Verification and Control of Partially Observable Probabilistic Systems

Gethin Norman[1], David Parker[2], and Xueyi Zou[3]

[1] School of Computing Science, University of Glasgow, UK
[2] School of Computer Science, University of Birmingham, UK
[3] Department of Computer Science, University of York, UK

Abstract. We present automated techniques for the verification and control of partially observable, probabilistic systems for both discrete and dense models of time. For the discrete-time case, we formally model these systems using partially observable Markov decision processes; for dense time, we propose an extension of probabilistic timed automata in which local states are partially visible to an observer or controller. We give probabilistic temporal logics that can express a range of quantitative properties of these models, relating to the probability of an event's occurrence or the expected value of a reward measure. We then propose techniques to either verify that such a property holds or synthesise a controller for the model which makes it true. Our approach is based on a grid-based abstraction of the uncountable belief space induced by partial observability and, for dense-time models, an integer discretisation of real-time behaviour. The former is necessarily approximate since the underlying problem is undecidable, however we show how both lower and upper bounds on numerical results can be generated. We illustrate the effectiveness of the approach by implementing it in the PRISM model checker and applying it to several case studies from the domains of task and network scheduling, computer security and planning.

1 Introduction

Guaranteeing the correctness of complex computerised systems often needs to take into account quantitative aspects of system behaviour. This includes the modelling of *probabilistic* phenomena, such as failure rates for physical components, uncertainty arising from unreliable sensing of a continuous environment, or the explicit use of randomisation to break symmetry. It also includes *timing* characteristics, such as time-outs or delays in communication or security protocols. To further complicate matters, such systems are often *nondeterministic* because their behaviour depends on inputs or instructions from some external entity such as a controller or scheduler.

Automated verification techniques such as probabilistic model checking have been successfully used to analyse quantitative properties of probabilistic systems across a variety of application domains, including wireless communication protocols, computer security and task scheduling. These systems are commonly

modelled using *Markov decision processes* (MDPs), if assuming a discrete notion of time, or *probabilistic timed automata* (PTAs), if using a dense model of time. On these models, we can consider two problems: *verification* that it satisfies some formally specified property for any possible resolution of nondeterminism; or, dually, *synthesis* of a controller (i.e., a means to resolve nondeterminism) under which a property is guaranteed to hold. For either case, an important consideration is the extent to which the system's state is *observable* to the entity controlling it. For example, to verify that a security protocol is functioning correctly, it may be essential to model the fact that some data held by a participant is not externally visible; or, when synthesising an optimal schedule for sending packets over a network, a scheduler may not be implementable in practice if it bases its decisions on information about the state of the network that is unavailable due to the delays and costs associated with probing it.

Partially observable MDPs (POMDPs) are a natural way to extend MDPs in order to tackle this problem. However, the analysis of POMDPs is considerably more difficult than MDPs since key problems are undecidable [38]. A variety of verification problems have been studied for these models (see, e.g., [1,5,16]) and the use of POMDPs is common in fields such as AI and planning [11], but there is limited progress in the development of practical techniques for probabilistic verification in this area, or exploration of their applicability.

In this paper, we present novel techniques for verification and control of partially observable, probabilistic systems under both discrete and dense models of time. We use POMDPs in the case of discrete-time models and, for dense time, propose a model called *partially observable probabilistic timed automata* (POPTAs), which extends the existing model of PTAs with a notion of partial observability. The semantics of a POPTA is an infinite-state POMDP. In order to specify verification and control problems on POMDPs and POPTAs, we define temporal logics to express properties of these models relating to the probability of an event (e.g., the probability of some observation eventually being made) or the expected value of various reward measures (e.g., the expected time until some observation). Nondeterminism in both a POMDP and a POPTA is resolved by a *strategy* that decides which actions to take and when to take them, based only on the history of observations (not states). The core problems we address are how to *verify* that a temporal logic property holds for all possible strategies, and how to *synthesise* a strategy under which the property holds.

In order to achieve this, we use a combination of techniques. To analyse a POMDP, we use grid-based techniques [37,54], which transform it to a fully observable but continuous-space MDP and then approximate its solution based on a finite set of grid points. We use this to construct and solve a strategy of the POMDP. The result is a pair of lower and upper bounds on the property of interest for the POMDP. If this is not precise enough, we can refine the grid and repeat. In the case of POPTAs, we develop a *digital clocks* discretisation, which extends the existing notion for PTAs [32]. The discretisation reduces the analysis to a *finite* POMDP, and hence we can use the techniques we have developed for analysing POMDPs. We define the conditions under which temporal logic

properties are preserved by the discretisation step and prove the correctness of the reduction under these conditions.

We implemented these methods in a prototype tool based on PRISM [31,44], and investigated their applicability by developing a number of case studies including: wireless network scheduling, a task scheduling problem, a covert channel prevention device (the NRL pump) and a non-repudiation protocol. Despite the undecidability of the POMDP problems we consider, we show that useful results can be obtained, often with precise bounds. In each case study, partial observability, nondeterminism, probability and, in the case of the dense-time models, real-time behaviour are all crucial ingredients to the analysis. This is a combination not supported by any existing techniques or tools.

A preliminary conference version of this paper, was published as [41].

1.1 Related work

POMDPs are common in fields such as AI and planning: they have many applications [11] and tool support exists [43]. However, unlike verification, the focus in these fields is usually on finite-horizon and discounted reward objectives. Early undecidability for key problems can be found in, e.g., [38]. POMDPs have also been applied to problems such as scheduling in wireless networks since, in practice, information about the state of wireless connections is often unavailable and varies over time; see e.g. [28,35,52,27,23].

POMDPs have also been studied by the formal verification community, see e.g. [1,5,16], establishing undecidability and complexity results for various qualitative and quantitative verification problems. In the case of qualitative analysis, [14] presents an approach for the verification and synthesis of POMDPs against LTL properties when restricting to finite-memory strategies. This has been implemented and applied to an autonomous system [49]. For quantitative properties, the recent work of [15] extends approaches developed for finite-horizon objectives to approximate the minimum expected reward of reaching a target (while ensuring the target is reached with probability 1), under the requirement that all rewards in the POMDP are positive.

Work in this area often also studies related models such as Rabin's probabilistic automata [5], which can be seen as a special case of POMDPs, and partially observable stochastic games (POSGs) [17], which generalise them. More practically oriented work includes: [22], which proposes a counter-example-driven refinement method to approximately solve MDPs in which components have partial observability of each other; and [13], which synthesises concurrent program constructs using a search over memoryless strategies in a POSG.

Theoretical results [8] and algorithms [12,20] have been developed for synthesis of partially observable timed games. In [8], it is shown that the synthesis problem is undecidable and, if the resources of the controller are fixed, decidable but prohibitively expensive. The algorithms require constraints on controllers: in [12], controllers only respond to changes made by the environment and, in [20],

their structure must be fixed in advance. We are not aware of any work for probabilistic real-time models in this area.

1.2 Outline

Section 2 describes the discrete-time models of MDPs and POMDPs, and Section 3 presents our approach for POMDP verification and strategy synthesis. In Section 4, we introduce the dense-time models of PTAs and POPTAs, and then, in Section 5, give our verification and strategy synthesis approach for POPTAs using digital clocks. Section 6 describes the implementation of our techniques for analysing POMDPs and POPTAs in a prototype tool, and demonstrates its applicability using several case studies. Finally, Section 7 concludes the paper.

2 Partially Observable Markov Decision Processes

In this section, we consider systems exhibiting probabilistic, nondeterministic and discrete-time behaviour. We first introduce MDPs, and then describe POMDPs, which extend these to include partial observability. For a more detailed tutorial on verification techniques for MDPs, we refer the reader to, for example, [21].

2.1 Markov decision processes

Let $Dist(X)$ denote the set of discrete probability distributions over a set X, δ_x the distribution that selects $x \in X$ with probability 1, and $\mathbb{R}$ the set of non-negative real numbers.

Definition 1 (MDP). *An MDP is a tuple* $\mathsf{M}=(S, \bar{s}, A, P, R)$ *where:*

- S *is a set of states;*
- $\bar{s} \in S$ *is an initial state;*
- A *is a set of actions;*
- $P : S \times A \to Dist(S)$ *is a (partial) probabilistic transition function;*
- $R=(R_S, R_A)$ *is a reward structure where* $R_S : S \to \mathbb{R}$ *is a state reward function and* $R_A : S \times A \to \mathbb{R}$ *an action reward function.*

An MDP M represents the evolution of a system exhibiting both probabilistic and nondeterministic behaviour through states from the set S. Each state $s \in S$ of M has a set $A(s) \stackrel{\text{def}}{=} \{a \in A \mid P(s,a) \text{ is defined}\}$ of *available* actions. The choice between which available action is chosen in a state is nondeterministic. In a state s, if action $a \in A(s)$ is selected, then the probability of moving to state s' equals $P(s,a)(s')$.

A *path* of M is a finite or infinite sequence $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$, where $s_i \in S$, $a_i \in A(s_i)$ and $P(s_i, a_i)(s_{i+1})>0$ for all $i \in \mathbb{N}$. The $(i+1)$th state s_i of path π is denoted $\pi(i)$ and, if π is finite, $last(\pi)$ denotes its final state. We write $FPaths_{\mathsf{M}}$ and $IPaths_{\mathsf{M}}$, respectively, for the set of all finite and infinite paths of

M starting in the initial state $\bar{s}$. MDPs are also annotated with *rewards*, which can be used to model a variety of quantitative measures of interest. A reward of $R(s)$ is accumulated when passing through state s and a reward of $R(s,a)$ when taking action a from state s.

A *strategy* of M (also called a *policy* or *scheduler*) is a way of resolving the choice of action in each state, based on the MDP's execution so far.

Definition 2 (Strategy). *A* strategy *of an MDP* $\mathsf{M}=(S,\bar{s},A,P,R)$ *is a function* $\sigma : FPaths_{\mathsf{M}} \to Dist(A)$ *such that, for any* $\pi \in FPaths_{\mathsf{M}}$*, we have* $\sigma(\pi)(a)>0$ *only if* $a \in A(last(\pi))$*. Let* Σ_{M} *denote the set of all strategies of* $\mathcal{M}$*.*

A strategy is *memoryless* if its choices only depend on the current state, *finite-memory* if it suffices to switch between a finite set of modes and *deterministic* if it always selects an action with probability 1.

When M is under the control of a strategy σ, the resulting behaviour is captured by a probability measure Pr^{σ}_{M} over the infinite paths of M [30]. Furthermore, given a random variable $f : IPaths_{\mathsf{M}} \to \mathbb{R}$ over the infinite paths of M, using the probability measure Pr^{σ}_{M}, we can define the expected value of the variable f with respect to the strategy σ, denoted $\mathbb{E}^{\sigma}_{\mathsf{M}}(f)$.

2.2 Partially observable Markov decision processes

POMDPs extend MDPs by restricting the extent to which their current state can be observed, in particular by strategies that control them. In this paper (as in, e.g., [5,16]), we adopt the following notion of observability.

Definition 3 (POMDP). *A POMDP is a tuple* $\mathsf{M}=(S,\bar{s},A,P,R,\mathcal{O},obs)$ *where:*

- $(S,\bar{s},A,P,R)$ *is an MDP;*
- $\mathcal{O}$ *is a finite set of* observations*;*
- $obs : S \to \mathcal{O}$ *is a labelling of states with observations;*

such that, for any states $s,s' \in S$ *with* $obs(s)=obs(s')$*, their available actions must be identical, i.e.,* $A(s)=A(s')$*.*

The current state s of a POMDP cannot be directly determined, only the corresponding observation $obs(s) \in \mathcal{O}$. The requirement on available actions in Definition 3 follows from the fact that, if states have different actions available, then they are not observationally equivalent as the available actions are not hidden, and hence should not have the same observation.

More general notions of observations are sometime used, e.g., that depend also on the previous action taken or are probabilistic. However, as demonstrated by [15], given a POMDP with the most general notion of observations (both probabilistic and dependent on the previous action), we can construct an equivalent (polynomially larger) POMDP of the form given in Definition 3. In addition, our analysis of probabilistic verification case studies where partial observation is needed (see, e.g., Section 6) suggests that this simpler notion of observability will often suffice in practice. To ease presentation, we assume that the initial state is observable, i.e., there exists $\bar{o} \in \mathcal{O}$ such that $obs(s)=\bar{o}$ if and only if $s=\bar{s}$.

The notions of paths, strategies and probability measures given above for MDPs transfer directly to POMDPs. However, the set Σ_{M} of all strategies for a POMDP M only includes *observation-based strategies*.

Definition 4 (Observation-based strategy). *A* strategy *of a POMDP* $\mathsf{M} = (S, \bar{s}, A, P, R, \mathcal{O}, obs)$ *is a function* $\sigma : FPaths_{\mathsf{M}} \to Dist(A)$ *such that:*

- σ *is a strategy of the MDP* $(S, \bar{s}, A, P, R)$*;*
- *for any paths* $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots s_n$ *and* $\pi' = s'_0 \xrightarrow{a'_0} s'_1 \xrightarrow{a'_1} \cdots s'_n$ *satisfying* $obs(s_i) = obs(s'_i)$ *and* $a_i = a'_i$ *for all* i*, we have* $\sigma(\pi) = \sigma(\pi')$*.*

Let Σ_{M} *denote the set of all (observation-based) strategies of* M*.*

Key properties for MDPs and POMDPs are the probability of reaching a target and the expected reward cumulated until this occurs (where we assume that the expected value is infinite if there is a non-zero probability of the target *not* being reached). Let O denote the target (i.e., a set of states for an MDP and a set of observations for a POMDP). Under a specific strategy σ of an MDP or POMDP M, we denote these two properties by $Pr^{\sigma}_{\mathsf{M}}(\mathtt{F}\ O)$ and $\mathbb{E}^{\sigma}_{\mathsf{M}}(\mathtt{F}\ O)$, respectively.

Usually, we are interested in the *optimal* (minimum or maximum) values $Pr^{opt}_{\mathsf{M}}(\mathtt{F}\ O)$ and $\mathbb{E}^{opt}_{\mathsf{M}}(\mathtt{F}\ O)$, where $opt \in \{\min, \max\}$. For MDP or POMDP M:

$$Pr^{\min}_{\mathsf{M}}(\mathtt{F}\ O) \stackrel{\text{def}}{=} \inf_{\sigma \in \Sigma_{\mathsf{M}}} Pr^{\sigma}_{\mathsf{M}}(\mathtt{F}\ O) \qquad \mathbb{E}^{\min}_{\mathsf{M}}(\mathtt{F}\ O) \stackrel{\text{def}}{=} \inf_{\sigma \in \Sigma_{\mathsf{M}}} \mathbb{E}^{\sigma}_{\mathsf{M}}(\mathtt{F}\ O)$$
$$Pr^{\max}_{\mathsf{M}}(\mathtt{F}\ O) \stackrel{\text{def}}{=} \sup_{\sigma \in \Sigma_{\mathsf{M}}} Pr^{\sigma}_{\mathsf{M}}(\mathtt{F}\ O) \qquad \mathbb{E}^{\max}_{\mathsf{M}}(\mathtt{F}\ O) \stackrel{\text{def}}{=} \sup_{\sigma \in \Sigma_{\mathsf{M}}} \mathbb{E}^{\sigma}_{\mathsf{M}}(\mathtt{F}\ O)$$

Note that the class of strategies Σ_{M} analysed in the above is different depending on whether M is an MDP or POMDP (see Definitions 2 and 4, respectively). In the case of MDPs, deterministic and memoryless strategies achieve optimal values. This allows the use of efficient computational techniques such as *policy iteration*, which builds a sequence of strategies until an optimal one is reached, and *value iteration*, which computes increasingly precise approximations to the optimal probability or expected value (see for example [45]). However, in the case of POMDPs, this no longer holds. In fact, determining the optimal probabilities and expected rewards defined above is undecidable [38], making exact solution intractable. Instead, the optimal value can be approximated, for example via analysis of the *belief MDP*, whose construction we will discuss shortly.

Example 1. As an example POMDP, we consider a maze, originally introduced by McCallum [40]. The example concerns a robot being placed uniformly at random in a maze and then trying to find its way to a certain target location. The maze is presented in Figure 1 and comprises 11 locations labelled from '0' to '10'. There are four actions that the robot can perform in each location, corresponding to the four directions it can move: *north*, *east*, *south* and *west*. Performing such an action moves the robot one location in that direction (if moving in that direction means hitting a wall, the robot remains where it is). The robot cannot see its current location, but only what walls surround it. Therefore, for example, the locations labelled '5', '6' and '7' yield the same observation, since the robot can only observe that there are walls to the east

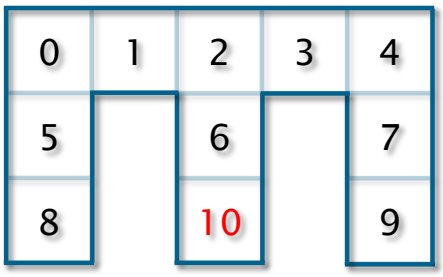

Fig. 1. McCallum's maze problem [40].

and west. The goal of the robot is to reach the target location labelled '10', and hence we associate a distinct observation with this location.

We find that the optimal (minimum) expected number of moves to reach the target is 4.3. If we instead consider a fully observable model (i.e., an MDP), then the optimal expected number of moves is 3.9. Considering a strategy of the POMDP that achieves the optimal value, if the robot initially observes that the only walls are on the east and west, then the strategy believes with equal probability that the robot is in one of the locations labelled '5', '6' and '7'. The strategy moves the robot *north* which allows it to learn which of these states the robot is actually in. More precisely, if the robot was in the location labelled '5', then, after moving north, it will observe walls to the north and west, if it was in the location '6' it will next observe only a wall to the north and, for the location labelled '7', next observe walls to the north and east.

Note that, if the strategy knew the robot was in the location labelled '6', the optimal move would be *south* as opposed to *north*. When the robot initially observes walls to the north and south, the strategy does not know if it is in the location labelled '1' or the one labelled '3'. Here the strategy can either choose *east* or *west*. When performing either action, the strategy will be able to learn the robot's position, while moving the robot closer to the target in one case and further away in the other. Once the strategy knows the robot's position, it can easily determine the optimal route for the robot to reach the target.

Beliefs. Given a POMDP M we can construct a corresponding *belief MDP* $\mathcal{B}(\mathsf{M})$: an equivalent (fully observable) MDP, whose (continuous) state space comprises *beliefs*, which are probability distributions over the state space of M. Intuitively, although we may not know which of several observationally-equivalent states we are currently in, we can determine the likelihood of being in each one, based on the probabilistic behaviour of M. The formal definition is given below, and we include details of the construction in Appendix A.

Definition 5 (Belief MDP). *Let* $\mathsf{M}=(S, \bar{s}, A, P, R, \mathcal{O}, obs)$ *be a POMDP. The* belief MDP *of* M *is given by* $\mathcal{B}(\mathsf{M})=(Dist(S), \delta_{\bar{s}}, A, P^{\mathcal{B}}, R^{\mathcal{B}})$ *where, for any beliefs*

$b, b' \in \mathit{Dist}(S)$ *and action* $a \in A$*:*

$$P^{\mathcal{B}}(b,a)(b') = \sum_{s\in S} b(s) \cdot \left(\sum_{o\in\mathcal{O}\wedge b^{a,o}=b'} \sum_{s'\in S\wedge obs(s')=o} P(s,a)(s')\right)$$
$$R^{\mathcal{B}}_S(b) = \sum_{s\in S} R_S(s)\cdot b(s)$$
$$R^{\mathcal{B}}_A(b,a) = \sum_{s\in S} R_A(s,a)\cdot b(s)$$

and $b^{a,o}$ *is the belief reached from* b *by performing action* a *and observing* o*, i.e.:*

$$b^{a,o}(s') = \begin{cases} \frac{\sum_{s\in S} P(s,a)(s')\cdot b(s)}{\sum_{s\in S} b(s)\cdot\left(\sum_{s''\in S\wedge obs(s'')=o} P(s,a)(s'')\right)} & \text{if } obs(s')=o \\ 0 & \text{otherwise.} \end{cases}$$

The optimal values for the probability and expected reward to reach a target in the belief MDP equal those for the POMDP, which is formally stated by the following proposition.

Proposition 1. *If* $\mathsf{M}=(S,\bar{s},A,P,R,\mathcal{O},obs)$ *is a POMDP and* $O\subseteq\mathcal{O}$ *a set of observations, then:*

$$Pr^{opt}_{\mathsf{M}}(\mathtt{F}\ O) = Pr^{opt}_{\mathcal{B}(\mathsf{M})}(\mathtt{F}\ T_O) \quad \text{and} \quad \mathbb{E}^{opt}_{\mathsf{M}}(\mathtt{F}\ O) = \mathbb{E}^{opt}_{\mathcal{B}(\mathsf{M})}(\mathtt{F}\ T_O)$$

where $T_O = \{b\in \mathit{Dist}(S) \mid \forall s\in S.\,(b(s)>0 \rightarrow obs(s)\in O)\}$ *and* $opt\in\{\min,\max\}$*.*

2.3 Parallel composition of POMDPs

To facilitate the modelling of complex systems, we introduce a notion of *parallel composition* for POMDPs, which allows us to define a system as set of interacting components. Our definition extends the standard definition for MDPs and probabilistic automata [47]. It is based on multi-way synchronisation over the same action by several components, as used in the process algebra CSP [46] and the PRISM model checker [31,44], but this can easily be generalised to incorporate more flexible definitions of synchronisation. We will use parallel composition of POMDPs for modelling the case studies that we present in Section 6.

Definition 6 (Parallel composition of POMDPs). *Consider any POMDPs* $\mathsf{M}_i=(S_i,\bar{s}_i,A_i,P_i,R_i,\mathcal{O}_i,obs_i)$*, for* $i=1,2$*. The parallel composition of* M_1 *and* M_2 *is the POMDP:*

$$\mathsf{M}_1 \| \mathsf{M}_2 = (S_1\times S_2, (\bar{s}_1,\bar{s}_2), A_1\cup A_2, R, \mathcal{O}_1\times\mathcal{O}_2, obs)$$

where, for any $s=(s_1,s_2)$ *and* $a\in A_1\cup A_2$*, we have:*

- *if* $a\in A_1\cap A_2$*, then* $a\in A(s_1,s_2)$ *if and only if* $a\in A(s_1)\cap A(s_2)$ *with*

$$P(s,a)(s') = P_1(s_1,a)(s'_1)\cdot P_2(s_2,a)(s'_2)$$

for all $s'=(s'_1,s'_2)\in S_1\times S_2$ *and* $R_A(s,a) = R_{A,1}(s_1,a) + R_{A,2}(s_2,a)$*;*

– *if* $a \in A_1 \backslash A_2$, *then* $a \in A(s_1, s_2)$ *if and only if* $a \in A(s_1)$ *with*

$$P(s,a)(s') = \begin{cases} P_1(s_1,a)(s_1') & \text{if } s_2{=}s_2' \\ 0 & \text{otherwise} \end{cases}$$

for all $s' = (s_1', s_2') \in S_1 \times S_2$ *and* $R_A(s,a) = R_{A,1}(s_1, a_1)$;
– *if* $a \in A_2 \backslash A_1$, *then* $a \in A(s_1, s_2)$ *if and only if* $a \in A(s_2)$ *with*

$$P(s,a)(s') = \begin{cases} P_2(s_2,a)(s_2') & \text{if } s_1{=}s_1' \\ 0 & \text{otherwise} \end{cases}$$

for all $s' = (s_1', s_2') \in S_1 \times S_2$ *and* $R_A(s,a) = R_{A,2}(s_2, a_2)$;
– $R_S(s) = R_{S,1}(s_1) + R_{S,2}(s_2)$;
– $obs(s) = (obs_1(s_1), obs_2(s_2))$.

As is standard in CSP-style parallel composition [46], an action which is in the action set of both components can only be performed when both components can perform it. Formally, using Definition 6, we see that, for any state $s{=}(s_1, s_2)$ of $\mathsf{M}_1 \| \mathsf{M}_2$, we have $A((s_1, s_2)) = (A(s_1) \cap A(s_2)) \cup (A(s_1) \backslash A_2) \cup (A(s_2) \backslash A_1)$. It therefore follows that, for any states s, s' of $\mathsf{M}_1 \| \mathsf{M}_2$ with $obs(s){=}obs(s')$, the available actions $A(s)$ and $A(s')$ are identical, thus satisfying the condition imposed on a POMDP's actions and observability in Definition 3.

In Definition 6 we have used addition to combine the reward values of the component POMDPs. However, depending on the system being modelled and its context, it may be more appropriate to combine the rewards in a different way, for example using multiplication or taking the maximum.

3 Verification and Strategy Synthesis for POMDPs

We now present our approach for verification and strategy synthesis for POMDPs.

3.1 Property specification

First, we define a temporal logic for the formal specification of quantitative properties of POMDPs. This is based on a subset (we omit temporal operator nesting) of the logic PCTL [24] and its reward-based extension in [21].

Definition 7 (POMDP property syntax). *The syntax of our temporal logic for POMDPs is given by the grammar:*

$$\begin{aligned} \phi &::= \mathtt{P}_{\bowtie p}[\psi] \mid \mathtt{R}_{\bowtie q}[\rho] \\ \alpha &::= \mathtt{true} \mid o \mid \neg\alpha \mid \alpha \wedge \alpha \\ \psi &::= \alpha \ \mathtt{U}^{\leqslant k} \ \alpha \mid \alpha \ \mathtt{U} \ \alpha \\ \rho &::= \mathtt{I}^{=k} \mid \mathtt{C}^{\leqslant k} \mid \mathtt{F} \ \alpha \end{aligned}$$

where o *is an observation,* $\bowtie \in \{\leqslant, <, \geqslant, >\}$, $p \in \mathbb{Q} \cap [0,1]$, $q \in \mathbb{Q}_{\geqslant 0}$ *and* $k \in \mathbb{N}$.

A POMDP property ϕ is an instance of either the probabilistic operator $\mathtt{P}_{\bowtie p}[\cdot]$ or the expected reward operator $\mathtt{R}_{\bowtie q}[\cdot]$. Intuitively, a state satisfies a formula $\mathtt{P}_{\bowtie p}[\psi]$ if the probability of the path formula ψ being satisfied is $\bowtie p$, and satisfies a formula $\mathtt{R}_{\bowtie q}[\rho]$ if the expected value of the reward formula ρ is $\bowtie q$.

For path formulae, we allow time-bounded ($\alpha\ \mathtt{U}^{\leqslant k}\ \alpha$) and unbounded ($\alpha\ \mathtt{U}\ \alpha$) until formulae, and adopt the usual equivalences such as $\mathtt{F}\ \alpha \equiv \mathtt{true}\ \mathtt{U}\ \alpha$ ("eventually α"). For reward formulae, we allow $\mathtt{I}^{=k}$ (state reward at k steps), $\mathtt{C}^{\leqslant k}$ (reward accumulated over the first k steps) and $\mathtt{F}\ \alpha$ (the reward accumulated until α becomes true). The propositional formulae (α) are Boolean combinations of observations of the POMDP.

We have omitted nesting of P and R operators in Definition 7 to allow consistent property specification for either verification or strategy synthesis problems (the latter is considerably more difficult in the context of nested formulae [6,10]).

Definition 8 (POMDP property semantics). *Let* $\mathsf{M}=(S,\bar{s},A,P,R,\mathcal{O},obs)$ *be a POMDP. We define satisfaction of a property* ϕ *from Definition 7 with respect to a strategy* $\sigma\in\Sigma_{\mathsf{M}}$ *as follows:*

$$\begin{array}{lcl}\mathsf{M},\sigma\models\mathtt{P}_{\bowtie p}[\,\psi\,] & \Leftrightarrow & Pr^{\sigma}_{\mathsf{M}}(\{\pi\in IPaths_{\mathsf{M}}\mid\pi\models\psi\})\bowtie p\\ \mathsf{M},\sigma\models\mathtt{R}_{\bowtie q}[\,\rho\,] & \Leftrightarrow & \mathbb{E}^{\sigma}_{\mathsf{M}}(rew(\rho))\bowtie q\end{array}$$

and, for any state $s\in S$ *and path* $\pi=s_0\xrightarrow{a_0}s_1\xrightarrow{a_1}\cdots\in IPaths_{\mathsf{M}}$*:*

$$\begin{array}{rcll} s\models\mathtt{true} & & \textit{always}\\ s\models o & \Leftrightarrow & o\in obs(s)\\ s\models\neg\alpha & \Leftrightarrow & s\not\models\alpha\\ s\models\alpha_1\wedge\alpha_2 & \Leftrightarrow & s\models\alpha_1\ \ \textit{and}\ \ s\models\alpha_2\end{array}$$

$$\begin{array}{rcl}\pi\models\alpha_1\ \mathtt{U}^{\leqslant k}\ \alpha_2 & \Leftrightarrow & \exists i\in\mathbb{N}.\,(\,i{\leqslant}k\wedge s_i\models\alpha_2\wedge\forall j{<}i.\,(s_j\models\alpha_1)\,)\\ \pi\models\alpha_1\ \mathtt{U}\ \alpha_2 & \Leftrightarrow & \exists i\in\mathbb{N}.\,(\,s_i\models\alpha_2\wedge\forall j{<}i.\,(s_j\models\alpha_1)\,)\end{array}$$

$$\begin{array}{rcll} rew(\mathtt{I}^{=k})(\pi) & = & R_S(s_k)\\ rew(\mathtt{C}^{\leqslant k})(\pi) & = & \sum_{j=0}^{k-1}\big(R_S(s_j)+R_A(s_j,a_j)\big)\\ rew(\mathtt{F}\ \alpha)(\pi) & = & \begin{cases}\infty & \textit{if}\ \forall j\in\mathbb{N}.\,s_j\not\models\alpha\\ \sum_{j=0}^{m_\alpha-1}\big(R_S(s_j)+R_A(s_j,a_j)\big) & \textit{otherwise}\end{cases}\end{array}$$

where $m_\alpha=\min\{j\mid s_j\models\alpha\}$.

3.2 Verification and strategy synthesis for POMDPs

Given a POMDP M and property ϕ, we are interested in solving the dual problems of *verification* and *strategy synthesis*.

Definition 9 (POMDP verification). *The* verification *problem for a POMDP* M *is: given a property* ϕ*, decide if* $\mathsf{M},\sigma\models\phi$ *holds for all strategies* $\sigma\in\Sigma_{\mathsf{M}}$.

Definition 10 (POMDP strategy synthesis). *The* strategy synthesis *problem for a POMDP* M *is: given a property* ϕ*, find, if it exists, a strategy* $\sigma \in \Sigma_{\mathsf{M}}$ *such that* $\mathsf{M}, \sigma \models \phi$*.*

The verification and strategy synthesis problems for a POMDP M and property ϕ can be solved similarly, by computing *optimal values* (i.e., minimum or maximum) for either path or reward objectives:

$$Pr^{\min}_{\mathsf{M}}(\psi) \stackrel{\text{def}}{=} \inf_{\sigma \in \Sigma_{\mathsf{M}}} Pr^{\sigma}_{\mathsf{M}}(\psi) \qquad \mathbb{E}^{\min}_{\mathsf{M}}(\rho) \stackrel{\text{def}}{=} \inf_{\sigma \in \Sigma_{\mathsf{M}}} \mathbb{E}^{\sigma}_{\mathsf{M}}(\rho)$$
$$Pr^{\max}_{\mathsf{M}}(\psi) \stackrel{\text{def}}{=} \sup_{\sigma \in \Sigma_{\mathsf{M}}} Pr^{\sigma}_{\mathsf{M}}(\psi) \qquad \mathbb{E}^{\max}_{\mathsf{M}}(\rho) \stackrel{\text{def}}{=} \sup_{\sigma \in \Sigma_{\mathsf{M}}} \mathbb{E}^{\sigma}_{\mathsf{M}}(\rho)$$

and, where required, also synthesising an *optimal strategy*. For example, verifying $\phi = \mathtt{P}_{\geqslant p}[\,\psi\,]$ requires computation of $Pr^{\min}_{\mathsf{M}}(\psi)$ since ϕ is satisfied by all strategies if and only if $Pr^{\min}_{\mathsf{M}}(\psi) \geqslant p$. Dually, consider synthesising a strategy for which $\phi' = \mathtt{P}_{<p}[\,\psi\,]$ holds. Such a strategy exists if and only if $Pr^{\min}_{\mathsf{M}}(\psi) < p$ and, if it does, we can use a strategy that achieves a value less than p. A common practice in probabilistic verification is to simply query the optimal values directly, by omitting the bounds $\bowtie p$ (for $\mathtt{P}$) or $\bowtie q$ (for $\mathtt{R}$) using *numerical* properties.

Definition 11 (Numerical POMDP property). *Let* ψ *and* ρ *be as specified in Definition 7. A* numerical POMDP property *is of the form* $\mathtt{P}_{\min=?}[\,\psi\,]$, $\mathtt{P}_{\max=?}[\,\psi\,]$, $\mathtt{R}_{\min=?}[\,\rho\,]$ *or* $\mathtt{R}_{\max=?}[\,\rho\,]$ *and yields the optimal value for the probability or reward formula.*

As mentioned earlier, when solving a POMDP, we may only be able to under- and over-approximate optimal values, which requires adapting the processes sketched above. For example, if we have determined lower and upper bounds $p^{\flat} \leqslant Pr^{\min}_{\mathsf{M}}(\psi) \leqslant p^{\sharp}$. We can verify that $\phi = \mathtt{P}_{\geqslant p}[\,\psi\,]$ holds for every strategy if $p^{\flat} \geqslant p$ or ascertain that ϕ does not hold if $p \geqslant p^{\sharp}$. But, if $p^{\flat} < p < p^{\sharp}$, we need to refine our approximation to produce tighter bounds. An analogous process can be followed for the case of strategy synthesis. The remainder of this section therefore focuses on how to (approximately) compute optimal values and strategies for POMDPs.

3.3 Numerical computation algorithms

Approximate numerical computation of either optimal probabilities $Pr^{opt}_{\mathsf{M}}(\psi)$ or expected reward values $\mathbb{E}^{opt}_{\mathsf{M}}(\rho)$ on a POMDP $\mathsf{M} = (S, \bar{s}, A, P, R, \mathcal{O}, obs)$ is performed with the sequence of steps given below, each of which is described in more detail subsequently. We compute both an under- and an over-approximation. For the former, we also generate a strategy which achieves this value.

(A) We modify POMDP M, reducing the problem to computing optimal values for a *probabilistic reachability* or *expected cumulative reachability* property;
(B) We build and solve a *finite abstraction* of the (infinite-state) belief MDP $\mathcal{B}(\mathsf{M})$ yielding an *over-approximation*;
(C) We synthesise and analyse a strategy for M, giving an *under-approximation*;

(D) If required, we *refine* the abstraction's precision and repeat (B) and (C).

(A) Property reduction. Checking $\mathtt{P}_{\bowtie p}[\psi]$ or $\mathtt{R}_{\bowtie q}[\rho]$ properties of the logic from Definition 7 can always be reduced to checking either a probabilistic reachability ($\mathtt{P}_{\bowtie p}[\mathtt{F}\ \alpha]$) or expected cumulative reachability reward ($\mathtt{R}_{\bowtie q}[\mathtt{F}\ \alpha]$) property on a modified POMDP $\mathsf{M}'=(S', \bar{s}', A', P', R', \mathcal{O}', obs')$. For the reduction in the case of MDPs, see for example [45].

(B) Over-approximation. We solve the modified POMDP M'. For simplicity, here and below, we describe the case of maximum reachability probabilities (the other cases are very similar) and thus need to compute $Pr^{\max}_{\mathsf{M}'}(\mathtt{F}\ O)$. We first compute an *over-approximation*, e.g., for maximum reachability probabilities $Pr^{\max}_{\mathsf{M}'}(\mathtt{F}\ O)$, we would find an *upper* bound. This is computed from an approximate solution to the belief MDP $\mathcal{B}(\mathsf{M}')$, whose construction we outlined in Section 2. This MDP has a continuous state space: the set of beliefs $Dist(S')$, where S' is the state space of M'.

To approximate its solution, we adopt the approach of [53,54] which computes values for a finite set of representative beliefs G whose convex hull is $Dist(S')$. Value iteration is applied to the belief MDP, using the computed values for beliefs in G and interpolating to get values for those not in G. The resulting values give the required upper bound. We use [53,54] as it works with *unbounded* (infinite horizon) and *undiscounted* properties. There are many other similar approaches [48], but these are formulated for discounted or finite-horizon properties.

The representative beliefs can be chosen in a variety of ways. We follow [37], where $G = \{\frac{1}{M} v \mid v \in \mathbb{N}^{|S'|} \wedge \sum_{i=1}^{|S'|} v(i){=}M\} \subseteq Dist(S')$, i.e. a uniform *grid* with *resolution* M. A benefit is that interpolation is very efficient, using a process called triangulation [19]. A downside is that the grid size is exponential in M. Efficiency might be improved with more complex grids that vary and adapt the resolution [48], but we found that [37] worked well enough for a prototype implementation.

(C) Under-approximation. Since it is preferable to have two-sided bounds, we also compute an *under-approximation*: here, a lower bound on $Pr^{\max}_{\mathsf{M}'}(\mathtt{F}\ O)$. To do so, we first synthesise a finite-memory strategy σ^* for M' (which is often a required output anyway). The choices of this strategy are built by stepping through the belief MDP and, for the current belief, choosing an action that achieves the values returned by value iteration in (B) above – see for example [48]. We then compute, by building and solving the finite discrete-time Markov chain induced by M' and σ^*, the value $Pr^{\sigma^*}_{\mathsf{M}'}(\mathtt{F}\ O)$ which is a lower bound for $Pr^{\max}_{\mathsf{M}'}(\mathtt{F}\ O)$.

(D) Refinement. Finally, when the computed approximations do not suffice to verify the required property (or, for strategy synthesis, σ^* does not satisfy the property), we *refine*, by increasing the grid resolution M and repeating steps (B) and (C). We note that no a priori bound can be given on the error between the generated under- and over-approximations (recall that the basic problem is

undecidable). Furthermore, just incrementing the resolution is not guaranteed to yield tighter bounds and in fact can yield worse bounds.

However, the abstraction approach that we use [53, Chap. 7], does provide an *asymptotic* guarantee on convergence. More precisely, convergence is shown for the case of expected total cumulative reward over models with non-negative rewards under the assumption that the cumulative reward is always finite. The case of probabilistic reachability can easily be reduced to the case of cumulative reward by assigning a one-off reward of 1 once the target is reached. For probabilistic reachability, finiteness of the cumulated reward is immediate. For expected cumulative reachability, reward finiteness is achieved by performing qualitative reachability analysis to remove states with infinite expected reward, i.e. the states that do not reach the target with probability 1. This is the standard approach for verifying MDPs against expected reachability properties [21] and is decidable for POMDPs [5].

Example 2. We return to the maze example from Example 1 and Figure 1. We can query the minimum expected number of steps to reach the target using the property $\mathtt{R}_{\min=?}[\,\mathtt{F}\ o_{target}\,]$, where o_{target} is the distinct observation corresponding to the target location labelled '10'. Following the approach described above, we obtain a precise answer (the bounds are $[4.300, 4.300]$) for grid resolution $M{=}2$ (for which the number of points in the grid is 19) and are able to synthesise the optimal strategy described in Example 1.

We now increase the size of the maze by adding an additional location to the southern end of each of the three north-south alignments of locations (i.e., to the locations labelled '8', '9' and '10') and keep the target as the southern most location of the middle such alignment. The resulting POMDP has 14 states and the same observation set as the original POMDP. Again considering the optimal expected number of steps to reach the target, we obtain the following results as the grid resolution is refined during the analysis:

- $M{=}2$ yields 34 grid points and the bounds $[4.3846, \infty]$;
- $M{=}3$ yields 74 grid points and the bounds $[4.8718, 5.3077]$;
- $M{=}4$ yields 150 grid points and the bounds $[4.8846, 5.3077]$;
- $M{=}5$ yields 283 grid points and the bounds $[5.0708, 5.3077]$;
- $M{=}6$ yields 501 grid points and the bounds $[5.3077, 5.3077]$.

The ∞ value for the case when $M{=}2$ follows from the fact that the synthesised strategy does not reach the target with probability 1, and hence the expected reward for this strategy is infinite (see Definition 8). As can be seen, the under-approximation (the upper bound, here), obtained from the value of the synthesised strategy in step (C), yields the optimal value almost immediately, while the over-approximation (the lower bound), obtained from the approximate solution to the belief MDP in step (B), takes more time to converge to the optimal value.

The synthesised optimal strategy is essentially the same as the one for the maze of Figure 1. For example, if the robot observes only walls on the east and west sides, then the strategy chooses to move the robot north until it reaches a

location labelled either '0', '2' or '4'. Then it knows where the robot is and the strategy can easily determine an optimal route to the target.

4 Partially Observable Probabilistic Timed Automata

In this section, we define *partially observable probabilistic timed automata* (POPTAs), which generalise the existing model of probabilistic timed automata (PTAs) with the notion of partial observability from POMDPs explained in Section 2. We define the syntax of a POPTA, explain some syntactic restrictions that we impose and formally define the semantics, which is given by a POMDP parameterised by a time domain $\mathbb{T}$. We also present a notion of parallel composition for POPTAs and give several illustrative examples of the model. The section begins with some background on the simpler model of PTAs and the notions used to define them. For more detailed tutorial material on this topic, we refer the interested reader to [42].

4.1 Time, clocks and clock constraints

Let $\mathbb{T} \in \{\mathbb{R}, \mathbb{N}\}$ be the time domain of either the non-negative reals or naturals. As in classic timed automata [2], we model real-time behaviour using non-negative, $\mathbb{T}$-valued variables called *clocks*, whose values increase at the same rate as real time. Assuming a finite set of clocks $\mathcal{X}$, a *clock valuation* v is a function $v : \mathcal{X} \to \mathbb{T}$ and we write $\mathbb{T}^{\mathcal{X}}$ for the set of all clock valuations over the time domain $\mathbb{T}$. Clock valuations obtained from v by incrementing all clocks by a delay $t \in \mathbb{T}$ and by resetting a set $X \subseteq \mathcal{X}$ of clocks to zero are denoted $v{+}t$ and $v[X{:=}0]$, respectively, and we write $\mathbf{0}$ if all clocks take the value 0. A (closed, diagonal-free) *clock constraint* ζ is either a conjunction of inequalities of the form $x \leqslant c$ or $x \geqslant c$, where $x \in \mathcal{X}$ and $c \in \mathbb{N}$, or `true`. We write $v \models \zeta$ if clock valuation v satisfies clock constraint ζ and use $CC(\mathcal{X})$ for the set of all clock constraints over $\mathcal{X}$.

4.2 Syntax of POPTAs

To explain the syntax of POPTAs, we first consider the simpler model of PTAs and then show how it extends to POPTAs.

Definition 12 (PTA syntax). *A* probabilistic timed automaton (PTA) *is a tuple* $\mathsf{P}{=}(L, \bar{l}, \mathcal{X}, A, \mathit{inv}, \mathit{enab}, \mathit{prob}, r)$ *where:*

- L *is a finite set of* locations *and* $\bar{l} \in L$ *is an* initial location;
- $\mathcal{X}$ *is a finite set of* clocks;
- A *is a finite set of* actions;
- $\mathit{inv} : L \to CC(\mathcal{X})$ *is an* invariant condition;
- $\mathit{enab} : L{\times}A \to CC(\mathcal{X})$ *is an* enabling condition;
- $\mathit{prob} : L{\times}A \to \mathit{Dist}(2^{\mathcal{X}}{\times}L)$ *is a* probabilistic transition function;

– $r{=}(r_L, r_A)$ *is a* reward structure *where* $r_L : L \to \mathbb{R}$ *is a* location reward function *and* $r_A : L{\times}A \to \mathbb{R}$ *is an* action reward function.

A state of a PTA is a pair (l, v) of location $l \in L$ and clock valuation $v \in \mathbb{T}^{\mathcal{X}}$. Time $t \in \mathbb{T}$ can elapse in the state only if the invariant $inv(l)$ remains continuously satisfied while time passes and the new state is then $(l, v{+}t)$, which we denote $(l, v){+}t$. An action a is enabled in the state if v satisfies $enab(l, a)$ and, if it is taken, then the PTA moves to location l' and resets the clocks $X \subseteq \mathcal{X}$ with probability $prob(l, a)(X, l')$. PTAs have two kinds of rewards:

– location rewards, which are accumulated at rate $r_L(l)$ while in location l;
– action rewards $r_A(l, a)$, which are accumulated when taking action a in location l.

PTAs equipped with such reward structures are a probabilistic extension of linearly-priced timed automata [7], also called weighted timed automata [7,4].

We now introduce POPTAs which extend PTAs by the inclusion of an observation function over locations.

Definition 13 (POPTA syntax). *A* partially observable PTA (POPTA) *is a tuple* $\mathsf{P} = (L, \bar{l}, \mathcal{X}, A, inv, enab, prob, r, \mathcal{O}_L, obs_L)$ *where:*

– $(L, \bar{l}, \mathcal{X}, A, inv, enab, prob, r)$ *is a* PTA;
– $\mathcal{O}_L$ *is a finite set of* observations;
– $obs_L : L \to \mathcal{O}_L$ *is a* location observation function.

For any locations $l, l' \in L$ *with* $obs_L(l){=}obs_L(l')$*, we require that* $inv(l){=}inv(l')$ *and* $enab(l, a){=}enab(l', a)$ *for all* $a \in A$.

The final condition of Definition 13 ensures the semantics of a POPTA yields a valid POMDP: recall states with the same observation are required to have identical available actions. Like for POMDPs, for simplicity, we also assume that the initial location is observable, i.e., there exists $\bar{o} \in \mathcal{O}_L$ such that $obs_L(l){=}\bar{o}$ if and only if $l{=}\bar{l}$.

The observability of clocks. The notion of observability for POPTAs is similar to the one for POMDPs, but applied to locations. Clocks, on the other hand, are always observable. The requirement that the same choices must be available in any observationally-equivalent states, implies the same delays must be available in observationally-equivalent states, and so unobservable clocks could not feature in invariant or enabling conditions. The inclusion of unobservable clocks would therefore necessitate modelling the system as a game with the elapse of time being under the control of a second (environment) player. The underlying semantic model would then be a partially observable stochastic game (POSG), rather than a POMDP. However, unlike POMDPs, limited progress has been made on efficient computational techniques for this model (belief space based techniques, for example, do not apply in general [17]). Even in the simpler case of non-probabilistic timed games, allowing unobservable clocks requires algorithmic analysis to restrict the class of strategies considered [12,20].

Encouragingly, however, we will later show in Section 6 that POPTAs with observable clocks were always sufficient for our modelling and analysis.

Restrictions on POPTAs. At this point, we need to highlight a few syntactic restrictions on the POPTAs treated in this paper.

Assumption 1 *For any POPTA* P*, all clock constraints appearing in* P*, i.e., in its invariants and enabling conditions, are required to be* closed *(no strict inequalities, e.g.* $x<c$*) and* diagonal-free *(no comparisons of clocks, e.g.,* $x<y$*).*

Assumption 2 *For any POPTA* $\mathsf{P}=(L,\bar{l},\mathcal{X},A,inv,enab,prob,r,\mathcal{O}_L,obs_L)$*, resets can only be applied to clocks that are non-zero. More precisely, for any* $l,l'\in L$*,* $a\in A$ *and* $X\subseteq\mathcal{X}$*, if* $prob(l,a)(X,l')>0$ *then for any* $v\in\mathbb{R}^{\mathcal{X}}$ *such that* $v(x)=0$ *for some* $x\in X$ *we have either* $v\not\models inv(l)$ *or* $v\not\models enab(l,a)$*.*

Assumption 1 is a standard restriction when using the digital clocks discretisation [32] which we work with in this paper. The reasoning behind Assumption 2 is demonstrated in Example 4. Checking both assumptions can easily be done syntactically – see Section 5.

4.3 Semantics of POPTAs

We now formally define the semantics of a POPTA P, which is given in terms of a POMDP. This extends the standard semantics of a PTA [32] with the same notion of observability we gave in Section 2 for POMDPs. The semantics, $[\![\mathsf{P}]\!]_{\mathbb{T}}$, is parameterised by a *time domain* $\mathbb{T}$, giving the possible values taken by clocks. Before giving the semantics for POPTAs we consider the simpler case of PTAs.

Definition 14 (PTA semantics). *Let* $\mathsf{P}=(L,\bar{l},\mathcal{X},A,inv,enab,prob,r)$ *be a probabilistic timed automaton. The* semantics of P with respect to the time domain $\mathbb{T}$ *is the MDP* $[\![\mathsf{P}]\!]_{\mathbb{T}}=(S,\bar{s},A\cup\mathbb{T},P,R)$ *such that:*

- $S=\{(l,v)\in L\times\mathbb{T}^{\mathcal{X}}\mid v\models inv(l)\}$ *and* $\bar{s}=(\bar{l},\mathbf{0})$*;*
- *for any* $(l,v)\in S$ *and* $a\in A\cup\mathbb{T}$*, we have* $P((l,v),a)=\mu$ *if and only if one of the following conditions hold:*
  - *(time transitions)* $a\in\mathbb{T}$*,* $\mu=\delta_{(l,v+a)}$ *and* $v+a\models inv(l)$ *for all* $0\leqslant t'\leqslant a$*;*
  - *(action transition)* $a\in A$*,* $v\models enab(l,a)$ *and for* $(l',v')\in S$*:*

$$\mu(l',v')=\textstyle\sum_{X\subseteq\mathcal{X}\wedge v'=v[X:=0]}prob(l,a)(X,l')$$

- *for any* $(l,v)\in S$ *and* $a\in A\cup\mathbb{T}$*:*

$$R_S(l,v)=r_L(l)$$
$$R_A((l,v),a)=\begin{cases}r_L(l)\cdot a & \text{if } a\in\mathbb{T}\\ r_A(l,a) & \text{if } a\in A.\end{cases}$$

For the standard (dense-time) semantics of a PTA, we take $\mathbb{T}=\mathbb{R}$. Since the semantics of a PTA is an infinite-state model, for algorithmic analysis, we first need to construct a *finite* representation. One approach for this is to use the *digital clocks* semantics for PTAs [32] which generalises the approach for timed automata [25]. This approach discretises a PTA model by transforming its real-valued clocks to clocks taking values from a bounded set of integers.

Before we give the definition we require the following notation. For any clock x of a PTA, let $\mathbf{k}_x$ denote the greatest constant to which x is compared in the clock constraints of the PTA. If the value of x exceeds $\mathbf{k}_x$, its exact value will not affect the satisfaction of any invariants or enabling conditions, and thus not affect the behaviour of the PTA.

Definition 15 (Digital clocks semantics). *The digital clocks semantics of a PTA* P*, written* $[\![\mathsf{P}]\!]_{\mathbb{N}}$*, can be obtained from Definition 14, taking* $\mathbb{T}$ *to be* $\mathbb{N}$ *and redefining the operation* $v{+}t$ *such that for any clock valuation* $v \in \mathbb{N}^{\mathcal{X}}$*, delay* $t \in \mathbb{N}$ *and clock* $x \in \mathcal{X}$ *we have* $(v{+}t)(x) = \min\{v(x){+}t, \mathbf{k}_x{+}1\}$.

We now extend Definition 14 and define the semantics of a POPTA.

Definition 16 (POPTA semantics). *Let* $\mathsf{P}{=}(L, \bar{l}, \mathcal{X}, A, inv, enab, prob, r, \mathcal{O}_L, obs_L)$ *be a POPTA. The* semantics *of* P*, with respect to the time domain* $\mathbb{T}$*, is the POMDP* $[\![\mathsf{P}]\!]_{\mathbb{T}}{=}(S, \bar{s}, A \cup \mathbb{T}, P, R, \mathcal{O}_L{\times}\mathbb{T}^{\mathcal{X}}, obs)$ *such that:*

- $(S, \bar{s}, A \cup \mathbb{T}, P, R)$ *is the semantics of the PTA* $(L, \bar{l}, \mathcal{X}, A, inv, enab, prob, r)$*;*
- *for any* $(l, v) \in S$*, we have* $obs(l, v) = (obs_L(l), v)$.

As for PTAs, we consider both the 'standard' dense-time semantics and the digital clocks semantics of a POPTA, by taking $\mathbb{T} = \mathbb{R}$ and $\mathbb{T} = \mathbb{N}$ respectively. The fact that the digital clocks semantics of a POPTA is finite, and the dense-time semantics is generally uncountable, can be derived from the definitions. Under the restrictions on POPTAs described above, as we will demonstrate in Section 5, the digital semantics of a POPTA preserves the key properties required in this paper, namely optimal probabilities and expected cumulative rewards for reaching a specified observation set.

Time divergence. As for PTAs and classic timed automata we restrict attention to *time-divergent* (or non-Zeno) strategies. Essentially this means that we restrict attention to strategies under which there are no unrealisable executions in which time does not advance beyond a certain point. There are syntactic and compositional conditions for PTAs for ensuring all strategies are time-divergent by construction [42]. These are derived from analogous results on timed automata [50,51] and carry over to our setting of POPTAs.

4.4 Parallel composition of POPTAs

As we did for POMDPs in Section 2, to aid the modelling of complex system, we now define a notion of parallel composition for POPTAs.

Definition 17 (Parallel composition of POPTAs). *Consider any POPTAs* $\mathsf{P}_i{=}(L_i, \bar{l}_i, \mathcal{X}_i, A_i, inv_i, enab_i, prob_i, r_i, \mathcal{O}_{L,i}, obs_{L,i})$ *for* $i \in \{1,2\}$ *such that* $\mathcal{X}_1 \cap \mathcal{X}_2 = \varnothing$. *The parallel composition of* M_1 *and* M_2, *denoted* $\mathsf{P}_1 \| \mathsf{P}_2$ *is the POPTA:*

$$\mathsf{P}_1 \| \mathsf{P}_2 = (L_1 \times L_2, (\bar{l}_1, \bar{l}_2), \mathcal{X}_1 \cup \mathcal{X}_2, A_1 \cup A_2, inv, enab, prob, r, \mathcal{O}_{L,1} \times \mathcal{O}_{L,2}, obs_L)$$

where for any $l{=}(l_1,l_2)$, $l'{=}(l'_1,l'_2) \in L_1 \times L_2$, $a \in A_1 \cap A_2$, $a_1 \in A_1 \setminus A_2$, $a_2 \in A_2 \setminus A_1$ *and* $X \subseteq \mathcal{X}_1 \cup \mathcal{X}_2$:

$$\begin{aligned}
inv(l) &= inv_1(l_1) \wedge inv_2(l_2) \\
enab(l,a) &= enab_1(l_1,a) \wedge enab_2(l_2,a) \\
enab(l,a_1) &= enab_1(l_1,a_1) \\
enab(l,a_2) &= enab_2(l_2,a_2) \\
prob(l,a)(X,l') &= prob_1(l_1,a)(X \cap \mathcal{X}_1, l'_1) \cdot prob_2(l_2,a)(X \cap \mathcal{X}_2, l'_2) \\
prob(l,a_1)(X,l') &= \begin{cases} prob_1(l_1,a_1)(X,l'_1) & \text{if } l_2 = l'_2 \text{ and } X \subseteq \mathcal{X}_1 \\ 0 & \text{otherwise} \end{cases} \\
prob(l,a_2)(X,l') &= \begin{cases} prob_2(l_2,a_2)(X,l'_2) & \text{if } l_1 = l'_1 \text{ and } X \subseteq \mathcal{X}_2 \\ 0 & \text{otherwise} \end{cases} \\
r_A(l,a) &= r_{A,1}(l_1,a) + r_{A,2}(l_2,a) \\
r_A(l,a_1) &= r_{A,1}(l_1,a_1) \\
r_A(l,a_2) &= r_{A,2}(l_2,a_2) \\
r_L(l) &= r_{L,1}(l_1) + r_{L,2}(l_2) \\
obs_L(l) &= (obs_{L,1}(l_1), obs_{L,2}(l_2)) \,.
\end{aligned}$$

For POPTAs, it follows from Definitions 17 and 13 that, for any locations l, l' of $\mathsf{P}_1 \| \mathsf{P}_2$ such that $obs_L(l){=}obs_L(l')$ and action a of $\mathsf{P}_1 \| \mathsf{P}_2$ we have $inv(l){=}inv(l')$ and $enab(l,a){=}enab(l',a)$. In addition the following lemma holds.

Lemma 1. *If* P_1 *and* P_2 *are POPTAs satisfying Assumptions 1 and 2, then* $\mathsf{P}_1 \| \mathsf{P}_2$ *satisfies Assumptions 1 and 2.*

Proof. Consider any POPTAs P_1 and P_2 which satisfy Assumptions 1 and 2. Since the conjunction of closed and diagonal-free clock constraints are closed and diagonal-free, it follows that $\mathsf{P}_1 \| \mathsf{P}_2$ satisfies Assumption 1.

For Assumption 2, consider any locations $l{=}(l_1,l_2)$ and $l'{=}(l'_1,l'_2)$, action a, set of clocks X and clock valuation v of $\mathsf{P}_1 \| \mathsf{P}_2$ such that $prob(l,a)(X,l'){>}0$ and $v(x){=}0$ for some clock $x \in X$. We have the following cases to consider.

- If $a \in A_1 \cap A_2$, then since $X \subseteq \mathcal{X}_1 \cup \mathcal{X}_2$ either $x \in \mathcal{X}_1$ or $x \in \mathcal{X}_2$. When $x \in \mathcal{X}_1$, since P_1 satisfies Assumption 2, it follows that $v \not\models inv_1(l_1)$ or $v \not\models enab_1(l_1,a)$. On the other hand, when $x \in \mathcal{X}_2$, since P_2 satisfies Assumption 2, it follows that $v \not\models inv_2(l_2)$ or $v \not\models enab_2(l_2,a)$. In either case, if follows from Definition 17 that $v \not\models inv(l)$ or $v \not\models enab(l,a)$.
- If $a \in A_1$, then by Definition 17 and since $prob(l,a)(X,l'){>}0$ we have $X \subseteq \mathcal{X}_1$ and $prob(l_1,a)(X,l'_1){>}0$. Therefore $x \in \mathcal{X}_1$ using the fact that P_1 satisfies Assumption 2 it follows that $v \not\models inv_1(l_1)$ or $v \not\models enab_1(l_1,a)$. Again using Definition 17 it follows that $v \not\models inv(l)$ or $v \not\models enab(l,a)$.

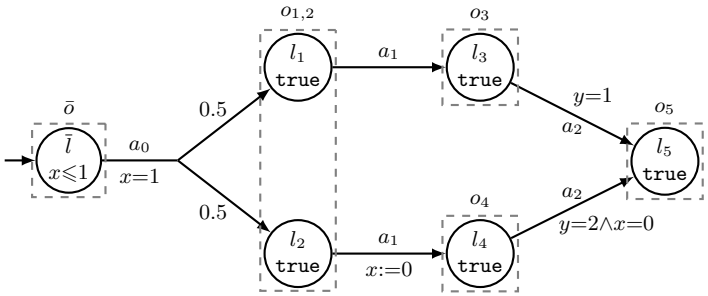

Fig. 2. Example of a partially observable PTA (see Example 3).

– If $a \in A_2$, then using similar arguments to the case above and the fact P_2 satisfies Assumption 2 we have $v \not\models \mathit{inv}(l)$ or $v \not\models \mathit{enab}(l, a)$.

Since these are all the cases to consider, it follows that $\mathsf{P}_1 \| \mathsf{P}_2$ satisfies Assumption 2 as required. □

Similarly to POMDPs (see Section 2), the reward values of the component POPTAs can be combined using alternative arithmetic operators depending on the system under study. As for PTAs [32], the semantics of the parallel composition of two POPTAs corresponds to the parallel composition of their individual semantic POMDPs using Definition 6. Formally, for POPTAs $\mathsf{P}_1, \mathsf{P}_2$ and time domain $\mathbb{T}$, we have that $[\![\mathsf{P}_1 \| \mathsf{P}_2]\!]_{\mathbb{T}} = [\![\mathsf{P}_1]\!]_{\mathbb{T}} \| [\![\mathsf{P}_2]\!]_{\mathbb{T}}$.

Additional modelling constructs to aid higher level modelling for PTAs also carry over to the case of POPTAs. These include discrete variables, urgent and committed locations and urgent actions. For further details, see [42].

4.5 Example POPTAs

Finally in this section, we present two example POPTAs. The second of these demonstrates why we have imposed Assumption 2 on POPTAs when using the digital clocks semantics.

Example 3. Consider the POPTA in Figure 2 with clocks x, y. Locations are grouped according to their observations, and we omit enabling conditions equal to `true`. We aim to maximise the probability of eventually observing o_5. If the locations were fully observable, i.e. the model was a PTA, we would leave the initial location $\bar{l}$ when $x{=}y{=}1$ and then, depending on whether the random choice resulted in a transition to location l_1 or l_2, wait 0 or 1 time units, respectively, before leaving the location. This would allow us to move immediately from the locations l_3 or l_4 to the location l_5, meaning we eventually observe o_5 with probability 1. However, in the POPTA, we need to make the same choice in l_1 and l_2 since they yield the same observation. As a result, at most one of the transitions leaving locations l_3 and l_4 is enabled when reaching these locations (the transition from l_3 will be enabled if we wait 0 time units before leaving both

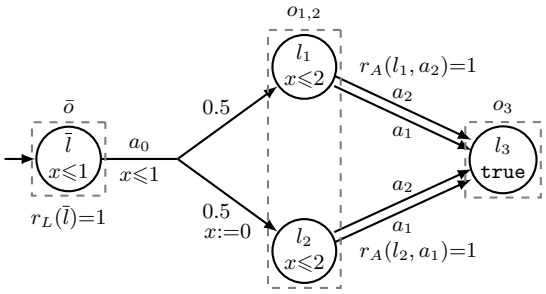

Fig. 3. Example POPTA for only resetting non-zero clocks (see Example 4).

l_1 and l_2, while the transition from l_4 will be enabled if we wait 1 time units before leaving both l_1 and l_2), and hence the maximum probability of eventually observing o_5 is 0.5.

Example 4. The POPTA P in Figure 3 demonstrates why our digital clocks approach (Theorem 1) is restricted to POPTAs which reset only non-zero clocks. We aim to minimise the expected reward accumulated before observing o_3 (the non-zero reward values are shown in Figure 3). If the model was a PTA and locations were fully observable, the minimum reward would be 0, achieved by leaving the initial location $\bar{l}$ immediately and then choosing a_1 in location l_1 and a_2 in location l_2. However, in the POPTA model, if we leave $\bar{l}$ immediately, the locations l_1 and l_2 are indistinguishable (we observe $(o_{1,2}, (0))$ when arriving in either), so we must choose the same action in these locations. Since we must leave the locations l_1 and l_2 when the clock x reaches the value 2, it follows that, when leaving the initial location immediately, the expected reward equals 0.5.

Now consider the strategy that waits $\varepsilon \in (0, 1)$ before leaving the initial location $\bar{l}$, accumulating a reward of ε. Clearly, since $\varepsilon \in \mathbb{R} \setminus \mathbb{N}$, this is possible only in the dense-time semantics. We then observe either $(o_{1,2}, (\varepsilon))$ when entering the location l_1, or $(o_{1,2}, (0))$ when entering the location l_2. Thus, observing whether the clock x was reset, allows a strategy to determine if the location reached is l_1 or l_2, and hence which of the actions a_1 or a_2 needs to be taken to observe o_3 without accumulating any additional reward. This yields a strategy that accumulates a total reward of ε before observing o_3. Now, since ε can be arbitrarily small, it follows that the minimum (infimum) expected reward for $[\![\mathsf{P}]\!]_{\mathbb{R}}$ is 0. On the other hand, for the digital clocks semantics, we can only choose a delay of 0 or 1 before leaving the initial location $\bar{l}$. In the former case, the expected reward is 0.5, as described above; for the latter case, we can again distinguish which of the locations l_1 or l_2 was reached by observing whether the clock x was reset. Hence, we can choose either a_1 or a_2 such that no further reward is accumulated, yielding a total expected reward of 1. Hence the minimum expected reward for $[\![\mathsf{P}]\!]_{\mathbb{N}}$ is 0.5, as opposed to 0 for $[\![\mathsf{P}]\!]_{\mathbb{R}}$.

5 Verification and Strategy Synthesis for POPTAs

We now present our approach for verification and strategy synthesis for POPTAs using the digital clock semantics given in the previous section.

5.1 Property specification

Quantitative properties of POPTAs are specified using the following logic.

Definition 18 (POPTA property syntax). *The syntax of our logic for POPTAs is given by the grammar:*

$$\begin{aligned}\phi &::= \mathtt{P}_{\bowtie p}[\psi] \mid \mathtt{R}_{\bowtie q}[\rho]\\ \alpha &::= \mathtt{true} \mid \zeta \mid o \mid \neg\alpha \mid \alpha \wedge \alpha\\ \psi &::= \alpha\ \mathtt{U}^{\leqslant k}\ \alpha \mid \alpha\ \mathtt{U}\ \alpha\\ \rho &::= \mathtt{I}^{=k} \mid \mathtt{C}^{\leqslant k} \mid \mathtt{F}\ \alpha\end{aligned}$$

where ζ is a clock constraint, o is an observation, $\bowtie \in \{\leqslant, <, \geqslant, >\}$, $p \in \mathbb{Q}\cap[0,1]$, $q \in \mathbb{Q}_{\geqslant 0}$ and $k \in \mathbb{N}$.

This property specification language is similar to the one we proposed earlier for POMDPs (see Definition 7), but we allow clock constraints to be included in propositional formulae. However, as for PTAs [42], the bound k in path formulae ($\alpha\ \mathtt{U}^{\leqslant k}\ \alpha$) and reward formulae ($\mathtt{I}^{=k}$ and $\mathtt{C}^{\leqslant k}$) corresponds to a time bound, as opposed to a bound on the number of discrete steps.

In the case of POPTAs, omitting the nesting of P and R operators is further motivated by the fact that the digital clocks approach is not applicable to nested properties (see [32] for details). Before we give the property semantics for POPTAs, we define the duration and position of a path in a POPTA.

Definition 19 (Duration of a POPTA path). *For a POPTA* P*, time domain* $\mathbb{T}$*, path* $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \in IPaths_{[\![\mathsf{P}]\!]_{\mathbb{T}}}$ *and* $i \in \mathbb{N}$*, the* duration *of* π *up to the* $(i{+}1)$*th state is given by:*

$$dur_\pi(i) = \textstyle\sum_{0\leqslant j<i \wedge a_j \in \mathbb{T}} a_j\,.$$

Definition 20 (Position of a POPTA path). *For a POPTA* P*, time domain* $\mathbb{T}$ *and path* $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \in IPaths_{[\![\mathsf{P}]\!]_{\mathbb{T}}}$*, a* position *of* π *is a pair* $(i,t) \in \mathbb{N}\times\mathbb{T}$ *such that* $t \leqslant dur_\pi(i{+}1){-}dur_\pi(i)$*. We say that position* (j,t') precedes *position* (i,t)*, written* $(j,t') \prec (i,t)$*, if* $j{<}i$ *or* $j{=}i$ *and* $t'{<}t$*.*

Definition 21 (POPTA property semantics). *Let* P *be a POPTA and* $\mathbb{T}$ *a time domain. We define satisfaction of a property* ϕ *from Definition 18 with respect to a strategy* $\sigma \in \Sigma_{[\![\mathsf{P}]\!]_{\mathbb{T}}}$ *as follows:*

$$\begin{aligned}[\![\mathsf{P}]\!]_{\mathbb{T}}, \sigma \models \mathtt{P}_{\bowtie p}[\,\psi\,] &\;\Leftrightarrow\; Pr^{\sigma}_{[\![\mathsf{P}]\!]_{\mathbb{T}}}(\{\pi \in IPaths_{[\![\mathsf{P}]\!]_{\mathbb{T}}} \mid \pi \models \psi\}) \bowtie p\\ [\![\mathsf{P}]\!]_{\mathbb{T}}, \sigma \models \mathtt{R}_{\bowtie q}[\,\rho\,] &\;\Leftrightarrow\; \mathbb{E}^{\sigma}_{[\![\mathsf{P}]\!]_{\mathbb{T}}}(rew(\rho)) \bowtie q\end{aligned}$$

and for any state $(l,v) \in L \times \mathbb{T}^{\mathcal{X}}$ *and path* $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \in IPaths_{[\![\mathsf{P}]\!]_{\mathbb{T}}}$*:*

$$
\begin{array}{rcl}
(l,v) \models \mathtt{true} & & \textit{always} \\
(l,v) \models o & \Leftrightarrow & o \in obs_L(l) \\
(l,v) \models \zeta & \Leftrightarrow & v \models \zeta \\
(l,v) \models \neg\alpha & \Leftrightarrow & (l,v) \not\models \alpha \\
(l,v) \models \alpha_1 \wedge \alpha_2 & \Leftrightarrow & (l,v) \models \alpha_1 \;\; \textit{and} \;\; (l,v) \models \alpha_2
\end{array}
$$

$$
\begin{array}{rcl}
\pi \models \alpha_1 \;\mathtt{U}^{\leqslant k}\; \alpha_2 & \Leftrightarrow & \textit{there exists a position } (i,t) \textit{ of } \pi \textit{ such that } \pi(i){+}t \models \alpha_2, \\
 & & dur_\pi(i){+}t \leqslant k \textit{ and } \pi(j){+}t' \models \alpha_1 \vee \alpha_2 \\
 & & \textit{for all positions } (j,t') \prec (i,t) \textit{ of } \pi \\
\pi \models \alpha_1 \;\mathtt{U}\; \alpha_2 & \Leftrightarrow & \textit{there exists a position } (i,t) \textit{ of } \pi \textit{ such that } \pi(i){+}t \models \alpha_2 \\
 & & \textit{and } \pi(j){+}t' \models \alpha_1 \vee \alpha_2 \textit{ for all positions } (j,t') \prec (i,t) \textit{ of } \pi
\end{array}
$$

$$
\begin{array}{rcl}
rew(\mathtt{I}^{=k})(\pi) & = & R_S(s_{m_k}) \\
rew(\mathtt{C}^{\leqslant k})(\pi) & = & \sum_{j=0}^{m_k-1} R_A(s_j,a_j) + R_S(s_{m_k}){\cdot}(k - dur_\pi(m_k)) \\
rew(\mathtt{F}\;\alpha)(\pi) & = & \begin{cases} \sum_{j=0}^{m_\alpha-1} R_A(s_j,a_j) + R_S(s_{m_\alpha}){\cdot}t_\alpha & \textit{if } (m_\alpha,t_\alpha) \textit{ exists} \\ \infty & \textit{otherwise} \end{cases}
\end{array}
$$

where $m_0{=}0$ *and* $m_k{=}\max\{j \mid dur_\pi(i){<}k\}$ *if* $k{>}0$ *and, when it exists,* (m_α,t_α) *is is the minimum position of the path* π *under the ordering* $\prec$ *for which* $s_{m_\alpha}{+}t_\alpha \models \alpha$*.*

In the case of the until operator, as for timed automata [26], due to the dense nature of time we require that the disjunction $\alpha_1 \vee \alpha_2$, as opposed to the formula α_1, holds at all positions preceding the first position at which α_2 is satisfied.

For a POPTA P and time domain $\mathbb{T}$, the action rewards of $[\![\mathsf{P}]\!]_{\mathbb{T}}$ (see Definitions 16 and 14) encode both the accumulation of state rewards when a time transition is taken and the action rewards of P. It follows that for cumulative reward properties, we only need to consider the action rewards of $[\![\mathsf{P}]\!]_{\mathbb{T}}$ together with the reward accumulated in the location we are in when either the time bound or the goal is first reached.

5.2 Verification and strategy synthesis

Given a POPTA P and property ϕ, as for POMDPs we are interested in solving the dual problems of *verification* and *strategy synthesis* (see Definitions 9 and 10) for the 'standard' dense-time semantics of P:

- decide if $[\![\mathsf{P}]\!]_{\mathbb{R}}, \sigma \models \phi$ holds for all strategies $\sigma \in \Sigma_{[\![\mathsf{P}]\!]_{\mathbb{R}}}$;
- find, if it exists, a strategy $\sigma \in \Sigma_{[\![\mathsf{P}]\!]_{\mathbb{R}}}$ such that $[\![\mathsf{P}]\!]_{\mathbb{R}}, \sigma \models \phi$.

Again, in similar fashion to POMDPs, these can be solved by computing optimal values for either path or reward objectives:

$$
\begin{array}{ll}
Pr^{\min}_{[\![\mathsf{P}]\!]_{\mathbb{R}}}(\psi) \stackrel{\text{def}}{=} \inf_{\sigma \in \Sigma_{[\![\mathsf{P}]\!]_{\mathbb{R}}}} Pr^{\sigma}_{[\![\mathsf{P}]\!]_{\mathbb{R}}}(\psi) & \mathbb{E}^{\min}_{[\![\mathsf{P}]\!]_{\mathbb{R}}}(\rho) \stackrel{\text{def}}{=} \inf_{\sigma \in \Sigma_{[\![\mathsf{P}]\!]_{\mathbb{R}}}} \mathbb{E}^{\sigma}_{[\![\mathsf{P}]\!]_{\mathbb{R}}}(\rho) \\
Pr^{\max}_{[\![\mathsf{P}]\!]_{\mathbb{R}}}(\psi) \stackrel{\text{def}}{=} \sup_{\sigma \in \Sigma_{[\![\mathsf{P}]\!]_{\mathbb{R}}}} Pr^{\sigma}_{[\![\mathsf{P}]\!]_{\mathbb{R}}}(\psi) & \mathbb{E}^{\max}_{[\![\mathsf{P}]\!]_{\mathbb{R}}}(\rho) \stackrel{\text{def}}{=} \sup_{\sigma \in \Sigma_{[\![\mathsf{P}]\!]_{\mathbb{R}}}} \mathbb{E}^{\sigma}_{[\![\mathsf{P}]\!]_{\mathbb{R}}}(\rho)
\end{array}
$$

and, where required, also synthesising an optimal strategy. The remainder of this section therefore focuses on how to (approximately) compute optimal values and strategies for POPTAs.

5.3 Numerical computation algorithms

Approximate numerical computation of either optimal probabilities or expected reward values on a POPTA P is performed with the sequence of steps given below, As for POMDPs we compute both an under- and an over-approximation. For the former, we also generate a strategy which achieves this value.

(A) We modify POPTA P, reducing the problem to computing optimal values for a *probabilistic reachability* or *expected cumulative reward* property [42];
(B) We apply the *digital clocks* discretisation of Section 4 to reduce the infinite-state semantics $[\![\mathsf{P}]\!]_{\mathbb{R}}$ of P to a *finite-state POMDP* $[\![\mathsf{P}]\!]_{\mathbb{N}}$;
(C) We build and solve a *finite abstraction* of the (infinite-state) belief MDP $\mathcal{B}([\![\mathsf{P}]\!]_{\mathbb{N}})$ of the POMDP from (B), yielding an *over-approximation*;
(D) We synthesise and analyse a strategy for $[\![\mathsf{P}]\!]_{\mathbb{N}}$, giving an *under-approximation*;
(E) If required, we *refine* the abstraction's precision and repeat (C) and (D).

(A) Property reduction. As discussed in [42] (for PTAs), checking $\mathtt{P}_{\bowtie p}[\psi]$ or $\mathtt{R}_{\bowtie q}[\rho]$ properties of the logic from Definition 18 can always be reduced to checking either a probabilistic reachability ($\mathtt{P}_{\bowtie p}[\mathtt{F}\ \alpha]$) or expected cumulative reachability reward ($\mathtt{R}_{\bowtie q}[\mathtt{F}\ \alpha]$) property on a modified model. For example, time-bounded probabilistic reachability ($\mathtt{P}_{\bowtie p}[\mathtt{F}^{\leqslant t}\ \alpha]$) can be transformed into probabilistic reachability ($\mathtt{P}_{\bowtie p}[\mathtt{F}\ (\alpha \wedge y \leqslant t)]$) where y is a new clock added to P which is never reset and does not appear in any invariant or enabling conditions. We refer to [42] for full details.

(B) Digital clocks. Assuming the POPTA P satisfies Assumptions 1 and 2, we can construct a finite POMDP $[\![\mathsf{P}]\!]_{\mathbb{N}}$ representing P by treating clocks as bounded integer variables. The correctness of this reduction is demonstrated below. The translation itself is relatively straightforward, involving a syntactic translation of the PTA (to convert clocks), followed by a systematic exploration of its finite state space. At this point, we also syntactically check satisfaction of the restrictions (Assumptions 1 and 2) that we require of POPTAs.

(C–E) POMDP analysis. This follows the approach for analysing probabilistic and expected cumulative reachability queries of POMDPs given in Section 3.

5.4 Correctness of the digital clocks reduction

We now prove that the digital clocks reduction preserves optimal probabilistic and expected reachability values of POPTAs. A direct corollary of this is that, for the logic presented in Definition 21, we can perform both verification and strategy synthesis using the finite-state digital clocks semantics.

Theorem 1. *If* P *is a POPTA satisfying Assumptions 1 and 2, then, for any set of observations* O_L *of* P *and* $opt \in \{\min, \max\}$*, we have:*

$$Pr^{opt}_{[\![\mathsf{P}]\!]_{\mathbb{R}}}(\mathtt{F}\ O_L) = Pr^{opt}_{[\![\mathsf{P}]\!]_{\mathbb{N}}}(\mathtt{F}\ O_L) \quad \textit{and} \quad \mathbb{E}^{opt}_{[\![\mathsf{P}]\!]_{\mathbb{R}}}(\mathtt{F}\ O_L) = \mathbb{E}^{opt}_{[\![\mathsf{P}]\!]_{\mathbb{N}}}(\mathtt{F}\ O_L).$$

Corollary 1. *If* P *is a POPTA satisfying Assumptions 1 and 2, and* ϕ *is a property from Definition 18, then:*

- $[\![\mathsf{P}]\!]_{\mathbb{R}}, \sigma \models \phi$ *holds for all strategies* $\sigma \in \Sigma_{[\![\mathsf{P}]\!]_{\mathbb{R}}}$ *if and only if* $[\![\mathsf{P}]\!]_{\mathbb{N}}, \sigma \models \phi$ *holds for all strategies* $\sigma \in \Sigma_{[\![\mathsf{P}]\!]_{\mathbb{N}}}$*;*
- *there exists a strategy* $\sigma \in \Sigma_{[\![\mathsf{P}]\!]_{\mathbb{R}}}$ *such that* $[\![\mathsf{P}]\!]_{\mathbb{R}}, \sigma \models \phi$ *if and only if there exists a strategy* $\sigma' \in \Sigma_{[\![\mathsf{P}]\!]_{\mathbb{N}}}$ *such that* $[\![\mathsf{P}]\!]_{\mathbb{N}}, \sigma' \models \phi$*;*
- *if a strategy* $\sigma \in \Sigma_{[\![\mathsf{P}]\!]_{\mathbb{N}}}$ *is such that* $[\![\mathsf{P}]\!]_{\mathbb{N}}, \sigma \models \phi$*, then* $\sigma \in \Sigma_{[\![\mathsf{P}]\!]_{\mathbb{R}}}$ *and* $[\![\mathsf{P}]\!]_{\mathbb{R}}, \sigma \models \phi$*.*

Proof. In each case, the proof follows straightforwardly from [42] which demonstrates that checking a property ϕ of the logic given in Definition 18 can always be reduced to checking either a probabilistic reachability ($\mathtt{P}_{\bowtie p}[\mathtt{F}\ \alpha]$) or expected cumulative reachability reward ($\mathtt{R}_{\bowtie q}[\mathtt{F}\ \alpha]$) property and using Theorem 1. The generalisation of results in [42] from PTAs to POPTAs relies on the fact that propositional formulae α in the logic are based on either observations or clock valuations, both of which are observable. □

Before we give the proof of Theorem 1 we require the following definitions and preliminary result. Consider a POPTA $\mathsf{P}=(L, \bar{l}, \mathcal{X}, A, inv, enab, prob, r, \mathcal{O}_L, obs_L)$. If v, v' are clock valuations and X, Y sets of clocks such that $X \neq Y$ and $v(x)>0$ for any $x \in X \cup Y$, then $v[X:=0] \neq v[Y:=0]$. Therefore, since we restrict our attention to POPTAs which reset only non-zero clocks (see Assumption 2), for a time domain $\mathbb{T}$, if there exists a transition from (l, v) to (l', v') in $[\![\mathsf{P}]\!]_{\mathbb{T}}$, then there is a unique (possibly empty) set of clocks which are reset when this transition is taken. We formalise this through the following definition. For any clock valuations $v, v' \in \mathbb{T}^{\mathcal{X}}$, let:

$$X_{[v \mapsto v']} \stackrel{\text{def}}{=} \{x \in \mathcal{X} \mid v(x)>0 \wedge v'(x)=0\}. \tag{1}$$

Using (1), the probabilistic transition function of $[\![\mathsf{P}]\!]_{\mathbb{T}}$ is such that, for any $(l, v) \in S$ and $a \in A$, we have $P((l, v), a)=\mu$ if and only if $v \models enab(l, a)$ and for any $(l', v') \in S$:

$$\mu(l', v') = \begin{cases} prob(l, a)(X_{[v \mapsto v']}, l') & \text{if } v[X_{[v \mapsto v']}:=0]=v' \\ 0 & \text{otherwise.} \end{cases}$$

We next introduce the concept of a belief PTA.

Definition 22 (Belief PTA). *If* $\mathsf{P}=(L, \bar{l}, \mathcal{X}, A, inv, enab, prob, r, \mathcal{O}_L, obs_L)$ *is a POPTA, the* belief PTA *of* P *is given by the tuple:*

$$\mathcal{B}(\mathsf{P}) = (Dist(L, obs_L), \delta_{\bar{l}}, \mathcal{X}, A, inv^{\mathcal{B}}, enab^{\mathcal{B}}, prob^{\mathcal{B}}, r^{\mathcal{B}})$$

where:

- *$Dist(L, obs_L)$ denotes the subset of $Dist(L)$ where $\lambda \in Dist(L, obs_L)$ if and only if, for $l, l' \in L$ such that $\lambda(l)>0$ and $\lambda(l')>0$ we have $obs_L(l)=obs_L(l')$;*
- *the invariant condition $inv^{\mathcal{B}} : Dist(L, obs_L) \to CC(\mathcal{X})$ and enabling condition $enab^{\mathcal{B}} : Dist(L, obs_L) \times A \to CC(\mathcal{X})$ are such that, for $\lambda \in Dist(L, obs_L)$ and $a \in A$, we have $inv^{\mathcal{B}}(\lambda)=inv(l)$ and $enab^{\mathcal{B}}(\lambda, a)=enab(l, a)$ where $l \in L$ and $\lambda(l)>0$;*
- *the probabilistic transition function:*

$$prob^{\mathcal{B}} : Dist(L, obs_L) \times A \to Dist(2^{\mathcal{X}} \times Dist(L, obs_L))$$

is such that, for any $\lambda, \lambda' \in Dist(L, obs_L)$, $a \in A$ and $X \subseteq \mathcal{X}$ we have:

$$prob^{\mathcal{B}}(\lambda, a)(\lambda', X) = \sum_{l \in L} \lambda(l) \cdot \left(\sum_{o \in O \wedge \lambda^{a,o,X} = \lambda'} \; \sum_{l' \in L \wedge obs_L(l')=o} prob(l, a)(l', X) \right)$$

and, for any $l' \in L$:

$$\lambda^{a,o,X}(l') = \begin{cases} \frac{\sum_{l \in L} prob(l,a)(l',X) \cdot \lambda(l)}{\sum_{l \in L} \lambda(l) \cdot \left(\sum_{l' \in L \wedge obs_L(l')=o} prob(l,a)(l',X)\right)} & \text{if } obs_L(l')=o \\ 0 & \text{otherwise;} \end{cases}$$

- *the reward structure $r^{\mathcal{B}}=(r_L^{\mathcal{B}}, r_A^{\mathcal{B}})$ consists of a location reward function $r_L^{\mathcal{B}} : Dist(L, obs_L) \to \mathbb{R}$ and action reward function $r_A^{\mathcal{B}} : Dist(L, obs_L) \times A \to \mathbb{R}$ such that, for any $\lambda \in Dist(L, obs_L)$ and $a \in A$:*

$$r_L^{\mathcal{B}}(\lambda) = \textstyle\sum_{l \in L} \lambda(l) \cdot r_L(l) \qquad \text{and} \qquad r_A^{\mathcal{B}}(\lambda, a) = \textstyle\sum_{l \in L} \lambda(l) \cdot r_A(l, a)\,.$$

For the above to be well defined, we require the conditions on the invariant condition and observation function given in Definition 13 to hold. For any $\lambda \in Dist(L, obs_L)$, we let o_λ be the unique observation such that $obs_L(l)=o_\lambda$ and $\lambda(l)>0$ for some $l \in L$.

We now show that, for a POPTA P, the semantics of its belief PTA is isomorphic to the belief MDP of the semantics of P.

Proposition 2. *For any POPTA P satisfying Assumption 2, time domain $\mathbb{T}$ we have that the MDPs $[\![\mathcal{B}(\mathsf{P})]\!]_{\mathbb{T}}$ and $\mathcal{B}([\![\mathsf{P}]\!]_{\mathbb{T}})$ are isomorphic.*

Proof. Consider any POPTA $\mathsf{P}=(L, \bar{l}, \mathcal{X}, A, inv, enab, prob, r, \mathcal{O}_L, obs_L)$ which satisfies Assumption 2, time domain $\mathbb{T}$ and let $[\![\mathsf{P}]\!]_{\mathbb{T}}=(S, \bar{s}, A \cup \mathbb{T}, P, R)$. To show the MDPs $[\![\mathcal{B}(\mathsf{P})]\!]_{\mathbb{T}}$ and $\mathcal{B}([\![\mathsf{P}]\!]_{\mathbb{T}})$ are isomorphic we first give a bijection between their state spaces and then use this bijection to show that the probabilistic transition and reward functions of $[\![\mathcal{B}(\mathsf{P})]\!]_{\mathbb{T}}$ and $\mathcal{B}([\![\mathsf{P}]\!]_{\mathbb{T}})$ are isomorphic.

Considering the belief MDP $\mathcal{B}([\![\mathsf{P}]\!]_{\mathbb{T}})$, see Definitions 5 and 16, and using the fact that $obs(l, v) = (obs_L(l), v)$, for any belief states b, b' and action a:

$$P^{\mathcal{B}}(b, a)(b') = \sum_{\substack{(o, v_o) \in O \times \mathbb{T}^{\mathcal{X}} \\ b^{a,(o,v_o)}=b'}} \; \sum_{(l,v) \in S} b(l, v) \cdot \left(\sum_{l' \in L \wedge obs_L(l')=o} P((l, v), a)(l', v_o) \right)$$

where, for any belief b, action a, observation (o, v_o) and state (l', v'), we have $b^{a,(o,v_o)}(l', v')$ equals:

$$\begin{cases} \frac{\sum_{(l,v)\in S} P((l,v),a)(l',v')\cdot b(l,v)}{\sum_{(l,v)\in S} b(l,v)\cdot\left(\sum_{l''\in L\wedge obs_L(l'')=o} P((l,v),a)(l'',v')\right)} & \text{if } obs_L(l')=o \text{ and } v'=v_o \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and $R^{\mathcal{B}}(b,a) = \sum_{(l,v)\in S} R((l,v),a)\cdot b(l,v)$. Furthermore, by Definition 16 and since P satisfies Assumption 2, if $a \in A$:

$$P((l,v),a)(l',v') = \begin{cases} prob(l,a)(X_{[v\mapsto v']}, l') & \text{if } v[X_{[v\mapsto v']}:=0]=v' \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

while if $a \in \mathbb{T}$:

$$P((l,v),a)(l',v') = \begin{cases} 1 & \text{if } l'=l \text{ and } v'=v+a \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

We see that $b^{a,(o,v_o)}(l', v')$ is zero if $v' \neq v_o$, and therefore we can write the belief as (λ, v_o) where $\lambda \in Dist(L)$ and $\lambda(l)=b^{a,(o,v_o)}(l, v_o)$ for all $l \in L$. In addition, for any $l' \in L$, if $\lambda(l')>0$, then $obs_L(l')=o$. Since the initial belief $\bar{b}$ can be written as $(\delta_{\bar{l}}, \mathbf{0})$ and we assume $obs_L(\bar{l}) \neq obs_L(l)$ for any $l \neq \bar{l} \in L$, it follows that we can write each belief b of $\mathcal{B}(\llbracket \mathsf{P} \rrbracket_{\mathbb{T}})$ as a tuple $(\lambda, v) \in Dist(L) \times \mathbb{T}^{\mathcal{X}}$ such that for any $l, l' \in L$, if $\lambda(l)>0$ and $\lambda(l')>0$, then $obs_L(l)=obs_L(l')$. Hence, it follows from Definitions 22 and 14 that there is a bijection between the states of $\mathcal{B}(\llbracket \mathsf{P} \rrbracket_{\mathbb{T}})$ and the states of $\llbracket \mathcal{B}(\mathsf{P}) \rrbracket_{\mathbb{T}}$.

We now use this bijection between the states to show that the probabilistic transition function and reward functions of $\llbracket \mathcal{B}(\mathsf{P}) \rrbracket_{\mathbb{T}}$ and $\mathcal{B}(\llbracket \mathsf{P} \rrbracket_{\mathbb{T}})$ are isomorphic. Using Definitions 5 and 16, for the probabilistic transition and the action reward functions we have the following two cases to consider.

– For any belief states (λ, v) and (λ', v') and action $a \in A$:

$$\begin{aligned} P^{\mathcal{B}}((\lambda,v),a)(\lambda',v') &= \sum_{\substack{o\in\mathcal{O}_L \\ \lambda^{a,(o,v')}=\lambda'}} \sum_{l\in L} \lambda(l)\cdot\left(\sum_{\substack{l'\in L \\ \mathcal{O}(l')=o}} P((l,v),a)(l',v')\right) \\ &= \sum_{\substack{o\in\mathcal{O}_L \\ \lambda^{a,(o,v')}=\lambda'}} \sum_{l\in L} \lambda(l)\cdot\left(\sum_{\substack{l'\in L \\ \mathcal{O}(l')=o}} prob(l,a)(X_{[v\mapsto v']}, l')\right) && \text{by (3)} \\ &= \sum_{l\in L} \lambda(l)\cdot\left(\sum_{\substack{o\in\mathcal{O}_L \\ \lambda^{a,(o,v')}=\lambda'}} \sum_{\substack{l'\in L \\ \mathcal{O}(l')=o}} prob(l,a)(X_{[v\mapsto v']}, l')\right) && \text{rearranging} \end{aligned}$$

where for any $l' \in L$:

$$\lambda^{a,(o,v')}(l') = \begin{cases} \frac{\sum_{l\in L} P((l,v),a)(l',v')\cdot\lambda(l)}{\sum_{l\in L}\lambda(l)\cdot\left(\sum_{l''\in L\wedge obs_L(l'')=o} P((l,v),a)(l'',v')\right)} & \text{if } obs_L(l')=o \\ 0 & \text{otherwise} \end{cases}$$

$$= \begin{cases} \frac{\sum_{l\in L} prob(l,a)(X_{[v\mapsto v']},l')\cdot\lambda(l)}{\sum_{l\in L}\lambda(l)\cdot\left(\sum_{l''\in L\wedge obs_L(l'')=o} prob(l,a)(X_{[v\mapsto v']},l')\right)} & \text{if } obs_L(l')=o \\ 0 & \text{otherwise} \end{cases} \quad \text{by (3)}$$

$$= \lambda^{a,o,X_{[v\mapsto v']}} \quad \text{by Definition 22.}$$

Using this result, together with Definitions 22 and 14, it follows that the probabilistic transition functions are isomorphic in the case. For the action reward functions, we have:

$$R^{\mathcal{B}}_A((\lambda,v),a) = \textstyle\sum_{l\in L} r_A(l,a)\cdot\lambda(l)$$

which, again from Definitions 22 and 14, shows that the reward functions are isomorphic in this case.

– For any belief states (λ, v) and (λ', v') and time duration $t \in \mathbb{T}$:

$$P^{\mathcal{B}}((\lambda,v),t)(\lambda',v') = \begin{cases} \sum_{l\in L}\lambda(l)\cdot P((l,v),a)(l,v') & \text{if } \lambda^{t,(o_\lambda,v')}=\lambda' \\ 0 & \text{otherwise} \end{cases}$$

where for any $l' \in L$:

$$\lambda^{t,(o_\lambda,v')}(l') = \begin{cases} \frac{\lambda(l')}{\sum_{l\in L}\lambda(l)} & \text{if } v'=v+t \\ 0 & \text{otherwise} \end{cases}$$

$$= \begin{cases} \lambda(l') & \text{if } v'=v+t \\ 0 & \text{otherwise} \end{cases} \quad \text{since } \lambda \text{ is a distribution.}$$

Substituting this expression for $\lambda^{t,(o_\lambda,v')}$ into that of $P^{\mathcal{B}}((\lambda,v),t)$ we have:

$$P^{\mathcal{B}}((\lambda,v),t)(\lambda',v')$$

$$= \begin{cases} \sum_{l\in L}\lambda(l)\cdot\left(\sum_{l'\in L} P((l,v),a)(l',v')\right) & \text{if } \lambda=\lambda' \text{ and } v'=v+t \\ 0 & \text{otherwise} \end{cases}$$

$$= \begin{cases} \sum_{l\in L}\lambda(l) & \text{if } \lambda=\lambda' \text{ and } v'=v+t \\ 0 & \text{otherwise} \end{cases} \quad \text{by (4)}$$

$$= \begin{cases} 1 & \text{if } \lambda=\lambda' \text{ and } v'=v+t \\ 0 & \text{otherwise} \end{cases} \quad \text{since } \lambda \text{ is a distribution}$$

which, from Definitions 22 and 14, shows the probabilistic transition functions are isomorphic. For the action reward function of $\mathcal{B}(\llbracket \mathsf{P} \rrbracket_{\mathbb{T}})$, we have $R^{\mathcal{B}}_A((\lambda,v),t) = \sum_{l\in L}(r_L(l)\cdot t)\cdot\lambda(l)$ and, from Definitions 22 and 14, this implies that the action reward functions are isomorphic.

Since these are the only cases to consider, both the probabilistic transition and action reward functions of $\mathcal{B}(\llbracket \mathsf{P} \rrbracket_{\mathbb{T}})$ and $\llbracket \mathcal{B}(\mathsf{P}) \rrbracket_{\mathbb{T}}$ are isomorphic.

To complete the proof it remains to show that the state reward functions are isomorphic. Since, by Definition 5, for any belief state (λ, v), we have $R^{\mathcal{B}}_S(\lambda, v) = \sum_{l \in L} r_L(l) \cdot \lambda(l)$, the result follows from Definitions 22 and 14. □

We are now in a position to present the proof of Theorem 1.

Proof (of Theorem 1). Consider any POPTA P satisfying Assumptions 1 and 2 and set of observables O_L of P. Since the PTA $\mathcal{B}(\mathsf{P})$ satisfies Assumption 1, using results presented in [32], we have that:

$$Pr^{opt}_{[\![\mathcal{B}(\mathsf{P})]\!]_{\mathbb{R}}}(\mathtt{F}\ T_{O_L}) = Pr^{opt}_{[\![\mathcal{B}(\mathsf{P})]\!]_{\mathbb{N}}}(\mathtt{F}\ T_{O_L}) \tag{5}$$

$$\mathbb{E}^{opt}_{[\![\mathcal{B}(\mathsf{P})]\!]_{\mathbb{R}}}(\mathtt{F}\ T_{O_L}) = \mathbb{E}^{opt}_{[\![\mathcal{B}(\mathsf{P})]\!]_{\mathbb{N}}}(\mathtt{F}\ T_{O_L}) \tag{6}$$

for $opt \in \{\min, \max\}$ and where $T_{O_L} = \{(l, v) \in L \times \mathbb{T}^{\mathcal{X}} \mid obs(l) \in O_L\}$. Note that, although [32] considers only PTAs with a finite set of locations, the proofs corresponding to the above results do not rely this fact, and hence the results carry over to $\mathcal{B}(\mathsf{P})$ which has an uncountable number of locations.

Due to the relationship we have given between the optimal probabilistic and expected reachability values of a POMDP and its belief MDP (see Proposition 1), it follows that:

$$Pr^{opt}_{[\![\mathsf{P}]\!]_{\mathbb{T}}}(\mathtt{F}\ O_L) = Pr^{opt}_{\mathcal{B}([\![\mathsf{P}]\!]_{\mathbb{T}})}(\mathtt{F}\ T_{O_L}) \quad \text{and} \quad \mathbb{E}^{opt}_{[\![\mathsf{P}]\!]_{\mathbb{T}}}(\mathtt{F}\ O_L) = \mathbb{E}^{opt}_{\mathcal{B}([\![\mathsf{P}]\!]_{\mathbb{T}})}(\mathtt{F}\ T_{O_L})\,. \tag{7}$$

Using Proposition 2 and, since P satisfies Assumption 2, it follows therefore that $[\![\mathcal{B}(\mathsf{P})]\!]_{\mathbb{T}} = \mathcal{B}([\![\mathsf{P}]\!]_{\mathbb{T}})$ for $\mathbb{T} \in \{\mathbb{R}, \mathbb{N}\}$. Combining this result with (5), (6) and (7), the theorem follows. □

6 Implementation and Case Studies

We have built a prototype tool for verification and strategy synthesis of POMDPs and POPTAs as an extension of the PRISM model checker [31,44]. Models are described in an extension of the existing PRISM modelling language, described in Section 6.1 below. For a specified POMDP or POPTA and property, the tool performs the steps outlined in Sections 3 and 5, computing a pair of bounds for a given property and synthesising a corresponding strategy.

We have developed a number of POMDP and POPTA case studies, from a variety of different application domains, to evaluate the tool and techniques. In each case, partial observability, nondeterminism, probability and, in the case of POPTAs, real-time behaviour are all essential aspects required for the analysis. The case studies are described in detail in Sections 6.2 to 6.7, and we summarise the experimental results from these examples in Section 6.8.

The software, details of all case studies, parameters and properties are available from [55]. Also available through this link are the details for the POMDPs in Examples 1 and 2 and the POPTAs in Examples 3 and 4.

6.1 Modelling POMDPs and POPTAs in PRISM

Models in PRISM are specified in a high-level language based on guarded commands, which is a variant of Reactive Modules [3]. A model is constructed as a set of modules which can interact with each other. A module contains a number of finite-valued variables which define the module's state. It's behaviour is described by a set of guarded commands containing an (optional) action label, a guard and a probabilistic choice between updates:

$$[\texttt{<action>}]\ \texttt{<guard>} \rightarrow\ \texttt{<prob>} : \texttt{<update>}\ +\ \cdots\ +\ \texttt{<prob>} : \texttt{<update>};$$

A guard is a predicate over the variables of all modules and an update specifies, using primed variables, how the module's own variables are updated. Interaction is both through the guards (guards can refer to variables of other modules) and the action labels (which allow modules to synchronise over commands). PRISM includes support for reward structures through reward items of the form:

$$\texttt{<guard>} : \texttt{<reward>}; \quad \text{or} \quad [\texttt{<action>}]\ \texttt{<guard>} : \texttt{<reward>};$$

representing state and action rewards respectively. In the case of real-time models, modules can also contain `clock` variables which can appear in guards and be reset by updates. In addition, the `invariant` keyword is used to allow for the specification of location invariants.

We have extended the existing modelling language for MDPs and PTAs to allow specification of which variables are observables (the unspecified variables are considered hidden) through the keyword `observables`.

6.2 Wireless network scheduling

Our first case study is based on [52] and concerns the wireless downlink scheduling of traffic to a number of different users with hard deadlines and where packets have priorities. The system is time-slotted: time is divided into periods and each period is divided into an equal number of slots. The system is parameterised by the total number of time periods (K) and the number of slots (T) per time period. At the start of each time period, a new packet is generated for each user with a priority assigned randomly. The goal of scheduling is to, in each period, deliver the packets to each user before the period ends. Packets not delivered by the end of a period are dropped.

There are c users and each one has a separate channel which can be in two states: one in which it is able to decode packets and one where it cannot. The state of each channel remains fixed within a time slot and between slots is Markovian, i.e., it changes randomly based only on the state in the previous slot. It is assumed that the conditions of the channels are unavailable to the system when scheduling packets. This corresponds to the real world situation where perfect channel information is not normally available since it requires non-negligible network resources.

```
pomdp

observables
  sched, k, t, pack1, pack2, pack3, prio1, prio2, prio3
endobservables

// timing constants
const int K; // total number of time periods
const int T; // number of slots per time period

// probabilities that channels change status
// channel of user 1
const double p1 = 0.8; // probability channel remains on
const double r1 = 0.2; // probability channel moves from off to on
// channel of user 2
const double p2 = 0.6; // probability channel remains on
const double r2 = 0.4; // probability channel moves from off to on
// channel of user 3
const double p3 = 0.7; // probability channel remains on
const double r3 = 0.3; // probability channel moves from off to on
```

Fig. 4. Initial fragment of the PRISM model for the network scheduling case study.

```
module scheduler

  k : [0..K−1]; // current time period
  t : [0..T−1]; // correct slot
  sched : [0..1]; // local state

  // next slot/time period
  [slot] sched=0 & t<T−1 → (sched′=1) & (t′=t+1);
  [slot] sched=0 & t=T−1 & k<K−1 → (sched′=1) & (t′=0) & (k′=k+1);

  // make scheduling choice
  [idle]   sched=1 → (sched′=0);
  [send1] sched=1 → (sched′=0);
  [send2] sched=1 → (sched′=0);
  [send3] sched=1 → (sched′=0);

  // loop when finished
  [] sched=0 & t=T−1 & k=K−1 → true;

endmodule
```

Fig. 5. PRISM module for the scheduler in the network scheduling case study.

The system is modelled in PRISM as a POMDP through the parallel composition of $3{\cdot}c{+}1$ modules (one module for the packet, priority and status of each channel and one module representing the scheduler). We show here the PRISM code for the case of $c{=}3$ users (and hence 3 channels). Figure 4 presents the first parts of the corresponding PRISM model. This defines the model type (POMDP), states which variables are observable and defines some constants used to describe the model. All variables except those representing the status of the channels are defined as observable, and hence the scheduler can observe the elapsed time, which packets need to be sent and their priorities. The constants include the numbers of time periods (K), of slots per period (T), and the probabilities that the channels change state after each time slot.

The module for the scheduler is presented in Figure 5. The scheduler has two local states: in the first (when $sched{=}0$), it updates the timing variables, i.e., either moves to the next slot or to the next period; in the second local state

```
// packets for channel 1
module packet1

  pack1 : [0..1]; // packet to send in current period

  // next slot
  [slot] t=0 → (pack1'=1); // new period so new packet
  [slot] t>0 → true;
  // sending
  [send1]  pack1=1 & chan1=1 → (pack1'=0); // channel up
  [send1]  pack1=1 & chan1=0 → true; // channel down

endmodule

// construct further channels' packets through renaming
module packet2=packet1[pack1=pack2, send1=send2, chan1=chan2] endmodule
module packet3=packet1[pack1=pack3, send1=send3, chan1=chan3] endmodule

// priority of the packets for channel 1
module priority1

  prio1 : [0..3]; // three priority values

  // new period so new packet and randomly assign priority
  [slot] t=0 → 0.1 : (prio1'=1) + 0.3 : (prio1'=2) + 0.6 : (prio1'=3);
  // priority already assigned for this period
  [slot] t>0 → true;

  // reset priority when packet has been sent
  [send1] chan1=0 → true;
  [send1] chan1=1 → (prio1'=0);

endmodule

// construct further priorities through renaming
module priority2 = priority1[prio1=prio2, chan1=chan2, send1=send2] endmodule
module priority3 = priority1[prio1=prio3, chan1=chan3, send1=send3] endmodule

// channel 1 status
module channel1

  chan1 : [0..1]; // status of channel (off/on)

  // initialise
  [slot] t=0 & k=0 → 0.5 : (chan1'=0) + 0.5 : (chan1'=1);
  // next slot
  [slot] chan1=0 & !(t=0 & k=0) → 1−r1 : (chan1'=0) + r1 : (chan1'=1);
  [slot] chan1=1 & !(t=0 & k=0) → 1−p1 : (chan1'=0) + p1 : (chan1'=1);

endmodule

// construct further channels through renaming
module channel2=channel1[chan1=chan2, p1=p2, r1=r2] endmodule
module channel3=channel1[chan1=chan3, p1=p3, r1=r3] endmodule
```

Fig. 6. PRISM modules for the channels in the network scheduling case study.

(when *sched*=1), it decides which packet to schedule for delivery in the current time slot by (nondeterministically) selecting one of the actions *send1*, *send2* and *send3* corresponding to the three channels. The scheduler can also choose not to try and send a packet by instead choosing the action *idle*.

The modules for the channels are presented in Figure 6. Each channel has three modules representing:

```
// reward structure for number of dropped packs
rewards "dropped_packets"
   [slot] t=0 & k>0 : ((pack1=0)?0:1) + ((pack2=0)?0:1) + ((pack3=0)?0:1);
   [idle] t=T−1 & k=K−1 : ((pack1=0)?0:1) + ((pack2=0)?0:1) + ((pack3=0)?0:1);
   [send1] t=T−1 & k=K−1 & chan1=0 : ((pack1=0)?0:1) + ((pack2=0)?0:1) + ((pack3=0)?0:1);
   [send2] t=T−1 & k=K−1 & chan2=0 : ((pack1=0)?0:1) + ((pack2=0)?0:1) + ((pack3=0)?0:1);
   [send3] t=T−1 & k=K−1 & chan3=0 : ((pack1=0)?0:1) + ((pack2=0)?0:1) + ((pack3=0)?0:1);
   [send1] t=T−1 & k=K−1 & chan1=1 : ((pack2=0)?0:1) + ((pack3=0)?0:1);
   [send2] t=T−1 & k=K−1 & chan2=1 : ((pack1=0)?0:1) + ((pack3=0)?0:1);
   [send3] t=T−1 & k=K−1 & chan3=1 : ((pack1=0)?0:1) + ((pack2=0)?0:1);
endrewards

// reward structure based on priorities
rewards "priority"
   [send1] chan1=1 & prio1=1 : 1;
   [send2] chan2=1 & prio2=1 : 1;
   [send3] chan3=1 & prio3=1 : 1;
   [send1] chan1=1 & prio1=2 : 10;
   [send2] chan2=1 & prio2=2 : 10;
   [send3] chan3=1 & prio3=2 : 10;
   [send1] chan1=1 & prio1=3 : 20;
   [send2] chan2=1 & prio2=3 : 20;
   [send3] chan3=1 & prio3=3 : 20;
endrewards
```

Fig. 7. PRISM specification of reward structures for the network scheduling case study.

- if the packet for the current time period has been sent;
- the priority of the current packet to be sent;
- the status of the channel.

As can be seen in Figure 6, we only give the full specification of the modules for the first channel; the modules for the remaining channels are defined through renaming. In the module *packet1*, commands labelled by the action *send1* are only enabled when *pack1*=1, and hence, as the modules synchronise, the scheduler can only choose to send a packet if it has not yet been delivered. This module also specifies that, if a packet is sent and the channel is down (*chan1*=0), the packet does not get delivered and still needs to be sent. In the modules *packet1* and *priority1*, we can see that at the start of each period there is a new packet to send on each channel and the priority of these packets is chosen at random. The module *channel1* specifies that in the initial state the status of the channel is selected uniformly at random and after this the status of the channel follows the probabilities given in Figure 4.

Finally, the reward structures for the model are presented in Figure 7. The first reward structure is used to count the number of dropped packets, i.e., the number of packets that remain to be sent at the end of each period. This is achieved by counting the number of the variables *pack1*, *pack2* and *pack3* that equal 1 when a time period ends. The second reward structure is used to accumulate the priorities of delivered packets, and therefore each time a packet gets delivered we assign an action reward equal to the corresponding priority.

For this case study, we synthesise strategies that maximise the expected cumulative reward based on the priorities of the packets using the reward structure of [52] and, for a simpler model where the priorities of packets are not considered (by removing the modules *priority1*, *priority2* and *priority3* and related

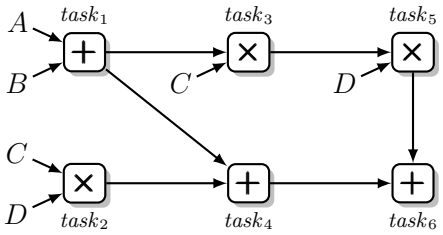

Fig. 8. Processor task graph for computing $D \times (C \times (A + B)) + ((A + B) + (C \times D))$.

reward structure), that minimise the expected number of dropped packets. These requirements can be specified in PRISM as follows:

- R{"*priority*"}max=?[F (*sched*=0 & $t{=}T{-}1$ & $k{=}K{-}1$)];
- R{"*dropped_packets*"}min=?[F (*sched*=0 & $t{=}T{-}1$ & $k{=}K{-}1$)].

In [52] the analysis is through handwritten proofs while here we construct a formal model and perform automated analysis, in addition in [52] discounted objectives are considered while we analyse undiscounted reachability objectives.

In [52] it is demonstrated that, due to hard deadlines and unknown channel status, idling, i.e. not sending a packet in certain slots even when there is a packet to send, is the optimal choice in certain situations. The reasoning given is that this allows the scheduler to learn the status of the channels and then improve the success of future transmissions of packets. Our analysis confirms this to be the case when priorities are considered. For example, when $T{=}3$ and $K{=}2$ which are the parameter values [52] use, we find that disallowing the scheduler to idle causes the maximum expected accumulated reward interval to decrease from $[36.322, 36.324]$ to $[36.316, 36.318]$ when the grid resolution is 48.

Our results also demonstrate that, when priorities of packets are not considered, idling is not an optimal choice. By using the presented approach this analysis was easy to perform as we only needed to make a simple change to the PRISM model removing the option for the scheduler to idle unless all packets in the current slot have been delivered, i.e. in the module for the scheduler (see Figure 5) the command labelled by the action *idle* becomes:

$$[idle]sched{=}1 \;\&\; pack1{=}0 \;\&\; pack2{=}0 \;\&\; pack3{=}0 \;\rightarrow\; (sched'{=}0);$$

6.3 Task-graph scheduler

Next, we consider a task-graph scheduling problem adapted from [9], where the goal is to minimise the *time* or *energy consumption* required to evaluate the arithmetic expression $D{\times}(C{\times}(A{+}B)){+}((A{+}B){+}(C{\times}D))$ using two processors (P_1 and P_2) that have different speed and energy requirements. Figure 8 presents a task graph for this computation showing the tasks that need to be performed and the dependencies between the tasks. The specification of the processors, as given in [9], is as follows:

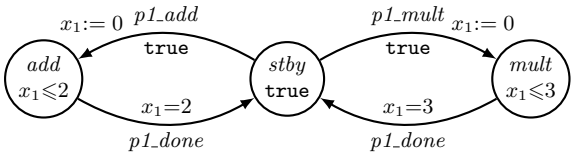

Fig. 9. Original timed automaton of processor P_1 from [9].

- *time for addition:* 2 and 5 picoseconds for processors P_1 and P_2;
- *time for multiplication:* 3 and 7 picoseconds for processors P_1 and P_2;
- *idle energy usage:* 10 and 20 Watts for processors P_1 and P_2;
- *active energy usage:* 90 and 30 Watts for processors P_1 and P_2.

The system is formed as the parallel composition of three timed automata: one for each processor and one for the scheduler. In Figure 9 we give the timed automaton representing P_1. The labels *p1_add* and *p1_mult* on the transitions represent an addition and multiplication task being scheduled on P_1 respectively, while the label *p1_done* indicates that the current task has been completed. The timed automaton includes the clock x_1 which is used to keep track of the time that a task has been running. It is reset when a task starts and the invariants and guards correspond to the time required to complete the tasks of addition and multiplication for P_1. The reward structure for computing the expected energy consumption associates a reward of 10 with the *stdby* location and reward 90 with the locations *add* and *mult* (corresponding to the energy usage of process P_1 when idle and active respectively) and all action rewards are 0.

The timed automaton and reward structure for processor P_2 are similar except that the names of the labels, invariants, guards and reward values correspond to the specification of P_2. The automata for the scheduler keeps track of the tasks that have been completed and nondeterministically decides how tasks get allocated to processes, subject to meeting the dependencies between tasks. After forming the parallel composition, the reward structure for the expected energy consumption then includes the addition of the reward structures for energy consumption of P_1 and P_2. The reward structure for computing the expected time associates a reward of 1 with all locations of the composed system.

We extend both the basic model of [9] described above and the extension from [42] which uses PTAs to model *probabilistic* task execution times. In both models we extend the processor P_1 with a new 'low power' state allowing it to save energy when not in use, but which incurs a delay of 4 picoseconds when waking up to execute a new task. This state is entered with probability *sleep* after each task is completed. We assume that the scheduler cannot observe whether the processor enters this lower power state, and hence the model is a POPTA. The POPTA for P_1 including this lower power state (labelled *low*) is given in Figure 10. We model the scheduler inability to observe if the processor is in the standby or lower power state by assigning the same observation (o_{idle}) to the locations labelled *stdby* and *low*. To model the 4 picosecond delay when waking from the low power state, we introduce the locations $wake_1$ and $wake_2$ corresponding to waking up

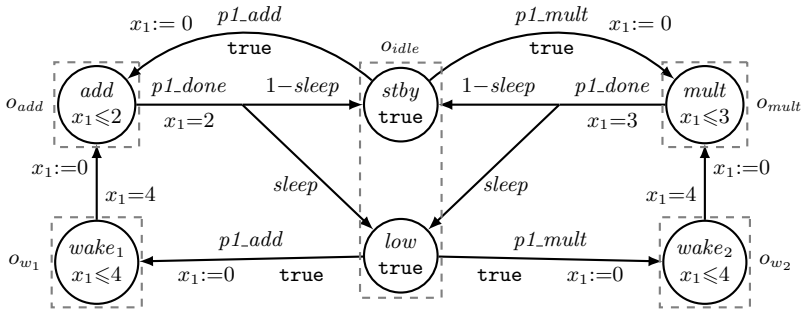

Fig. 10. POPTA of processor P_1 with a low power state.

to perform an add and a multiplication operation respectively. Not included in Figure 10 is the initial location, from which we immediately move, by adding the guard $x{=}0$ to this location, to either the location *low* or *stby* each with probability 0.5. The PRISM module representing P_1 is given in Figure 11 with the variable *sleep1* specified as unobservable. The PTA model with probabilistic task execution times given in [42] can be extended similarly. For both models, we generate optimal schedulers (minimising expected execution time or energy usage) using strategy synthesis.

6.4 The NRL pump

The NRL (Naval Research Laboratory) pump [29] is designed to provide reliable and secure communication over networks of nodes with '*high*' and '*low*' security levels. It prevents a covert channel leaking information from '*high*' to '*low*' through the *timing* of messages and acknowledgements. Communication is buffered and *probabilistic* delays are added to acknowledgements from '*high*' in such a way that the potential for information leakage is minimised, while maintaining network performance. A PTA model is considered in [33].

We model the pump as a POPTA using a hidden variable for a secret value $z \in \{0, 1\}$ (initially set uniformly at random) which '*high*' tries to covertly communicate to '*low*'. The model is the parallel composition of three POPTAs representing '*high*', '*low*' and the pump. This communication is attempted by adding a delay of h_0 or h_1, depending on the value of z, whenever sending an acknowledgement to '*low*'. In the model, '*low*' sends N messages to '*high*' and tries to guess z based on the time taken for its messages to be acknowledged. We consider the maximum probability '*low*' can (either eventually or within some time frame) correctly guess z. We also study the expected time to send all messages and acknowledgements. These properties measure the security and performance aspects of the pump. Results are presented in Figure 12 varying h_1 and N (we fix $h_0{=}2$). They show that increasing either the difference between h_0 and h_1 (i.e., increasing h_1) or the number N of messages sent improve the chance of '*low*' correctly guessing the secret z, at the cost of a decrease in network performance. On the other hand, when $h_0{=}h_1$, however many messages are sent,

```
module P1

  p1 : [0..5];
  // 0 - initial location
  // 1 - inactive (idle or sleep)
  // 2,3 - waking (adding and multiplying)
  // 4 - adding
  // 5 - multiplying
  sleep1 : [0..1]; // 0 - idle and 1 - sleep
  x1 : clock; // local clock

  invariant
     (p1=0 ⇒ x1⩽0) &
     (p1=1 ⇒ true) &
     (p1=2 ⇒ x1⩽4) &
     (p1=3 ⇒ x1⩽4) &
     (p1=4 ⇒ x1⩽2) &
     (p1=5 ⇒ x1⩽3)
  endinvariant

  // initialise
  [start] p1=0 → 0.5 : (p1'=1) & (sleep1'=0) + 0.5 : (p1'=1) & (sleep1'=1);

  // start from sleep state
  [p1_add]  p1=1 & sleep1=1 → (p1'=2) & (x1'=0) & (sleep1'=0); // add
  [p1_mult] p1=1 & sleep1=1 → (p1'=3) & (x1'=0) & (sleep1'=0); // multiply

  // start from idle state
  [p1_add]  p1=1 & sleep1=0 → (p1'=4) & (x1'=0); // add
  [p1_mult] p1=1 & sleep1=0 → (p1'=5) & (x1'=0); // multiply

  // wake from sleep
  [] p1=2 & x1=4 → (p1'=4) & (x1'=0); // add
  [] p1=3 & x1=4 → (p1'=5) & (x1'=0); // multiply

  // finish
  [p1_done] p1=4 & x1=2 → (1−sleep) : (p1'=1) + sleep : (p1'=1) & (sleep1'=1); // add
  [p1_done] p1=5 & x1=3 → (1−sleep) : (p1'=1) + sleep : (p1'=1) & (sleep1'=1);  // multiply

endmodule
```

Fig. 11. PRISM module of processor P_1 with a low power state.

'*low*', as expected, learns nothing of the value being sent and at best can guess correctly with probability 0.5.

6.5 Non-repudiation protocol

The next case study is a non-repudiation protocol for information transfer due to Markowitch & Roggeman [39]. It is designed to allow an originator O to send information to a recipient R while guaranteeing *non-repudiation*, that is, neither party can deny having participated in the information transfer. The initialisation step of the protocol requires O to *randomly* select an integer N in the range $1, \ldots, K$ that is never revealed to R during execution.

In previous analyses [34,42], modelling this step was not possible since no notion of (non-)observability was used. We resolve this by building a POPTA model of the protocol including this step, thus matching Markowitch & Roggeman's original specification. In particular, we include a hidden variable to store

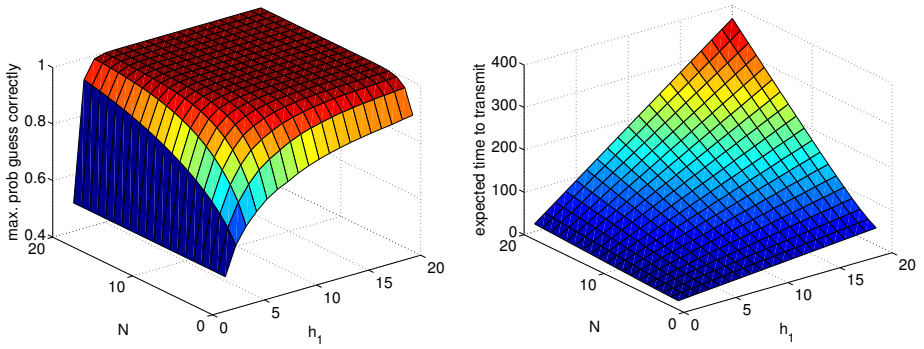

Fig. 12. Analysing security/performance of the NRL pump: (a) maximum probability of covert channel success; (b) maximum expected transmission time.

the random value N. The model is the parallel composition of two component POPTAs representing the originator and the recipient.

We build two POPTA models: a basic model, where R's only malicious behaviour corresponds to stopping early; and a more complex model, where R also has access to a decoder. We also consider a simpler discrete-time POMDP model where the timing information is abstracted and R's only malicious behaviour corresponds to stopping early. We compute the maximum probability that R gains an unfair advantage (obtains the information from O while being able to deny participating). Our results (see Tables 1 and 2) show that, for the basic models, this probability equals $1/K$ when convergence is achieved and that R is more powerful in the complex model.

6.6 The dining cryptographers protocol

This protocol, due to Chaum [18], solves the following problem. A group of N cryptographers are having dinner at their favourite restaurant. The waiter informs them that arrangements have been made for the bill to be paid anonymously: one of the cryptographers might be paying for the dinner, or it might be their master. The cryptographers respect each other's privacy, but would like to know if the master is paying for dinner. The protocol proceeds as follows.

- Each cryptographer flips an unbiased coin and only informs the cryptographer on the right of the outcome.
- Each cryptographer states whether the two coins that it can see (the one it flipped and the one the left-hand neighbour flipped) are the same ('*agree*') or different ('*disagree*'). However, if a cryptographer actually paid for dinner, then the cryptographer instead states the opposite ('*disagree*' if the coins are the same and '*agree*' if the coins are different).

An even number of '*agrees*' indicates the master paid, an odd number that a cryptographer paid. But this provides no additional information as to which cryptographer actually paid.

We model the protocol as a parallel composition of POMDPs: one for each cryptographer and one representing the master. The observable behaviour of the POMDP is with respect to a specific cryptographer. In particular, all the '*agree*' and '*disagree*' announcements are visible to this cryptographer, but only the values of its own and its left-hand neighbour's coins are visible.

In the model we do not impose any requirement on the ordering in which the cryptographers state '*agree*' or '*disagree*', in case this can be used to provide information to the specific cryptographer as to who actually pays. In the initialisation phase, we assume that the master selects, uniformly at random, one of the other cryptographers to pay.

We analyse both the minimum and maximum probability that the specified cryptographer can guess which of the other cryptographers actually pays. We find that, when the approach converges, the maximum probability that the cryptographer can correctly guess which of the other cryptographers pays is the same both before and after the protocol is run, i.e., by selecting one of the other cryptographers uniformly at random. Hence we have demonstrated that the protocol does indeed satisfy the privacy requirement in these cases. Privacy had previously been analysed with PRISM using MDPs (see [44]), however in this work an exponential number of properties needed to be verified, as opposed to the single maximum probabilistic reachability property required when modelling the protocol as a POMDP.

6.7 Grid-world robot

The final case study is based on the POMDP example given in [36]. There is a robot placed randomly an $n \times n$ grid and its goal is to reach the south east corner location. All locations of the grid look identical, i.e., have the same observation, except the target. The robot can perform four different actions corresponding to moving in the four compass directions. There is no change in location if the chosen action would take the robot off the grid. We have constructed POMDP models for the cases when n equals 3 and 4. For both models we have synthesised a controller that optimises (i.e., minimises) the expected number of steps to reach the target and a controller that optimises (i.e., maximises) the probability of reaching the target within k steps.

6.8 Experimental results

Tables 1 and 2 summarise a representative set of experimental results from the analysis of the POMDP and POPTA case studies, respectively. All were run on a 2.8 GHz PC with 8GB RAM. The table shows the parameters used for each model (see [55] for details), the property analysed and various statistics from the analysis: the size of the POMDP (in the case of POPTAs this is the POMDP that is obtained through the digital clocks semantics); number of observations; number of hidden values (i.e., the maximum number of states with the same observation); the grid size (resolution M and total number of points); the time taken; and the results obtained. For comparison, in the rightmost column, we

Case study (parameters)		Property	Verification/strategy synthesis of POMDP							MDP result
			States	Num. obs.	Num. hidd.	Res. (M)	Grid points	Time (s)	Result (bounds)	
wireless network sched. (c T K)	2 2 20	$\mathtt{R}_{\min=?}[\mathtt{F}\ done]$ (dropped packets)	754	214	4	16	151,811	34.2	[19.3, 19.3]	15.9
	2 4 20		2,029	533	4	16	457,233	190.3	[10.2, 10.3]	8.26
	2 8 20		4,589	1,173	4	16	1,077,393	893.3	[3.2, 3.2]	2.56
	3 3 8		1,714	234	8	8	1,171,699	428.4	[9.4, 9.9]	6.61
	3 4 8		4,777	617	8	6	967,729	654.6	[3.63, 4.20]	2.51
	3 5 8		6,825	873	8	6	1,407,025	1,461	[2.00, 2.34]	1.33
wireless network sched. (c T K)	2 2 8	$\mathtt{R}_{\max=?}[\mathtt{F}\ done]$ (priorities cumul.)	1,534	410	4	12	158,159	43.5	[125, 125]	143
	2 4 8		3,577	921	4	12	387,193	174.3	[180, 180]	191
	2 8 8		7,673	1,945	4	12	853,113	707.9	[222, 222]	225
	3 3 2		3,932	524	8	4	133,915	90.51	[44.1, 47.2]	56.8
	3 4 2		5,971	779	8	4	215,899	167.8	[55.5, 58.9]	67.0
	3 5 2		8,019	1,035	8	4	300,259	277.6	[64.1, 67.4]	73.6
nrp discrete (K)	4	$\mathtt{P}_{\max=?}[\mathtt{F}\ unfair]$	39	21	5	4	173	0.1	[0.25, 0.375]	1.0
	4		39	21	5	12	1,685	0.2	[0.25, 0.25]	1.0
	8		125	41	9	4	2,385	0.7	[0.13, 0.38]	1.0
	8		125	41	9	16	1,038,321	124.0	[0.13, 0.18]	1.0
dining crypt (N)	3	$\mathtt{P}_{\min=?}[\mathtt{F}\ paid]$	179	90	6	4	606	0.16	[0.5, 0.5]	0.0
	4		964	282	15	8	674,398	210.5	[0.082, 0.333]	0.0
	5		4,741	842	36	4	746,020	907.5	[0.0, 0.25]	0.0
	6		22,406	2,458	85	2	210,256	505.7	[0.0, 0.2]	0.0
dining crypt (N)	3	$\mathtt{P}_{\max=?}[\mathtt{F}\ paid]$	179	90	6	4	606	0.16	[0.5, 0.5]	1.0
	4		964	90	15	8	674,398	209.8	[0.333, 0.568]	1.0
	5		4,741	842	36	4	746,020	1,044	[0.25, 1.0]	1.0
	6		22,406	2,458	85	2	210,256	1,046	[0.2, 1.0]	1.0
grid (n)	3	$\mathtt{R}_{\min=?}[\mathtt{F}\ target]$	11	3	9	4	331	0.1	[2.63, 2.88]	2.0
	3		11	3	9	8	6,436	1.0	[2.84, 2.88]	2.0
	4		17	3	16	4	3,061	0.8	[3.27, 4.13]	2.73
	4		17	3	16	8	319,771	60.6	[3.91, 4.13]	2.73
grid (n k)	3 1	$\mathtt{P}_{\max=?}[\mathtt{F}^{\leqslant k}\ target]$	18	4	9	8	12,871	1.7	[0.13, 0.13]	0.25
	3 2		27	6	9	4	991	0.4	[0.38, 0.38]	0.75
	3 3		36	8	9	4	1321	0.3	[0.75, 0.75]	1.0
	3 4		45	10	9	2	181	0.1	[1.0, 1.0]	1.0
	4 2		48	6	16	3	2,041	0.7	[0.2, 0.2]	0.33
	4 4		80	10	16	6	193,801	24.5	[0.53, 0.63]	1.0
	4 5		96	12	16	6	232,561	33.5	[0.80, 0.85]	1.0
	4 6		112	14	16	2	841	0.6	[1.0, 1.0]	1.0

Table 1. Experimental results from verification/strategy synthesis of POMDPs.

show what result is obtained if the POMDP or POPTA is treated as an MDP or PTA (by making everything observable).

On the whole, we find that the performance of our prototype is good, especially considering the complexity of the POMDP solution methods and the fact that we use a relatively simple grid mechanism. We are able to analyse POPTAs whose integer semantics yields POMDPs of up to 60,000 states, with experiments usually taking just a few seconds and, at worst, 20 minutes. These are, of course, smaller than the standard PTA or MDP models that can be verified, but we were still able to obtain useful results for several case studies.

The values in the rightmost column of Tables 1 and 2 illustrate that the results obtained with POMDPs and POPTAs would not have been possible using an MDP or PTA model, i.e., where all states of the model are observable. In the *wireless network* case study in the MDP model the scheduler can see the status of the channels, and hence use this information to decrease the number of dropped packets and increase the cumulate reward based on the priorities of

Case study (parameters)		Property	Verification/strategy synthesis of POPTA							PTA result
			States ($\llbracket \mathsf{P} \rrbracket_{\mathbb{N}}$)	Num. obs.	Num. hidd.	Res. (M)	Grid points	Time (s)	Result (bounds)	
scheduler	0.25	$\mathtt{R}_{\mathtt{min=?}}[\mathtt{F}\ done]$ (exec. time)	2,090	1,619	2	2	2,537	1.6	[15.84, 15.84]	15.59
basic	0.5		2,090	1,619	2	2	2,537	1.3	[21.1, 21.1]	18.0
(*sleep*)	0.75		2,090	1,619	2	4	3,463	1.9	[19.25, 19.25]	20.38
scheduler	0.25	$\mathtt{R}_{\mathtt{min=?}}[\mathtt{F}\ done]$ (energy cons.)	2,090	1,619	2	2	2,537	1.0	[1.849, 1.849]	1.834
basic	0.5		2,090	1,619	2	2	2,537	1.8	[2.149, 2.149]	2.119
(*sleep*)	0.75		2,090	1,619	2	4	3,463	2.1	[2.444, 2.444]	2.399
scheduler	0.25	$\mathtt{R}_{\mathtt{min=?}}[\mathtt{F}\ done]$ (exec. time)	5,484	4,204	2	2	6,662	4.5	[16.21, 16.21]	15.96
prob	0.5		5,484	4,204	2	2	6,662	3.4	[18.73, 18.73]	18.23
(*sleep*)	0.75		5,484	4,204	2	4	9,154	5.2	[21.29, 21.29]	20.53
scheduler	0.25	$\mathtt{R}_{\mathtt{min=?}}[\mathtt{F}\ done]$ (energy cons.)	5,484	4,204	2	4	6,662	3.5	[1.890, 1.890]	1.875
prob	0.5		5,484	4,204	2	2	6,662	4.0	[2.177, 2.177]	2.147
(*sleep*)	0.75		5,484	4,204	2	4	9,154	4.7	[2.461, 2.461]	2.416
pump (h_1 N)	16 2	$\mathtt{P}_{\mathtt{max=?}}[\mathtt{F}\ guess]$	243	145	3	2	342	0.7	[0.940, 0.992]	1.0
	16 2		243	145	3	40	4,845	4.0	[0.940, 0.941]	1.0
	16 16		1,559	803	3	2	2,316	16.8	[0.999, 0.999]	1.0
pump (h_1 N D)	8 4 50	$\mathtt{P}_{\mathtt{max=?}}[\mathtt{F}^{\leqslant D}\ guess]$	12,167	7,079	3	2	17,256	11.0	[0.753, 0.808]	1.0
	8 4 50		12,167	7,079	3	12	68,201	36.2	[0.763, 0.764]	1.0
	16 8 50		26,019	13,909	3	2	38,130	52.8	[0.501, 0.501]	1.0
	16 8 100		59,287	31,743	3	2	86,832	284.8	[0.531, 0.532]	1.0
nrp basic (K)	4	$\mathtt{P}_{\mathtt{max=?}}[\mathtt{F}\ unfair]$	365	194	5	8	5,734	0.8	[0.25, 0.281]	1.0
	4		365	194	5	24	79,278	5.9	[0.25, 0.25]	1.0
	8		1,273	398	9	4	23,435	4.8	[0.125, 0.375]	1.0
	8		1,273	398	9	8	318,312	304.6	[0.125, 0.237]	1.0
nrp complex (K)	4	$\mathtt{P}_{\mathtt{max=?}}[\mathtt{F}\ unfair]$	1,501	718	5	4	7,480	2.1	[0.438, 0.519]	1.0
	4		1,501	718	5	12	72,748	14.8	[0.438, 0.438]	1.0
	8		5,113	1,438	9	2	16,117	6.1	[0.344, 0.625]	1.0
	8		5,113	1,438	9	4	103,939	47.1	[0.344, 0.520]	1.0

Table 2. Experimental results from verification/strategy synthesis of POPTAs.

packets. In the *crypt* and *pump* case studies, the MDP and PTA give probability 1 of guessing correctly (e.g., in the *pump* example, '*low*' can simply read the value of the secret). Similarly, for the *nrp* models, the PTA gives probability 1 of unfairness because the recipient can read the random value the originator selects. For the *scheduler* example, the PTA model gives a scheduler with better time/energy consumption but which cannot be implemented in practice since the power state is not visible. In similar fashion, for the *grid* example, we see that optimal strategy is improved if the precise location on the grid is available.

Another positive aspect is that, in many cases, the bounds generated are very close (or even equal, in which case the results are exact). For the *pump* and *scheduler* case studies, we included results for the smallest grid resolution M required to ensure the difference between the bounds is at most 0.001. In many cases, this is achieved with relatively small values (for the *scheduler* case study, in particular, M is at most 4). For the cases we were unable to do this we have instead included the results for the largest grid resolution for which POMDP solution was possible: higher values could not be handled within the memory constraints of our test machine. We anticipate being able to improve this in the future by adapting more advanced approximation methods for POMDPs [48]. For the *crypt* case study, as we increase the number of cryptographers, we find that the over approximations obtained through the approximate solution of the belief MDP are coarse (0.0 and 1.0 for minimum and maximum probabilities,

respectively) while the under approximations obtained through synthesis are precise. This appears to be the due to large number of hidden values in the POMDP compared to the other case studies and our prototype implementation using only a basic approximation method.

7 Conclusions

We have proposed novel methods for the verification and control of partially observable, probabilistic systems for both discrete and dense models of time. We have used temporal logics to define probabilistic, timed properties and reward measures. For discrete-time models, the techniques developed are based on a belief space approximation. For dense-time models we have demonstrated that the digital clocks discretisation preserves the properties of interest, which allows us to employ the techniques developed for the discrete-time case. We have implemented this work in an extension of the probabilistic model checker PRISM and demonstrated the effectiveness on several case studies.

Future directions include more efficient approximation schemes, zone-based implementations and development of the theory for unobservable clocks. Allowing unobservable clocks, as mentioned previously, will require moving to partially observable stochastic games and restricting the class of strategies.

Acknowledgments. This work was partly supported by the EPSRC grant "Automated Game-Theoretic Verification of Security Systems" (EP/K038575/1) and the PRINCESS project, funded by the DARPA BRASS programme. We also gratefully acknowledge support from Google Summer of Code 2014 and thank the anonymous referees for their helpful comments.

References

1. de Alfaro, L.: The verification of probabilistic systems under memoryless partial-information policies is hard. In: Proc. 2nd Probabilistic Methods in Verification Workshop (PROBMIV'99). pp. 19–32 (1999), Birmingham University Research Report CSR-99-9
2. Alur, R., Dill, D.: A theory of timed automata. Theoretical Computer Science 126, 183–235 (1994)
3. Alur, R., Henzinger, T.: Reactive modules. Formal Methods in System Design 15(1), 7–48 (1999)
4. Alur, R., La Torre, S., Pappas, G.: Optimal paths in weighted timed automata. Theoretical Computer Science 318(3), 297–322 (2004)
5. Baier, C., Bertrand, N., Größer, M.: On decision problems for probabilistic Büchi automata. In: Amadio, R. (ed.) Proc. 11th Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS'08). LNCS, vol. 4962, pp. 287–301. Springer (2008)
6. Baier, C., Größer, M., Leucker, M., Bollig, B., Ciesinski, F.: Controller synthesis for probabilistic systems. In: Lévy, J.J., Mayr, E., Mitchell, J. (eds.) Proc. 3rd IFIP Int. Conf. Theoretical Computer Science (TCS 2006). pp. 493–506. Kluwer (2004)

7. Behrmann, G., Fehnker, A., Hune, T., Larsen, K., Pettersson, P., Romijn, J., Vaandrager, F.: Minimum-cost reachability for linearly priced timed automata. In: Benedetto, M.D., Sangiovanni-Vincentelli, A. (eds.) Proc. 4th Int. Conf. Hybrid Systems: Computation and Control (HSCC 2001). LNCS, vol. 2034, pp. 147–162. Springer (2001)
8. Bouyer, P., D'Souza, D., Madhusudan, P., Petit, A.: Timed control with partial observability. In: Proc. 15th Int. Conf. Computer Aided Verification (CAV'03). LNCS, vol. 2725, pp. 180–192 (2003)
9. Bouyer, P., Fahrenberg, U., Larsen, K., Markey, N.: Quantitative analysis of real-time systems using priced timed automata. Communications of the ACM 54(9), 78–87 (2011)
10. Brázdil, T., Brožek, V., Forejt, V., Kučera, A.: Stochastic games with branching-time winning objectives. In: Proc. 21th Int. Symp. Logic in Computer Science (LICS 2006). pp. 349–358. IEEE Computer Society (2006)
11. Cassandra, A.: A survey of POMDP applications (1998), presented at the AAAI Fall Symposium, 1998 (pomdp.org/pomdp/papers/applications.pdf)
12. Cassez, F., David, A., Larsen, K., Lime, D., Raskin, J.F.: Timed control with observation based and stuttering invariant strategies. In: Namjoshi, K., Yoneda, T., Higashino, T., Okamura, Y. (eds.) Proc. 5th Int. Symp. Automated Technology for Verification and Analysis (ATVA'07). LNCS, vol. 4762, pp. 192–206. Springer (2007)
13. Cerný, P., Chatterjee, K., Henzinger, T., Radhakrishna, A., Singh, R.: Quantitative synthesis for concurrent programs. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proc. 23rd Int. Conf. Computer Aided Verification (CAV'11). LNCS, vol. 6806, pp. 243–259. Springer (2011)
14. Chatterjee, K., Chmelík, M., Gupta, R., Kanodia, A.: Qualitative analysis of POMDPs with temporal logic specifications for robotics applications. In: Proc. Int. Conf. Robotics and Automation, (ICRA'15). pp. 325–330. IEEE Computer Society (2015)
15. Chatterjee, K., Chmelík, M., Gupta, R., Kanodia, A.: Optimal cost almost-sure reachability in POMDPs. Artificial Intelligence 234, 26–48 (2016)
16. Chatterjee, K., Chmelík, M., Tracol, M.: What is decidable about partially observable Markov decision processes with omega-regular objectives. In: Proc. 22nd EACSL Annual Conf. Computer Science Logic (CSL'13). LIPIcs, vol. 23, pp. 165–180. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2013)
17. Chatterjee, K., Doyen, L.: Partial-observation stochastic games: How to win when belief fails. ACM Transactions on Computational Logic 15(2) (2014)
18. Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology 1, 65–75 (1988)
19. Eaves, B.: A course in triangulations for solving equations with deformations. Springer (1984)
20. Finkbeiner, B., Peter, H.: Template-based controller synthesis for timed systems. In: Flanagan, C., König (eds.) Proc. 18th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12). LNCS, vol. 7214, pp. 392–406 (2012)
21. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated verification techniques for probabilistic systems. In: Bernardo, M., Issarny, V. (eds.) Formal Methods for Eternal Networked Software Systems (SFM'11). LNCS, vol. 6659, pp. 53–113. Springer (2011)

22. Giro, S., Rabe, M.: Verification of partial-information probabilistic systems using counterexample-guided refinements. In: Chakraborty, S., Mukund, M. (eds.) Proc. 10th Int. Symp. Automated Technology for Verification and Analysis (ATVA'12). LNCS, vol. 7561, pp. 333–348. Springer (2012)
23. Gopalan, A., Caramanis, C., Shakkottai, S.: Wireless scheduling with partial channel state information: large deviations and optimality. Queueing Systems 80(4), 293–340 (2015)
24. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6(5), 512–535 (1994)
25. Henzinger, T., Manna, Z., Pnueli, A.: What good are digital clocks? In: Kuich, W. (ed.) Proc. 19th Int. Coll. Automata, Languages and Programming (ICALP'92). LNCS, vol. 623, pp. 545–558. Springer (1992)
26. Henzinger, T., Nicollin, X., Sifakis, J., Yovine, S.: Symbolic model checking for real-time systems. Information and Computation 111(2), 193–244 (1994)
27. Jagannathan, K., Menache, I., Modiano, E., Mannor, S.: A state action frequency approach to throughput maximization over uncertain wireless channels. Internet Mathematics 9(2–3), 136–160 (2013)
28. Johnston, L., Krishnamurthy, V.: Opportunistic file transfer over a fading channel: A POMDP search theory formulation with optimal threshold policies. IEEE Trans. Wireless Communications 5(2), 394–405 (2006)
29. Kang, M., Moore, A., Moskowitz, I.: Design and assurance strategy for the NRL pump. Computer 31(4), 56–64 (1998)
30. Kemeny, J., Snell, J., Knapp, A.: Denumerable Markov Chains. Springer, 2nd edn. (1976)
31. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proc. 23rd Int. Conf. Computer Aided Verification (CAV'11). LNCS, vol. 6806, pp. 585–591. Springer (2011)
32. Kwiatkowska, M., Norman, G., Parker, D., Sproston, J.: Performance analysis of probabilistic timed automata using digital clocks. Formal Methods in System Design 29, 33–78 (2006)
33. Lanotte, R., Maggiolo-Schettini, A., Tini, S., Troina, A., Tronci, E.: Automatic analysis of the NRL pump. In: Proc MEFISTO Project 2003: Formal Methods for Security and Time. ENTCS, vol. 99, pp. 245–266. Elsevier (2014)
34. Lanotte, R., Maggiolo-Schettini, A., Troina, A.: Automatic analysis of a non-repudiation protocol. In: Proc. 2nd Int. Workshop Quantitative Aspects of Programming Languages (QAPL'04). ENTCS, vol. 112, pp. 113–129. Elsevier (2005)
35. Li, C., Neely, M.J.: Network utility maximization over partially observable Markovian channels. In: Proc. Int. Symp. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt'11). pp. 17–24. IEEE Computer Society (2011)
36. Littman, M., Cassandra, A., Kaelbling, L.: Learning policies for partially observable environments: Scaling up. Tech. Rep. CS-95-11, Department of Computer Science, Brown University (1995)
37. Lovejoy, W.: Computationally feasible bounds for partially observed Markov decision processes. Operations Research 39(1), 162–175 (1991)
38. Madani, O., Hanks, S., Condon, A.: On the undecidability of probabilistic planning and related stochastic optimization problems. Artificial Intelligence 147(1–2), 5–34 (2003)
39. Markowitch, O., Roggeman, Y.: Probabilistic non-repudiation without trusted third party. In: Proc. 2nd Workshop Security in Communication Networks (1999)

40. McCallum, R.: Overcoming incomplete perception with utile distinction memory. In: Proc. 10th Int. Conf. Machine Learning (ICML 1993). pp. 190–196. Morgan Kauffman (1993)
41. Norman, G., Parker, D., Zou, X.: Verification and control of partially observable probabilistic real-time systems. In: Sankaranarayanan, S., Vicario, E. (eds.) Proc. 13th Int. Conf. Formal Modelling and Analysis of Timed Systems (FORMATS'15). LNCS, vol. 9268, pp. 240–255. Springer (2015)
42. Norman, G., Parker, D., Sproston, J.: Model checking for probabilistic timed automata. Formal Methods in System Design 43(2), 164–190 (2013)
43. Poupart, P.: Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes. Ph.D. thesis, University of Toronto (2005)
44. PRISM web site, www.prismmodelchecker.org
45. Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley and Sons (1994)
46. Roscoe, A.W.: The theory and practice of concurrency. Prentice-Hall (1997)
47. Segala, R., Lynch, N.: Probabilistic simulations for probabilistic processes. Nordic Journal of Computing 2(2), 250–273 (1995)
48. Shani, G., Pineau, J., Kaplow, R.: A survey of point-based POMDP solvers. Autonomous Agents and Multi-Agent Systems 27(1), 1–51 (2013)
49. Svoreňová, M., Chmelík, M., Leahy, K., Eniser, H., Chatterjee, K., Černá, I., Belta, C.: Temporal logic motion planning using POMDPs with parity objectives: Case study paper. In: Proc. 18th Int. Conf. Hybrid Systems: Computation and Control (HSCC'15). pp. 233–238. ACM (2015)
50. Tripakis, S.: Verifying progress in timed systems. In: Katoen, J.P. (ed.) Proc. 5th Int. AMAST Workshop Real-Time and Probabilistic Systems (ARTS'99). LNCS, vol. 1601, pp. 299–314. Springer (1999)
51. Tripakis, S., Yovine, S., Bouajjan, A.: Checking timed Büchi automata emptiness efficiently. Formal Methods in System Design 26(3), 267–292 (2005)
52. Yang, L., Murugesan, S., Zhang, J.: Real-time scheduling over Markovian channels: When partial observability meets hard deadlines. In: Proc. Global Telecommunications Conference (GLOBECOM'11). pp. 1–5. IEEE Computer Society (2011)
53. Yu, H.: Approximate Solution Methods for Partially Observable Markov and Semi-Markov Decision Processes. Ph.D. thesis, MIT (2006)
54. Yu, H., Bertsekas, D.: Discretized approximations for POMDP with average cost. In: Proc. 20th Conf. Uncertainty in Artificial Intelligence (UAI'04). pp. 619–627. AUAI Press (2004)
55. Case studies, examples and prototype tool for verification and strategy synthesis of POMDPs and POPTAs, www.prismmodelchecker.org/files/rts-poptas/

A Construction of the Belief MDP

For the convenience of the reader, we have included below, the construction of the belief MDP for a POMDP $\mathsf{M}=(S, \bar{s}, A, P, R, \mathcal{O}, obs)$ (see Definition 5). We use the standard notation from the POMDP literature, for example $\mathbf{Pr}[\, s', a, b\,]$ denotes the probability that the next state is s', the action a is performed and the current belief is b and $\mathbf{Pr}[\, o \,|\, a, b\,]$ denote the probability of observing o conditioned on the action being a is performed and that the current belief state is b.

First, let us suppose we are in a belief state b, perform action a and observe o. Based on this new information we move to a new belief state $b^{a,o}$ using the observation function of M. Let us now construct this new belief state. First, by construction, we have for any $s' \in S$:

$$\begin{aligned} b^{a,o}(s') &= \mathbf{Pr}[\, s' \,|\, o, a, b \,] \\ &= \frac{\mathbf{Pr}[\, s', o, a, b \,]}{\mathbf{Pr}[\, o, a, b \,]} && \text{by definition of conditional probabilities} \\ &= \begin{cases} \dfrac{\mathbf{Pr}[\, s', a, b \,]}{\mathbf{Pr}[\, o, a, b \,]} & \text{if } obs(s')=o \\ 0 & \text{otherwise} \end{cases} && \text{by definition of } obs. \end{aligned}$$

Now considering the numerator in the first case, since the value of s' is dependent on the other values:

$$\begin{aligned} \mathbf{Pr}[\, s', a, b \,] &= \mathbf{Pr}[\, s' \,|\, a, b \,] \cdot \mathbf{Pr}[\, a, b \,] \\ &= \mathbf{Pr}[\, s' \,|\, a, b \,] \cdot 1 && \text{since } b \text{ and } a \text{ are fixed} \\ &= \textstyle\sum_{s \in S} \mathbf{Pr}[\, s' \,|\, a, s \,] \cdot b(s) && \text{definition of } b \\ &= \textstyle\sum_{s \in S} P(s,a)(s') \cdot b(s) && \text{definition of } P. \end{aligned}$$

For the denominator since o is dependent on b and a we have:

$$\begin{aligned} \mathbf{Pr}[\, o, a, b \,] &= \mathbf{Pr}[\, o \,|\, a, b \,] \cdot \mathbf{Pr}[\, a, b \,] \\ &= \mathbf{Pr}[\, o \,|\, a, b \,] \cdot 1 && \text{since } b \text{ and } a \text{ are fixed} \\ &= \textstyle\sum_{s \in S} \mathbf{Pr}[\, o \,|\, a, s \,] \cdot b(s) && \text{by definition of } b \\ &= \textstyle\sum_{s \in S} \left(\sum_{s' \in S} \mathbf{Pr}[\, o \,|\, s', a \,] \cdot P(s,a)(s') \right) \cdot b(s) && \text{by definition of } P \\ &= \textstyle\sum_{s \in S} \left(\sum_{s' \in S \wedge obs(s')=o} P(s,a)(s') \right) \cdot b(s) && \text{by definition of } obs. \end{aligned}$$

Combining these results (and rearranging) we have:

$$b^{a,o}(s') = \begin{cases} \dfrac{\sum_{s \in S} P(s,a)(s') \cdot b(s)}{\sum_{s \in S} b(s) \cdot \left(\sum_{s'' \in S \wedge obs(s'')=o} P(s,a)(s'') \right)} & \text{if } obs(s')=o \\ 0 & \text{otherwise.} \end{cases}$$

Now using this we can define the probabilistic transition function of the belief MDP $\mathcal{B}(\mathsf{M})$. Suppose we are in a belief state b and we perform action a. Now the probability we move to belief b' is given by:

$$P_T^{\mathcal{B}}(b,a)(b') = \mathbf{Pr}[\, b' \,|\, a, b \,] = \textstyle\sum_{o \in O} \mathbf{Pr}[\, b' \,|\, o, a, b \,] \cdot \mathbf{Pr}[\, o \,|\, a, b \,].$$

The first term in the summation ($\mathbf{Pr}[\, b' \,|\, o, a, b \,]$) is the probability of being in belief b' after being in belief b, performing action a and observing o. Therefore by definition of $b^{a,o}$, this probability will equal 1 if b' equals $b^{a,o}$ and 0 otherwise.

For the second term, as in the derivation of the denominator above, we have:

$$\mathbf{Pr}[\, o \,|\, a, b \,] = \textstyle\sum_{s \in S} b(s) \cdot \left(\sum_{s' \in S \wedge obs(s')=o} P(s,a)(s') \right)$$

This completes the construction of the transition function of the belief MDP.

It remains to construct the reward structure $R^{\mathcal{B}}$ of the belief MDP. In this case, we just have to take the expectation of the state and action reward functions with respect to the current belief, i.e. for any belief state b and action a:

$$R_S^{\mathcal{B}}(b) = \sum_{s \in S} R_S(s) \cdot b(s)$$
$$R_A^{\mathcal{B}}(b, a) = \sum_{s \in S} R_A(s, a) \cdot b(s)\,.$$