# UNIVERSITYOF <br> BIRMINGHAM <br> University of Birmingham Research at Birmingham 

# Finite-Horizon Bisimulation Minimisation for Probabilistic Systems 

Kamaleson, Nishanthan; Parker, David; Rowe, Jonathan

DOI:
10.1007/978-3-319-32582-8_10

License:
None: All rights reserved

## Document Version

Peer reviewed version
Citation for published version (Harvard):
Kamaleson, N, Parker, D \& Rowe, J 2016, Finite-Horizon Bisimulation Minimisation for Probabilistic Systems. in Model Checking Software: 23rd International Symposium, SPIN 2016, Co-located with ETAPS 2016, Eindhoven, The Netherlands, April 7-8, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9641, Springer, pp. 147-164, 23rd International Symposium, SPIN 2016, Co-located with ETAPS 2016, Eindhoven, Netherlands, 7/04/16. https://doi.org/10.1007/978-3-319-32582-8_10

Link to publication on Research at Birmingham portal

## Publisher Rights Statement:

The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-32582-8_10
Checked for eligibility: 04/03/2016

## General rights

Unless a licence is specified above, all rights (including copyright and moral rights) in this document are retained by the authors and/or the copyright holders. The express permission of the copyright holder must be obtained for any use of this material other than for purposes permitted by law.

- Users may freely distribute the URL that is used to identify this publication.
- Users may download and/or print one copy of the publication from the University of Birmingham research portal for the purpose of private study or non-commercial research.
$\bullet$ User may use extracts from the document in line with the concept of 'fair dealing' under the Copyright, Designs and Patents Act 1988 (?)
-Users may not further distribute the material nor use it for the purposes of commercial gain.
Where a licence is displayed above, please note the terms and conditions of the licence govern your use of this document.
When citing, please reference the published version.


## Take down policy

While the University of Birmingham exercises care and attention in making items available there are rare occasions when an item has been uploaded in error or has been deemed to be commercially or otherwise sensitive.
If you believe that this is the case for this document, please contact UBIRA@lists.bham.ac.uk providing details and we will remove access to the work immediately and investigate.

# Finite-Horizon Bisimulation Minimisation for Probabilistic Systems 

Nishanthan Kamaleson, David Parker, and Jonathan E. Rowe<br>School of Computer Science, University of Birmingham


#### Abstract

We present model reduction techniques to improve the efficiency and scalability of verifying probabilistic systems over a finite time horizon. We propose a finite-horizon variant of probabilistic bisimulation for discrete-time Markov chains, which preserves a bounded fragment of the temporal logic PCTL. In addition to a standard partitionrefinement based minimisation algorithm, we present on-the-fly finitehorizon minimisation techniques, which are based on a backwards traversal of the Markov chain, directly from a high-level model description. We investigate both symbolic and explicit-state implementations, using SMT solvers and hash functions, respectively, and implement them in the PRISM model checker. We show that finite-horizon reduction can provide significant reductions in model size, in some cases outperforming PRISM's existing efficient implementations of probabilistic verification.


## 1 Introduction

Probabilistic verification is an automated technique for the formal analysis of quantitative properties of systems that exhibit stochastic behaviour. A probabilistic model, such as a Markov chain or a Markov decision process, is systematically constructed and then analysed against properties expressed in a formal specification language such as temporal logic. Mature tools for probabilistic verification such as PRISM [15] and MRMC [13] have been developed, and the techniques have been applied to a wide range of application domains, from biological reaction networks [11] to car airbag controllers [1].

A constant challenge in this area is the issue of scalability: probabilistic models, which are explored and constructed in an exhaustive fashion, are typically huge for real-life systems, which can limit the practical applicability of the techniques. A variety of approaches have been proposed to reduce the size of these models. One that is widely used is probabilistic bisimulation [18], an equivalence relation over the states of a probabilistic model which can be used to construct a smaller quotient model that is equivalent to the original one (in the sense that it preserves key properties of interest to be verified).

Typically, it preserves both infinite-horizon (long-run) properties, e.g., "the probability of eventually reaching an error state", finite-horizon (transient, or time-bounded) properties, e.g. "the probability of an error occurring within $k$ time-steps", and, more generally, any property expressible in an appropriate
temporal logic such as PCTL [10]. It has been shown that, in contrast to nonprobabilistic verification, the effort required to perform bisimulation minimisation can pay off in terms of the total time required for verification [12].

In this paper, we consider model reduction techniques for finite-horizon properties of Markov chains. We propose a finite-horizon variant of probabilistic bisimulation, which preserves stepwise behaviour over a finite number of steps, rather than indefinitely, as in standard probabilistic bisimulation. This permits a more aggressive model reduction, but still preserves satisfaction of PCTL formulae of bounded depth (i.e., whose interpretation requires only a bounded exploration of the model). Time-bounded properties are commonly used in probabilistic verification, e.g., for efficiency ("the probability of task completion within $k$ steps") or for reliabilty ("the probability of an error occurring within time $k$ ").

We formalise finite-horizon probabilistic bisimulation, define the subset of PCTL that it preserves and then give a partition-refinement based algorithm for computing the coarsest possible finite-horizon bisimulation relation, along with a corresponding quotient model. The basic algorithm is limited by the fact it requires the full Markov chain to be constructed before it is minimised, which can be a bottleneck. So, we then develop on-the-fly approaches, which construct the quotient model directly from a high-level model description of the Markov chain, based on a backwards traversal of its state space. We propose two versions: one symbolic, based on SMT solvers, and one explicit-state.

We implemented all algorithms in PRISM and evaluated them on a range of examples. First, we apply the partition-refinement based approach to some standard benchmarks to investigate the size of the reduction that can be obtained in a finite-horizon setting. Then, we apply the on-the-fly approach to a class of problems to which it is particularly well suited: models with a large number of possible initial configurations, on which we ask questions such as "from which initial states does the probability of an error occurring within 10 seconds exceed $0.01 ?$ ?. We show that on-the-fly finite-horizon bisimulation can indeed provide significant gains in both verification time and scalability, demonstrated in each case by outperforming the existing efficient implementations in PRISM.

Related Work. For the standard notion of probabilistic bisimulation on Markov chains [18], various decision procedure and minimisation algorithms have been developed. Derisavi et al. [9] proposed an algorithm with optimal complexity, assuming the use of splay trees and, more recently, a simpler solution was put forward in [20]. Signature-based approaches, which our first, partition-refinement algorithm adapts, have been studied in, for example, [9,22]. Also relevant is the SMT-based bisimulation minimisation technique of [6] which, like our on-the-fly algorithm, avoids construction of the full model when minimising. Our SMTbased algorithm has an additional benefit in that it works on model descriptions with state-dependent probabilities. Other probabilistic verification methods have been developed based on backwards traversal of a model, for example for probabilistic timed automata [16], but this is for a different class of models and does not perform minimisation. Della Penna et al. considered finite-horizon verification of Markov chains [7], but using disk-based methods, not model reduction.

## 2 Preliminaries

We start with some background on probabilistic verification of Markov chains.

### 2.1 Discrete-time Markov Chains

A discrete-time Markov chain (DTMC) can be thought of as a state transition system where transitions between states are annotated with probabilities.

Definition 1 (DTMC). A DTMC is a tuple $\mathcal{D}=\left(\mathcal{S}, \mathcal{S}_{\text {init }}, \mathbf{P}, \mathcal{A P}, \mathcal{L}\right)$, where:
$-\mathcal{S}$ is a finite set of states and $\mathcal{S}_{\text {init }} \subseteq \mathcal{S}$ is a set of initial states;
$-\mathbf{P}: \mathcal{S} \times \mathcal{S} \rightarrow[0,1]$ is a transition probability matrix, where, for all states $s \in \mathcal{S}$, we have $\sum_{s^{\prime} \in \mathcal{S}} \mathbf{P}\left(s, s^{\prime}\right)=1$;
$-\mathcal{A P}$ is a set of atomic propositions and $\mathcal{L}: \mathcal{S} \rightarrow 2^{\mathcal{A P}}$ is a labelling function giving the set of propositions from $\mathcal{A P}$ that are true in each state.

For each pair $s, s^{\prime}$ of states, $\mathbf{P}\left(s, s^{\prime}\right)$ represents the probability of going from $s$ to $s^{\prime}$. If $\mathbf{P}\left(s, s^{\prime}\right)>0$, then $s$ is a predecessor of $s^{\prime}$ and $s^{\prime}$ is a successor of $s$. For a state $s$ and set $C \subseteq \mathcal{S}$, we will often use the notation $\mathbf{P}(s, C):=\sum_{s^{\prime} \in C} \mathbf{P}\left(s, s^{\prime}\right)$.

A path $\sigma$ of a DTMC $\mathcal{D}$ is a finite or infinite sequence of states $\sigma=s_{0} s_{1} s_{2} \ldots$ such that $\forall i \geq 0, s_{i} \in \mathcal{S}$ and $\mathbf{P}\left(s_{i}, s_{i+1}\right)>0$. The $i^{t h}$ state of the path $\sigma$ is denoted by $\sigma[\bar{i}]$. We let $\operatorname{Path}^{\mathcal{D}}(s)$ denote the set of infinite paths of $\mathcal{D}$ that begin in a state $s$. To reason formally about the behaviour of a DTMC, we define a probability measure $\operatorname{Pr}_{s}$ over the set of infinite paths $\operatorname{Path}^{\mathcal{D}}(s)$ [14]. We usually consider the behaviour from some initial state $s \in \mathcal{S}_{\text {init }}$ of $\mathcal{D}$.

### 2.2 Probabilistic Computation Tree Logic

Properties of probabilistic models can be expressed using Probabilistic Computation Tree Logic (PCTL) [10] which extends Computation Tree Logic (CTL) with time and probabilities. In PCTL, state formulae $\Phi$ are interpreted over states of a DTMC and path formulae $\phi$ are interpreted over paths.

Definition 2 (PCTL). The syntax of PCTL is as follows:

$$
\begin{aligned}
& \Phi::=\text { true }|a| \neg \Phi|\Phi \wedge \Phi| \mathrm{P}_{\bowtie p}[\phi] \\
& \phi::=\Phi_{1} \mathrm{U}^{\leq k} \Phi_{2}
\end{aligned}
$$

where $a$ is an atomic proposition, $\bowtie \in\{<, \leq, \geq,>\}, p \in[0,1]$ and $k \in \mathbb{N} \cup\{\infty\}$.
The main operator in PCTL, in addition to those that are standard from propositional logic, is the probabilistic operator $\mathrm{P}_{\bowtie p}[\phi]$, which means that the probability measure of paths that satisfy $\phi$ is within the bound $\bowtie p$. For path formulae $\phi$, we allow the (bounded) until operator $\Phi_{1} U^{\leq k} \Phi_{2}$. If $\Phi_{2}$ becomes true within $k$ time steps and $\Phi_{1}$ is true until that point, then $\Phi_{1} U^{\leq k} \Phi_{2}$ is true. In the case where $k$ equals $\infty$, the bounded until operator becomes the unbounded until operator and is denoted by $U$. For simplicity of presentation, in this paper, we omit the next ( $\mathrm{X} \Phi$ ) operator, but this could easily be added.

Definition 3 (PCTL semantics). Let $\mathcal{D}=\left(\mathcal{S}, \mathcal{S}_{\text {init }}, \mathbf{P}, \mathcal{A} \mathcal{P}, \mathcal{L}\right)$ be a $D T M C$. The satisfaction relation $\vDash_{\mathcal{D}}$ for $P C T L$ formulae on $\mathcal{D}$ is defined by:

```
\(-s \vDash_{\mathcal{D}}\) true \(\quad \forall s \in \mathcal{S}\)
\(-s \vDash_{\mathcal{D}} a \quad\) iff \(a \in \mathcal{L}(s)\)
\(-s \vDash_{\mathcal{D}} \neg \Phi \quad\) iff \(s \nvdash_{\mathcal{D}} \Phi\)
\(-s \vDash_{\mathcal{D}} \Phi_{1} \wedge \Phi_{2} \quad\) iff \(s \vDash_{\mathcal{D}} \Phi_{1}\) and \(s \vDash_{\mathcal{D}} \Phi_{2}\)
\(-s \vDash_{\mathcal{D}} \mathrm{P}_{\bowtie p}[\phi] \quad\) iff \(\operatorname{Pr}_{s}\left\{\sigma \in \operatorname{Path}^{\mathcal{D}}(s) \mid \sigma \vDash_{\mathcal{D}} \phi\right\} \bowtie p\)
\(-\sigma \vDash_{\mathcal{D}} \Phi_{1} \mathrm{U} \leq k \Phi_{2} \quad\) iff \(\exists i \in \mathbb{N} .\left(i \leq k \wedge \sigma[i] \vDash_{\mathcal{D}} \Phi_{2} \wedge\left(\forall j .0 \leq j<i . \sigma[j] \vDash_{\mathcal{D}} \Phi_{1}\right)\right)\)
```

For example, a PCTL formula such as $\mathrm{P}_{<0.01}\left[\neg\right.$ fail $_{1} \mathrm{U} \leq k$ fail $\left.{ }_{2}\right]$ means that the probability of a failure of type 2 occurring within $k$ time-steps, and before a failure of type 1 does, is less than 0.01 . Common derived operators are $\mathrm{F} \Phi \equiv$ true $U \Phi$, which means that $\Phi$ eventually becomes true, and $\mathrm{F}^{\leq k} \Phi \equiv \operatorname{true} \mathrm{U}^{\leq k} \Phi$, which means that $\Phi$ becomes true within $k$ steps.

### 2.3 Probabilistic Bisimulation

Larsen and Skou [18] defined (strong) probabilistic bisimulation for discrete probabilistic transition systems, which is an equivalence relation used to identify states with identical labellings and (probabilistic) step-wise behaviour.

Definition 4 (Probabilistic bisimulation). Let $\mathcal{D}=\left(\mathcal{S}, \mathcal{S}_{\text {init }}, \mathbf{P}, \mathcal{A P}, \mathcal{L}\right)$ be a DTMC and $\mathcal{R}$ an equivalence relation on $\mathcal{S}$. Then $\mathcal{R}$ is a (strong) probabilistic bisimulation on $\mathcal{D}$ if, for $\left(s_{1}, s_{2}\right) \in \mathcal{R}$ :
(i) $\mathcal{L}\left(s_{1}\right)=\mathcal{L}\left(s_{2}\right)$ and (ii) for all $C \in \mathcal{S} / \mathcal{R}: \mathbf{P}\left(s_{1}, C\right)=\mathbf{P}\left(s_{2}, C\right)$
where $\mathcal{S} / \mathcal{R}$ denotes the set of equivalence classes of set $\mathcal{S}$ by relation $\mathcal{R}$. States $s_{1}, s_{2}$ are bisimilar if there exists a bisimulation on $\mathcal{D}$ containing $\left(s_{1}, s_{2}\right)$.

Two states that are probabilistically bisimilar will satisfy the same properties, including both infinite-horizon (long-run) and finite-horizon (transient) properties. Aziz et al. [3] proved that any property in the temporal logic PCTL is also preserved in this manner. Thanks to these results, the analysis of the original Markov chain, such as probabilistic model checking of PCTL, can be equivalently performed on the quotient Markov chain, in which equivalence classes of bisimilar states are lumped together into a single state.

Usually, we are interested in the coarsest possible probabilistic bisimulation for a DTMC $\mathcal{D}$ (or, in other words, the union of all possible bisimulation relations). We denote the coarsest possible probabilistic bisimulation by $\sim$. The quotient model $\mathcal{D} / \sim$ derived using this relation is defined as follows.

Definition 5 (Quotient DTMC). Given $D T M C \mathcal{D}=\left(\mathcal{S}, \mathcal{S}_{\text {init }}, \mathbf{P}, \mathcal{A P}, \mathcal{L}\right)$, the quotient DTMC is defined as $\mathcal{D} / \sim=\left(\mathcal{S}^{\prime}, \mathcal{S}_{\text {init }}^{\prime}, \mathbf{P}^{\prime}, \mathcal{A} \mathcal{P}, \mathcal{L}^{\prime}\right)$ where:
$-\mathcal{S}^{\prime}=\mathcal{S} / \sim=\left\{[s]_{\sim} \mid s \in \mathcal{S}\right\}$
$-\mathcal{S}_{\text {init }}^{\prime}=\left\{[s]_{\sim} \mid s \in \mathcal{S}_{\text {init }}\right\}$
$-\mathbf{P}^{\prime}\left([s]_{\sim},\left[s^{\prime}\right]_{\sim}\right)=\mathbf{P}\left(s,\left[s^{\prime}\right]_{\sim}\right)$
$-\mathcal{L}^{\prime}\left([s]_{\sim}\right)=\mathcal{L}(s)$
and $[s]_{\sim}$ denotes the unique equivalence class of relation $\sim$ containing $s$.

## 3 Finite-Horizon Bisimulation

We now formalise the notion of finite-horizon bisimulation, a step-bounded variant of standard probabilistic bisimulation for Markov chains [18]. We fix, from this point on, a DTMC $\mathcal{D}=\left(\mathcal{S}, \mathcal{S}_{\text {init }}, \mathbf{P}, \mathcal{A} \mathcal{P}, \mathcal{L}\right)$. Intuitively, a $k$-step finitehorizon bisimulation, for non-negative integer $k$, preserves the stepwise behaviour of $\mathcal{D}$ over a finite horizon of $k$ steps. We use the following inductive definition.

Definition 6 (Finite-horizon bisimulation). $A k$-step finite-horizon bisimulation, for $k \in \mathbb{N}_{\geq 0}$, is an equivalence relation $\mathcal{R}_{k} \subseteq \mathcal{S} \times \mathcal{S}$ such that, for all states $\left(s_{1}, s_{2}\right) \in \mathcal{R}_{k}$, the following two conditions are satisfied:
(i) $\mathcal{L}\left(s_{1}\right)=\mathcal{L}\left(s_{2}\right)$;
(ii) $\mathbf{P}\left(s_{1}, C\right)=\mathbf{P}\left(s_{2}, C\right)$ for each equivalence class $C \in \mathcal{S} / \mathcal{R}_{k-1}$,
where $\mathcal{R}_{k-1}$ is a $(k-1)$-step finite-horizon bisimulation. A 0-step finite-horizon bisimulation is an equivalence relation $\mathcal{R}_{0}$ satisfying only condition (i) above.

Definition 7 (Finite-horizon bisimulation equivalent). We say states $s_{1}, s_{2}$ are ( $k$-step) finite-horizon bisimulation equivalent (bisimilar), denoted $s_{1} \sim_{k} s_{2}$, if there exists a $k$-step finite-horizon bisimulation $\mathcal{R}_{k}$ such that $\left(s_{1}, s_{2}\right) \in \mathcal{R}_{k}$.

Two states $s_{1}$ and $s_{2}$ satisfying $s_{1} \sim_{k} s_{2}$ have the same stepwise behaviour over $k$ steps. The following simple, but useful, properties hold.

Proposition 1. Let $s_{1}, s_{2} \in \mathcal{S}$ be two states. Then:
(a) if $s_{1} \sim_{k} s_{2}$, then $s_{1} \sim_{j} s_{2}$ for any $0 \leq j \leq k$.
(b) if $s_{1} \sim s_{2}$, then $s_{1} \sim_{k} s_{2}$ for any $k \geq 0$.
(c) if $s_{1} \sim_{k} s_{2}$ and $s_{1} \rightarrow s_{1}^{\prime}$, then $s_{1}^{\prime} \sim_{k-1} s_{2}^{\prime}$ for some state $s_{2}^{\prime}$ such that $s_{2} \rightarrow s_{2}^{\prime}$.

From a model checking perspective, if $s_{1} \sim_{k} s_{2}$, then $s_{1}$ and $s_{2}$ satisfy the same PCTL formulae up to a bounded depth $k$. We formalise this as follows.

Definition 8 (Formula depth). The depth of a PCTL formula $\Phi$, denoted $d(\Phi)$, is a value in $\mathbb{N} \cup\{\infty\}$ defined inductively as follows:
$-d($ true $)=d(a)=0$ for atomic proposition $a ;$
$-d(\neg \Phi)=d(\Phi)$;
$-d\left(\Phi_{1} \wedge \Phi_{2}\right)=\max \left(d\left(\Phi_{1}\right), d\left(\Phi_{2}\right)\right) ;$
$-d\left(\mathrm{P}_{\bowtie p}\left[\Phi_{1} \mathrm{U}^{\leq j} \Phi_{2}\right]\right)=j+\max \left(d\left(\Phi_{1}\right)-1, d\left(\Phi_{2}\right)\right)$.
For example, if $a$ and $b$ are atomic propositions, we have $d\left(\mathrm{P}_{\bowtie p}[\right.$ true $\left.\mathrm{U} \leq 5 a]\right)=5$, $d\left(\mathrm{P}_{\bowtie p}\left[\operatorname{true} \mathrm{U}^{\leq 5} a\right] \wedge \mathrm{P}_{\bowtie p}\left[\operatorname{true} \mathrm{U}^{\leq 6} a\right]\right)=6$, and $d\left(\mathrm{P}_{\bowtie p}\left[\operatorname{true} \mathrm{U}^{\leq 5} \mathrm{P}_{\bowtie p}[a \mathrm{U} \leq 3 b]\right]\right)=8$.

If states $s_{1}$ and $s_{2}$ are ( $k$-step) finite-horizon bisimilar, then they satisfy exactly the same PCTL formulae of depth at most $k$.

Theorem 1. Let $s_{1}$ and $s_{2}$ be two states such that $s_{1} \sim_{k} s_{2}$, and $\Phi$ be a PCTL formula with depth $d(\Phi) \leq k$, then $s_{1}=\Phi$ if and only if $s_{2}=\Phi$.

Proof. We prove the result by induction over the structure (see Definition 2) of PCTL formula $\Phi$. Propositional operators are straightforward since $s_{1}$ and $s_{2}$ satisfy the same atomic propositions, by the definition of $\sim_{k}$, and, for $\Phi=\neg \Phi_{1}$ or $\Phi=\Phi_{1} \wedge \Phi_{2}$, the subformulae $\Phi_{1}$ and $\Phi_{2}$ have depth at most $k$ so, by induction, we can assume that $s_{1} \models \Phi_{i} \Leftrightarrow s_{2} \models \Phi_{i}$ for $i \in\{1,2\}$.

The remaining case to consider is $\Phi=\mathrm{P}_{\bowtie p}\left[\Phi_{1} \mathrm{U}^{\leq j} \Phi_{2}\right]$. We know, from Definition 8 , that the depths $d\left(\Phi_{1}\right)$ and $d\left(\Phi_{2}\right)$ of the two subformulae are at most $k-j+1$ and $k-j$. From the semantics of PCTL, we have that, for any state $s$ :

$$
s \models \mathrm{P}_{\bowtie p}\left[\Phi_{1} \mathrm{U}^{\leq j} \Phi_{2}\right] \Leftrightarrow \operatorname{Pr}_{s}\left(\Phi_{1} \mathrm{U}^{\leq j} \Phi_{2}\right) \bowtie p
$$

which means it suffices to show that:

$$
\begin{equation*}
\operatorname{Pr}_{s_{1}}\left(\Phi_{1} \mathrm{U}^{\leq j} \Phi_{2}\right)=\operatorname{Pr}_{s_{2}}\left(\Phi_{1} \mathrm{U}^{\leq j} \Phi_{2}\right) \tag{1}
\end{equation*}
$$

We in fact show this to be true for any states $s_{1}, s_{2}$, values $j \leq k$ and PCTL subformulae $\Phi_{1}, \Phi_{2}$ satisfying $s_{1} \sim_{k} s_{2}$ and $\max \left(d\left(\Phi_{1}\right)-1, d\left(\Phi_{2}\right)\right) \leq k-j$, which we prove inductively over $j$. From the model checking algorithm for PCTL [10], we know that, for any state $s$ :

$$
\operatorname{Pr}_{s}\left(\Phi_{1} \mathrm{U}^{\leq j} \Phi_{2}\right)=\left\{\begin{array}{cl}
1 & \text { if } s \neq \Phi_{2} \\
0 & \text { if } s \neq \neg \Phi_{1} \wedge \neg \Phi_{2} \\
0 & \text { if } s \neq \Phi_{1} \wedge \neg \Phi_{2} \text { and } j=0 \\
\sum_{s^{\prime} \in \mathcal{S}} \mathbf{P}\left(s, s^{\prime}\right) \operatorname{Pr}_{s^{\prime}}\left(\Phi_{1} \mathrm{U} \leq j-1\right. \\
\left.\Phi_{2}\right) & \text { if } s
\end{array}=\Phi_{1} \wedge \neg \Phi_{2} \text { and } j>0 . ~ \$\right.
$$

For the base case $j=0$, only the first three cases of the definition above can apply, and we know that $s_{1} \models \Phi_{i} \Leftrightarrow s_{2} \models \Phi_{i}$ for $i \in\{1,2\}$, so we have that $\operatorname{Pr}_{s_{1}}\left(\Phi_{1} \mathrm{U} \leq{ }^{\leq 0} \Phi_{2}\right)=\operatorname{Pr}_{s_{2}}\left(\Phi_{1} \mathrm{U} \mathrm{U}^{\leq 0} \Phi_{2}\right)$. For the inductive case, where $j>0$, we can assume that $\operatorname{Pr}_{s_{1}}\left(\Phi_{1} \mathrm{U} \leq j-1 \Phi_{2}\right)=\operatorname{Pr}_{s_{2}}\left(\Phi_{1} \mathrm{U} \leq j-1 \Phi_{2}\right)$, as long as $s_{1} \sim_{j-1} s_{2}$. Considering again the possible cases in the above definition, the first two follow as for $j=0$ and the third cannot apply since $j>0$. For the fourth case, since $j>0$, we know there exists a $(j-1)$-step finite-horizon bisimulation $\mathcal{R}_{j-1}$. Let us further assume an (arbitrary) function rep: $\mathcal{S} / \mathcal{R}_{j-1} \rightarrow \mathcal{S}$, which selects a unique representative from each equivalence class of $\mathcal{R}_{j-1}$. We have:

$$
\begin{aligned}
& \operatorname{Pr}_{s_{1}}\left(\Phi_{1} \mathrm{U}^{\leq j} \Phi_{2}\right) \\
& =\sum_{s^{\prime} \in \mathcal{S}} \mathbf{P}\left(s_{1}, s^{\prime}\right) \operatorname{Pr}_{s^{\prime}}\left(\Phi_{1} \mathrm{U} \leq j-1 \Phi_{2}\right) \quad \text { by definition } \\
& =\sum_{C \in \mathcal{S} / \sim_{j-1}}^{s^{\prime} \in \mathcal{S}} \sum_{s^{\prime} \in C} \mathbf{P}\left(s_{1}, s^{\prime}\right) P r_{s^{\prime}}\left(\Phi_{1} \mathrm{U} \leq j-1 \Phi_{2}\right) \quad \text { since } \sim_{j-1} \text { partitions } \mathcal{S} \\
& =\sum_{C \in \mathcal{S} / \sim_{j-1}} \operatorname{Pr}_{r e p(C)}\left(\Phi_{1} \mathrm{U} \leq j-1 \Phi_{2}\right) \sum_{s^{\prime} \in C} \mathbf{P}\left(s_{1}, s^{\prime}\right) \quad \text { by induction on } j \\
& =\sum_{C \in \mathcal{S} / \sim_{j-1}} \operatorname{Pr}_{r e p(C)}\left(\Phi_{1} \mathrm{U}^{\leq j-1} \Phi_{2}\right) \mathbf{P}\left(s_{1}, C\right) \\
& =\sum_{C \in \mathcal{S} / \sim_{j-1}} \operatorname{Pr}_{r e p(C)}\left(\Phi_{1} \mathbb{U} \leq j-1 \Phi_{2}\right) \mathbf{P}\left(s_{2}, C\right) \quad \text { since } s_{1} \sim_{j} s_{2} \\
& =\sum_{C \in \mathcal{S} / \sim_{j-1}} \operatorname{Pr}_{r e p(C)}\left(\Phi_{1} \mathrm{U} \leq j-1 \Phi_{2}\right) \sum_{s^{\prime} \in C} \mathbf{P}\left(s_{2}, s^{\prime}\right) \\
& =\sum_{C \in \mathcal{S} / \sim_{j-1}} \sum_{s^{\prime} \in C} \mathbf{P}\left(s_{2}, s^{\prime}\right) \operatorname{Pr}_{s^{\prime}}\left(\Phi_{1} \mathrm{U} \leq j-1 \Phi_{2}\right) \quad \text { since } s^{\prime} \sim_{j-1} \operatorname{rep}(C) \\
& =\sum_{s^{\prime} \in \mathcal{S}} \mathbf{P}\left(s_{2}, s^{\prime}\right) P r_{s^{\prime}}\left(\Phi_{1} \mathbb{U} \leq j-1 \Phi_{2}\right) \quad \text { since } \sim_{j-1} \text { partitions } \mathcal{S} \\
& =\operatorname{Pr}_{s_{2}}\left(\Phi_{1} \mathrm{U}^{\leq j} \Phi_{2}\right) \quad \text { by definition }
\end{aligned}
$$

which proves (1), as required, and concludes the proof.


Fig. 1. (a) Example DTMC; (b-c) Finite-horizon quotient DTMCs for $k=0,1$.
In similar fashion to the standard (non-finite-horizon) case, we are typically interested in the coarsest possible $k$-step finite-horizon bisimulation relation for a given DTMC (labelled with atomic propositions) and time horizon $k$, which we denote by $\sim_{k}$. We can also define this as the union of all possible $k$-step finite-horizon bisimulation relations. Furthermore, for $\sim_{k}$ (or any other finitehorizon bisimulation relation), we can define a corresponding quotient DTMC, whose states are formed from the equivalence classes of $\sim_{k}$, and whose $k$-step behaviour is identical to the original DTMC $\mathcal{D}$.

This is similar, but not identical, to the process of building the quotient Markov chain corresponding to a full minimisation (see Definition 5). We must take care since, unlike for full bisimulation, given a state $B \in \mathcal{S} / \sim_{k}$ of the quotient model, the probabilities $\mathbf{P}\left(s, B^{\prime}\right)$ of moving to other equivalence classes $B^{\prime} \in \mathcal{S} / \sim_{k}$ can be different for each state $s \in B$ (according to the definition of $\sim_{k}$, probabilities are the same to go states with the same $(k-1)$-step, not $k$-step, behaviour). However, when they do differ, it suffices to pick an arbitrary representative from $B$. We formalise the quotient DTMC construction below, and then present some examples.

Definition 9 (Finite-horizon quotient DTMC). If $\mathcal{D}=\left(\mathcal{S}, \mathcal{S}_{\text {init }}, \mathbf{P}, \mathcal{A} \mathcal{P}, \mathcal{L}\right)$ is a DTMC and $\sim_{k}$ is a finite-horizon bisimulation on $\mathcal{D}$, then a quotient DTMC can be constructed as $\mathcal{D} / \sim_{k}=\left(\mathcal{S}^{\prime}, \mathcal{S}_{\text {init }}^{\prime}, \mathbf{P}^{\prime}, \mathcal{A} \mathcal{P}, \mathcal{L}^{\prime}\right)$ where:
$-\mathcal{S}^{\prime}=\mathcal{S} / \sim_{k}=\left\{[s]_{\sim_{k}} \mid s \in \mathcal{S}\right\}$
$-\mathcal{S}_{\text {init }}^{\prime}=\left\{[s]_{\sim_{k}} \mid s \in \mathcal{S}_{\text {init }}\right\}$
$-\mathbf{P}^{\prime}\left(B, B^{\prime}\right)=\mathbf{P}\left(\operatorname{rep}(B), B^{\prime}\right)$ for any $B, B^{\prime} \in \mathcal{S}^{\prime}$
$-\mathcal{L}^{\prime}(B)=\mathcal{L}(\operatorname{rep}(B))$ for any $B \in \mathcal{S}^{\prime}$,
where rep $: \mathcal{S} / \sim_{k} \rightarrow \mathcal{S}$ is an arbitrary function that selects a unique representative from each equivalence class of $\sim_{k}$, i.e., $B=[\operatorname{rep}(B)]_{\sim_{k}}$ for all $B \in \mathcal{S}^{\prime}$.

Example 1. Fig. 1 illustrates finite-horizon bisimulation on an example DTMC, shown in part (a). Fig.s 1 (b) and (c) show quotient DTMCs for 0-step and 1step finite-horizon bisimulation minimisation, respectively, where quotient state names indicate their corresponding equivalence class (e.g., $B_{23}$ corresponds to DTMC states $s_{2}$ and $s_{3}$ ). For 2-step minimisation (not shown), blocks $B_{23}$ and $B_{01}$ are both split in two, and only the states $s_{4}$ and $s_{5}$ remain bisimilar.

From the above, we see that $s_{2} \sim_{1} s_{3}$, but $s_{2} \not \chi_{2} s_{3}$. Consider the PCTL formula $\Phi=\mathrm{P}_{\bowtie p}\left[\operatorname{true} \mathrm{U}^{\leq k} a\right]$, which has depth $d(\Phi)=k$. Satisfaction of $\Phi$ is equivalent in states $s_{2}$ and $s_{3}$ for $k=1$, but not for $k=2$. To give another example, for $\Phi^{\prime}=\mathrm{P}_{>0}\left[\mathrm{P}_{>0.5}\left[\right.\right.$ true $\left.\left.\mathrm{U}^{\leq 2} a\right] \mathrm{U}^{\leq 1} a\right]$, which has $d\left(\Phi^{\prime}\right)=1+2-1=2$, we have $s_{3} \vDash \Phi^{\prime}$, but $s_{2} \not \vDash \Phi^{\prime}$.

In constructing the 1 -step quotient model (Fig. 1 (c)), we used $s_{1}$ as a representative of equivalence class $B_{01}=\left\{s_{0}, s_{1}\right\}$, which is why there is a transition to $B_{23}$. We could equally have used $s_{0}$, which would yield a different quotient DTMC, but which still preserves 1-step behaviour.

## 4 Finite-Horizon Bisimulation Minimisation

Bisimulation relations have a variety of uses, but our focus here is on using them to minimise a probabilistic model prior to verification, in order to improve the efficiency and scalability of the analysis. More precisely, we perform finite-horizon bisimulation minimisation, determining the coarsest possible finite-horizon bisimulation relation $\sim_{k}$, for a given $k$, and then constructing the corresponding quotient Markov chain. Theorem 1 tells us that it is then safe to perform verification on the smaller quotient model instead.

We begin, in this section, by presenting a classical partition-refinement based minimisation algorithm, which is based on an iterative splitting of an initially coarse partition of the state space until the required probabilistic bisimulation has been identified. In the next section, we will propose on-the-fly approaches which offer further gains in efficiency and scalability.

### 4.1 A Partition-Refinement Based Minimisation Algorithm

The standard approach to partition refinement is to use splitters [19,9], individual blocks in the current partition which show that one or more other blocks contain states that should be split into distinct sub-blocks. An alternative approach is to use a so-called signature-based method [8]. The basic structure of the algorithm remains the same, however the approach to splitting differs: rather than using splitters, a signature corresponding to the current partition is computed at each iteration for each state $s$. This signature comprises the probability of moving from $s$ in one step to each block in the partition. In the next iteration, all states with different signatures are placed in different blocks.

Because each iteration of the signature-based algorithm considers the onestep behaviour of every state in the model, it is relatively straightforward to adapt to finite-horizon bisimulation minimisation. Algorithm 1 shows the finitehorizon minimisation algorithm MinimiseFiniteHorizon. It takes a DTMC $\mathcal{D}$ and the time horizon $k$ as input. The partition $\Pi$ is first initialised to group states based on the different combinations of atomic propositions, i.e., states with identical labellings are placed in one block. ${ }^{1}$ The partition is then repeatedly split, each time by computing the signatures for each state and splitting

[^0]accordingly. The loop terminates either when $k$ iterations have been completed or no further splitting is possible. Finally, the quotient model is constructed, as described in the previous section.

Correctness. The correctness of MinimiseFiniteHorizon, i.e. that it generates the coarsest $k$-step finite-horizon bisimulation, can be argued with direct reference to Definition 6. For $k=0$, only the initialisation step at the start of the algorithm is needed. For $k>0$ the $i$ th iteration of the loop produces a partition $\Pi$ which groups precisely the equivalence classes of $\sim_{i}$, which are constructed from those of $\sim_{i-1}$, as in Definition 6. It is also clear that we group all equivalent states at each step, yielding the coarsest relation. If the algorithm terminates early, at step $j$, then $\sim_{i}=\sim_{k}$ for all $j \leq i \leq k$.

```
Algorithm 1: MinimiseFiniteHorizon
    Data: \(\mathcal{D}=\left(\mathcal{S}, \mathcal{S}_{\text {init }}, \mathbf{P}, \mathcal{A P}, \mathcal{L}\right), k\)
    \(\Pi, \Pi^{\prime}:=\emptyset\);
    // Initialise partition
    for \(A \subseteq \mathcal{A P}\) do
        \(B_{A}:=\{s \in \mathcal{S} \mid L(s)=A\}\)
        if \(B_{A} \neq \emptyset\) then \(\Pi:=\Pi \cup\left\{\left(\left\{B_{A}\right\},\langle \rangle\right)\right\}\);
    \(i:=1 ; \quad\) // Splitting loop
    while \(i \leq k \wedge \Pi \neq \Pi^{\prime}\) do
        \(\Pi^{\prime}:=\Pi ; \Pi:=\emptyset\)
        for \(s \in \mathcal{S}\) do
            Sig \(:=\langle \rangle\); // Compute signature
                for \(B \in \Pi^{\prime}\) do \(\operatorname{Sig}(B):=0\);
                for \(s \rightarrow s^{\prime}\) do
                    \(B_{s^{\prime}}:=\) block of \(\Pi^{\prime}\) containing \(s^{\prime}\)
                    \(\operatorname{Sig}\left(B_{s^{\prime}}\right):=\operatorname{Sig}\left(B_{s^{\prime}}\right)+\mathbf{P}\left(s, s^{\prime}\right)\)
                \(B_{s}:=\) block of \(\Pi^{\prime}\) containing \(s\)
                if \(\exists\left(B^{\prime}\right.\), Sig \() \in \Pi \wedge B^{\prime} \subseteq B_{s}\) then
                    \(B^{\prime}:=B^{\prime} \cup\{s\} ; \quad / /\) New blocks
                else
                    \(\Pi:=\Pi \cup\{(\{s\}\), Sig \()\}\)
        \(i:=i+1\)
    \(\mathcal{S}^{\prime}:=\emptyset ; \mathcal{S}_{\text {init }}^{\prime}:=\emptyset ; \quad\) // Build quotient
    for \((B, \operatorname{Sig}) \in \Pi\) do
        \(\mathcal{S}^{\prime}:=\mathcal{S}^{\prime} \cup\{B\}\)
        if \(B \cap \mathcal{S}_{\text {init }} \neq \emptyset\) then \(\mathcal{S}_{\text {init }}^{\prime}:=\mathcal{S}_{\text {init }}^{\prime} \cup\{B\}\);
        \(\mathbf{P}^{\prime}(B, \cdot):=\operatorname{Sig}\)
        \(\mathcal{L}^{\prime}(B):=\mathcal{L}(s)\) for any \(s \in B\)
    return \(\mathcal{D}^{\prime}=\left(\mathcal{S}^{\prime}, \mathcal{S}_{\text {init }}^{\prime}, \mathbf{P}^{\prime}, \mathcal{A} \mathcal{P}, \mathcal{L}^{\prime}\right)\)
```


## 5 On-the-Fly Finite-Horizon Minimisation

A key limitation of the partition-refinement approach presented in the previous section is that it takes as input the full DTMC to be minimised, the construction of which can be expensive in terms of both time and space. This can remove any potential gains in terms of scalability that minimisation can provide.

To resolve this, we now propose methods to compute a finite-horizon bisimulation minimisation in an on-the-fly fashion, where the minimised model is constructed directly from a high-level modelling language description of the original model, bypassing construction of the full, un-reduced DTMC. In our case, the probabilistic models are described using the modelling language of the PRISM model checker [15], which is based on guarded commands.

Our approach works through a backwards traversal of the model, which allows us to perform bisimulation minimisation on the fly. For simplicity, we focus on preserving the subclass of PCTL properties comprising a single P operator, more precisely, those of the form $\mathrm{P}_{\bowtie p}\left[b_{1} \mathrm{U} \leq k\right.$ 数] for atomic propositions $b_{1}$ and $b_{2}$. This is the kind of property most commonly found in practice.

### 5.1 The On-the-Fly Minimisation Algorithm

The basic approach to performing finite-horizon minimisation on the fly is shown as FiniteHorizonOnTheFly, in Algorithm 2. This takes model, which is a description of the DTMC, $B_{1}$ and $B_{2}$, the sets of states satisfying $b_{1}$ and $b_{2}$, respectively, in the property $\mathrm{P}_{\bowtie p}\left[b_{1} \mathrm{U}^{\leq k} b_{2}\right]$, and the time horizon $k$. The algorithm does not make any assumptions about how sets of states are represented or manipulated. Below, we will discuss two separate instantiations of it.

The algorithm is based on a backwards traversal of the model. It uses a separate algorithm FindMergedPredecessors(model, target, restrict), which queries the DTMC (model) to find all (immediate) predecessors of states in target that are also in restrict (the restrict set will be used to restrict attention to the set $B_{1}$ corresponding to the left-hand side $b_{1}$ of the until formula). The algorithm also groups the predecessor states in blocks according to the probabilities with which they transition to target and returns these too. As above, each instantiation of Algorithm 2 will use a separate implementation of the FindMergedPredecessors algorithm.

The main loop of the algorithm iterates backwards through the model: after the $i$ th iteration, it has found all states that can reach the target set $B_{2}$ within $i$ steps with positive probability. The new predecessors for each iteration are stored in a set of blocks $P$. A separate set $P^{\prime}$ is used to store predecessors of blocks in $P$, which will then be considered in the next iteration.

More precisely, $P$ (and $P^{\prime}$ ) store, like in Algorithm 1, a list of pairs ( $B, D$ ) where $B$ is a block (a set of states) and $D$ is a (partial) probability distribution storing probabilities of outgoing transitions (from $B$, to other blocks). The set $\Pi$, which is used to construct the partition representing the finite-horizon bisimulation relation, is also stored as a list of pairs.

```
Algorithm 2: FiniteHorizonOnTheFly
    Data: model, \(B_{1}, B_{2}, k\)
    \(P:=\left\{\right.\) FindMergedPredecessors \(\left(\right.\) model \(\left.\left., B_{2}, B_{1}\right)\right\} ; P^{\prime}:=\emptyset\)
    \(\Pi:=\left\{\left(B_{2},\langle \rangle\right)\right\}\)
    \(i:=1\)
    while \(P \neq \emptyset \wedge i \leq k\) do
        \((B, D):=\operatorname{pop}(P) ; \quad / /\) block \(B\), (sub)distribution \(D\)
        for \(\left(B^{\prime}, D^{\prime}\right) \in \Pi \wedge B \neq \emptyset\) do
            if \(B^{\prime} \cap B \neq \emptyset\) then
                replace \(\left(B^{\prime}, D^{\prime}\right)\) in \(\Pi\) with \(\left.\left(B^{\prime} \backslash B\right), D^{\prime}\right)\) and \(\left(B^{\prime} \cap B, D^{\prime} \cup D\right)\)
                \(B:=B \backslash B^{\prime}\)
                refine all \(\left(B^{\prime \prime}, D^{\prime \prime}\right) \in \Pi\) and \((B, D)\) with respect to the split of \(B^{\prime}\)
            end
        end
        if \(B \neq \emptyset\) then
            \(\Pi:=\Pi \cup\{(B, D)\}\)
            \(P^{\prime}:=P^{\prime} \cup\left\{\right.\) FindMergedPredecessors \(\left(\right.\) model \(\left.\left., B, B_{1}\right)\right\}\)
        end
        if \(\left(P=\emptyset \wedge P^{\prime} \neq \emptyset\right)\) then
            \(P:=P^{\prime} ; P^{\prime}:=\emptyset\)
            \(i:=i+1\)
        end
    end
    return FiniteHorizonQuotient \((\Pi)\)
```

```
Algorithm 3: FiniteHorizonQuotient
    Data: \(\Pi\)
    \(\mathcal{S}^{\prime}:=\left\{B_{\text {sink }}\right\} ; \mathcal{L}^{\prime}\left(B_{\text {sink }}\right)=\emptyset ; \mathcal{S}_{\text {init }}^{\prime}:=\emptyset ; \mathbf{P}^{\prime}\left(B_{\text {sink }}, \cdot\right):=\left\langle B_{\text {sink }} \rightarrow 1\right\rangle ;\)
    for \((B, D) \in \Pi\) do
        \(\mathcal{S}^{\prime}:=\mathcal{S}^{\prime} \cup\{B\}\)
        if \(B \cap \mathcal{S}_{\text {init }} \neq \emptyset\) then \(\mathcal{S}_{\text {init }}^{\prime}:=\mathcal{S}_{\text {init }}^{\prime} \cup\{B\}\);
        \(p_{\text {sink }}=1-\sum_{\left(B^{\prime}, D^{\prime}\right) \in \Pi} D\left(B^{\prime}\right)\)
        \(\mathbf{P}^{\prime}(B, \cdot):=D \cup\left\langle B_{\text {sink }} \rightarrow p_{\text {sink }}\right\rangle\)
        \(\mathcal{L}^{\prime}(B):=\mathcal{L}(s)\) for any \(s \in B\)
    end
    return \(\mathcal{D}^{\prime}=\left(\mathcal{S}^{\prime}, \mathcal{S}_{\text {init }}^{\prime}, \mathbf{P}^{\prime}, \mathcal{A P}, \mathcal{L}^{\prime}\right)\)
```

Algorithm 2 begins by finding all immediate predecessors of states in $B_{2}$ that are also in $B_{1}$ and putting them in $P$. In each iteration, it takes each block-distribution pair $(B, D)$ from $P$ one by one: it will add this to the current partition $\Pi$. But, before doing so, it checks whether $B$ overlaps with any existing blocks $B^{\prime}$ in $\Pi$. If so, $B^{\prime}$ is split in two, and the overlap is removed from $B$. At this point, the partition $\Pi$ is refined to take account of the splitting of block $B^{\prime}$. We repeatedly recompute the probabilities associated with each block in $\Pi$ and, if these are then different for states within that block, it is also split.

Each iteration of the main loop finishes when all pairs $(B, D)$ from $P$ have been dealt with. If $i<k$, then newly found predecessors $P^{\prime}$ are copied to $P$ and the process is repeated. If $i=k$, then the time horizon $k$ has been reached and the finite-horizon bisimulation has been computed.

Finally, the quotient model is built. The basic construction is as in Algorithm 1 but, since on-the-fly construction only partially explores the model, we need to add an extra sink state to complete the DTMC.

Computing predecessors. One of the main challenges in implementing the on-the-fly algorithm is determining the predecessors of a given set of states from the high-level modelling language description. The PRISM language, used here, is based on guarded commands, for example:

$$
c>0 \rightarrow c / K:\left(c^{\prime}=c-1\right)+1-c / K:\left(c^{\prime}=c+1\right)
$$

The meaning is that, when a state satisfies the guard $(c>0)$, the updates (decrementing or incrementing variable $c$ ) can be executed, each with an associated probability $(c / K$ or $1-c / K)$. We assume here a single PRISM module of commands (multiple modules can be syntactically expanded into a single one [23]).

In the following sections, we describe two approaches to finding predecessors: one symbolic, which represents blocks (sets of states) as predicates and uses an SMT (satisfiability modulo theories) [5] based implementation; and one explicitstate, which explicitly enumerates the states in each block.

### 5.2 Symbolic (SMT-based) Minimisation

Our first approach represents state sets (i.e., blocks of the bisimulation partition) symbolically, as predicates over PRISM model variables. If target is a predicate representing a set of states, their predecessors, reached by applying some guarded command update update, can be found using the weakest precondition, denoted $\mathbf{w p}$ (update, target). More precisely, if the guard of the command is guard, and bounds represents the lower and upper bounds of all model variables, the following expression captures the set of states, if any, that are predecessors:

$$
\text { bounds } \wedge \text { guard } \wedge \mathbf{w p}(\text { update, target })
$$

We determine, for each guarded command update in the model description, whether states can reach target via that update by checking the satisfiability of the expression above using an SMT solver. FindMergedPredecessors (see Algorithm 4) is used to determine predecessors in this way. It also restricts attention to states satisfying a further expression restrict.

The probability attached to an update in a guarded command is in general a state-dependent expression prob (see the earlier example command) so this must be analysed when FindMergedPredecessors groups states according to the probability with which they transition to target. If the SMT query in the algorithm is satisfiable, a valid probability is also obtained from the corresponding valuation ( $p^{\prime}$ in Algorithm 4). The conjunction of the expression predecessor and

```
Algorithm 4: FindMergedPredecessors (SMT-based)
    Data: model, target, restrict
    \(P:=\emptyset\)
    bounds := variable bounds from model
    foreach (guard, updates) in model do
        foreach (prob, update) in updates do
            predecessor := restrict \(\wedge\) bounds \(\wedge\) guard \(\wedge \mathbf{w p}(\) update, target \()\)
            \(q u e r y:=\) predecessor \(\wedge(p=p r o b)\)
            while query is satisfiable do
                \(p^{\prime}:=\) value of \(p\) in query
                if \(\left(B,\left\langle\right.\right.\) target \(\left.\left.\rightarrow p^{\prime}\right\rangle\right) \in P\) for some \(B\) then
                replace ( \(B,\left\langle\right.\) target \(\left.\rightarrow p^{\prime}\right\rangle\) ) in \(P\) with
                ( \(B \vee\) predecessor, \(\left\langle\left\langle\right.\right.\) target \(\left.\rightarrow p^{\prime}\right\rangle\) )
            else
                \(P:=P \cup\left\{\left(\right.\right.\) predecessor,\(\left\langle\right.\) target \(\left.\left.\left.\rightarrow p^{\prime}\right\rangle\right)\right\}\)
                end
                query \(:=\) query \(\wedge\left(\right.\) prob \(\left.\neq p^{\prime}\right)\)
            end
        end
    end
    return \(P\)
```

$p=p r o b$ denotes the set of predecessors with the same probability. To obtain all such probabilities, the algorithm adds a blocking expression $\operatorname{prob} \neq p^{\prime}$ to the query and repeats the process.

SMT-based methods for probabilistic bisimulation minimisation have been developed previously [6]. One key difference here is that our approach handles transition probabilities expressed as state-dependent expressions, rather than fixed constants, which are needed for some of the models we later evaluate.

### 5.3 Explicit-State Minimisation

As an alternative to the symbolic approach using SMT, we developed an explicitstate implementation of finite-horizon minimisation in which the blocks of equivalent states are represented by explicitly listing the states that comprise them. As in the previous algorithm, the blocks are refined at each time step such that states residing in the same block have equal transition probabilities to the required blocks. To improve performance and store states compactly, we hash them based on the valuation of variables that define them. This is done in such a way that the hash values are bi-directional (one-to-one).

The algorithm explicitly computes the predecessor state for each update and each state in the set target, the transition probability is then computed for each predecessor state and these are collected in order to group states into sets. The set restrict is not stored explicitly, but rather as a symbolic expression which is then evaluated against each state's variable values to compute the intersection.

## 6 Experimental Results

We have implemented the bisimulation minimisation techniques presented in this paper as an extension of the PRISM model checker [15], and applied them to a range of benchmark models. For both the partition-refinement based minimisation of Sect. 4, and the on-the-fly methods in Sect. 5, we build on PRISM's "explicit" model checking engine. For the SMT-based variant, we use the Z3 solver [4], through the Z3 Java API. All our experiments were run on an Intel Core i7 2.8 GHz machine, using 2 GB of RAM.

Our investigation is in two parts. First, we apply the partition-refinement algorithm to several DTMCs from the PRISM benchmark suite [17] to get an idea of the size of reductions that can be obtained on some standard models. We use: Crowds (an anonymity protocol), $E G L$ (a contract signing protocol) and $N A N D$ (NAND multiplexing). Details of all models, parameters and properties used can be found at [24]. A common feature of these models is that they have a single initial state, from which properties are verified. Since on-the-fly approaches explore backwards from a target set, we would usually need to consider time horizons $k$ high enough such that the whole model was explored.

So, to explore in more depth the benefits of the on-the-fly algorithms, we consider another common class of models in probabilistic verification: those in which we need to exhaustively check whether a property is true over a large set of possible configurations. We use Approximate majority [2], a population protocol for computing a majority value amongst a set of $K$ agents, and two simple models of genetic algorithms [21] in which a population of $K$ agents evolves over time, competing to exist according to a fitness value in the range $0, \ldots, N-1$. In the first variant, tournament, the agent with the highest value wins; in the second, modulo, the sum of the two scores is used modulo N. Again, details of all models, parameters and properties used can be found at [24].

### 6.1 The partition-refinement algorithm

Fig. 2 shows results for the partition-refinement algorithm. The top row of plots shows the number of blocks in the partition built by finite-horizon bisimulation minimisation for different values of $k$ on the first three benchmark examples. For the largest values of $k$ shown, we have generated the partition corresponding to the full (non-finite-horizon) bisimulation. In most cases, the growth in the number of blocks is close to linear in $k$, although it is rather less regular for the $N A N D$ example. In all cases, it seems that the growth is slow enough that verifying finite-horizon properties for a range of values of $k$ can be done on a considerably smaller model than the full bisimulation.

The bottom row of plots shows, for the same examples, the time required to perform bisimulation minimisation and then verify a $k$-step finite-horizon property (details at [24]). The black lines show the time for finite-horizon minimisation, the grey lines for full minimisation. The latter are relatively flat, indicating that the time for verification (which is linear in $k$ ) is very small compared to the time needed for minimisation. However, we see significant gains in the total time required for finite-horizon minimisation compared to full minimisation.


Fig. 2. Results for partition-refinement. Top: quotient size for varying time horizon $k$. Bottom: time for finite-horizon (black) and full (grey) minimisation/verification.

However, despite these gains, the times to minimise and verify the quotient model are still larger than to simply build and verify the full model. This is primarily because the partition refinement algorithm requires construction of the complete model first, the time for which eclipses any gains from minimisation. This was the motivation for the on-the-fly algorithms, which we evaluate next.

### 6.2 On-the-Fly Algorithms

Table 1 shows model sizes and timings for the on-the-fly algorithms on a range of models and scenarios. The left four columns show the model (and which on-the-fly algorithm was used), any parameters required ( $N$ or $K$ ) and the time horizon $k$. Next, under the headings 'Full Red.' and 'Finite Horiz.', we show the reductions in model size obtained using full (non-finite-horizon) and finitehorizon minimisation (for several $k$ ), respectively. In the first case, 'States' and 'Blocks' show the size of the full DTMC and the fully reduced quotient model, respectively. For the second case, 'Blocks' is the size of the finite-horizon quotient model and, to give a fair comparison, 'States' is the number of states in the full DTMC that can reach the target of the property within $k$ steps (i.e., the number of states across all blocks). The rightmost three columns show the time required to build the model in three scenarios: 'Finite Horiz.' uses the on-the-fly approach over $k$ steps; 'Full Red.' builds the full (non-finite-horizon) quotient by repeating the on-the-fly algorithm until all states have been found; and 'PRISM' builds the full model using its most efficient (symbolic) construction engine.

First, we note that finite-horizon minimisation yields useful reductions in model size in all cases, both with respect to the full model and to normal (nonfinite horizon) minimisation. Bisimulation reduces models by a factor of roughly 2 and 5, for the Approximate majority and Modulus examples, respectively. For Tournament, a very large reduction is obtained since, for the property checked, the model ends up being abstracted to only distinguish two fitness values. Finitehorizon minimisation gives models that are smaller again, by a factor of between

| $\begin{gathered} \text { Model } \\ \text { (method) } \end{gathered}$ | Param.s |  | $k$ | Full Red. |  | Finite Horiz. <br> States Blocks |  | Time (s) |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $N$ | K |  | States | Blocks |  |  | PRISM | Full Red. | Finite Horiz. |
| Approx. majority (explicit) | n/a | 100 | 20 | 20300 | 10201 | 242 | 122 | 11.0 | 14.2 | 0.2 |
|  |  |  | 40 |  |  | 882 | 442 |  |  | 0.3 |
|  |  |  | 60 |  |  | 1922 | 962 |  |  | 0.4 |
|  |  | 150 | 100 | 45450 | 22801 | 5202 | 2602 | 46.1 | 83.1 | 1.2 |
|  |  |  | 150 |  |  | 11552 | 5777 |  |  | 5.1 |
|  |  |  | 200 |  |  | 20402 | 10202 |  |  | 15.6 |
|  |  | 200 | 250 | 80600 | 40401 | 31752 | 15877 | memout | 293.5 | 40.8 |
|  |  |  | 300 |  |  | 45602 | 22802 |  |  | 93.9 |
|  |  |  | 350 |  |  | 61952 | 30977 |  |  | 180.8 |
|  |  | 250 | 375 | 125750 | 63001 | 71064 | 35533 | memout | 773.5 | 247.8 |
|  |  |  | 400 |  |  | 80802 | 40402 |  |  | 323.2 |
|  |  |  | 425 |  |  | 91164 | 45583 |  |  | 416.6 |
| Genetic alg. tournament (explicit) | 8 | 22 | 8 | 1184040 | 22 | 6435 | 10 | 19.2 | 5.3 | 0.3 |
|  |  |  | 9 |  |  | 11440 | 11 |  |  | 0.4 |
|  |  |  | 10 |  |  | 19448 | 12 |  |  | 0.4 |
|  |  | 23 | 8 | 1560780 | 23 | 6435 | 10 | 31.1 | 7.0 | 0.3 |
|  |  |  | 9 |  |  | 11440 | 11 |  |  | 0.4 |
|  |  |  | 10 |  |  | 19448 | 12 |  |  | 0.4 |
|  | 10 | 21 | 8 | 10015005 | 21 | 24310 | 10 | 59.0 | 43.6 | 0.5 |
|  |  |  | 9 |  |  | 48620 | 11 |  |  | 0.6 |
|  |  |  | 10 |  |  | 92378 | 12 |  |  | 0.7 |
|  |  | 22 | 8 | 14307150 | 22 | 24310 | 10 | 61.3 | 51.3 | 0.5 |
|  |  |  | 9 |  |  | 48620 | 11 |  |  | 0.6 |
|  |  |  | 10 |  |  | 92378 | 12 |  |  | 0.7 |
| Genetic alg. tournament (SMT) | 4 | 9 | 3 | 165 | 9 | 20 | 5 | 0.03 | 155 | 4.5 |
|  |  |  | 4 |  |  | 35 | 6 |  |  | 11.1 |
|  |  |  | 5 |  |  | 56 | 7 |  |  | 23.5 |
|  |  | 10 | 3 | 220 | 10 | 20 | 5 | 0.03 | 215 | 9.3 |
|  |  |  | 4 |  |  | 35 | 6 |  |  | 15.1 |
|  |  |  | 5 |  |  | 56 | 7 |  |  | 31.1 |
|  | 5 | 9 | 3 | 330 | 9 | 35 | 5 | 0.04 | 723.4 | 22.1 |
|  |  |  | 4 |  |  | 70 | 6 |  |  | 70.7 |
|  |  |  | 5 |  |  | 126 | 7 |  |  | 180.9 |
|  |  | 10 | 3 | 495 | 10 | 35 | 5 | 0.04 | 1998.7 | 48.8 |
|  |  |  | 4 |  |  | 70 | 6 |  |  | 82.0 |
|  |  |  | 5 |  |  | 126 | 7 |  |  | 233.7 |
| Genetic alg. modulus (explicit) | 7 | 19 | 8 | 177100 | 29565 | 22179 | 3638 | 0.4 | 475.3 | 6.8 |
|  |  |  | 9 |  |  | 39404 | 6491 |  |  | 21.6 |
|  |  |  | 10 |  |  | 66002 | 10914 |  |  | 64.3 |
|  |  | 20 | 8 | 230230 | 38431 | 22179 | 3637 | 0.5 | 778.6 | 6.9 |
|  |  |  | 9 |  |  | 39404 | 6488 |  |  | 20.3 |
|  |  |  | 10 |  |  | 66068 | 10914 |  |  | 65.9 |
|  | 9 | 11 | 6 | 75582 | 12707 | 24822 | 3435 | 0.3 | 79.9 | 7.7 |
|  |  |  | 7 |  |  | 51756 | 8084 |  |  | 32.3 |
|  |  |  | 8 |  |  | 70448 | 11745 |  |  | 58.3 |
|  |  | 12 | 6 | 125970 | 21145 | 24906 | 3450 | 0.3 | 253.5 | 7.8 |
|  |  |  | 7 |  |  | 54440 | 8482 |  |  | 37.4 |
|  |  |  | 8 |  |  | 88642 | 14207 |  |  | 102.4 |

Table 1. Experimental results for on-the-fly bisimulation minimisation.
2 and 10 on these examples, even for relatively large values of $k$ on the Approximate majority models. Comparing columns 7 and 8 in Table 1 shows that much of the reduction is indeed due to merging of bisimilar states, not just to a $k$-step truncation of the state space from the backwards traversal.

Regarding performance and scalability, we first discuss results for the SMTbased implementation. We were only able to apply this to the Tournament example, where a very large reduction in state space is achieved. On a positive note, the SMT-based approach successfully performs minimisation here and gives a symbolic (Boolean expression) representation for each block. However, the process is slow, limiting applicability to DTMCs that can already be verified without
minimisation. Our experiments showed that the slow performance was largely caused by testing for overlaps between partition blocks resulting in a very large number of calls to the SMT solver.

The explicit-state on-the-fly implementation performed much better and Table 1 shows results for all three models. In particular, for the Tournament example, finite-horizon minimisation and verification is much faster than verifying the full model using the fastest engine in PRISM. This is because we can bypass construction of the full models, which have up to 14 million states for this example. For the Modulus example, the model reductions obtained are much smaller and, as a result, PRISM is able to build and verify the model faster. However, for the Approximate Majority example, the minimisation approach can be applied to larger models than can be handled by PRISM. For this example, although the state spaces of the full model are manageable, the models prove poorly suited to PRISM's model construction implementation (which is based on binary decision diagram data structures).

## 7 Conclusions

We have presented model reduction techniques for verifying finite-horizon properties on discrete-time Markov chains. We formalised the notion of $k$-step finitehorizon bisimulation mininisation and clarified the subset of PCTL that it preserves. We have given both a partition-refinement algorithm and an on-the-fly approach, implemented in both a symbolic (SMT-based) and explicit-state manner as an extension of PRISM. Experimental results demonstrated that significant model reductions can be obtained in this manner, resulting in improvements in both execution time and scalability with respect to the existing efficient implementations in PRISM.

Future work in this area will involve extending the techniques to other classes of probabilistic models, and adapting the on-the-fly approaches to preserve the full time-bounded fragment of PCTL, including nested formulae.

Acknowledgements. This work has been supported by the EU-FP7-funded project HIERATIC.

## References

1. Aljazzar, H., Fischer, M., Grunske, L., Kuntz, M., Leitner, F., Leue, S.: Safety analysis of an airbag system using probabilistic FMEA and probabilistic counterexamples. In: Proc. QEST'09 (2009)
2. Angluin, D., Aspnes, J., Eisenstat, D.: A simple population protocol for fast robust approximate majority. Distributed Computing 21(2), 87-102 (2008)
3. Aziz, A., Singhal, V., Balarin, F., Brayton, R.K., Sangiovanni-Vincentelli, A.L.: It usually works: The temporal logic of stochastic systems. In: CAV'95 (1995)
4. De Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Proc. TACAS'08 (2008)
5. De Moura, L., Bjørner, N.: Satisfiability modulo theories: Introduction and applications. Communications of the ACM 54(9), 69-77 (2011)
6. Dehnert, C., Katoen, J.P., Parker, D.: SMT-based bisimulation minimisation of Markov models. In: Proc. VMCAI'13. pp. 28-47 (2013)
7. Della Penna, G., Intrigila, B., Melatti, I., Tronci, E., Zilli, M.V.: Finite horizon analysis of markov chains with the mur $\phi$ verifier. STTT 8(4-5), 397-409 (2006)
8. Derisavi, S.: Signature-based symbolic algorithm for optimal Markov chain lumping. In: Proc. QEST'07. pp. 141-150. IEEE Computer Society (2007)
9. Derisavi, S., Hermanns, H., Sanders, W.H.: Optimal state-space lumping in Markov chains. Information Processing Letters 87(6), 309-315 (2003)
10. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. FAC 6(5), 512-535 (1994)
11. Heath, J., Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic model checking of complex biological pathways. In: Proc. CMSB'06 (2006)
12. Katoen, J.P., Kemna, T., Zapreev, I., Jansen, D.: Bisimulation minimisation mostly speeds up probabilistic model checking. In: Proc. TACAS'07. vol. 4424 (2007)
13. Katoen, J.P., Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.: The ins and outs of the probabilistic model checker MRMC. Perform. eval. 68(2), 90-104 (2011)
14. Kemeny, J., Snell, J., Knapp, A.: Denumerable Markov Chains. Springer-Verlag, 2nd edn. (1976)
15. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Proc. CAV'11. LNCS, vol. 6806, pp. 585-591. Springer (2011)
16. Kwiatkowska, M., Norman, G., Sproston, J., Wang, F.: Symbolic model checking for probabilistic timed automata. Information and Computation 205(7) (2007)
17. Kwiatkowska, M., Norman, G., Parker, D.: The PRISM benchmark suite. In: Proc. QEST'12. pp. 203-204 (2012)
18. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. Information and Computation 94(1), 1-28 (1991)
19. Paige, R., Tarjan, R.E.: Three partition refinement algorithms. SIAM Journal on Computing 16(6), 973-989 (1987)
20. Valmari, A., Franceschinis, G.: Simple o(m logn) time Markov chain lumping. In: Proc. TACAS'10, pp. 38-52. Springer (2010)
21. Vose, M.: The simple genetic algorithm: Foundations and theory. MIT Press (1999)
22. Wimmer, R., Becker, B.: Correctness issues of symbolic bisimulation computation for markov chains. In: Proc. MMB'10 (2010)
23. http://www.prismmodelchecker.org/doc/semantics.pdf
24. http://www.prismmodelchecker.org/files/spin16fh

[^0]:    ${ }^{1}$ In the algorithm, we store the signatures with the partition, so $\Pi$ is a list of pairs of blocks (state-sets) and signatures (distributions).

