Model Checking Branching Time Properties for Incomplete Markov Chains

Shiraj Arora(✉) and M. V. Panduranga Rao

Indian Institute of Technology Hyderabad, Kandi, India
{cs14resch11010,mvp}@iith.ac.in

Abstract. In this work, we discuss a numerical model checking algorithm for analyzing incompletely specified models of stochastic systems, specifically, Discrete Time Markov Chains (DTMC). Models of a system could be incompletely specified for several reasons. For example, they could still be under development or, there could be some doubt about the correctness of some components. We restrict ourselves to cases where incompleteness can be captured by expanding the logic of atomic propositions to a three valued logic that includes an *unknown* truth value. We seek to answer meaningful model checking queries even in such circumstances.

The approach we adopt in this paper is to develop the model checking algorithm from first principles. We develop a tool based on the algorithm and compare the performance of this approach with the indirect approach of invoking a binary model checker.

Keywords: Probabilistic model checking · Discrete Time Markov Chains · Probabilistic computational tree logic · Three valued logic · Incomplete models

1 Introduction

Probabilistic models are widely used to represent real-world systems that exhibit stochastic behaviour, like cyber-physical systems, biological processes, network and security protocols. Examples of such probabilistic models are Markov chains like Discrete and Continuous Time Markov Chains (DTMC and CTMC respectively) [3], Markov Decision Processes (MDP), Constrained Markov Chains [8] and Probabilistic Automata [13]. Verification of these types of probabilistic models involves asserting whether or not the system design exhibits the required behavior. The required behavior or property is formally specified as statements in logics like Probabilistic Computation Tree Logic (PCTL) [16] and Continuous Stochastic Language (CSL) [4].

Probabilistic Model Checking is a formal technique to analyze and verify the required behaviour of stochastic systems. Given a probabilistic model that describes a stochastic system, and a formal specification of the required behaviour of the system, the goal of probabilistic model checking is to decide

F. Biondi et al. (Eds.): SPIN 2019, LNCS 11636, pp. 20–37, 2019.
https://doi.org/10.1007/978-3-030-30923-7_2

whether the system exhibits the required behaviour or not. Probabilistic model checking has been explored in the literature using either numerical [4,12,16] or statistical algorithms [22,23,25].

Traditionally, systems are analyzed through model checking once the entire information about a model is available, at least in principle. An interesting question is if this analysis can be done when the model has incomplete information. The incompleteness may arise because of either (i) nonavailability of input information about either state space or transitions between states, (ii) if the correctness of some module is in doubt or (iii) loss of information due to abstraction of models, or some combination of the three. To capture incomplete information, a natural choice is to expand binary logic to include a third *unknown* truth value. We denote this by the question mark "?". Depending upon the type of incompleteness, different techniques have been reported to verify both stochastic as well as non stochastic incomplete models.

There exists a significant body of work on model checking with three valued logics. Bruns and Godefroid [6,7] use three valued modal logic to represent models with partial state space. These models are then verified by using model checking algorithms for binary truth values. Similarly, Chechik et al. [11] used multi-valued logic to represent incomplete and inconsistent information in a model. They verified such a model using a symbolic multi-valued CTL model checker. Abstraction is often used to deal with problems of state space explosion in model checking. However, such an approach may cause loss of information in the model. This incompleteness in the abstracted model can be represented using a third truth value. Godefroid et al. [15] and Chechik [10] discussed the verification of abstracted models using three valued LTL and CTL model checkers, respectively. Abstraction is also commonly used in verification of stochastic models like Markov chains, to overcome the problem of state-space explosion [14,17,19]. Besides abstraction, incompleteness in stochastic systems may arise from imprecise transition probabilities obtained from statistical experiments. For example, discrete-time Markov chains have been defined wherein transition probabilities are intervals instead of exact values to handle imprecision [18,20]. Verification of such interval based discrete-time Markov chains works by either reducing it to a class of discrete-time Markov chains or to a Markov decision process [5,9,24].

Another way of capturing incomplete information, the one on which this paper is based, is to allow some atomic propositions to assume the "?" truth value in some states [2]. An interesting question is to determine which properties of the system can be verified in the absence of complete information–is there sufficient information in the model to evaluate a particular property to either True or False? It is possible to answer this question by invoking PCTL model checking algorithms twice [2]. We refer to this approach as *2MC*.

In this paper, we adopt a direct approach that solves the numerical model checking problem from first principles as exposited in [4,21]. We call this approach *1MC*. While this approach is an adaptation of the standard numerical model checking algorithm, the fact that true and false are no longer

complementary to each other raises some complications, which we address. We also show examples to illustrate the approach and practical applications of the problem formulation and the solution.

Intuitively, one expects the *1MC* algorithm to perform better in cases when the *2MC* algorithm has to invoke the model checker twice and *2MC* to perform better when it needs to invoke it only once. We back this intuition up with experimental evidence.

The rest of the paper is arranged as follows. Section 2 discusses the syntax and semantics of the modeling and specification formalism for incomplete models. Section 3 discusses the model checking algorithm. Implementation details, results and comparison with *2MC* are discussed in Sect. 4. Section 5 concludes the paper with a brief discussion on future directions.

2 qDTMC and qPCTL

2.1 Discrete Time Markov Chain with Question Marks (qDTMC)

A Discrete Time Markov Chain with question marks (qDTMC) extends the traditional DTMC to account for incomplete information in a model.

Definition 1. *A qDTMC is a tuple* $\mathcal{M} = (S, i_{init}, \mathbb{P}, AP, L)$ *where*

- S *is a finite non-empty set of states,*
- $i_{init} : S \to [0, 1]$ *is the initial distribution, such that* $\sum_{s \in S} i_{init}(s) = 1$,
- $\mathbb{P} : S \times S \to [0, 1]$ *gives the transition probability between two states in* S *such that:*
$$\forall s \in S : \sum_{s' \in S} \mathbb{P}(s, s') = 1,$$
- AP *is a set of atomic propositions, and*
- $L : S \times AP \to \{T, F, \mathbf{?}\}$ *assigns a truth value from the set* $\{T, F, \mathbf{?}\}$ *to each atomic proposition* $a \in AP$ *in a state* $s \in S$.

A qDTMC differs from a DTMC only in terms of its labelling function. The truth values T and F correspond that an atomic proposition being true and false respectively in the state s. If it is not known whether an atomic proposition is true or false in s, then the truth value **?** is assigned to the atomic proposition.

Figure 1 illustrates an example qDTMC $\mathcal{M}$ wherein the state s_0 is the initial state with probability 1 in the state space $S = \{s_0, s_1, ..., s_6\}$. The weight on each transition between the states denotes the probability of the transition. $AP = \{p, q\}$ is a set of atomic propositions in $\mathcal{M}$. The truth value of each atomic proposition in a state is also represented in the qDTMC. For instance, $\neg pq?$ denotes p is false and q is unknown in the state s_0 and pq denotes both p and q are true in the state s_2. We consider qDTMC $\mathcal{M}$ as a running example in the paper.

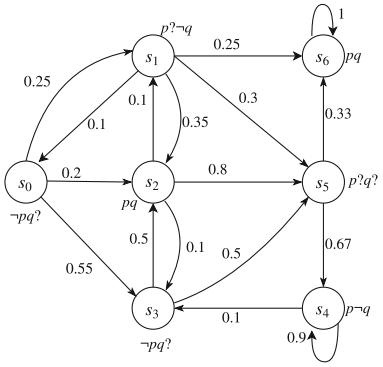


Fig. 1. Example of a qDTMC

Definition 2. *A path π in a qDTMC $\mathcal{M}$ is a sequence of states $s_0, s_1, s_2, \ldots$ such that for all $i = 0, 1, 2, \ldots$ $s_i \in S$ and $\mathbb{P}(s_i, s_{i+1}) > 0$. The $i + 1^{th}$ state in a path π is denoted by $\pi[i]$. The set $Path(s)$ is the set of all infinite paths starting from state s in $\mathcal{M}$.*

Definition 3. *A cylinder set $C(\omega)$ is the set of all infinite paths with a common finite prefix $\omega = s_0, s_1, \ldots, s_n$. The probability measure μ of $C(\omega)$ in the qDTMC $\mathcal{M}$ can be defined as*

$$\mu(C(\omega)) = \prod_{i=0}^{n-1} \mathbb{P}(s_i, s_{i+1})$$

2.2 qPCTL

Probabilistic Computation Tree Logic with question marks (qPCTL) is an extension of PCTL [16], defined to formally express a property required of a qDTMC model. In what follows, our notational convention will be similar to that of [2]. The syntax of qPCTL is the same as that of PCTL:

Syntax:

$$\begin{aligned} \Phi &::= \top \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid Pr_{\bowtie\theta}[\psi] \\ \psi &::= X\Phi \mid \Phi_1 U \Phi_2 \mid \Phi_1\ U^{\leq k}\ \Phi_2 \end{aligned}$$

where Φ, Φ_1, and Φ_2 are state formulas, ψ is a path formula, a is an atomic proposition, $\theta \in [0, 1]$ defines the probability constraint, $\bowtie \in \{ <, >, \leq, \geq\}$ represents the set of comparison operators, and $k \in \mathbb{N}$ is the time bound. The X, U, and $U^{\leq k}$ operators are called *Next*, *Until* and *Bounded Until* respectively.

Recall that an atomic proposition a can have one of the three truth values $\{T, F, ?\}$ in a qPCTL formula. Thus, in addition to verifying a property as true (T) or false (F), a qPCTL formula can also be evaluated to "unknown" (**?**). The conditions for which the logic returns **?** are incorporated into the semantics of qPCTL.

For the truth values (T, F, ?) in qPCTL, the logical operations ($\wedge$, $\vee$, $\neg$) are as defined in Tables 1, 2 and 3.

Table 1. The AND operator

$\wedge$	T	?	F
T	T	?	F
?	?	?	F
F	F	F	F

Table 2. The OR operator

$\vee$	T	?	F
T	T	T	T
?	T	?	?
F	T	?	F

Table 3. The NOT operator

$\neg$	
T	F
?	?
F	T

Semantics:
Each state formula Φ in qPCTL is verified to either T, F, or **?** in a state $s \in S$ that is $(s, \Phi) = \{T, F, ?\}$ as:

1. The qPCTL formula Φ can trivially be $\top$.

$$(s, \top) = T; \qquad (s, \neg\top) = \neg(s, \top) = \ F.$$

2. An atomic proposition a in a qDTMC can have three possible truth values $\{T, F, ?\}$. Thus, a qPCTL formula $\Phi = a$ in a state s is evaluated to $\{T, F, ?\}$ if

$$(s, a) = \begin{cases} T \text{ iff } L(s, a) = T \\ F \text{ iff } L(s, a) = F \\ ? \text{ iff } \ L(s, a) =? \end{cases}$$

3. Using the *NOT* and *AND* operator from Tables 3 and 1 respectively, we have:

$$(s, \neg\Phi) = \begin{cases} T \text{ iff } (s, \Phi) = F \\ F \text{ iff } (s, \Phi) = T \\ ? \text{ iff } \ (s, \Phi) =? \end{cases} \text{ and } (s, \Phi_1 \wedge \Phi_2) = \begin{cases} T \text{ iff } (s, \Phi_1) = T \wedge (s, \Phi_2) = T \\ F \text{ iff } (s, \Phi_1) = F \vee (s, \Phi_2) = F \\ ? \text{ otherwise} \end{cases}$$

4. If a qPCTL formula contains a probabilistic operator that is $\Phi = Pr_{\bowtie\theta}[\psi]$, then the probability measure of paths starting from state s that evaluate path formula ψ to true or false is calculated separately. The formula Φ is then verified as follows:

$$(s, Pr_{\bowtie\theta}[\psi]) = \begin{cases} T \text{ if } \mu\{\pi \in Path(s) : (\pi, \psi) = T\} \bowtie \theta \\ F \text{ if } \mu\{\pi \in Path(s) : (\pi, \psi) = F\} \bowtie 1 - \theta \\ ? \text{ otherwise} \end{cases}$$

A path formula ψ for a path $\pi \in Path(s)$ has the following semantics:

1. *Next operator*: A path formula of form $\psi = X\Phi$ is verified to $\{T, F, ?\}$ for a path π if the state formula Φ is evaluated to $\{T, F, ?\}$, respectively, in the second state of π.

$$(\pi, X\Phi) = \begin{cases} T \text{ if } (\pi[1], \Phi) = T \\ F \text{ if } (\pi[1], \Phi) = F \\ ? \text{ if } (\pi[1], \Phi) = ? \end{cases}$$

2. *Until operator*:

$$(\pi, \Phi_1 U \Phi_2) = \begin{cases} T \text{ if } \exists i : (\pi[i], \Phi_2) = T \ \wedge \ \forall i' < i : (\pi[i'], \Phi_1) = T \\ F \text{ if } (\forall i : (\pi[i], \Phi_2) = F) \\ \quad \vee \ [\exists i : (\pi[i], \Phi_2) = T \ \wedge \ \exists i' < i : (\pi[i'], \Phi_1) = F] \\ ? \text{ otherwise.} \end{cases}$$

Thus, ψ evaluates to ? if one of the following occurs:
- Φ_2 is ? for all the states in π.
- Φ_2 is ? for at least one state in π and is never T in any of the states along the path and Φ_1 is never F.
- Φ_2 is T for some state $\pi[i]$ and Φ_1 is ? for at least one state $\pi[k]$ with $k < i$ but never F in any of the states upto $\pi[i]$.

3. *Bounded Until*: A path formula of the form $\psi = \Phi_1 U^{\leq k} \Phi_2$ is verified to $\{T, F, ?\}$ same as that for *until* operator, but only for paths of finite length k.

$$(\pi, \Phi_1 U^{\leq k} \Phi_2) = \begin{cases} T \text{ if } \exists i \leq k : (\pi[i], \Phi_2) = T \ \wedge \ \forall i' < i : (\pi[i'], \Phi_1) = T \\ F \text{ if } (\forall i \leq k : (\pi[i], \Phi_2) = F) \\ \quad \vee \ [\exists i \leq k : (\pi[i], \Phi_2) = T \ \wedge \exists i' < i : (\pi[i'], \Phi_1) = F] \\ ? \text{ otherwise.} \end{cases}$$

3 qPCTL Model Checking

For a qDTMC $\mathcal{M} = (S, \mathbb{P}, i_{init}, AP, L)$, a state $s \in S$ and a qPCTL state formula Φ, we want to determine if $(s, \Phi) = T$. We note that if $(s, \Phi) \neq T$, then Φ need not be false in the state s. Unlike for DTMCs, the two statements are not complementary for qDTMCs. Hence we define three satisfaction sets $Sat_T(\Phi)$, $Sat_F(\Phi)$, and $Sat_?(\Phi)$.

Definition 4. *A state $s \in Sat_T(\Phi)$ if and only if $(s, \Phi) = T$. A state $s \in Sat_F(\Phi)$ if and only if $(s, \Phi) = F$. Finally, $s \in Sat_?(\Phi)$ if and only if $(s, \Phi) =?$.*

We now describe an algorithm 1MC, to compute these satisfaction sets $Sat_T(\Phi)$ and $Sat_F(\Phi)$ by performing a bottom-up traversal of the syntax tree of Φ. The remaining satisfaction set $Sat_?(\Phi)$ can be computed as $Sat_?(\Phi) = S \setminus [Sat_T(\Phi) \cup Sat_F(\Phi)]$. For a given state formula Φ, these satisfaction sets will partition the state space S. We now discuss the 1MC algorithm in detail. Algorithm 1 lists the pseudocode.

Non-probabilistic Operators: The satisfaction sets for each of the non-probabilistic operators in the qPCTL is based on the logical operations described in Tables 1, 2 and 3. Cases 1 through 4 of the 1MC algorithm compute the satisfaction sets for non-probabilistic operators. It is easy to see that:

Lemma 1. *For the non-probabilistic state formulas, the 1MC Algorithm (cases 1 through 4) is correct.*

Probabilistic Operators: Construction of the satisfaction sets for probabilistic operators is somewhat more complicated. We need a few definitions first.

Definition 5. *True satisfaction probability for a state s is defined as the probability measure of paths starting from s which evaluate the path formula ψ as T. It is denoted by $Pr((s, \psi) = T)$.*

Definition 6. *False satisfaction probability for a state s is defined as the probability measure of paths starting from s which evaluate the path formula ψ as F, and is denoted by $Pr((s, \psi) = F)$.*

Now, the satisfaction sets can be defined as:

$$\begin{aligned} Sat_T(Pr_{\bowtie\theta}[\psi]) &= \{s \in S \mid Pr((s,\psi) = T) \bowtie \theta\} \\ Sat_F(Pr_{\bowtie\theta}[\psi]) &= \{s \in S \mid Pr((s,\psi) = F) \bowtie 1-\theta\} \\ Sat_?(Pr_{\bowtie\theta}[\psi]) &= S \setminus [Sat_T(Pr_{\bowtie\theta}[\psi]) \cup Sat_F(Pr_{\bowtie\theta}[\psi])] \end{aligned}$$

To construct these sets, we have to calculate $Pr((s, \psi) = T/F/?)$ for path formula ψ. There are three path formulas in qPCTL–*Next*, *Until* and *Bounded Until*. We now discuss the algorithms for computing the satisfaction probabilities for these path formulas:

1. **Next operator** – $[X\Phi]$: The satisfaction probabilities for *next* operator are calculated by adding transition probabilities of the state s to the states which

Algorithm 1. Algorithm **1MC**

Function: ComputeSat(Φ)
switch (Φ)
case $\top$:
  $Sat_T(\Phi) \leftarrow S$; $Sat_F(\Phi) \leftarrow \emptyset$; $Sat_?(\Phi) \leftarrow \emptyset$
case a:
  for all $s \in S$ **do**
    if $L(s, a) = T$ **then**
      $Sat_T(\Phi) \leftarrow Sat_T(\Phi) \cup \{s\}$
    else if $L(s, a) = F$ **then**
      $Sat_F(\Phi) \leftarrow Sat_F(\Phi) \cup \{s\}$
    else
      $Sat_?(\Phi) \leftarrow Sat_?(\Phi) \cup \{s\}$
    end if
  end for
case $\neg\Phi_1$
  ComputeSat(Φ_1)
  $Sat_T(\Phi) \leftarrow Sat_F(\Phi_1)$; $Sat_F(\Phi) \leftarrow Sat_T(\Phi_1)$; $Sat_?(\Phi) \leftarrow Sat_?(\Phi_1)$
case $\Phi_1 \wedge \Phi_2$
  ComputeSat(Φ_1); ComputeSat(Φ_2)
  $Sat_T(\Phi) \leftarrow Sat_T(\Phi_1) \cap Sat_T(\Phi_2)$
  $Sat_F(\Phi) \leftarrow Sat_F(\Phi_1) \cup Sat_F(\Phi_2)$
  $Sat_?(\Phi) \leftarrow S \setminus [Sat_T(\Phi) \cup Sat_F(\Phi)]$

case $Pr_{\bowtie\theta}[\psi]$
  switch (ψ)
  case $X\Phi_1$
    ComputeSat(Φ_1)
    for all $s, s' \in S$ **do**
      if $s' \in Sat_T(\Phi_1)$ **then**
        $Pr_T(s,\psi) \leftarrow Pr_T(s,\psi) + \mathbb{P}(s,s')$
      else if $s' \in Sat_F(\Phi_1)$ **then**
        $Pr_F(s,\psi) \leftarrow Pr_F(s,\psi) + \mathbb{P}(s,s')$
      else
        $Pr_?(s,\psi) \leftarrow Pr_?(s,\psi) + \mathbb{P}(s,s')$
      end if
    end for
  case $[\Phi_1 U \Phi_2]$
    $ComputeSat(\Phi_1)$; $ComputeSat(\Phi_2)$
    $S_{=0} \leftarrow Compute\_Until\_S_{=0}$
    $S_{=?} \leftarrow Compute\_Until\_S_{=?}$
    $S_{=1} \leftarrow Compute\_Until\_S_{=1}$
    $S_{find} \leftarrow S \setminus [S_{=0} \cup S_{=1} \cup S_{=?}]$
    for all $s \in S$ **do**
      $Pr_T(s,\psi) \leftarrow Compute\_Until\_Pr_T(s)$
        ▷ Computes true satisfaction probability using equation 1.
      $Pr_F(s,\psi) \leftarrow Compute\_Until\_Pr_F(s)$
        ▷ Computes false satisfaction probability using equation 2.
    end for
  case $[\Phi_1 U^{\leq k} \Phi_2]$
    $ComputeSat(\Phi_1)$; $ComputeSat(\Phi_2)$
    $S_{=0} \leftarrow Sat_F(\Phi_2) \cap Sat_F(\Phi_1)$
    $S_{=?} \leftarrow [(Sat_?(\Phi_1) \setminus Sat_T(\Phi_2)) \ \cup \ (Sat_?(\Phi_2) \setminus Sat_T(\Phi_1))\ ]$
    $S_{=1} \leftarrow Sat_T(\Phi_2)$
    $S_{find} \leftarrow Sat_T(\Phi_1) \setminus Sat_T(\Phi_2)$
    for all $s \in S$ **do**
      $Pr_T(s,\psi) \leftarrow Compute\_BUntil\_Pr_T(s,k)$
        ▷ Computes true satisfaction probability using equation 4
      $Pr_F(s,\psi) \leftarrow Compute\_BUntil\_Pr_F(s,k)$
        ▷ Compute false satisfaction probability using equation 5
    end for
  end switch
  for all $s \in S$ **do**
    if $Pr_T(s,\psi) \bowtie \theta$ **then**
      $Sat_T(\Phi) \leftarrow Sat_T(\Phi) \cup \{s\}$
    else if $Pr_F(s,\psi) \bowtie 1-\theta$ **then**
      $Sat_F(\Phi) \leftarrow Sat_F(\Phi) \cup \{s\}$
    else
      $Sat_?(\Phi) \leftarrow Sat_?(\Phi) \cup \{s\}$
    end if
  end for
end switch
return $Sat_T(\Phi)$, $Sat_F(\Phi)$, $Sat_?(\Phi)$

satisfy Φ as T, F or ?.

$$Pr((s, X\Phi) = T) = \sum_{s' \in Sat_T(\Phi)} \mathbb{P}(s, s')$$
$$Pr((s, X\Phi) = F) = \sum_{s' \in Sat_F(\Phi)} \mathbb{P}(s, s')$$
$$Pr((s, X\Phi) = ?) = \sum_{s' \in Sat_?(\Phi)} \mathbb{P}(s, s')$$

Based on satisfaction probabilities, each state s in S belongs to only one of the three satisfaction sets $Sat_T(Pr_{\bowtie\theta}[X\Phi])$, $Sat_F(Pr_{\bowtie\theta}[X\Phi])$ or $Sat_?(Pr_{\bowtie\theta}[X\Phi])$. If $Pr((s, X\Phi) = T) \bowtie \theta$ then $s \in Sat_T(Pr_{\bowtie\theta}[X\Phi])$. Else if $Pr((s, X\Phi) = F) \bowtie 1-\theta$ then $s \in Sat_F(Pr_{\bowtie\theta}[X\Phi])$. Otherwise, s belongs to the set $Sat_?(Pr_{\bowtie\theta}[X\Phi])$.

Example 1. In Fig. 1, a qDTMC $\mathcal{M}_1$ with state space $S = \{s_0, s_1, \ldots, s_6\}$ with an initial state s_0 is given. If we want to verify a qPCTL state formula $\Phi = Pr_{\geq 0.5}[Xq]$ for $\mathcal{M}_1$, then the satisfaction sets will be: $Sat_T(\Phi_1) = \{s_1, s_3, s_6\}$, $Sat_F(\Phi_1) = \{s_4, s_5\}$ and $Sat_?(\Phi_1) = \{s_0, s_2\}$. Thus for the initial state s_0, the qPCTL state formula Φ_1 will be evaluated to ?.

2. **Until operator** – $[\Phi_1 U \Phi_2]$: Identifying B with the set of states where Φ_2 is true and C as the set of states where Φ_1 is true, a graph theoretic notion of constrained reachability is useful:

Definition 7. *Constrained reachability in a qDTMC $(s, C\ U\ B)$ is defined as an event of reaching the destination set $B \subseteq S$ from the state s such that all the preceding states in the path belong to a constraint set $C \subseteq S$.*

Thus the true satisfaction probability $Pr((s, \Phi_1 U \Phi_2) = T)$ can be calculated as the probability of constrained reachability being true for paths starting from state s. We can also denote this probability as $Pr((s, C\ U\ B) = T)$. Similarly, the false satisfaction probability $Pr((s, \Phi_1 U \Phi_2) = F) = Pr((s, C\ U\ B) = F)$ is the probability of constrained reachability being false.

Notably, it is possible that constrained reachability for a path in a qDTMC is neither true nor false. For such paths, constrained reachability is evaluated to *unknown* (?). This requires us to calculate all the reachability probabilities separately for each state in $\mathcal{M}$. For each state s, the probability that the constrained reachability is $T/F/?$ is denoted by $x_s^{(T/F/?)}$ respectively, and is defined as:

$$x_s^{(T/F/?)} = Pr((s, C\ U\ B) = T/F/?) = Pr((s, \Phi_1 U \Phi_2) = T/F/?).$$

To calculate these constrained reachability probabilities, we first partition the state space S into $\{S_{=0}, S_{=1}, S_{=?}$ and $S_{find}\}$. We now define each of these partition sets and discuss how to construct these sets later.

$$S_{=0} = \{s \in S \mid Pr((s, C\ U\ B) = F) = 1\}$$
$$B \subseteq S_{=1} \subseteq \{s \in S | Pr((s, C\ U\ B) = T) = 1\}$$
$$S_{=?} = \{s \in S \mid Pr((s, C\ U\ B) =?) = 1\}$$

The remaining states in S form the set S_{find}:

$$S_{find} = S \setminus [S_{=0} \cup S_{=1} \cup S_{=?}]$$

Thus, for states $s \in S_{=1}$, $x_s^{(T)} = 1$, $x_s^{(F)} = 0$ and $x_s^{(?)} = 0$. Similarly for states $s \in S_{=0}$, $x_s^{(F)} = 1$, $x_s^{(T)} = 0$ and $x_s^{(?)} = 0$ and for states $s \in S_{=?}$, $x_s^{(?)} = 1$, $x_s^{(T)} = 0$ and $x_s^{(F)} = 0$.

For a state s in the set S_{find}, the constrained reachability probabilities are neither exactly 1 nor 0. We now identify the paths starting from s that eventually reach the set $S_{=1}$ such that all the preceding states are from set S_{find}. These paths will evaluate $C\ U\ B$ (or $\Phi_1 U \Phi_2$) to T and the probability measure of these paths gives $x_s^{(T)}$. Similarly, $x_s^{(F)}$ is the probability measure of paths from s that reach $S_{=0}$ through the states from S_{find} and hence evaluate $C\ U\ B$ (or $\Phi_1 U \Phi_2$) to F.

The satisfaction probabilities for the path formula $[\Phi_1 U \Phi_2]$ is then calculated using the following sets of linear equations.

$$x_s^{(T)} = \begin{cases} 1 & \text{if } s \in S_{=1} \\ 0 & \text{if } s \in S_{=0} \\ 0 & \text{if } s \in S_{=?} \\ \sum\limits_{t \in S} \mathbb{P}(s,t).x_t^{(T)} & \text{if } s \in S_{find} \end{cases} \tag{1}$$

$$x_s^{(F)} = \begin{cases} 0 & \text{if } s \in S_{=1} \\ 1 & \text{if } s \in S_{=0} \\ 0 & \text{if } s \in S_{=?} \\ \sum\limits_{t \in S} \mathbb{P}(s,t).x_t^{(F)} & \text{if } s \in S_{find} \end{cases} \tag{2}$$

$$Pr((s, \Phi_1 U \Phi_2) =?) = x_s^{(?)} = 1 - [x_s^{(T)} + x_s^{(F)}] \tag{3}$$

We now discuss how to construct $S_{=0}$, $S_{=?}$ and $S_{=1}$. Pseudocode listing for these subroutines is provided in the Algorithms 2, 3 and 4. Then, the set $S_{find} = S \setminus [S_{=0} \cup S_{=1} \cup S_{=?}]$ is computed.

We first compute $S_{=0}$, the set of states that have the false satisfaction probability $Pr((s, \Phi_1 U \Phi_2) = F) = 1$. We identify the states that have false satisfaction probability less than 1. To do this, we first identify the set R of states for which Φ_2 is either T or ?. Then we do a backward search on the paths leading to R, to find states where Φ_1 is not F. We add these states to R. When no more states can be added to R, we remove R from the state space to get the set $S_{=0}$.

Example 2. For the qDTMC $\mathcal{M}_1$ in Fig. 1 and a qPCTL state formula $\Phi = Pr_{\geq 0.5}[\neg p\ U\ q]$, we first identify the satisfaction sets for the state formulas $\Phi_1 = \neg p$ and $\Phi_2 = q$:
$Sat_T(\Phi_1) = \{s_0, s_3\}$, $Sat_T(\Phi_2) = \{s_2, s_6\}$,
$Sat_F(\Phi_1) = \{s_2, s_4, s_6\}$, $Sat_F(\Phi_2) = \{s_1, s_4\}$
$Sat_?(\Phi_1) = \{s_1, s_5\}$ and $Sat_?(\Phi_2) = \{s_0, s_3, s_5\}$.

We can now compute the set $R = \{s_0, s_1, s_2, s_3, s_5, s_6\}$. Thus the partition set $S_{=0}$ will be $\{s_4\}$.

We follow a similar procedure to identify the set of states for which the probability $Pr((s, \Phi_1 U \Phi_2) =?)$ will be 1. We start with the states that belong to either $S_{=0}$ or $Sat_T(\Phi_2)$ and then identify the paths leading to these states. The probability $Pr((s, \Phi_1 U \Phi_2) =?)$ for such paths will always be less than 1. We thus exclude these states from the set $S_{=?}$.

Example 3. In continuation of Example 2, the set R is now computed as $R = \{s_4, s_2, s_6\} \cup \{s_0, s_3\} = \{s_0, s_2, s_3, s_4, s_6\}$. Thus the set $S_{=?}$ is computes to $\{s_1, s_5\}$.

The set $S_{=1}$ consists of states that have the true satisfaction probability $Pr((s, \Phi_1 U \Phi_2) = T) = 1$. As before, we first identify the set of states that have true satisfaction probability less than 1, and then remove them from the state space to compute $S_{=1}$.

Example 4. The set R in Algorithm 4 for $\mathcal{M}_1$ is now computed as $R = \{s_4, s_1, s_5\} \cup \{s_0, s_3\} = \{s_0, s_1, s_3, s_4, s_5\}$. Thus the set $S_{=1}$ will be $\{s_2, s_6\}$. Also, the set S_{find} can now be computed as $S_{find} = \{s_0, s_3\}$.

Algorithm 2. Algorithm to compute $S_{=0}$ for until operator

Function: *Compute_Until_*$S_{=0}$
$R \leftarrow Sat_T(\Phi_2) \cup Sat_?(\Phi_2)$
while true **do**
  $R' \leftarrow R \cup \{s \in S \setminus Sat_F(\Phi_1) \mid \exists s' \in R, \mathbb{P}(s, s') > 0\}$
  if $R' = R$ **then**
    break
  else
    $R \leftarrow R'$
  end if
end while
$S_{=0} \leftarrow S \setminus R$ **return** $S_{=0}$

Algorithm 3. Algorithm to compute $S_{=?}$ for until operator

Function: *Compute_Until_*$S_{=?}$
$R \leftarrow S_{=0} \cup Sat_T(\Phi_2)$
while true **do**
  $R' \leftarrow R \cup \{s \in Sat_T(\Phi_1) \setminus Sat_T(\Phi_2) \mid \exists s' \in R, \mathbb{P}(s, s') > 0\}$
  if $R' = R$ **then**
    break
  else
    $R \leftarrow R'$
  end if
end while
$S_{=?} \leftarrow S \setminus R$ **return** $S_{=?}$

Algorithm 4. Algorithm to compute $S_{=1}$ for until operator

Function: *Compute_Until_S*$_{=1}$
$R \leftarrow S_{=0} \cup S_{=?}$
while true **do**
  $R' \leftarrow R \cup \{s \in Sat_T(\Phi_1) \setminus Sat_T(\Phi_2) | \exists s' \in R, \mathbb{P}(s, s') > 0\}$
  if $R' = R$ **then**
    break
  else
    $R \leftarrow R'$
  end if
end while
$S_{=1} \leftarrow S \setminus R$

Example 5. Now for $\mathcal{M}_1$ and the qPCTL formula $\Phi = Pr_{\geq 0.5}[\neg p \; U \; q]$, we can compute the satisfaction sets using the true and false satisfaction probabilities: $Sat_T(\Phi) = \{s_2, s_3, s_6\}$, $Sat_F(\Phi) = \{s_4\}$ and $Sat_?(\Phi) = \{s_0, s_1, s_5\}$. Thus, for the initial state s_0, Φ will be evaluated to ?.

3. **Bounded Until operator** – $[\Phi_1 U^{\leq k} \Phi_2]$: The satisfaction probabilities for $[\Phi_1 U^{\leq k} \Phi_2]$ can be directly computed by evaluating k transitions of the qDTMC. Depending on the truth values of Φ_1 and Φ_2 in a state, the state space S is partitioned into sets $S_{=0}$, $S_{=1}$, $S_{=?}$ and S_{find} in the 1MC algorithm. This partition is illustrated in Fig. 2.

$$\begin{aligned} S_{=0} &= Sat_F(\Phi_2) \cap Sat_F(\Phi_1), \\ S_{=?} &= [\ (Sat_?(\Phi_1) \setminus Sat_T(\Phi_2)) \ \cup \ (Sat_?(\Phi_2) \setminus Sat_T(\Phi_1))\ ], \\ S_{=1} &= Sat_T(\Phi_2), \\ S_{find} &= S \setminus [S_{=0} \cup S_{=1} \cup S_{find}] \ = \ Sat_T(\Phi_1) \setminus Sat_T(\Phi_2) \end{aligned}$$

A state $s \in S_{=1}$ if the truth value of Φ_2 is T in s. Now irrespective of the truth value of Φ_1, the path formula $\Phi_1 U^{\leq k} \Phi_2$ will be evaluated as T for all paths starting from this state s, because Φ_2 is T in the initial state of the path. Thus, the true (false) satisfaction probability for states in $S_{=1}$ will be 1 (0).

A state belonging to set $S_{=0}$ will have truth values of both Φ_1 and Φ_2 as F. All paths starting from such a state $s \in S_{=0}$ will have both Φ_1 and Φ_2 false in the initial state itself and will evaluate $\Phi_1 U^{\leq k} \Phi_2$ to F. Thus, the true (false) satisfaction probabilities for states in $S_{=0}$ will be 0 (1).

A state s belongs to the set $S_{=?}$ if the truth value of at least one of Φ_1 or Φ_2 is ? in s and the other is not T. No path starting from such a state $s \in S_{=?}$ can evaluate $\Phi_1 U^{\leq k} \Phi_2$ to either T or F–the truth or falsehood of $\Phi_1 U^{\leq k} \Phi_2$ for a path starting in s is cannot be determined if (i) Φ_1 is ? in s and Φ_2 is not T or (ii) Φ_1 is not T and Φ_2 is ?. For such states, both true and false satisfaction probabilities are 0.

Now, the remaining states in the state space will have Φ_1 as T, but Φ_2 is either F or ?. Since Φ_2 is not T in starting state s of the path, we need to find the value of Φ_1 and Φ_2 in the subsequent states. Thus, these states form the set S_{find}.

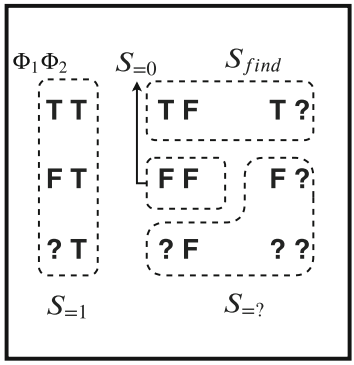


Fig. 2. A table illustrating possible combinations of truth values for Φ_1 and Φ_2 at a state s and corresponding partition set

Example 6. Given a qDTMC $\mathcal{M}_1$ and qPCTL formula $\Phi = Pr_{\geq 0.5}[\neg p\ U^{\leq 3}\ q]$, we can compute the partition sets of the state space S as: $S_{=0} = \{s_4\}$, $S_{=?} = \{s_1, s_5\}$, $S_{=1} = \{s_2, s_6\}$ and $S_{find} = \{s_0, s_3\}$.

We now calculate the satisfaction probabilities of the path formula $[\Phi_1 U^{\leq k}\ \Phi_2]$ for the paths starting at a state s using the 1MC algorithm. We denote the true satisfaction probability for state s as $Pr((s, \Phi_1 U^{\leq k}\ \Phi_2) = T)$ or $x_s^{(T),k}$. We know that the true satisfaction probability for states in $S_{=1}$ is 1. Also for states in $S_{=0}$ and $S_{=?}$, the true satisfaction probability is 0.

To evaluate the path formula $\Phi_1 U^{\leq k} \Phi_2$, a path with initial state $s \in S_{find}$ is traversed until either a state in one of $S_{=0}$, $S_{=1}$ or $S_{=?}$, or the bound k is reached. If a state $s \in S_{find}$ when bound k is reached, then the probability of $[\Phi_1 U^{\leq k}\ \Phi_2]$ being evaluated to T is 0. The 1MC algorithm thus computes the true satisfaction probability as follows.

$$x_s^{(T),k} = \begin{cases} 1 & \text{if} \quad s \in S_{=1} \\ 0 & \text{if} \quad s \in S_{=0} \\ 0 & \text{if} \quad s \in S_{=?} \\ 0 & \text{if } s \in S_{find} \wedge k = 0 \\ \sum\limits_{t \in S} \mathbb{P}(s,t).x_t^{(T),k-1} & \text{if } s \in S_{find} \wedge k > 0 \end{cases} \tag{4}$$

Similarly, we now calculate the false satisfaction probability and denote it as $Pr((s, \Phi_1 U^{\leq k}\ \Phi_2) = F)$ or $x_s^{(F),k}$. Recall that the path formula $[\Phi_1 U^{\leq k}\ \Phi_2]$ evaluates to F if either the formula Φ_2 is F at all states of k-length path, or if at some state Φ_2 evaluates to T but at some preceding state, Φ_1 evaluated to F.

We know that the false satisfaction probability for states in $S_{=0}$ is 1 and is 0 for states in $S_{=1}$ and $S_{=?}$. We also know that the states in set S_{find} will have Φ_1 as T and Φ_2 is either F or ?. Now no path starting from a state that has Φ_1 as T and Φ_2 as ? ($S_{find} \cap Sat_?(\Phi_2)$) will evaluate $\Phi_1 U^{\leq k} \Phi_2$ as F. Thus, false satisfaction probability for such states will be 0.

We now traverse paths with initial state $s \in S_{find} \cap Sat_F(\Phi_2)$ until either a state in one of $S_{=0}$, $S_{=1}$, $S_{=?}$ or $S_{find} \cap Sat_?(\Phi_2)$, or the bound k is reached. If a state $s \in S_{find} \cap Sat_F(\Phi_2)$ when bound k is reached, then the probability of $[\Phi_1 U^{\leq k}\ \Phi_2]$ being evaluated to F is 1. This correlates to Φ_2 evaluating to F at all states in the path, and thus path formula $[\Phi_1 U^{\leq k}\ \Phi_2]$ being evaluated to F.

$$x_s^{(F),k} = \begin{cases} 0 & \text{if} \quad s \in S_{=1} \\ 1 & \text{if} \quad s \in S_{=0} \\ 0 & \text{if} \quad s \in S_{=?} \\ 0 & \text{if} \quad s \in (S_{find} \cap Sat_?(\Phi_2)) \\ 1 & \text{if } s \in (S_{find} \cap Sat_F(\Phi_2)) \wedge k = 0 \\ \sum\limits_{t \in S} \mathbb{P}(s,t).x_t^{(F),k} & \text{if } s \in (S_{find} \cap Sat_F(\Phi_2)) \wedge k > 0 \end{cases} \tag{5}$$

$$x_s^{(?),k} = 1 - [x_s^{(T),k} + x_s^{(F),k}] \tag{6}$$

Example 7. For the given qDTMC $\mathcal{M}_1$ and qPCTL formula $\Phi = Pr_{\geq 0.5}[\neg p\ U^{\leq 3}\ q]$, the true satisfaction probability for state s_0, $Pr((s_0, \neg p\ U^{\leq 3}\ q) = T)$ is 0.475. Also, the false satisfaction probability $Pr((s_0, \neg p\ U^{\leq 3}\ q) = F)$ is 0. Thus, the initial state $s_0 \in Sat_?(\Phi)$ and the formula Φ is evaluated to ? at state s_0.

From the above arguments, we conclude that:

Lemma 2. *For the path formulas $X\Phi$, $\Phi_1 U^{\leq k} \Phi_2$ and $\Phi_1 U \Phi_2$, Algorithm 1 is correct for the corresponding probabilistic state formulas:*

- *1MC($s, Pr_{\bowtie\theta}[X\Phi]$)=T (alt., F or ?) iff ($s, Pr_{\bowtie\theta}[X\Phi]$)=T (resp., F or ?)*
- *1MC($s, Pr_{\bowtie\theta}[\Phi_1 U \Phi_2]$)=T (alt., F or ?) iff ($s, Pr_{\bowtie\theta}[\Phi_1 U \Phi_2]$)=T (resp., F or ?)*
- *1MC($s, Pr_{\bowtie\theta}[\Phi_1 U^{\leq k} \Phi_2]$)=T (alt., F or ?) iff ($s, Pr_{\bowtie\theta}[\Phi_1 U^{\leq k} \Phi_2]$)=T (resp., F or ?)*

From lemmas 1 and 2, we have:

Theorem 1. *1MC(s, Φ)=T (alt., F or ?) iff (s, Φ) =T (resp., F or ?)*

Complexity of qPCTL Model Checking: Unlike for the standard PCTL model checking algorithm [4], the 1MC algorithm for qPCTL model checking computes an additional partition set $S_{=?}$ before solving the system of linear equations for the states in the set S_{find}. The computation of set $S_{=?}$ is done in $\Theta(|S|)$ time. Other than that, the asymptotic time complexity is polynomial with respect to the size of the model $\mathcal{M}$ and linear in terms of the size of the query Φ. Thus, the time complexity of the 1MC qPCTL model checking algorithm is the same as that of the PCTL algorithm and the 2MC algorithm for

qPCTL–$\mathcal{O}(poly(size(M)).n_{max}.|\Phi|)$, where n_{max} is the maximum step bound for bounded *until* and 1 if the formula does not contain a bounded *until*. However, in many cases, a drop in the number of runs results in a significant performance improvement over the 2MC algorithm. This is more pronounced for larger models. We validate this with extensive experimentation in the next section.

4 Implementation and Results

We have implemented the 1MC algorithm as a Java tool. The tool takes as an input the qDTMC model and the qPCTL query. The tool supports both qualitative and quantitative qPCTL queries. A qualitative query only checks if the required probability threshold is met, and results in T, F or ?. A quantitative query on the other hand computes the exact probability of the query being T, F and ?.

We will begin by mentioning that the 1MC approach discussed in this paper yields matching results for the case studies reported in [1] and [2]– (i) a model representing an incomplete program code, and (ii) a network model that has incomplete information about its nodes respectively.

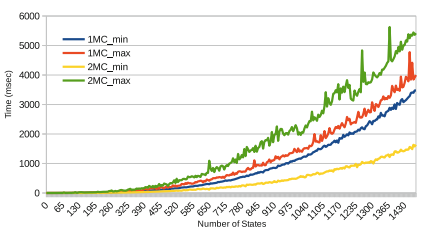
We now compare the performance of the proposed 1MC algorithm with the 2MC algorithm in [2] for model checking different qDTMCs. For the sake of fairness, we also implemented the standard PCTL model checker from scratch. The 2MC algorithm calls this bare-bones model checker as a subroutine.

For a fixed size of state space, we randomly generate 50 qDTMCs with different transition probability matrices and labeling functions. We study the variation in time taken to verify models with different structures for different qPCTL queries, and record the minimum and maximum time taken.

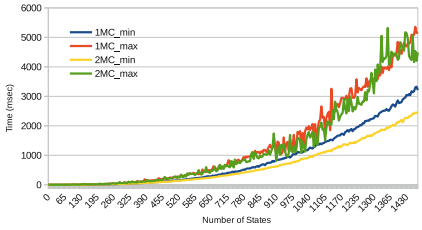
We repeat this for different sizes of state spaces starting from qDTMCs with 5 states, and up to 1500 states. Thus the algorithms were compared in terms of the time taken to verify the models of varying sizes.

Figure 3 plots the minimum and maximum times taken by the 1MC and 2MC algorithms to verify the properties $\Phi_1 = Pr_{\geq 0.8}[p_0\ U\ p_1]$, $\Phi_2 = Pr_{\geq 0.7}[\ p_0\ U\ Pr_{\geq 0.6}[X\ p_1]]$ and $\Phi_3 = Pr_{\geq 0.7}[\ p_0\ U\ Pr_{\geq 0.6}[\ p_1\ U\ p_2]]$ where p_0, p_1 and p_2 are atomic propositions in the incomplete models.

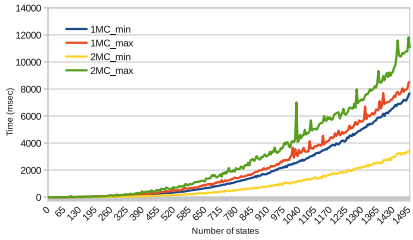
It can be seen from the results that the minimum time curve for 1MC algorithm is higher than that for the 2MC algorithm. This is due to the fact that the 2MC algorithm need not compute both its steps in all cases. For instance, if the probability of a property being true meets the required threshold in the first step itself, the model checker need not calculate the probability of a property being false. However, the 1MC algorithm calculates both true as well as false probability in single step, which increases the computation overhead. In many cases, however, both steps of 2MC algorithm are needed; thus making it very expensive for models with large state space. The proposed 1MC algorithm generates results faster for such models and has a lower maximum time curve than that for 2MC algorithm.



(a) Time taken to verify property Φ_1



(b) Time taken to verify property Φ_2



(c) Time taken to verify property Φ_3

Fig. 3. Minimum and maximum time taken by the algorithms to verify properties Φ_1, Φ_2 and Φ_3 for models of varying sizes and topology.

5 Conclusion and Future Work

We believe that model checking for incomplete models will be of immense practical use. While it is possible to design algorithms that use existing techniques designed for binary logic, it is useful to have algorithms designed exclusively for three-valued logics. We discussed an algorithm and its application for model checking PCTL queries against incomplete DTMCs that accommodate a three-valued logic.

Future efforts in this direction would be (i) applying these algorithms for interim analysis of incomplete models in practice and (ii) designing similar algorithms and tools for other models and systems of logic.

References

1. Arora, S., Legay, A., Richmond, T., Traonouez, L.-M.: Statistical model checking of incomplete stochastic systems. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11245, pp. 354–371. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03421-4_23
2. Arora, S., Rao, M.V.P.: Probabilistic model checking of incomplete models. CoRR (2017). http://arxiv.org/abs/1706.05082
3. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time Markov chains. IEEE Trans. Softw. Eng. **29**(6), 524–541 (2003). https://doi.org/10.1109/TSE.2003.1205180

4. Baier, C., Katoen, J.P.: Principles of Model Checking (Representation and Mind Series). The MIT Press (2008)
5. Benedikt, M., Lenhardt, R., Worrell, J.: LTL model checking of interval Markov chains. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013. LNCS, vol. 7795, pp. 32–46. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36742-7_3
6. Bruns, G., Godefroid, P.: Model checking partial state spaces with 3-valued temporal logics. In: Halbwachs, N., Peled, D. (eds.) CAV 1999. LNCS, vol. 1633, pp. 274–287. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48683-6_25
7. Bruns, G., Godefroid, P.: Generalized model checking: reasoning about partial state spaces. In: Palamidessi, C. (ed.) CONCUR 2000. LNCS, vol. 1877, pp. 168–182. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44618-4_14
8. Caillaud, B., Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wąsowski, A.: Constraint Markov chains. Theor. Comput. Sci. **412**(34), 4373–4404 (2011). https://doi.org/10.1016/j.tcs.2011.05.010
9. Chakraborty, S., Katoen, J.-P.: Model checking of open interval Markov chains. In: Gribaudo, M., Manini, D., Remke, A. (eds.) ASMTA 2015. LNCS, vol. 9081, pp. 30–42. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18579-8_3
10. Chechik, M.: On interpreting results of model-checking with abstraction. University of Toronto, Technical report (2000)
11. Chechik, M., Easterbrook, S., Petrovykh, V.: Model-checking over multi-valued logics. In: Oliveira, J.N., Zave, P. (eds.) FME 2001. LNCS, vol. 2021, pp. 72–98. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45251-6_5
12. Courcoubetis, C., Yannakakis, M.: Verifying temporal properties of finite-state probabilistic programs. In: 29th Annual Symposium on Foundations of Computer Science, pp. 338–345. IEEE (1988)
13. Delahaye, B., et al.: Abstract probabilistic automata. Inf. Comput. **232**, 66–116 (2013). https://doi.org/10.1016/j.ic.2013.10.002
14. Fecher, H., Leucker, M., Wolf, V.: *Don't Know* in probabilistic systems. In: Valmari, A. (ed.) SPIN 2006. LNCS, vol. 3925, pp. 71–88. Springer, Heidelberg (2006). https://doi.org/10.1007/11691617_5
15. Godefroid, P., Piterman, N.: LTL generalized model checking revisited. Int. J. Softw. Tools Technol. Transfer **13**(6), 571–584 (2011)
16. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Asp. Comput. **6**(5), 512–535 (1994). https://doi.org/10.1007/BF01211866
17. Huth, M., Piterman, N., Wagner, D.: Three-valued abstractions of Markov chains: completeness for a sizeable fragment of PCTL. In: Kutyłowski, M., Charatonik, W., Gębala, M. (eds.) FCT 2009. LNCS, vol. 5699, pp. 205–216. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03409-1_19
18. Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: Proceedings 1991 Sixth Annual IEEE Symposium on Logic in Computer Science, pp. 266–277. IEEE (1991)
19. Klink, D.: Three-valued abstraction for stochastic systems. Verlag Dr, Hut (2010)
20. Kozine, I.O., Utkin, L.V.: Interval-valued finite Markov chains. Reliable Comput. **8**(2), 97–113 (2002)
21. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic model checking. In: Bernardo, M., Hillston, J. (eds.) SFM 2007. LNCS, vol. 4486, pp. 220–270. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72522-0_6
22. Sen, K., Viswanathan, M., Agha, G.: Statistical model checking of black-box probabilistic systems. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 202–215. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27813-9_16

23. Sen, K., Viswanathan, M., Agha, G.: On statistical model checking of stochastic systems. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 266–280. Springer, Heidelberg (2005). https://doi.org/10.1007/11513988_26
24. Sen, K., Viswanathan, M., Agha, G.: Model-checking Markov chains in the presence of uncertainties. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006. LNCS, vol. 3920, pp. 394–410. Springer, Heidelberg (2006). https://doi.org/10.1007/11691372_26
25. Younes, H.L.S., Simmons, R.G.: Probabilistic verification of discrete event systems using acceptance sampling. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 223–235. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45657-0_17