Rate Reduction for State-labelled Markov Chains with Upper Time-bounded CSL Requirements

Bharath Siva Kumar Tati   Markus Siegle
Universität der Bundeswehr München
Germany
Bharath.Tati@unibw.de   Markus.Siegle@unibw.de

This paper presents algorithms for identifying and reducing a dedicated set of transition rates of a state-labelled continuous-time Markov chain model. The purpose of the reduction is to make states to satisfy a given requirement, specified as a CSL upper time-bounded Until formula. We distinguish two different cases, depending on the type of probability bound. A natural partitioning of the state space allows us to develop possible solutions, leading to simple algorithms for both cases.

1 Introduction

In a production plant, there can be the requirement that "once started, the production process should be completed within 1 hour in 95% of all cases", or that "the probability of an alarm during the first 30 min of operation should be at most 5%". If the system is modelled as a state-labelled continuous-time Markov chain (SMC) and the requirements are formulated with the help of continuous stochastic logic (CSL), they can be verified automatically by stochastic model checking [1], supported by efficient tools such as the probabilistic model checker PRISM [10] or MRMC [9].

This paper addresses the question of how to improve a given system (also called plant $\mathscr{P}$) when it has been found that $\mathscr{P}$ violates such a given formal requirement. In an earlier paper [13] we presented solutions for the case of *untimed* probabilistic requirements, and building on that work we now present solutions for the case of *upper time-bounded Until*-type requirements (without nested or multiple *Until* operators). In general, one could think of many ways of how to modify a system in order to make it satisfy a given requirement. We decided to restrict ourselves to *rate reduction*, which means that some of the system's transition rates may be reduced, but the structure of the system remains untouched. We only allow rate *reductions*, as opposed to increasing any rates, motivated by the fact that it is usually easily possible to slow down some technical process (machine, processor, etc.), while it may not be possible to accelerate it. We work with a partitioning of the state space into classes, based on the requirement at hand. Our strategy then is to reduce all transition rates between some source class and target class by a *common reduction factor*. Depending on the case, different sets of transitions may be reduced by different reduction factors to achieve the goal. Before the start of the adaptation process, the value of all reduction factors is 1, and in the end all reduction factors will be still at most 1, but strictly greater than 0 (which means that no transitions are completely disabled). Throughout, our intention is to make as many as possible states of $\mathscr{P}$ satisfy the user requirement, but it is not always possible to make all states satisfying. This paper develops simple, intuitive algorithms, which are first motivated by examples. We show the correctness of the algorithms (while pointing out the limitations of Algorithm 2) and also analyze their complexity.

Related work: A related topic is model checking of parametric Markov chains, where reachability probabilities take the form of rational functions [5, 6, 7]. Here the goal is to find valid parameter values

Submitted to:
SynCoP 2016

© B. S. K. Tati & M. Siegle
This work is licensed under the
Creative Commons Attribution License.

in a multi-dimensional search space, as described in [8], where a discretization strategy was proposed together with refinement and/or sampling. Synthesizing optimal rate parameter values is also the goal of [3] (in the context of Markov models of biochemical systems), where time-bounded properties are considered. Closely related is the so-called *model repair* problem which occurs when a system violates a given requirement, which is to be fixed by modifying transition parameters while at the same time keeping cost at a minimum. Model repair has been addressed e.g. in [2, 4, 12], for parametric DTMC or MDP models, where solutions are obtained with the help of nonlinear optimization, sampling/refinement or greedy strategies. Our approach described here is different from all of the above, since we work with CTMC models which are a priori not parametric, but come with fixed rates. Once some requirement is violated, we seek to identify sets of transitions and reduce their rates by a common reduction factor in order to make as many states as possible satisfy the requirement. We do currently not consider the cost of rate reduction, and we restrict ourselves to simple algorithms which avoid expensive multi-dimensional parameter searches.

The rest of the paper is structured as follows: Sec. 2 provides the basic definitions and recalls an earlier result for untimed requirements. In Sec. 3, an example is elaborated on in order to explain the idea of rate reduction. It illustrates the benefits, but also the limitations of the proposed approach. The general algorithms are discussed in Sec. 4, which also includes their complexity analysis. Finally, Sec. 5 summarizes the main findings of the paper and touches on possible future work.

2 Preliminaries

A State labelled Markov chain (SMC) is defined as follows:

Definition 1 (SMC) *A SMC $\mathscr{P}$ is a tuple $(S_{\mathscr{P}}, R_{\mathscr{P}}, L_{\mathscr{P}})$ where*

- *$S_{\mathscr{P}}$ is a finite set of states*
- *$R_{\mathscr{P}} : S_{\mathscr{P}} \times S_{\mathscr{P}} \mapsto \mathbb{R}_{\geq 0}$, is the transition function (rate matrix)*
- *$L_{\mathscr{P}} : S_{\mathscr{P}} \to 2^{AP}$ is a state labelling function, where AP is a finite set of atomic propositions*

This definition does not impose any special structural conditions (such as irreducibility) on the state graph of the Markov chain. A finite timed *path* σ in a SMC $\mathscr{P}$ is a finite sequence $\sigma = [(s_0, t_0), (s_1, t_1), \cdots, (s_{n-1}, t_{n-1}), s_n] \in (S_{\mathscr{P}} \times \mathbb{R}_{>0})^* \times S_{\mathscr{P}}$, and with $Paths(s)$ we denote the set of all finite paths originating from state s. Probabilities are assigned to sets of finite timed paths by the usual cylinder set construction on sets of infinite timed paths. In order to specify user requirements and characterize execution paths of SMCs, we use a subset of CSL (continuous stochastic logic) [1].

Definition 2 (CSL with upper time bound) *The grammar for CSL state formulas Φ, Φ' and path formulas φ is given as:*

$$\Phi ::= q \,|\, \neg\Phi \,|\, \Phi \vee \Phi \,|\, P_{\sim b}(\varphi), \qquad \varphi ::= \Phi' \,\mathscr{U}^{\leq t}\, \Phi', \qquad \Phi' ::= q \,|\, \neg\Phi' \,|\, \Phi' \vee \Phi'$$

In the definition, $q \in AP$ is an atomic proposition, $\neg$ denotes negation, $\vee$ denotes disjunction, $b \in (0,1)$ is a probability value, and $\sim \in \{\leq, \geq\}$ a comparison operator. $P_{\sim b}(\varphi)$ asserts that the probability measure of the set of paths satisfying φ meets the bound given by $\sim b$. The path formula φ is constructed using the *Until* ($\mathscr{U}$) operator and an upper time bound $t > 0$. Note that we do not consider CSL requirements with nested *Until* operators (that's why we distinguish between Φ and Φ'), since parameter adaptations for *Until* operators at different levels are interdependent in a complex way. For similar reasons we do not consider requirements with multiple *Until* operators (although the grammar in Def. 2 does not exclude them). We also do not consider the CSL *next* operator since it would be rather trivial to handle.

Remark 1 This paper considers probability bounds $b \in (0,1)$ instead of $b \in [0,1]$, since the approach presented here does not aim to turn a non-zero probability into zero, or to turn a probability smaller than one into one. So, we do not treat requirements of the form $P_{\leq 0}(\varphi)$ or $P_{\geq 1}(\varphi)$, and for similar reasons we also do not treat requirements of the form $P_{>0}(\varphi)$ or $P_{<1}(\varphi)$. Furthermore, there is no need to distinguish between $< b$ and $\leq b$ (or $> b$ and $\geq b$), because in the continuous-time setting the probabilities are the same.

Definition 3 (Semantics of time-bounded Until path formula) *The satisfaction relation $\models$ for time-bounded* Until *path formulas is defined as in [1]:*

$$\sigma \models \Phi \,\mathscr{U}^{\leq t}\, \Psi \quad \textit{if} \quad \exists t' \in [0,t].(\sigma@t' \models \Psi \wedge \forall t'' \in [0,t').\sigma@t'' \models \Phi)$$

where $\sigma@t$ denotes the state occupied by the path σ at time t.

The untimed *Until* formula is obtained as: $\Phi \,\mathscr{U}\, \Psi = \Phi \,\mathscr{U}^{<\infty}\, \Psi$. Let $Sat(\Phi)$ denote the set of states satisfying state formula Φ, and let $Pr(s,\varphi) = Pr(\{\sigma \in Paths(s) \mid \sigma \models \varphi\})$ denote the probability of the set of φ-satisfying paths originating in state s. In order to accomplish the process of rate reduction, we will use a partitioning of the SMC state space:

Definition 4 (Partitioning of SMC) *Given an SMC $\mathscr{P}$ and a CSL requirement $\Phi_t = P_{\sim b}(\Phi \,\mathscr{U}^{\leq t}\, \Psi)$. Let $\varphi = \Phi \,\mathscr{U}\, \Psi$, the untimed version of the path formula. Then, states belonging to*

- $Sat(\neg\Phi \wedge \neg\Psi)$ *are placed in class* invalid
- $Sat(\Psi)$ *are placed in class* target
- $Sat(\Phi \wedge \neg\Psi)$ *are placed in class* transit

The transit class is further partitioned as:

- $Pr(s,\varphi) = 1$ *are placed in class* gototarget
- $Pr(s,\varphi) = 0$ *are placed in class* gotoinvalid
- $0 < Pr(s,\varphi) < 1$ *are placed in class* gobothways

This partitioning is illustrated in Fig. 1[1]. Starting from states of class *gototarget*, the Markov chain will eventually reach the *target* class via Φ-states (almost surely within finite time). Conversely, states of class *gotoinvalid* do not possess any path satisfying the given (untimed, and therefore also time-bounded) *Until* requirement. From states of class *gobothways*, both of these behaviours are possible. During the parameter adaptation procedure, our attention will be on the states of the *transit* class. The Partitioning of the state space can be obtained efficiently, based on the state labelling and by applying standard graph algorithms [11]. Note that the time bound $\leq t$ and the probability bound $\sim b$ in the given CSL formula have no influence on the partitioning.

As a simple but important fact we emphasize that the probability of a state satisfying a time-bounded *Until* property will always be less than or equal the probability of that state satisfying the corresponding *untimed* property, i.e.

$$\forall s.\, \forall t.\; (Pr(s,\Phi \,\mathscr{U}^{\leq t}\, \Psi) \leq Pr(s,\Phi \,\mathscr{U}\, \Psi)) \tag{1}$$

Furthermore, for an SMC $\mathscr{P}$, a subset $X \subseteq S$ of its state set, and a timed or untimed CSL state formula Φ, we introduce the following notation: $(\mathscr{P} \models_X \Phi) \iff (\forall s \in X : s \models \Phi)$. As an example, $\mathscr{P} \models_{gobothways} \Phi$ means that all states from class *gobothways* satisfy the requirement Φ.

[1]For the purpose of this paper, there is no need to distinguish between classes *invalid* and *gotoinvalid*, but we prefer to separate them for reasons of symmetry.

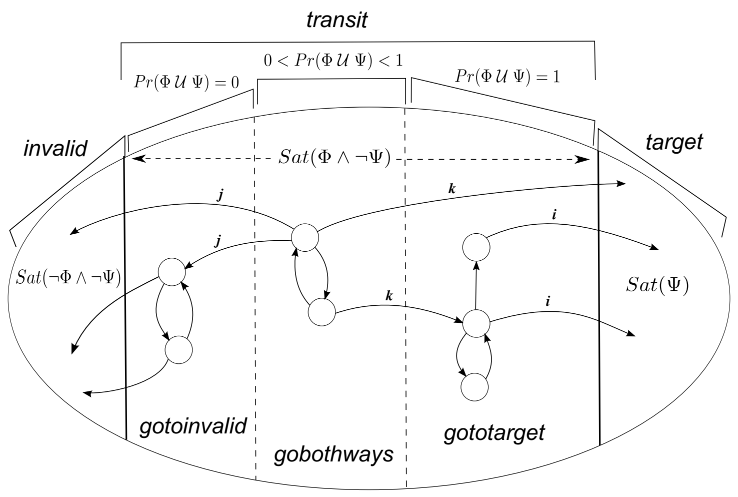

transit
$0 < Pr(\Phi\,\mathcal{U}\,\Psi) < 1$
$Pr(\Phi\,\mathcal{U}\,\Psi) = 0$
$Pr(\Phi\,\mathcal{U}\,\Psi) = 1$
invalid
target
$Sat(\Phi \wedge \neg\Psi)$
$Sat(\neg\Phi \wedge \neg\Psi)$
$Sat(\Psi)$
j
j
k
k
i
i
gotoinvalid
gobothways
gototarget

Figure 1: Partitioning of the state space

For the purpose of rate reduction, we construct a reduced (parametric, see below) SMC $\mathscr{G}$ from SMC $\mathscr{P}$, where certain transition rates are reduced by factors i, j and k and all the states in classes *invalid* and *target* are made absorbing.

Definition 5 (Reduced parametric SMC $\mathscr{G}$) *Given SMC $\mathscr{P}$ as in Def. 1, a CSL requirement $\Phi_t = P_{\sim b}(\Phi\ \mathscr{U}^{\leq t}\ \Psi)$ and three reduction factors (parameters) $0 < i, j, k \leq 1$. The reduced SMC $\mathscr{G}(\Phi,\Psi)$ is defined as a tuple $(S_\mathscr{G}, R_\mathscr{G}, L_\mathscr{G})$ where:*

- *$S_\mathscr{G} = S_\mathscr{P}$, and the state space partitioning into classes is taken from $\mathscr{P}$*
- $R_\mathscr{G}(s,s') = \begin{cases} 0 & s \in target \vee s \in invalid \\ i \cdot R_\mathscr{P}(s,s') & s \in gototarget \wedge s' \in target \\ j \cdot R_\mathscr{P}(s,s') & s \in gobothways \wedge s' \in (gotoinvalid \cup invalid) \\ k \cdot R_\mathscr{P}(s,s') & s \in gobothways \wedge s' \in (gototarget \cup target) \\ R_\mathscr{P}(s,s') & otherwise \end{cases}$
- $L_\mathscr{G} = L_\mathscr{P}$

Let T_i denote the set of transitions whose rate is multiplied by reduction factor i (and analogously for T_j and T_k).

Considering the reduction factors i, j and k as variables, the reduced SMC $\mathscr{G}$ is a parametric SMC. Once they are fixed, $\mathscr{G}$ is a standard SMC. Fig. 1 shows how the transition rates between certain state classes will be multiplied by the reduction factors. The purpose of this multiplication will be explained in the course of the paper.

2.1 A result for untimed CSL

In [13], we presented algorithms for the rate reduction problem for untimed *Until* requirements, based on the following principle: In case of an upper probability bound ($\leq b$), all transition rates from *gobothways* to *gototarget* and to *target* are reduced by a global factor of $0 < k \leq 1$. Similarly, in case of a lower probability bound ($\geq b$), all rates from *gobothways* to *gotoinvalid* and to *invalid* are reduced by a factor of $0 < j \leq 1$. The idea is to thereby influence the branching probabilities of states from class *gobothways* in the desired direction. Theorem 3.2 of [13] states that following this strategy, the parameter synthesis problem for untimed *Until* requirements can always be solved for all states of class *gobothways*. We are going to use this result in the sequel, in combination with Eq. (1).

2.2 Binary Search Method (BSM)

In the time-bounded CSL scenario, we use the Binary Search Method (BSM) to approximate the maximum satisfying value of the reduction factor, where we search the range between 0 and 1 up to a precision of a given $\varepsilon > 0$. Since a closed-form solution for the transient state probabilities for a parametric system is not available, we use an approximation via BSM with multiple evaluations using uniformisation. At the start of the reduction process, the search interval is $(0,1]$. BSM continues to halve the search interval until its width is at most the predefined precision ε. In pseudo code, BSM reads as follows, the initial call being `BSM(0,1)`:

```
BSM(lower, upper)
  {
   while upper-lower > epsilon
   {
     middle = (lower + upper) / 2;
        if "middle satisfies the requirement" then
          lower = middle;
        else
          upper = middle;
   }
   return lower;
  }
```

As a result, BSM returns the lower bound of the final search interval, where the search is considered unsuccessful if that returned value is zero.

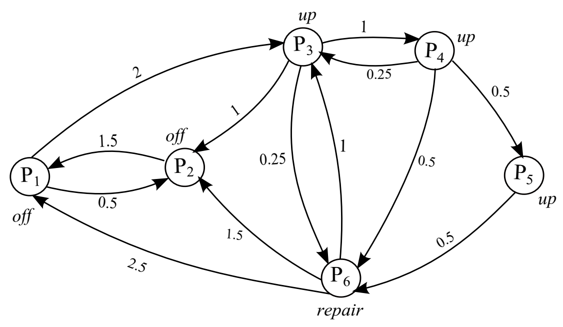
3 Example

Before giving the general rate reduction algorithms, we will explain our method with the help of an example for the two cases of upper time-bounded CSL *Until* requirements for the model SMC $\mathscr{P}$ given in Fig. 2. Plant $\mathscr{P}$ (SMC) models an abstract machine with 6 states, which can be *off*, *up* or under *repair*. When the machine is *up*, it can go into *off* or *repair* with some predefined rates. We present our new heuristics/algorithms, partly based on earlier heuristics given in our paper [13], to find reduction factors i, j and k (in case $\Phi = P_{\sim b}(up\ \mathscr{U}^{\leq t}\ repair)$ is violated). In order to create the reduced parametric SMC $\mathscr{G}$, we need to partition $\mathscr{P}$ by using the CSL path formula $\varphi = up\ \mathscr{U}\ repair$ according to Def. 4. SMC $\mathscr{G}$ for this example is given in Fig. 3, which also shows the partitioning. Note that for this example, class *gotoinvalid* is empty.

3.1 Case 1

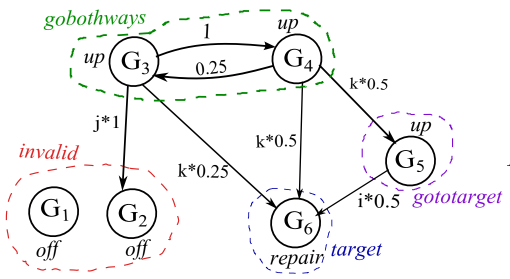
We have an abstract model as shown in $\mathscr{P}$ and the user property Φ_1 in this case shall be given as in Eq. (2).

$$\Phi_1 := P_{\leq 0.2}(\varphi), \text{where } \varphi = up\ \mathscr{U}^{\leq 5}\ repair \tag{2}$$



$$R_{\mathcal{P}} = \begin{pmatrix} 0 & 0.5 & 2 & 0 & 0 & 0 \\ 1.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0.25 \\ 0 & 0 & 0.25 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0.5 \\ 2.5 & 1.5 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Figure 2: Plant $\mathscr{P}$ along with rate matrix $R_{\mathscr{P}}$



$$R_{\mathcal{G}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & j*1 & 0 & 1 & 0 & k*0.25 \\ 0 & 0 & 0.25 & 0 & k*0.5 & k*0.5 \\ 0 & 0 & 0 & 0 & 0 & i*0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 3: Reduced SMC $\mathscr{G}$ and its rate matrix

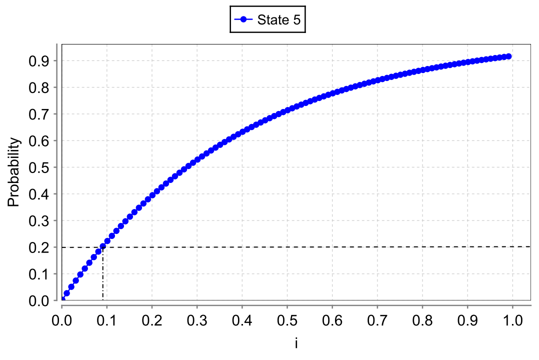
Model checking with PRISM [10], we get the probabilities for each state as follows:

$$Pr(P_1, \varphi) = 0 \leq 0.2 \tag{3}$$
$$Pr(P_2, \varphi) = 0 \leq 0.2 \tag{4}$$
$$Pr(P_3, \varphi) = 0.47323 > 0.2 \tag{5}$$
$$Pr(P_4, \varphi) = 0.83443 > 0.2 \tag{6}$$
$$Pr(P_5, \varphi) = 0.91791 > 0.2 \tag{7}$$
$$Pr(P_6, \varphi) = 1 > 0.2 \tag{8}$$

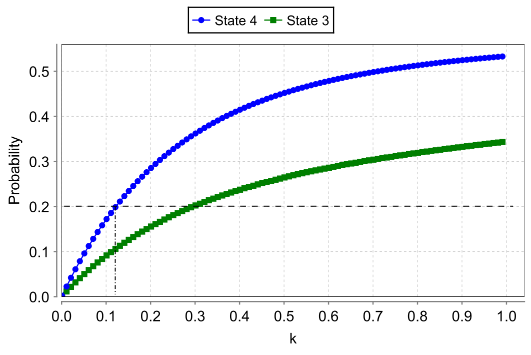
From the above probability values, we can see that $Sat(\Phi_1) = \{P_1, P_2\}$. Our focus is only on classes *gobothways* and *gototarget*, as the states from the other classes trivially satisfy/violate the requirement Φ_1. The user property Φ_1 is violated for all the states inside our classes of interest, i.e., P_3, P_4 and P_5. Hence, the process of rate reduction is required. Before the start of adaptation procedure, the values of i, k and j are equal to 1. Now, since the probability of reaching the *target* class within the specified time bound needs to be reduced, we have to slow down the rates going towards class *target*, i.e., adapt reduction factors i and k, such that the probabilities of the states in classes *gototarget* and *gobothways* to satisfy φ will fall below 0.2.

Initially, we adapt the i value, because the probability of the states in *gototarget* class will not get affected by adapting the k value, whereas the vice versa is not true. To graphically show the process of finding the satisfying range of the reduction factor i, we plotted[2] the probabilities at equidistant discrete points of i. The curve for state P_5 of *gototarget* class is shown in the Fig. 4a.

[2]All the graphs are created using PRISM tool [10].



(a) Adapting i for State 5; y-axis is $Pr(P_5, \varphi)$



(b) Adapting k for states 3 and 4, for fixed $i = 0.089$; y-axis is $Pr(\star, \varphi)$

Figure 4: Graphs for Case 1

Finding the reduction factors by solving a set of equations as in the case of untimed CSL is not feasible[3] for the time-bounded *Until* case. Hence, we used BSM to approximately find the satisfaction range of i, which is $0 < i \leq 0.089$.

Now, we will focus on the states from class *gobothways* (states P_3 and P_4). In order to make the states from this class satisfying, we need to adapt the k value. In case 1, BSM considers the state with highest probability (here it is state 4) within the class. While adapting k, we keep the i value fixed to its maximum from the range which we found earlier. By doing so, we obtain the range of k to satisfy Eq. (2) to be $(0 < k \leq 0.122)$. The graph in Fig. 4b, shows the probability plots for states 3 and 4 of $\mathscr{P}$ while reducing factor k. The k value can also be read from the graph in Fig. 4b, the curve for state 4 (blue curve) falls below the required probability bound 0.2 at $k = 0.122$. Hence, such a k range will be the solution for the whole $\mathscr{P}$. After applying the reduction factors i and k, the probabilities of the states in *gobothways* and *gototarget* class modify as follows, and therefore satisfy the user requirement.

$$Pr(P_3, \varphi) = 0.10675 \leq 0.2, \qquad Pr(P_4, \varphi) = 0.19988 \leq 0.2, \qquad Pr(P_5, \varphi) = 0.19948 \leq 0.2$$

From Eq. (1), we know that the probability for a time-bounded user requirement is always less than or equal to the probability of untimed user requirement. Therefore, since the untimed problem can be always solved (see Sec. 2.1) we can conclude that the time-bounded variant can also always be solved.

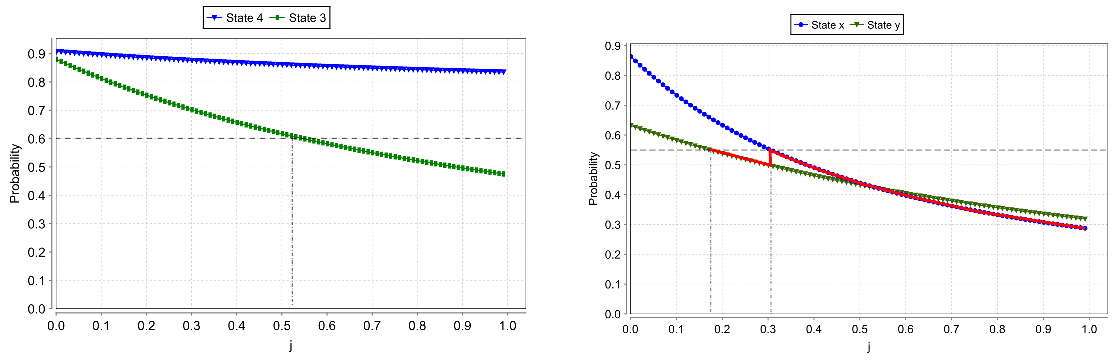
3.2 Case 2

The user requirement in case 2 shall be as in Eq. (9).

$$\Phi_2 := P_{\geq 0.95}(\varphi), \text{where again } \varphi = up\ \mathscr{U}^{\leq 5}\ repair \tag{9}$$

The actual probabilities of each state for φ are the same as shown in equations from (3) to (8). We can see that probabilities of state P_3 and P_4 (and P_5) are lower than the required probability bound i.e., ≥ 0.95, and thus the user requirement is violated for those states. In case 2, our interest will be only on states from *gobothways* class for the following reasons: 1) All states from classes *invalid* and *gotoinvalid* trivially

[3]The transient probabilities for a fixed value of the reduction factor are computed numerically via uniformisation, a parametric closed-form solution is not available.



(a) (b)

Figure 5: (a) Adapting j for states 3 and 4 for Case 2; y-axis is $Pr(\star,\varphi)$,
(b) Intersecting curves while adapting j (for a different example)

cannot satisfy the requirement. 2) States from *gototarget* may or may not satisfy the requirement, but if they don't, nothing can be improved by reducing any rate (one would have to increase the rate from *gototarget* to *target*, but this is not one of our options).

Here, since the probability should be increased beyond 0.95, the transition rates from *gobothways* to *gotoinvalid* or *invalid* should be reduced, in order to make the transitions between *gobothways* to *target* more likely to happen. For BSM, we have to reduce j, and so we choose the state with least probability (here it is state 3) within the *gobothways* class. The graph in Fig. 5a shows the probability plots of states 3 and 4 for φ. As it can be seen, while j approaches 0, the probability for φ reaches a value of 0.89 (but it does not reach 1.0 due to the time bound), but as per the requirement it should be greater than 0.95. In this case, our algorithm fails to find a satisfying solution.

However, let us take another user requirement for the same case 2 as given in Eq. (10),

$$\Psi_2 := P_{\geq 0.6}(\varphi), \text{where again } \varphi = up\ \mathscr{U}^{\leq 5}\ repair \tag{10}$$

Notice the change in the probability bound. Now, from the graph in Fig. 5a, we can observe that the satisfying range of j to be approximately $0 < j \leq 0.52$, where the curve is above required probability bound (i.e., ≥ 0.6). For the current j value, check all the states from *gobothways* class just to make sure they satisfy Φ_2. In general, reducing j makes the unbounded requirement $Pr(\Phi_1\ \mathscr{U}\ \Phi_2)$ larger (even reaches prob. 1), but it slows down the process. Hence the existence of a solution in Case 2 depends on the combination of time and probability bounds given in the user requirement.

4 Algorithms

This section explains the general rate reduction algorithms for cases 1 and 2 of upper time-bounded CSL *Until* properties. But before we present the algorithms, we need to discuss the important phenomenon of intersecting curves. In the example from Sec. 3, the probability curves for different states never showed any intersection points. However, in general the probability curves for different states may intersect, as we can observe in other examples. Fig. 5b shows such an example of intersecting curves for varying reduction factor j, but intersection of curves may also occur when varying factors i or k. For that reason, once the value of the reduction factor has been determined for the state with the highest (in case 1) or lowest (in case 2) original probability, one needs to recheck all the other states from the class. As long as at least one of them violates the requirement, the value of the reduction factor has to be reduced further. This is the reason why our algorithms in this section employ while-loops for determining the reduction factors.

4.1 Case 1

The CSL requirement in this case is given as $\Phi_1 := P_{<b}(\Phi\ \mathscr{U}^{\leq t}\ \Psi)$, and the state space of $\mathscr{P}$ is partitioned according to Def. 4. For any state s from *invalid* $\cup$ *gotoinvalid*, it holds that $Pr(s, \Phi\ \mathscr{U}^{\leq t}\ \Psi) = 0$, and since $0 < b$ those states trivially satisfy the requirement. Similarly, for states from class *target* the probability to follow a satisfying path is 1, thus *target*-states trivially violate the requirement (and nothing can be done about that).

Therefore, the algorithm will only address states from classes *gobothways* and *gototarget*. Based on the fact stated by Eq. (1) and the fact that the untimed problem can always be solved (see Sec. 2.1), the problem for upper time-bounded *Until* as in Φ_1 can always be solved for these two classes. The algorithm proceeds by first adapting factor i on transitions from *gototarget* to *target*. It starts to reduce the probability for s_h, the state from class *gototarget* with the highest probability. Since the probability curves of different states within class *gototarget* may intersect with each other, once a satisfying i value is found for s_h, we need to check whether all the other states of class *gototarget* also satisfy Φ_1. As long as any of the states violates Φ_1, we continue the process of reducing i (while loop) by selecting the state with highest probability at that instance. Once all states from class *gototarget* satisfy Φ_1 factor i will remain fixed. As a second step, a similar procedure is followed for class *gobothways*, where the algorithm reduces factor k on transitions from *gobothways* to *gototarget* $\cup$ *target*. Again, because of possible intersection of probability curves, the process needs to be continued for class *gobothways* as motivated above. Algorithm 1 formalizes this rate reduction process for Case 1.

4.2 Case 2

In this case, the general form of CSL property to be checked is given as $\Phi_2 := P_{\geq b}(\Phi\ \mathscr{U}^{\leq t}\ \Psi)$. As already stated in Case 1, for any state s from *invalid* $\cup$ *gotoinvalid*, it holds that $Pr(s, \Phi\ \mathscr{U}^{\leq t}\ \Psi) = 0$, and since $0 \ngeq b$ those states trivially violate the requirement (and nothing can be done about it). Similarly, for states from class *target*, the probability to follow a satisfying path is 1, thus *target*-states trivially satisfy the requirement. For any state from class *gototarget*, the probability of satisfying paths may be above or below the bound b. For states of the former type, the requirement is satisfied, but for states of the latter type, it is not and nothing can be done about it, since one would have to accelerate the paths towards *target*, which is not possible (since we only allow rate reductions).

Hence, we only focus on the remaining class *gobothways* for rate reduction. If the given Φ_2 is violated, then the branching probability (to eventually reach class *target* instead of class *invalid*) is too low, or the speed of moving to class *target* is too slow. From the result for untimed *Until* (cf. Sec. 2.1) we know that the branching probability can be increased to any desired value (arbitrarily close to 1) by the reduction factor j. Following the similar strategy in case of time-bounded *Until* may lead to a solution, but not always (because of the speed being too slow), as demonstrated by example in Sec. 3.2. So the strategy is to try reducing factor j, and return j if a satisfying value is found, else return *fail*. Again, since probability curves may intersect, the search for a satisfying value of j needs to be performed in a while loop. The general algorithm for this case is given in Algorithm 2.

Algorithm 1 Rate reduction for $\Phi_1 := P_{\leq b}(\Phi\, \mathscr{U}^{\leq t}\, \Psi)$

```
**Input:** SMC $\mathscr{P}$ and time-bounded CSL *Until* property $\Phi_1$.
**Output:** Satisfying $i$, $k$ reduction factors and corresponding sets of transitions $T_i, T_k$
if $gobothways \cup gototarget = \emptyset$ then
    quit                                          ▷ No states whose probabilities can be modified
else
    if $\mathscr{P} \models_{gobothways \cup gototarget} \Phi_1$ then      ▷ No need for rate reduction
        quit
    else
        $\mathscr{G} = (S_\mathscr{G}, R_\mathscr{G}, L_\mathscr{G})$; $i = 1$; $k = 1$    ▷ Construct reduced SMC (Def. 5), initialize red. factors
        while TRUE do                                     ▷ Find i for class gototarget
            for current $i$, find $s_h = \arg\max_s\{Pr(s, \Phi\mathscr{U}^{\leq t}\Psi) \mid s \in gototarget\}$
            Apply BSM to $s_h$ to find satisfying $i$
            Check all $s \in gototarget$ for current $i$
            if any of the states still violates $\Phi_1$ then
                continue
            else
                fix factor $i$ and corresponding set $T_i$ ; break
            end if
        end while
        while TRUE do                                     ▷ Find k for class gobothways
            for current $k$ value, find $s'_h = \arg\max_s\{Pr(s, \Phi\mathscr{U}^{\leq t}\Psi) \mid s \in gobothways\}$
            Apply BSM to $s'_h$ to find satisfying $k$
            Check all $s \in gobothways$ for current $k$
            if any of the states still violates $\Phi_1$ then
                continue
            else
                break
            end if
        end while
        return factors $i$, $k$ and corresponding sets $T_i$, $T_k$
    end if
end if
```

4.3 A comment on optimality of Algorithm 2

As argued in Sec. 4.1, Algorithm 1 always succeeds in making all states of $gobothways \cup gototarget$ satisfying, whereas Algorithm 2 does not always find as solution. If Algorithm 2 fails, then there does not exist a common reduction factor j that will make all the states of class *gobothways* to satisfy Φ_2. In this case, some states from *gobothways* may indeed satisfy Φ_2, but not all of them. Algorithm 2 is not always optimal in the following sense: There exist cases where the algorithm fails but where it would be possible to use a more general form of rate reduction in order to make more (or maybe even all) states of *gobothways* to satisfy Φ_2. In those cases, it is possible to further increase the probability of certain states from *gobothways* to satisfy the path formula $\Phi\, \mathscr{U}^{\leq t}\, \Psi$ by reducing not only the rates from class *gobothways* to class $gotoinvalid \cup invalid$ (as Algorithm 2 does), but also reducing the rates of

Algorithm 2 Rate reduction for $\Phi_2 := P_{\geq b}(\Phi\, \mathscr{U}^{\leq t}\, \Psi)$

```
**Input:** SMC $\mathscr{P}$ and time-bounded CSL *Until* property $\Phi_2$.
**Output:** Satisfying $j$ reduction factor and corresponding set of transitions $T_j$, or *fail*
if $gobothways = \emptyset$ then
    quit                                              ▷ No states whose probabilities can be modified
else
    if $\mathscr{P} \models_{gobothways} \Phi_2$ then                    ▷ No need for rate reduction
        quit
    else
        $\mathscr{G} = (S_{\mathscr{G}}, R_{\mathscr{G}}, L_{\mathscr{G}})$; $j = 1$       ▷ Construct reduced SMC (Def. 5), initialize red. factor
        while TRUE do                                         ▷ Find j for class gobothways
            for current $j$ value find, $s_l = \arg\min_s \{Pr(s, \Phi\, \mathscr{U}^{\leq t}\, \Psi) \mid s \in gobothways\}$
            For state $s_l$ apply BSM to reduce $j$
            if solution found then                                      ▷ i.e., if $j > 0$
                Check all $s \in gobothways$ with current $j$ value
                if any of the states still violates $\Phi_2$ then
                    continue
                else
                    return factor $j$ and corresponding set $T_j$
                end if
            else
                return *fail*
            end if
        end while
    end if
end if
```

certain transitions *among* the states of class *gobothways* (i.e. transitions within *gobothways*). However, selecting individual transitions among states of class *gobothways* for rate reduction is a very difficult task for which there are currently no known efficient algorithms or even heuristics, so this is beyond the scope of this paper.

4.4 Complexity

We assume a SMC with N states and $M = O(N^2)$ transitions. The time complexity for model checking time-bounded *Until* is the same as for CTMC transient solution by uniformisation, which is known to be $O(M \cdot q \cdot t)$, where q is the uniformisation rate and t is the time bound [1] (note that the uniformisation rate, which is at least the maximum of the states exit rates, actually decreases as we reduce transitions rates). For constructing the reduced SMC $\mathscr{G}$ the state classes need to be determined and the reduction factors need to be inserted, all of which can be done in time $O(M)$. Given the desired precision ε, each run of BSM requires $O(\log_2 \frac{1}{\varepsilon})$ steps. Therefore, the overall time complexity of Algorithms 1 and 2 is $O(M \cdot q \cdot t \cdot N \cdot \log_2 \frac{1}{\varepsilon})$, where the factor N reflects the iterations of the while loop(s).

5 Conclusion

In this paper, we have considered systems (also called "plants") specified as state-labelled Markov chains, and user requirements given as upper time-bounded CSL formulas (without multiple or nested *Until* operators). Whenever a user requirement is violated by some or all states of the plant, we try to *repair* the plant by reducing some dedicated sets of its transition rates, such that eventually the user requirement will be satisfied. We only allow for a slowdown of transition rates (but without completely disabling any transitions), since in most practical situations increasing the transition rates is not feasible. Upon model checking, some states in the plant's state space may already satisfy the requirement, whereas some others may not. Some states will have constant probability which cannot be modified by reducing the rates. Depending on the type of probability bound for the *Until* formula, we identified two possible cases for which we have devised simple and intuitive algorithms along with necessary and sufficient conditions for the solutions to exist. The algorithms partition the state space into different classes and find the appropriate sets of transitions between those classes whose rates need to be reduced, as well as the respective reduction factors.

Our ideal goal is to identify the maximum number of states which can be made to satisfy the user requirement. We have shown that Algorithm 1 (for the upper probability bound) always achieves this goal of maximality, but as discussed in Sec. 4.3, Algorithm 2 (for the lower probability bound) is not always optimal.

In this paper, we have not addressed lower time-bounded CSL user requirements, i.e. the case of $\Phi = P_{\sim b}(\Phi\, \mathscr{U}^{\geq t}\, \Psi)$. Based upon the probability bound, lower time-bounded CSL requirements can be further divided into two categories to which we refer as cases 3 and 4, and we are planning to elaborate on these cases in a forthcoming paper. Another interesting topic for future work would be to consider time-bounded CSL formulas with multiple or nested *Until* operators, a difficult problem, as already hinted at in Sec. 2. Furthermore, there is the interesting open question of how to repair a given plant at the level of the high-level modelling formalism, instead of at the level of Markov chain.

References

[1] C. Baier, B. R. Haverkort, H. Hermanns & J.-P. Katoen (2003): *Model-Checking Algorithms for Continuous-Time Markov Chains*. *IEEE Trans. Software Eng.* 29(6), pp. 524–541, doi:10.1109/TSE.2003.1205180.

[2] E. Bartocci, R. Grosu, P. Katsaros, C.R. Ramakrishnan & S.A. Smolka (2011): *Model Repair for Probabilistic Systems*. In: *TACAS 2011*, *Lecture Notes in Computer Science* 6605, Springer, pp. 326–340, doi:10.1007/978-3-642-19835-9_30.

[3] M. Ceska, F. Dannenberg, M. Z. Kwiatkowska & N. Paoletti (2014): *Precise Parameter Synthesis for Stochastic Biochemical Systems*. In: *Computational Methods in Systems Biology - 12th International Conference, CMSB*, pp. 86–98, doi:10.1007/978-3-319-12982-2_7.

[4] T. Chen, E.M. Hahn, T. Han, M. Kwiatkowska, H. Qu & L. Zhang (2013): *Model Repair for Markov Decision Processes*. In: *Theoretical Aspects of Software Engineering (TASE), 2013*, pp. 85–92, doi:10.1109/TASE.2013.20.

[5] C. Daws (2004): *Symbolic and Parametric Model Checking of Discrete-Time Markov Chains*. In: *Theoretical Aspects of Computing - ICTAC 2004*, *Lecture Notes in Computer Science* 3407, Springer Berlin Heidelberg, pp. 280–294, doi:10.1007/978-3-540-31862-0_21.

[6] E. M. Hahn, H. Hermanns, B. Wachter & L. Zhang (2010): *PARAM: A Model Checker for Parametric Markov Models*. In: *Computer Aided Verification, 22nd International Conference, CAV*, pp. 660–664, doi:10.1007/978-3-642-14295-6_56.

[7] E. M. Hahn, H. Hermanns & L. Zhang (2011): *Probabilistic reachability for parametric Markov models*. *STTT* 13(1), pp. 3–19, doi:10.1007/s10009-010-0146-x.

[8] T. Han, J.-P. Katoen & A. Mereacre (2008): *Approximate Parameter Synthesis for Probabilistic Time-Bounded Reachability*. In: *Proceedings of the 29th IEEE Real-Time Systems Symposium, RTSS*, pp. 173–182, doi:10.1109/RTSS.2008.19.

[9] J.-P. Katoen, I.S. Zapreev, E.M. Hahn, H. Hermanns & D.N. Jansen (2011): *The Ins and Outs of the Probabilistic Model Checker MRMC*. *Perform. Eval.* 68(2), pp. 90–104, doi:10.1016/j.peva.2010.04.001.

[10] M. Z. Kwiatkowska, G. Norman & D. Parker (2011): *PRISM 4.0: Verification of Probabilistic Real-Time Systems*. In: *Computer Aided Verification - 23rd International Conference, CAV*, pp. 585–591, doi:10.1007/978-3-642-22110-1_47.

[11] K. Mehlhorn (1984): *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness*. *EATCS Monographs on Theoretical Computer Science* 2, Springer. Available at `http://www.mpi-sb.mpg.de/~mehlhorn/DatAlgbooks.html`.

[12] S. Pathak, E. Ábrahám, N. Jansen, A. Tacchella & J.-P. Katoen (2015): *A Greedy Approach for the Efficient Repair of Stochastic Models*. In: *NASA Formal Methods - 7th Int. Symposium*, pp. 295–309, doi:10.1007/978-3-319-17524-9_21.

[13] B.-S.-K. Tati & M. Siegle (2015): *Parameter and Controller Synthesis for Markov Chains with Actions and State Labels*. In: *2nd International Workshop on Synthesis of Complex Parameters (SynCoP'15)*, *OASIcs* 44, pp. 63–76, doi:10.4230/OASIcs.SynCoP.2015.63.