \text{ for all } s_i \in \gamma(\hat{s}_i), \text{ with } S_j = \{s_j : s_j \in \gamma \hat{s}_j\} \qquad (1)$$

$$\hat{P}(s_i, S_k) - v_0 \geq 0, \text{ for all } s_i \in \gamma(\hat{s}_i), \text{ with } S_k = \{s_k : s_k \in S \setminus \gamma(\hat{s}_j)\} \qquad (2)$$

i.e. there is a concretization strategy for the given v_0, such that a DTMC exists (stochastic constraints are satisfied). Def. 14 proposes a particular concretization strategy: from all concrete s_i (yellow states of M) of the abstract $\hat{s}_i$, their outgoing transition probabilities to s_j (red states of M) of the abstract $\hat{s}_j$ are increased uniformly to fulfill Def. 10 for the ADTMC with the increased lower bound probability. All other probabilities for outgoing transitions from s_i to s_k (blue state of M) are decreased uniformly, in order to preserve the stochastic conditions; if this is not feasible for the given v_0, then the IncreaseLowerBound operation fails to return a DTMC M'.

In fact, the ADTMC entails multiple DTMCs, and thus a repaired ADTMC allows for a multitude of ways to repair the DTMC. Apart from the concretization strategy of Def. 14, other strategies could be applied either, (i) interactively by the user, or (ii) automatically, in order to find a DTMC, i.e. the stochastic conditions to be satisfied.

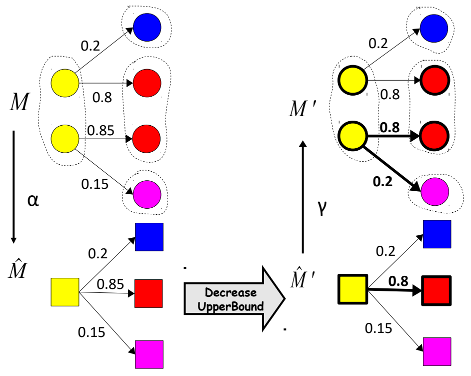


Figure 9: Decreasing the upper bound of an ADTMC and the resulting concrete DTMC. (The upper bound probabilities of the ADTMC are only shown.)

Definition 15 (DecreaseUpperBound). Let $M = (S, s_{init}, P, L)$ a DTMC and the ADTMC derived from M as in Def. 10 be $\alpha(M) = (\hat{S}, \hat{s}_{init}, P^{\ell}, P^{u}, \hat{L})$. For a given $v_0 \in \mathbb{R}_+$ and a pair $\hat{s}_i, \hat{s}_j \in \hat{S}$, let $prob_{max} = P^{u}(\hat{s}_i, \hat{s}_j) - v_0$. Then, $\forall s_m \in S \setminus \gamma(\hat{s_i}), s_n \in S, P'(s_m, s_n) = P(s_m, s_n)$, whereas for all $s_i \in \gamma(\hat{s_i}), s_j \in \gamma(\hat{s_j}), s_k \in S \setminus \gamma(\hat{s_j})$:

$$P'(s_i, s_j) = \begin{cases} P(s_i, s_j) - \mathit{diff}/card_j & \text{if } \mathit{diff} = prob_{out} - prob_{max} > 0 \\ P(s_i, s_j) & \text{otherwise} \end{cases}$$

$$P'(s_i, s_k) = \begin{cases} P(s_i, s_k) + \mathit{diff}/card_k & \text{if } \mathit{diff} = prob_{out} - prob_{max} > 0 \\ P(s_i, s_k) & \text{otherwise} \end{cases}$$

with $prob_{out} = \sum_{s_i \in \gamma(\hat{s}_i), s_j \in \gamma(\hat{s}_j)} P(s_i, s_j)$, $card_j = |\{(s_i, s_j)\}|$ and $card_k = |\{(s_i, s_k)\}|$.

If P' fulfills the stochastic conditions, thus $P'(s_i, s_j), P'(s_i, s_k) \in [0, 1]$ for all s_i, s_j, s_k, then the DTMC $M' = (S, s_{init}, P', L) = \mathit{DecreaseUpperBound}$ $(M, \alpha(M), (\hat{s}_i, \hat{s}_j), v_0)$ is defined.

The DecreaseUpperBound operation, which is illustrated in Fig. 9, decreases the upper bound probability of a transition $(\hat{s}_i, \hat{s}_j)$ of the ADTMC (yellow to red transition of $\hat{M}$) by some given value $v_0 \in \mathbb{R}_+$ by modifying the DTMC M; this is feasible for the given v_0 if and only if equations (1) and (2) hold.

6. The PAMR algorithm

The algorithm is executed at Step 2 of the PAMR process. For brevity, we present the PAMR algorithm for the case of unbounded properties, whereas for the bounded properties there are minor differences that we discuss at the end. The algorithm accepts as input the concrete DTMC M, the abstract ADTMC $\hat{M} = \alpha(M)$, a state $\hat{s} = \alpha(s)$ and a reachability property $\varphi = P_{\geq p}[F\psi]$ or $\varphi = P_{\leq p}[F\psi]$ such that $(M, s) \not\models \varphi$. The main body of the algorithm with two possible execution paths, one for $\varphi = P_{\geq p}[F\psi]$ and the other for $\varphi = P_{\leq p}[F\psi]$, is shown in Algorithm 1.

If $\varphi = P_{\geq p}[F\psi]$, then AddV$_0$LowerToADTMC is initially called with M, $\hat{M}$, $\hat{s}$ and φ as arguments and returns a transition probability function $P^{\ell}_{v_0}$. This function modifies $\hat{M}$'s lower bound transition probabilities according to a repair strategy defining with a parameter v_0 which transition probabilities are

Algorithm 1 PAMR

```
Input: $M = (S, s_{init}, P, L)$, $\hat{M} = \alpha(M) = (\hat{S}, \hat{s}_{init}, P^{\ell}, P^{u}, \hat{L})$, $\hat{s} \in \hat{S}$ and a
    property $\varphi = P_{\geq p}[F\psi]$ or $\varphi = P_{\leq p}[F\psi]$ such that $(\hat{M}, \hat{s}) \not\models \varphi$.
Output: $M' = (S, s_{init}, P', L)$ such that $(M', s) \models \varphi$ or FAILURE.
1:  if $\varphi = P_{\geq p}[F\psi]$ then
2:      $P^{\ell}_{v_0}$ := AddV$_0$LowerToADTMC$(M, \hat{M}, \hat{s}, \varphi)$
3:      $eq\_set_{v_0}$ := LowerReachabilityEquations$(\hat{M}, P^{\ell}_{v_0}, \hat{s}, \varphi)$
4:      $v_0$ := NLPSolve$(min(v_0), eq\_set_{v_0}, x \geq p, v_0 > 0)$
5:      return ConcretizeLowerRepairedModel$(M, \hat{M}, P^{\ell}_{v_0}, \hat{s}, \varphi, v_0)$
6:  else if $\varphi = P_{\leq p}[F\psi]$ then
7:      $P^{u}_{v_0}$ := SubtractV$_0$UpperToADTMC$(\hat{M}, \hat{s}, \varphi)$
8:      $eq\_set_{v_0}$ := UpperReachabilityEquations$(\hat{M}, P^{u}_{v_0}, \hat{s}, \varphi)$
9:      $v_0$ := NLPSolve$(min(v_0), eq\_set_{v_0}, x \leq p, v_0 > 0)$
10:     return ConcretizeUpperRepairedModel$(M, \hat{M}, P^{u}_{v_0}, \hat{s}, \varphi, v_0)$
```

increased, in order for LowerReachabilityEquations to subsequently generate the nonlinear equations for the probability measure of φ. These equations, together with the objective function, the inequality constraints for v_0 and the probability measure of φ are then passed to NLPSolve. If a solution is returned for v_0, then ConcretizeLowerRepairedModel is called with arguments M, $\hat{M}$, $P^{\ell}_{v_0}$, $\hat{s}, \varphi$ and v_0, and either a repaired model M' is found or else it returns FAILURE. The returned value is eventually reported as the final result of the main PAMR algorithm.

More specifically, the body of AddV$_0$LowerToADTMC is described in Al-

Algorithm 2 AddV$_0$LowerToADTMC

```
Input: $M = (S, s_{init}, P, L)$, $\hat{M} = (\hat{S}, \hat{s}_{init}, P^{\ell}, P^{u}, \hat{L})$, $\hat{s} \in \hat{S}$ and a property
    $\varphi = P_{\geq p}[F\psi]$.
Output: $P^{\ell}_{v_0}$
1:  for all $\hat{s}_k$ such that $(\hat{M}, \hat{s}_k) \models \psi$ do
2:      for all $(\hat{s}_i, \hat{s}_j)$ in shortest maximal paths of the form
        $\pi = [\hat{s}, ..., \hat{s}_i, \hat{s}_j, ..., \hat{s}_k]$ where for all $(\hat{s}_i, \hat{s}_j)$ of $\pi$, $P(s_i, s_j) > 0$ for some
        $s_i, s_j \in S$ with $\hat{s}_i = \alpha(s_i)$, $\hat{s}_j = \alpha(s_j)$ do
3:          $P^{\ell}_{v_0}(\hat{s}_i, \hat{s}_j) := P^{\ell}(\hat{s}_i, \hat{s}_j) + v_0$ with $v_0 > 0$
```

Algorithm 3 LowerReachabilityEquations

```
Input: $\hat{M} = (\hat{S}, \hat{s}_{init}, P^{\ell}, P^{u}, \hat{L})$, $P^{\ell}_{v_0}$, $\hat{s} \in \hat{S}$ and a property $\varphi = P_{\geq p}[F\psi]$.
Output: The equations for the probability measure of $\varphi = P_{\geq p}[F\psi]$.
1: Generate the equations for $\varphi$ from the probabilities in $P^{\ell}_{v_0}$ as in Def. 5.
```

gorithm 2. The P^{ℓ} is modified as follows. For the shortest maximal paths (i.e. paths with transitions that appear at most once) from $\hat{s}$ to a state $\hat{s}_k$ which satisfies ψ, in all transitions $(\hat{s}_i, \hat{s}_j)$ for which there is transition between corresponding concrete states with non-zero probability, the $P^{\ell}(\hat{s}_i, \hat{s}_j)$ is increased by adding the parameter v_0.

The function LowerReachabilityEquations is described in Algorithm 3, where the equations for the probability measure x of φ being true in the ADTMC are generated as in Def. 5. We note here that *the number of generated equations is equal to the number of states of the ADTMC.*

Algorithm 4 NLPSolve

```
Input: An objective function $obj(v_0)$, a set of equations $eq\_set_{v_0}$, an inequality constraint for an unknown in $eq\_set_{v_0}$ and the $v_0$'s range.
Output: The value $v_0$ for which $obj$ is optimized or FAILURE.
1: Use the Sequential Quadratic Programming (SQP) method to solve the non-linear optimization problem for $v_0$.
```

The function NLPSolve is described in Algorithm 4. This function is called in line 4 of Algorithm 1 with $min(v_0)$ as the objective function, the equations $eq\_set_{v_0}$ generated by LowerReachabilityEquations, the inequality constraint $x \geq p$, where p is the given probability bound, and the inequality constraint which ensures positive solution for v_0.

For NLPSolve, the exact solution of constrained nonlinear optimization problems has been proved to be NP-Hard in the worst case [15]. We adopt the Sequential Quadratic Programming (SQP) [16] method, which looks for a *local* solution instead of a *global* one. The complexity of SQP algorithms is bound to the number of equations, which in NLPSolve is equal to the number of the abstract states, and can be exponential in the worst case [17]. *By applying SQP to the state-space of the ADTMC we achieve significant efficiency gains compared to applying it to the state-space of the concrete model M (Table 1).*

Algorithm 5 ConcretizeLowerRepairedModel

```
Input: $M = (S, s_{init}, P, L)$, $\hat{M} = (\hat{S}, \hat{s}_{init}, P^{\ell}, P^{u}, \hat{L})$, $P^{\ell}_{v_0}$, $\hat{s} \in \hat{S}$, a property
    $\varphi = P_{\geq p}[F\psi]$ and a $c \in \mathbb{R}_{+}$.
Output: $M_{Repaired} = (S, s_{init}, P', L)$ or FAILURE.
 1: $M' := M$
 2: for all $(\hat{s}_i, \hat{s}_j) \in \hat{S} \times \hat{S}$ with $P^{\ell}_{v_0}(\hat{s}_i, \hat{s}_j) = P^{\ell}(\hat{s}_i, \hat{s}_j) + v_0$ do
 3:     RET := IncreaseLowerBound$(M', \hat{M}, (\hat{s}_i, \hat{s}_j), c)$
 4:     if RET == FAILURE then
 5:         return FAILURE
 6:     else
 7:         $M'$ := RET
 8: $\hat{M}' := \alpha(M')$
 9: if AbstractModelCheck$(\hat{M}', \varphi)$ == TRUE then
10:     return $M'$
11: return FAILURE
```

The function ConcretizeLowerRepairedModel in Algorithm 5, which finds the repaired DTMC M', is called if NLPSolve computes a solution for v_0 that is passed as argument. The IncreaseLowerBound repair operation is invoked for all $(\hat{s}_i, \hat{s}_j)$, for which the transition probability in $P^{\ell}_{v_0}$ includes v_0. Each invocation of IncreaseLowerBound produces a modified DTMC or a FAILURE result, if the computed value for v_0 cannot yield a valid DTMC because the stochastic conditions are not fulfilled. For the M' produced by the final call of IncreaseLowerBound, the algorithm verifies if it really satisfies φ. This check is essential because IncreaseLowerBound operation alters the transition probabilities of the DTMC in order to preserve the stochastic conditions and these changes might affect the probability measure for φ. As opposed to other model repair approaches [18], which validate the solution by model checking the concrete model, we model check φ over the abstract ADTMC $\hat{M}'$. If the model checking result is true, then M' is returned by the PAMR algorithm as the repair solution.

The execution path of PAMR in Algorithm 1 for $\varphi = P_{\leq p}[F\psi]$ is implemented in an analogous way, with the only difference that instead of increasing the lower bounds, the upper bounds are decreased. The main body of the functions called in this execution path of PAMR are presented in Algorithms 6, 7 and 8.

Algorithm 6 SubtractV$_0$UpperToADTMC

```
Input: $\hat{M} = (\hat{S}, \hat{s}_{init}, P^\ell, P^u, \hat{L})$, $\hat{s} \in \hat{S}$ and a property $\varphi = P_{\leq p}[F\psi]$.
Output: $P^u_{v_0}$
1: for all $\hat{s}_k$ such that $(\hat{M}, \hat{s}_k) \models \psi$ do
2:     for all $(\hat{s}_i, \hat{s}_j)$ in shortest maximal paths of the form
       $\pi = [\hat{s}, ..., \hat{s}_i, \hat{s}_j, ..., \hat{s}_k]$ with $P^u(\hat{s}_i, \hat{s}_j) > 0$ for all $(\hat{s}_i, \hat{s}_j)$ of $\pi$ do
3:         $P^u_{v_0}(\hat{s}_i, \hat{s}_j) := P^u(\hat{s}_i, \hat{s}_j) - v_0$ with $v_0 > 0$
```

Algorithm 7 UpperReachabilityEquations

```
Input: $\hat{M} = (\hat{S}, \hat{s}_{init}, P^\ell, P^u, \hat{L})$, $P^u_{v_0}$, $\hat{s} \in \hat{S}$ and a property $\varphi = P_{\leq p}[F\psi]$.
Output: The equations for the probability measure of $\varphi = P_{\leq p}[F\psi]$.
1: Generate the equations for $\varphi$ from the probabilities in $P^u_{v_0}$ as in Def. 5.
```

The PAMR algorithm can be easily shown to be *sound* in the sense that if a DTMC M' is returned for the property φ being true at state s, then we have $(M', s) \models \varphi$. The proof is straightforward as a consequence of Theorem 1 and the fact that a DTMC is returned if and only if the result of model checking the ADTMC $\hat{M}'$ that is derived from M' is true.

For bounded properties of the form $P_{\bowtie p}[F^{\leq k}\psi]$, the AddV$_0$LowerToADTMC and SubtractV$_0$UpperToADTMC are modified, such that they work only for the paths of finite length k, whereas LowerReachabilityEquations is modified together with UpperReachabilityEquations, in order to produce the equations of Def. 6 instead of Def. 5.

7. Application and discusssion

We present the results of the repair of the DTMC in Fig. 1 for the robot system. In the illustrated scenario we are interested in achieving a probability level of at least 0.325 for the robot to reach the *green* state. The PCTL property for the mentioned goal is $\varphi = P_{\geq 0.325}[Fq]$, where q is the atomic proposition for *color* = *green*. This property is not satisfied in the model M of Fig. 1.

When the PAMR algorithm is applied to the ADTMC of Fig. 2 (initial abstract model) the NLPSolve fails to compute a solution for the parameter v_0 (Case 1 of Step 2 in the PAMR process of Section 5). The process continues with a refinement step (Case 1.1 of Step 2 in the PAMR process) and the

Algorithm 8 ConcretizeUpperRepairedModel

```
Input: $M = (S, s_{init}, P, L)$, $\hat{M} = (\hat{S}, \hat{s}_{init}, P^{\ell}, P^{u}, \hat{L})$, $P^{u}_{v_0}$, $\hat{s} \in \hat{S}$, a property
    $\varphi = P_{\leq p}[F\psi]$ and a $c \in \mathbb{R}_{+}$.
Output: $M_{Repaired} = (S, s_{init}, P', L)$ or FAILURE
 1: $M' := M$
 2: for all $(\hat{s}_i, \hat{s}_j) \in \hat{S} \times \hat{S}$ with $P^{u}_{v_0}(\hat{s}_i, \hat{s}_j) = P^{u}(\hat{s}_i, \hat{s}_j) - v_0$ do
 3:     RET := DecreaseUpperBound($M', \hat{M}, (\hat{s}_i, \hat{s}_j), c$)
 4:     if RET == FAILURE then
 5:         return FAILURE
 6:     else
 7:         $M'$ := RET
 8: $\hat{M}' := \alpha(M')$
 9: if AbstractModelCheck($\hat{M}', \varphi$) == TRUE then
10:     return $M'$
11: return FAILURE
```

refined ADTMC of Fig. 4 is obtained. The PAMR algorithm generates the following equations for the refined ADTMC and φ:

$$
\begin{aligned}
x_0 &= 0x_0 + (0 + v_0)x_1 \\
x_1 &= \left(\frac{2}{3}\right) x_1 + (0 + v_0)x_2 \\
x_2 &= 1
\end{aligned}
$$

The NLPSolve function is eventually called for the above equations and the inequality constraint $x_0 \geq 0.325$. In this case, a solution is returned for the parameter v_0, which is 0.329. The execution of the PAMR algorithm is completed with the computation of the repaired DTMC shown in Fig. 10. Thus, a solution is obtained by reducing the repair problem for a model with 16 states to that for an abstract model with only 3 states.

7.1. Discussion

The efficiency advantage of the PAMR algorithm is based on the fact that the non-linear optimization problem is solved over the state space of the abstract ADTMC, which may be orders of magnitude smaller than the state space of the concrete DTMC.

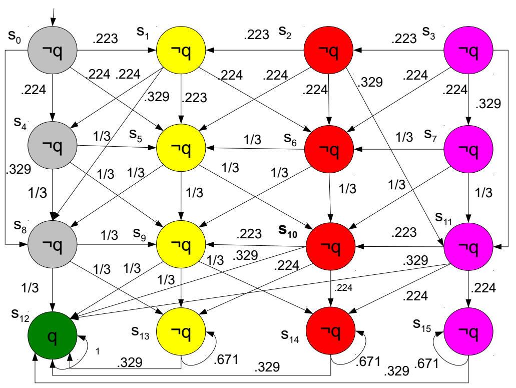


Figure 10: Repaired DTMC after the first refinement step.

The PAMR process of Section 5 does not deteriorate the possibility of obtaining a repair solution compared to the concrete model repair case. This happens due to the fact that the refinement process always converges to the concrete DTMC, if in the meantime a repair solution is not found.

If a repair solution M' is found, the distance of M' from the initial DTMC M (Def. 12) may be greater than the distance of a solution derived by directly applying the repair to the concrete model. This happens because minimizing v_0 only represents the best we can do regarding the Manhattan distance minimality at the degree of precision offered by the specific ADTMC. In essence, this is the price to pay for using abstraction though in practice this may be the only way to obtain a repair solution for models with large or even medium size state space. In our application, the differences between the first PAMR repair solution shown in Fig. 10 and the direct repair solution in Fig. 11 are noticeable.

Whatever the solution is, it is still possible for the refinement process to continue, such that the PAMR algorithm can be applied to ADTMCs with larger state spaces, in which case a more fine grained repair solution could be obtained. This is illustrated through applying the PAMR algorithm in the ADTMCs of Fig. 5 and Fig. 6, which depict respectively the ADTMCs after the second and the third refinement steps in the model for the robot system. The corresponding repaired DTMCs for these two cases are shown in Fig. 12 and Fig. 13.

The PAMR process can be effectively controlled through the refinement method, as well as through specifying how the transition probabilities are

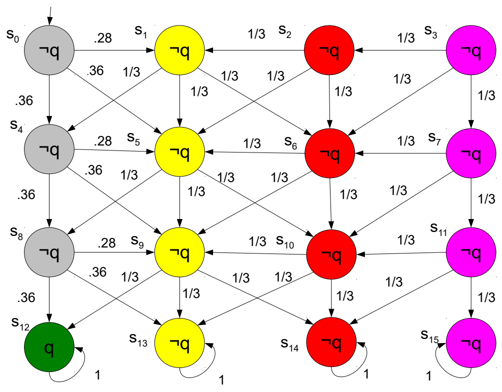


Figure 11: Repaired DTMC after applying repair directly to the concrete model.

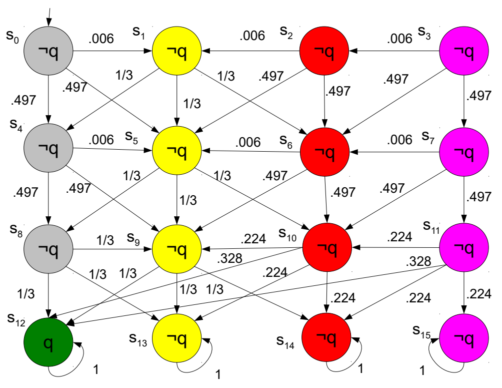


Figure 12: Repaired DTMC after the second refinement step.

modified. These two provide the means to follow alternative repair strategies. The refinement method can be replaced by any other method which may be considered as more adequate for a specific application domain [9, 10, 11, 12, 13, 14]. The solution from applying the PAMR algorithm to the first refined ADTMC of Fig. 7 is shown in Fig. 14. It is clear that this solution differs from the one in Fig. 10 that was obtained through the previous refinement approach.

$\mathsf{AddV_0ToLowerADTMC}$ and $\mathsf{SubtractV_0ToUpperADTMC}$ specify which transition probabilities can be changed and how. The implementation of these functions in our algorithm aims to find a repair solution that distributes the changes to many transitions instead of localizing them to just a few, and that the transition probabilities are evenly changed. This repair strategy, as well

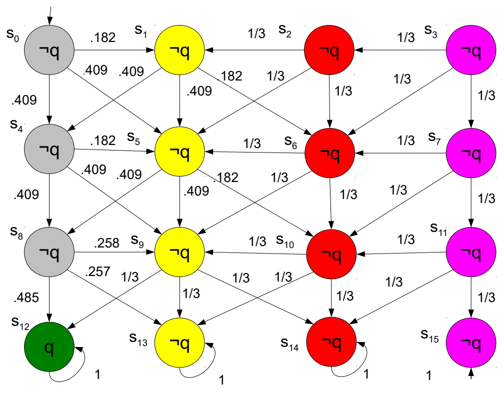


Figure 13: Repaired DTMC after the third refinement step.

as other strategies can result in sub-optimal solutions, i.e. the minimal repair according to the distance of Def. 11 may be missed. However, in practice it is not possible to adopt a single strategy for all domains where the repair problem can arise. Therefore, the aforementioned functions can be replaced by analogous user-defined functions, to specify the parameters $(v_0, v_1, ...)$ for modifying the transition probabilities, and which transition probabilities can be modified.

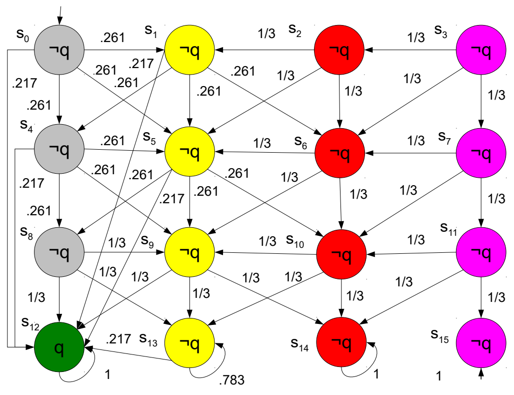


Figure 14: Repaired DTMC after the alternative first refinement step.

By using the same parameter v_0 for all transitions to be modified and the same implementation of IncreaseLowerBound, DecreaseUpperBound for all modified transitions, we ensure that the same parameter can affect multiple transitions in the same way. This is essential if the PAMR framework is to be applied to a probabilistic model specified in a high-level guarded command language like the one used in widely used model checking tools [19, 20].

8. Experimental Results

We have implemented the PAMR algorithm using MAPLE [21]. In particular, we utilized specific functions for model checking ADTMCs (the `solve()` linear equations' solver) [22] and for the solution of the constrained non-linear optimization problem (the `NLPSolve()` function called with the `sqp` parameter).

Using these functions we compared the efficiency of the PAMR algorithm with that of the same algorithm when it is directly applied to the DTMC, i.e. without abstraction. In particular, we experimented with four systems from various domains, whose state spaces have significantly different structure. These systems are the Craps game [23], the IPv4 Zeroconf protocol [2, 24], a message authentication protocol [25] and the gambler's ruin model, which demonstrates the worst-case behavior of the PAMR process in terms of its efficiency gains. The message authentication protocol is mentioned in [26], as a model repair scenario, where the primary aim is not finding the optimal repair solution, but it is sufficient to find some repair solution rapidly (fast model repair problem). Our PAMR framework fits ideally to this context, since a repair solution can be obtained even for very large model sizes, which can be afterwards refined with respect to the needs of the user.

The Craps game model shown in Fig. 15 refers to a dice game, where the player wins or loses based on the outcome of the roll of two dice. The outcomes 2, 3, 7, 11, 12 are "craps", i.e. the player loses. On any other outcome the dice are rolled again and the outcome of the come-out roll is remembered (the "point"). The dice are rolled repeatedly until the outcome is 7, in which case the player loses, or the outcome is the point, in which case the player wins.

The probability for winning the Craps game, i.e. reaching the *Win* state in Fig. 15 is approximately 0.27. In order to increase the probability to at least 0.3 for the player to win the game, we would like to find how much the dice should be biased each time they are rolled. Let us consider the atomic proposition q as $Win = true$, in which case the PCTL property of interest is $\varphi = P_{\geq 0.3}[Fq]$. The DTMC of Fig. 15 was eventually repaired after three refinement steps, which resulted in the ADTMC shown in Fig. 16. The repair solution given by the PAMR algorithm is shown in Fig. 17.

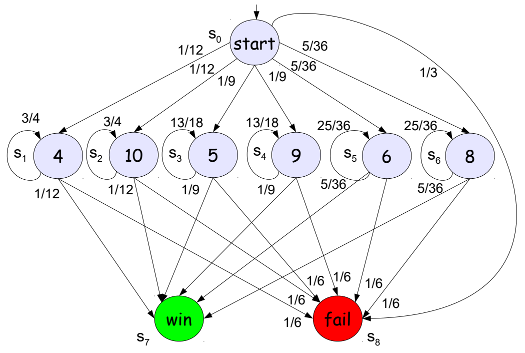


Figure 15: The DTMC for a version of the Craps game.

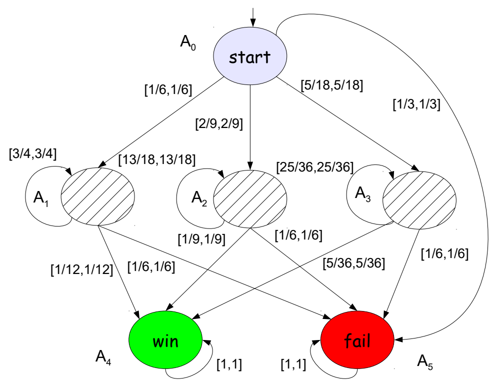


Figure 16: The ADTMC after the third refinement step for the Craps game.

Zeroconf is a protocol for assigning IP addresses in a network of hosts. When a new host joins the network it asks the other hosts if the newly selected IP is already in use from any other host. There is a probability that the new host will not get any answer and in this case the query is repeated. The host will assume falsely that the chosen address is valid, if after n tries no answer is received.

The DTMC for an instance of the Zeroconf protocol is shown in Fig. 18a, where the probability of not getting an answer is 0.99. We try to repair the DTMC for $n = 10$ tries, in order to ensure that the probability for reaching the OK state (which is 0.32) becomes at least 0.99. We consider the atomic proposition q as $OK = true$, in which case the reachability PCTL property of interest is $\varphi = P_{\geq 0.99}[Fq]$. The application of our PAMR framework resulted,

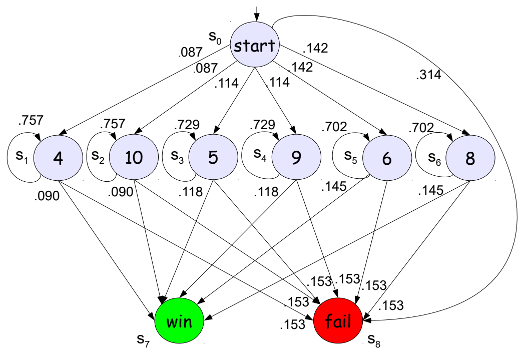


Figure 17: The repaired DTMC for the Craps game.

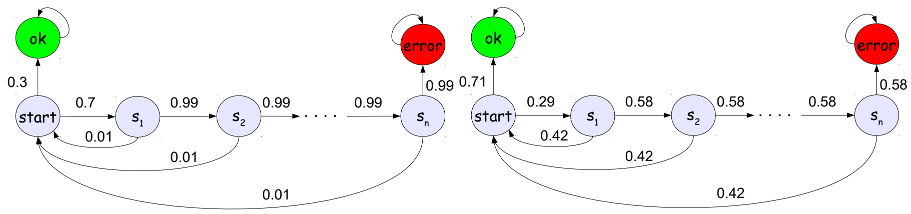
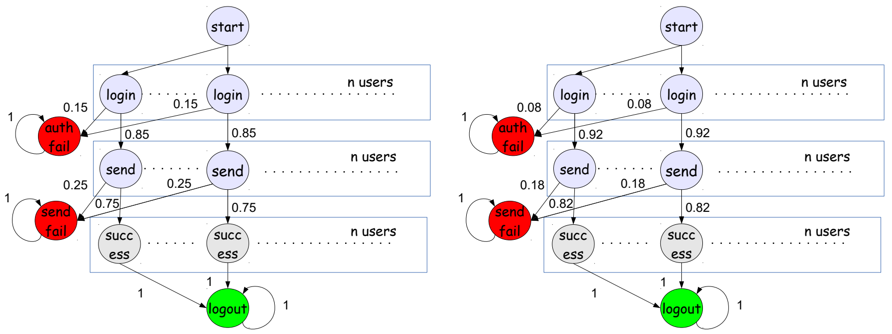

(a) The initial DTMC. (b) The repaired DTMC for $n = 10$.

Figure 18: The IPv4 Zeroconf protocol

after three refinement steps, to the repair solution presented in Fig. 18b.

The DTMC of the message authentication protocol for n users is depicted in Fig. 19a. An authentication process takes place for each user, which can fail with a probability of 0.15. After the successful authentication, each user can send a message with a failure probability of 0.25. The users log out from the system after having successfully sent a message. We applied our PAMR framework for $n = 5$ users, in order to get the repaired DTMC on which the probability for the users to log out is at least 0.75. A repaired DTMC is acquired after four refinement steps and is shown in Fig. 19b.

For the gambler's ruin model, let us consider a gambler who starts playing a game with initial wealth N coins and can bet at each time one coin. The gambler can win the bet with a probability p and can lose the bet with probability q. The game stops either when the gambler has no more money to bet or when he has earned a specific amount W. The DTMC for a version of the game with $N = 20$, $W = N \times 10\% = 2$, $p = 0.3$ and $q = 0.7$ is

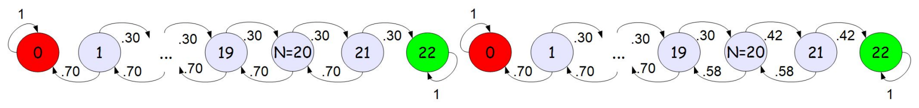


(a) The initial DTMC. (b) The repaired DTMC for $n = 5$.

Figure 19: A message authentication protocol

presented in Fig. 20a. The gambler's ruin model belongs to a special category of Markov Chains, called *birth-death*, where only a one-step transition to the nearest neighbors is permitted.

The probability for the gambler to earn the amount W is 0.183. After applying our PAMR framework with the aim of finding a repaired DTMC where the probability of the gambler to earn the amount W will be at least 0.25, we acquire the DTMC in Fig 20b.



(a) The initial DTMC. (b) The repaired DTMC

Figure 20: The gambler's ruin model for initial wealth $N = 20$.

Our experiments include extended versions for all the four models. For the Craps game, the model's state space is expanded by rolling more than two dice and also adapting the conditions for wining/losing the game. For the IPv4 Zeroconf protocol, for the message authentication protocol and for the gambler's ruin model, the state space is expanded by increasing the number of tries, the number of users and the initial wealth respectively. As shown in Table 1, the state space size varies between a few hundreds of states and transitions up to several tens of thousands states and transitions.

For all the models and for the same reachability properties, we applied our model repair algorithm directly to the DTMC (concrete model) and we

Models	Concr. States	Concr. Trans.	Concrete Repair	PAMR	Ref. Steps	Abstr. States	d_{concr}	d_{PAMR}
Craps (Ext.1)	123	483	$0.8s$	$0.1s$	3	6	7.4	10.4
Craps (Ext.2)	243	963	$5.1s$	$0.1s$	3	6	14.6	20.5
Craps (Ext.3)	483	1923	$38.3s$	$0.1s$	3	6	29	40.6
Craps (Ext.4)	963	3843	$5m13s$	$0.1s$	3	6	57.8	80.9
Craps (Ext.5)	1923	7683	$49m43s$	$0.1s$	3	6	115.4	161.6
Craps (Ext.6)	3843	15363	$8h30m23s$	$0.1s$	3	6	230.6	322.9
Craps (Ext.7)	7683	30723	*time-out*	$0.1s$	3	6	–	645.42
Craps (Ext.7)	7683	30723	*time-out*	$1.2s$	8	123	–	608.5
Craps (Ext.7)	7683	30723	*time-out*	$19.1s$	10	483	–	562.4
Craps (Ext.7)	7683	30723	*time-out*	$2h4m34s$	13	3843	–	488.7
Craps (Ext.7)	7683	30723	*time-out*	*time-out*	14	7683	–	–
Zconf (n=100)	103	204	$0.5s$	$0.2s$	3	5	10.1	82.8
Zconf (n=200)	203	404	$2.7s$	$0.2s$	3	5	17.7	164.8
Zconf (n=400)	403	804	$18.1s$	$0.2s$	3	5	28.9	328.8
Zconf (n=800)	803	1604	$2m38s$	$0.2s$	3	5	43.3	656.8
Zconf (n=1600)	1603	3204	$15m29s$	$0.2s$	3	5	51.2	1312.8
Zconf (n=3200)	3203	6404	$2h36m12s$	$0.2s$	3	5	57.6	2624.8
Zconf (n=6400)	6403	12804	*time-out*	$0.2s$	3	5	–	5248.2
Zconf (n=6400)	6403	12804	*time-out*	$2.1s$	10	104	–	4224.6
Zconf (n=6400)	6403	12804	*time-out*	$14.9s$	12	404	–	2048.3
Zconf (n=6400)	6403	12804	*time-out*	$1h36m43s$	15	3204	–	512.1
Zconf (n=6400)	6403	12804	*time-out*	*time-out*	16	6403	–	–
Auth.Prot.(n=30)	154	213	$1.2s$	$0.2s$	4	7	8.4	8.4
Auth.Prot.(n=60)	304	423	$7.8s$	$0.2s$	4	7	16.8	16.8
Auth.Prot.(n=120)	604	843	$57.5s$	$0.2s$	4	7	33.6	33.6
Auth.Prot.(n=240)	1204	1683	$9m43s$	$0.2s$	4	7	67.2	67.2
Auth.Prot.(n=480)	2404	3363	$1h12m11s$	$0.2s$	4	7	134.4	134.4
Auth.Prot.(n=960)	4804	6723	$12h19m52s$	$0.2s$	4	7	268.8	268.8
Auth.Prot.(n=1920)	9604	13443	*time-out*	$0.2s$	4	7	–	537.6
Auth.Prot.(n=1920)	9604	13443	*time-out*	$2.2s$	9	131	–	537.6
Auth.Prot.(n=1920)	9604	13443	*time-out*	$20s$	11	515	–	537.6
Auth.Prot.(n=1920)	9604	13443	*time-out*	$1h54m32s$	14	4099	–	537.6
Auth.Prot.(n=1920)	9604	13443	*time-out*	*time-out*	15	9604	–	–
Gambler(N=100)	111	220	$0.8s$	$0.2s$	4	58	0.38	2.93
Gambler(N=200)	221	440	$4.6s$	$0.9s$	5	113	0.84	5.94
Gambler(N=400)	441	880	$33.3s$	$4.8s$	6	223	2.48	12.28
Gambler(N=800)	881	1760	$4m59s$	$32.7s$	7	443	5.92	25.04
Gambler(N=1600)	1761	3520	$45m12s$	$4m48s.$	8	883	13.44	50.88
Gambler(N=3200)	3521	7040	$8h2m41s$	$45m17s$	9	1763	30.72	103.36
Gambler(n=7400)	7041	14080	*time-out*	$6h58m36s$	10	3523	–	207.36

Table 1: Experimental results of PAMR compared to concrete repair (in fourth column the accumulated times are shown, for all iterated refinement steps)

compared its execution time and the repair solution's distance with that of the PAMR algorithm for the ADTMC. For the Craps game and the IPv4 Zeroconf protocol, we get a repair solution after three refinement steps, whereas for the message authentication protocol a solution is obtained after four refinement steps. For the gambler's ruin model the number of needed refinement steps varies from 4 to 10 according to the size of the model. The results in Table 1 suggest that as the model's state space grows up, the needed time for the concrete model repair is increased dramatically or even a solution cannot be obtained within a time-out period of 14 hours, while the PAMR algorithm returns a repair solution efficiently. It is worth mentioning that the time for the PAMR algorithm remains constant for the Craps game, the IPv4 Zeroconf protocol and the message authentication protocol, because in all extended versions the repair solution was obtained from ADTMCs with the same size of state space after three or four refinement steps respectively. This is not true for the gambler's ruin model, where the repair solution is always acquired in the final refinement step, before converging to the concrete DTMC. These results confirm the fact that the efficiency gains of abstraction in model repair depends - as in model checking - on the model's state space structure and the property.

For all models except for the gambler's ruin model, we continued the application of our PAMR framework, after having found the first repair solution, until the PAMR converged to be applied to the initial concrete model, in which case the ADTMC would be the same with the DTMC. From the results presented in Table 1, we deduce that for all the examined models, our method gives a repair solution and it is significantly more efficient than the direct repair even for the ADTMC produced in the penultimate refinement step.

9. Related Work

The model repair problem for probabilistic systems has been first introduced in [2]. The authors use parametric model checking and transform the model repair problem to a non-linear optimization problem. The experimental results of their work indicate that their approach suffers from the state space explosion problem even for models with relatively small number of states. In comparison with [2], beyond the efficiency advantage of the PAMR framework, it is true that we do not aim at a direct repair solution, if any, but to gradually approach a suitable solution through a number of consecutive refinements that depends on the available computational resources. In such an iterative process, the designer can interactively develop his repair strategy, as described in Section 7.1, whereas in [2] the repair strategy is specified at once through the definition of a so-called controllable DTMC over a set of parameters that has to fulfill the stochasticity condition. However, such a parameterization approach is not easily applied in all state space topologies.

An effort for presenting a scalable method for the repair of DTMCs is also presented in [18]. In that work, the authors present a greedy approach where, starting from an initial parameter assignment, they apply local repair steps by iteratively changing the parameter values. The execution time for the local repairs is reduced with respect to the methods using non-linear optimization, but the fact that this approach includes a model-checking phase eventually increases the total execution time.

Regarding the model repair of Markov Decision Processes (MDPs), there are approximate solution techniques, which have been introduced in [26].

Some related work exists regarding the *parametric analysis* of Markov models. More specifically, an interesting and related to model repair problem is that of *parameter synthesis*, where the aim is to find ranges of parameter values such that a satisfaction probability of a formula meets a given threshold, is maximized, or minimized. In [27], the authors address the system design problem for Continuous Time Markov Chains (CTMCs), where their aim is to optimize some parameters of a stochastic model to maximize robustness of some given specifications. The state space explosion problem is also inherent in [28], where a tool is introduced using GPU-accelerated parameter synthesis to make the approach scalable. This work addresses the parallel implementation of the parameter space decomposition technique of [29]. A recent work [30] deals with the synthesis problem for CTMCs within the scope of optimizing specific quality criteria. Moreover, a tool called PROPhESY has been presented in [31], for scalable, incremental, and automatic parameter synthesis. Another tool called SEA-PARAM has been presented in [32], where the authors propose a parameter synthesis method for Parametric Markov Decision Processes (PMDPs).

Moreover, the authors in [29] have proposed a method for parameter interval decomposition for CSL formulas in CTMCs. This method is a means to reduce the inaccuracy created from a min-max approximation. If this idea could be lifted to DTMCs, it is a promising tool that could be adopted in our abstraction-refinement framework.

The model repair problem has attracted the interest of applied research initiatives in various problems, such as the formal verification and *fault recovery* of aerospace systems [33]. From this perspective, interesting applications in the context of temporal logics for non-probabilistic systems have been investigated in [34] and [35]. In [36], the authors attempt three different model repair techniques to efficiently compute the probability distribution for the minimum recovery time of an input randomized distributed self-stabilizing protocol. The first approach is based on solving symbolic linear algebraic equations, in order to identify the fastest state reachability in parametric discrete-time Markov chains. Their second approach applies parameter synthesis techniques to compute the rational function describing the average recovery time, like in [2]. In their third approach, they focus on finding sub-optimal solutions by computing over- and under-approximations of the result for a given parameter region and iteratively refining the regions with minimal recovery time up to the desired precision. Yet, all of the proposed techniques still suffer from the state space explosion problem.

For the abstraction of probabilistic models, some more techniques have been proposed apart from the adopted abstraction method with 3-valued semantics. In [11], the authors present an abstraction method for Markov Decision processes based on games. In [37], the authors propose a *model aggregation* technique to construct effectively the lumping quotient of a Markov chain. In [38], the authors examine the minimal distance problem of Markov chains using bisimulation metrics, i.e., given a finite DTMC and a positive integer k, they try to find the k-state DTMC with the minimal distance to the original. We note that in our abstraction framework based on [8], the abstract model is connected to the concrete model with a simulation – not with a (more restrictive) bisimulation – relation. A thorough presentation of the abstraction methods used for probabilistic systems is given in [39]. We are not aware of any other model repair approach, which exploits an abstraction technique for probabilistic systems.

However, abstraction has been used in model repair for systems with large state spaces, in the non-probabilistic setting [6]. In this work, the authors present an abstraction-refinement framework based on Kripke Structures as the concrete models, Kripke Modal Transition Systems as the abstract models, CTL as the specification language and 3-valued model checking. Their aim is that of reducing the upper bound complexity class of the repair process, such that it depends on the size of a much smaller abstract model. A more complete journal version of this work is presented in [7].

10. Conclusions

We presented an abstraction-refinement framework and algorithms for the repair of DTMCs with respect to reachability PCTL properties. The so-called PAMR framework and its algorithm aim to confront the state space explosion problem and to provide a solution that is applicable even in model repair problems with very large state spaces. After having described the PAMR framework, we illustrated the efficiency gains of our approach compared to the direct repair of the concrete model, including the discussion of custom repair strategies depending on the particular application domain. The practical utility of the PAMR framework was demonstrated in the repair of four DTMC models with diverse structures of their state space.

As a future research prospect, a PAMR tool would allow applying the method in more complex real-world applications. This will also open chances for investigating how PAMR can be used in various domains, such as in the design of *fault-tolerant* systems. Additional work is needed towards examining the way that the refinement process can be utilized for interactively driving the repair strategy. An interesting line of research for future work is the extension of our PAMR algorithm to support nested properties by using recursion and the notion of constraints, as they have been used in the abstract model repair for non probabilistic systems [7]. Finally, we also plan to work on the model repair for other types of systems, such as hybrid or real-time systems, for which the use of abstraction to cope with the state space explosion is also an attractive prospect.

Bibliography

[1] E. M. Clarke, E. A. Emerson, J. Sifakis, Model checking: Algorithmic verification and debugging, Communications of the ACM 52 (11) (2009) 74–84.

[2] E. Bartocci, R. Grosu, P. Katsaros, C. R. Ramakrishnan, S. A. Smolka, Model repair for probabilistic systems, in: Proceedings of the 17th international conference on Tools and algorithms for the construction and analysis of systems: part of the joint European conferences on theory and practice of software, TACAS'11/ETAPS'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 326–340.

[3] E. M. Clarke, O. Grumberg, D. E. Long, Model checking and abstraction, ACM Trans. Program. Lang. Syst. 16 (1994) 1512–1542.

[4] C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, S. Bensalem, D. Probst, Property preserving abstractions for the verification of concurrent systems, Formal Methods in System Design 6 (1995) 11–44.

[5] P. Godefroid, M. Huth, R. Jagadeesan, Abstraction-based model checking using modal transition systems, in: Proceedings of the 12th International Conference on Concurrency Theory, CONCUR '01, Springer-Verlag, London, UK, 2001, pp. 426–440.

[6] G. Chatzieleftheriou, B. Bonakdarpour, S. A. Smolka, P. Katsaros, Abstract model repair, in: Proceedings of the 4th international conference on NASA Formal Methods, Springer-Verlag, 2012, pp. 341–355.

[7] G. Chatzieleftheriou, B. Bonakdarpour, P. Katsaros, S. A. Smolka, Abstract Model Repair, Logical Methods in Computer Science Volume 11, Issue 3. doi:10.2168/LMCS-11(3:11)2015.

[8] J.-P. Katoen, D. Klink, M. Leucker, V. Wolf, Three-valued abstraction for probabilistic systems, The Journal of Logic and Algebraic Programming 81 (4) (2012) 356–389.

[9] H. Hermanns, B. Wachter, L. Zhang, Probabilistic cegar, in: International Conference on Computer Aided Verification, Springer, 2008, pp. 162–175.

[10] M. Kattenbelt, M. Kwiatkowska, G. Norman, D. Parker, Abstraction refinement for probabilistic software, in: International Workshop on Verification, Model Checking, and Abstract Interpretation, Springer, 2009, pp. 182–197.

[11] M. Kattenbelt, M. Kwiatkowska, G. Norman, D. Parker, A game-based abstraction-refinement framework for markov decision processes, Formal Methods in System Design 36 (3) (2010) 246–280.

[12] P. R. Dargenio, B. Jeannet, H. E. Jensen, K. G. Larsen, Reachability analysis of probabilistic systems by successive refinements, in: Process Algebra and Probabilistic Methods. Performance Modelling and Verification, Springer, 2001, pp. 39–56.

[13] A. Komuravelli, C. S. Păsăreanu, E. M. Clarke, Assume-guarantee abstraction refinement for probabilistic systems, in: International Conference on Computer Aided Verification, Springer, 2012, pp. 310–326.

[14] R. Chadha, M. Viswanathan, A counterexample-guided abstraction-refinement framework for markov decision processes, ACM Transactions on Computational Logic (TOCL) 12 (1) (2010) 1.

[15] S. A. Vavasis, Quadratic programming is in np, Information Processing Letters 36 (2) (1990) 73–77.

[16] P. T. Boggs, J. W. Tolle, Sequential quadratic programming, Acta numerica 4 (1995) 1–51.

[17] C. T. Lawrence, A. L. Tits, A computationally efficient feasible sequential quadratic programming algorithm, Siam Journal on optimization 11 (4) (2001) 1092–1118.

[18] S. Pathak, E. Ábrahám, N. Jansen, A. Tacchella, J.-P. Katoen, A greedy approach for the efficient repair of stochastic models, in: NASA Formal Methods Symposium, Springer, 2015, pp. 295–309.

[19] M. Kwiatkowska, G. Norman, D. Parker, Prism 4.0: Verification of probabilistic real-time systems, in: Proceedings of the 23rd International Conference on Computer Aided Verification, CAV'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 585–591.

[20] C. Dehnert, S. Junges, J. Katoen, M. Volk, A storm is coming: A modern probabilistic model checker, in: Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II, 2017, pp. 592–600.

[21] Maple, accessed: 2017-06-22.
URL `http://www.maplesoft.com/products/Maple/`

[22] M. Kwiatkowska, G. Norman, D. Parker, Stochastic model checking, in: SFM, Vol. 7, Springer, 2007, pp. 220–270.

[23] C. Baier, J.-P. Katoen, Principles of Model Checking (Representation and Mind Series), The MIT Press, 2008.

[24] M. Kwiatkowska, G. Norman, D. Parker, J. Sproston, Performance analysis of probabilistic timed automata using digital clocks, Formal Methods in System Design 29 (2006) 33–78.

[25] A. Filieri, C. Ghezzi, G. Tamburrelli, Run-time efficient probabilistic model checking, in: Proceedings of the 33rd International Conference on Software Engineering, ICSE '11, ACM, New York, NY, USA, 2011, pp. 341–350. doi:10.1145/1985793.1985840.

[26] T. Chen, E. M. Hahn, T. Han, M. Kwiatkowska, H. Qu, L. Zhang, Model repair for markov decision processes, in: Theoretical Aspects of Software Engineering (TASE), 2013 International Symposium on, IEEE, 2013, pp. 85–92.

[27] E. Bartocci, L. Bortolussi, L. Nenzi, G. Sanguinetti, System design of stochastic models using robustness of temporal properties, Theoretical Computer Science 587 (2015) 3–25.

[28] M. Češka, P. Pilař, N. Paoletti, L. Brim, M. Kwiatkowska, Prism-psy: precise gpu-accelerated parameter synthesis for stochastic systems, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2016, pp. 367–384.

[29] L. Brim, M. Češka, S. Dražan, D. Šafránek, Exploring parameter space of stochastic biochemical systems using quantitative model checking, in: Proceedings of the 25th International Conference on Computer Aided Verification, CAV'13, Springer-Verlag, Berlin, Heidelberg, 2013, pp. 107–123.

[30] R. C. Calinescu, M. Ceska, S. Gerasimou, M. Kwiatkowska, N. Paoletti, Designing robust software systems through parametric markov chain synthesis, in: IEEE International Conference on Software Architecture (ICSA 2017), IEEE, 2017.

[31] C. Dehnert, S. Junges, N. Jansen, F. Corzilius, M. Volk, H. Bruintjes, J.-P. Katoen, E. Abraham, Prophesy: A probabilistic parameter synthesis tool, in: International Conference on Computer Aided Verification, Springer, 2015, pp. 214–231.

[32] S. Arming, E. Bartocci, A. Sokolova, Sea-param: Exploring schedulers in parametric mdps, Quantitative Aspects of Programming Languages and Systems 2017.

[33] M. Bozzano, H. Bruintjes, A. Cimatti, J.-P. Katoen, T. Noll, S. Tonetta, Formal methods for aerospace systems, in: Cyber-Physical System Design from an Architecture Analysis Viewpoint, Springer, 2017, pp. 133–159.

[34] A. Arcuri, Evolutionary repair of faulty software, Applied Soft Computing 11 (4) (2011) 3494–3514.

[35] F. Buccafurri, T. Eiter, G. Gottlob, N. Leone, Enhancing model checking in verification by ai techniques, Artificial Intelligence 112 (1) (1999) 57–104.

[36] S. Aflaki, M. Volk, B. Bonakdarpour, J.-P. Katoen, A. Storjohann, Automated fine tuning of probabilistic self-stabilizing algorithms, In SRDS (2017).

[37] S. Derisavi, H. Hermanns, W. H. Sanders, Optimal state-space lumping in markov chains, Inf. Process. Lett. 87 (6) (2003) 309–315. doi:10.1016/S0020-0190(03)00343-0.

[38] G. Bacci, G. Bacci, K. G. Larsen, R. Mardare, On the Metric-Based Approximate Minimization of Markov Chains, in: I. Chatzigiannakis, P. Indyk, F. Kuhn, A. Muscholl (Eds.), 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017), Vol. 80 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017, pp. 104:1–104:14. doi:10.4230/LIPIcs.ICALP.2017.104.

[39] C. Dehnert, D. Gebler, M. Volpato, D. N. Jansen, On abstraction of probabilistic systems, in: Advanced Lectures of the International Autumn School on Stochastic Model Checking. Rigorous Dependability Analysis Using Model Checking Techniques for Stochastic Systems - Volume 8453, ROCKS 2012, Springer-Verlag New York, Inc., New York, NY, USA, 2014, pp. 87–116.