Adaptation in partially observable environments via model checking

Rocco De Nicola
IMT Institute for Advanced Studies Lucca, Italy
rocco.denicola@imtlucca.it

Michele Loreti
Università degli Studi di Firenze, Italy
michele.loreti@unifi.it

Marco Tinacci
IMT Institute for Advanced Studies Lucca, Italy
marco.tinacci@imtlucca.it

Modern software systems consist of a large number of agents that cooperate and compete to achieve *local* and *global* goals. Local goals are specific of single agents while the global ones refer to the expected emerging behaviour resulting from the interaction of many agents. Agents are often located in an environment that can impact on their *goals*. For this reason it is crucial to have mechanisms that allow each agent to adapt its behaviour to the changing environmental conditions while guaranteeing goals achievement. When a change in the environment is perceived, each agent has to use its knowledge to perform specific actions and *adapt* its behaviour. Unfortunately, only in rare cases agents have a complete view of the context where they are operating; often only a partial view is available. In this paper we show how formal tools, and specifically probabilistic model checking, can be used to support dynamic adaptation of agents when only a partial *observations* of the environment are available. The proposed approach guarantees that each agent is able to select those *adaptation actions* that maximizes the probability of achieving the required goal. In the paper, system behaviour is described in terms of POMDP (Partial Observable Markov Decision Processes) while goals are specified using an appropriate variant of LTL. A simple running example, based on a simple robotic scenario, is used to show the effectiveness of the proposed approach.

1 Introduction

Many modern software systems consist of a large number of agents that cooperate and compete [] to achieve *local* and *global* goals. Local goals are specific of single agents while the global ones refer to the expected emerging behaviour resulting from the interaction of many agents. Agents are often located in an environment that can impact on their *goals*. For this reason it is crucial to have mechanisms that allow each agent to adapt its behaviour to the changing environmental conditions while guaranteeing goals achievement. When a change in the environment is perceived, each agent has to use its knowledge to perform specific actions and *adapt* its behaviour. Unfortunately, only in rare cases agents have a complete view of the context where they are operating; often only a partial view is available. It is then important to develop tools to be used to support dynamic adaptation of agents also in presence of partial information about the operating environment.

Adaptive systems The issue of programming adaptive agents is present in many scenarios. Typically, each adaptive agent does not have complete information of the activities taking place in the surrounding environment. Thus, methodologies are needed to take decisions to adapt to circumstances. The information about the state of the environment are usually stored in a set of *control data* that are used

Submitted to:
QAPL 2016

for *adaptation*. A system is said to be adaptive when its behaviour and its run-time decisions depend directly upon this set of *control data* [6].

Examples of adaptive systems involve agents like robots that have to navigate towards a light source, gather resources, rescue victims while avoiding to fall into traps or to interfere with each other. Another example can be found in cloud computing scenarios where physical machines have to adapt in order to guarantee availability of specific virtual ones while guaranteeing availability and quality of service parameters agreed with the clients.

Methodology In this paper, we propose a general methodology to be used when dealing with adaptive agents that interacts with an external environment by executing actions and perceiving signals. The environment is formed by all the components that are external to the agent; the controller of each agent exploits the observable history (sequences of actions and observations) to decide the action to perform in order to achieve the specific target in an optimal way. The target is formally specified at design-time by means of a temporal logic formula that expresses properties of observation sequences in probabilistic and partially observable domains. A methodology is introduced to resolve agents' choices by relying on model checking [1]; the action to be performed are determined by taking into account the path that maximizes the probability of satisfying the provided target formula.

We consider systems whose behaviour is fully described but such that their current state is only partially observable. Agents are described by means of nondeterministic models and the environment with probabilistic ones. By composing these two systems we obtain a partially observable process in which every state is split in two parts: the controller, that is fully observable, and the environment, that is not directly observable. The history of performed actions and perceived observations is used to refine a probabilistic belief about the current state.

The classical model checking algorithm is "adapted" to partially observable models in order to check the probabilities to satisfy the temporal logic formula that describes the goal. The partially observable model is transformed into a fully observable one, observations are used to label states and the measure of satisfiability of a property is reduced to the one in a fully observable context. The adaptation strategy consists then in choosing the action that maximize the chances to satisfy the goal formula. Depending on the needs, it might be appropriate to minimize the risk or to maximize the opportunities.

Since the (storage and computing) resources available to the agents are often limited, the model checking approach may look expensive. But, it has to be considered that, we rely on a two steps algorithm; the first step has an exponential complexity while the second one is polynomial. The former one is computed just once and only the latter has to be computed at run-time.

A running example In the paper we will use a simple robotic scenario as a running example. We consider a robot that moves inside an arena with other robots. The proximity sensors of the robot analyse the surrounding area to locate obstacles and their positions; the acquired information is then used by the robot to determine the direction to take.

Figure 1 provides a pictorial rendering of the scenario; the arena is represented as a 5×5 grid and is bounded by four walls. The dashed grid represents the agents' walkable paths. We assume that at any step a robot can move from a cross to an adjacent one. The arena contains four agents; the main robot, the black one, and three white robots that, it is assumed, follow a random trajectory. The main robot monitors the

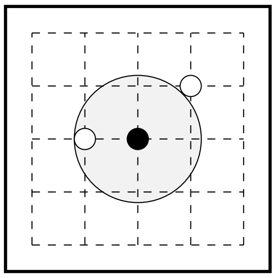

Figure 1: Bird's-eye view representation of the robot arena scenario

five local positions (north, south, east, west and the central position) via proximity sensors. It is able to perceive other agents within a range of visibility represented by a gray circle; in the figure, it can perceive just one agent at west.

In this kind of setting we may want to attribute different goals to the black robot, like collision-avoidance or tracking the white robots' movement. We use this simple scenario to show how goals can be formally represented and, starting from them, how good the results obtained by the synthesis of an adaptive controller are.

Structure of the paper The rest of the paper is organized as follows. We start by introducing the basic structures and languages that we have chosen to represent agents' behaviour and environment; then we extend them to model the adaptation problem and to express goals. Subsequently, we prove correctness of the proposed transformations, present the adaptation algorithm, split into an off-line and on-line phase, and discuss its complexity. Finally, we show some results of the proposed method applied to the robotics example.

2 Background

In this section we introduce the preliminary concepts that provide the foundational base for the proposed approach. First we will recall basic definitions of Labeled Transition System (LTS), Discrete-Time Markov Chain (DTMC) and Markov Decision Processes (MDPs). Then we consider the partially observable extensions of the last two models that are, respectively, Hidden Markov Models (HMMs) and Partially Observable Markov Decision Processes (POMDPs).

In this paper we will use the following notation: $[i..j] \equiv i, i+1, \ldots, j-1, j$ represents the sequence of natural numbers from i to j, $[..j] \equiv [0..j]$, $\Delta(X)$ is the set of all the discrete probability distributions over any domain X and Y^{ω} is the set of all the infinite sequences of elements of Y. We will adopt a name-space structure to ease the notation, when a structure defined as $X = \langle Y \rangle$ we can univocally index the internal element as Y_X. We omit the index when it is clear from the context.

Since we need to describe different kinds of paths for different models, we introduce a uniform notation. Let $Paths^X$ be the set of infinite paths of a model X, we denote with $FPaths^X$ the set of prefixes of every path in $Paths^X$. Let $\pi = \pi_0, \pi_1, \ldots$ be a path in $Paths^X$ where every element $\pi_i = (y_0, y_1, \ldots, y_n) \in Y_0 \times Y_1 \times \cdots \times Y_n$ $(i = 0, 1, \ldots)$ is a tuple of elements y_i from sets Y_j $(j = 0, 1, \ldots, n)$, we denote with $\pi_{i,Y_j} = y_j$ the specific element of the tuple and with $\pi_{Y_j} = \pi_{0,Y_j}, \pi_{1,Y_j}, \ldots$ the selection applied to every tuple of the path. Let $\pi = \pi_0, \ldots, \pi_n$, the notation $|\pi| = n+1$ is used to indicate the length of a finite path. Moreover, we use the previously introduced notation $[i..j]$ to select specific subsequences of a path π (finite or infinite): $\pi[i..j] = \pi_i, \ldots, \pi_j$, $\pi[..j] = \pi_0, \ldots, \pi_j$ and $\pi[i..] = \pi_i, \ldots$ ($\pi[i..]$ stays for $\pi[i, \ldots, |\pi|]$ for finite paths).

Transition systems

LTSs are typically used to describe potential behaviour of discrete systems. Each LTS consists of a set of *states*, representing the possible system configurations. a set of *actions*, identifying the activities that an be performed in the system, and a *labeled transition relation*, describing the possible evolution of a state to another when an activity is executed.

Definition 1 (LTS)**.** A *deterministic* LTS is $\mathscr{L} = \langle \mathscr{S}, \mathscr{A}, T \rangle$ where $\mathscr{S}$ is the set of states, $\mathscr{A}$ is the set of actions and $T : \mathscr{S} \times \mathscr{A} \to \mathscr{S} \cup \{\bot\}$ is the transition function.

As anticipated in the introduction, in our approach we consider our system composed by two parts. One that is completely known and the other that is only partially observable. A LTS is used to model the behaviour of the agent that is known and for which we aim at synthesizing a successful strategy that that maximizes the chance of achieving a required goal. Since the controller has a *deterministic behaviour*, in the definition above we limit our attention to only *deterministic* LTS. Let $\mathscr{L} = \langle \mathscr{S}, \mathscr{A}, T \rangle$, we will write $s_1 \xrightarrow{a}_{\mathscr{L}} s_2$ if and only if $T(s_1, a) = s_2 \neq \bot$ (we use $s_1 \xrightarrow{a} s_2$ when $\mathscr{L}$ is clear from the context). Moreover, we let $init(s) = \{a \in \mathscr{A} \mid T(s,a) \neq \bot\}$ denote the set of actions that can be performed starting from state s. Finally, a (memoryless) scheduler for $\mathscr{L}$ is a function $\eta : S \to \mathscr{A}$ and $Sched^{\mathscr{L}}$ denote the set of every scheduler of $\mathscr{L}$.

Example 1. In our running example the agent modeled via the LTS is the *black robot* of Figure 1. If we consider an arena of size $k \times k$, the states of the LTS consists of the set of pairs (i, j), with $1 \leq i, j \leq k$, representing the position of the robot in the arena. The set of actions is $\{\mathsf{north}, \mathsf{east}, \mathsf{south}, \mathsf{west}, \mathsf{here}\}$ where the first three actions represent possible robot movements while the last one indicates that the robot remains in the current position. Transition relation is:

$$(i,j) \xrightarrow{\mathsf{north}} (i, \min(k, j+1)) \quad (i,j) \xrightarrow{\mathsf{east}} (\min(i+1,k), j)$$

$$(i,j) \xrightarrow{\mathsf{sout}} (i, \max(1, j-1)) \quad (i,j) \xrightarrow{\mathsf{west}} (\max(1, i-1), j) \quad (i,j) \xrightarrow{\mathsf{here}} (i,j)$$

Markov chains

In a LTS possible evolutions of a system state are selected nondeterministically. However, it is sometime useful to render system evolution in terms of a *random process*. This is the case of DTMCs.

Definition 2 (DTMC)**.** A DTMC is $\mathscr{D} = \langle \mathscr{S}, T, \mathscr{L} \rangle$ where $\mathscr{S}$ is a finite set of states and $T : \mathscr{S} \to \Delta(\mathscr{S})$ is the transition function associating each state in T with a probability distribution in $\Delta(\mathscr{S})$ and $\mathscr{L} : \mathscr{S} \to 2^{AP}$ is the labeling function on a set of atomic propositions AP.

While in an LTS next state is selected by the executed action, in a DTMC next state is selected probabilistically. Paths in a DTMC $\mathscr{D}$ are defined as sequences of states $\pi = s_0, \ldots s_i, \dots \in Paths^{\mathscr{D}}$ such that for each $i \in [0..]$, $T(s_i)(s_{i+1}) > 0$.

Example 2. The behaviour of a single white robot of Figure 1 can be described by a DTMC. Like in Example 1, the states of the DTMC are the possible positions of the robot in the grid, i.e. pairs (i, j). The transition function associates each element (i, j) wth the uniform probability distribution among the adjacent positions. For instance, $T(1,1) = \{(0,1) : \frac{1}{5}, (1,0) : \frac{1}{5}, (2,1) : \frac{1}{5}, (1,2) : \frac{1}{5}, (1,1) : \frac{1}{5}\}$.

Markov Processes

MDPs mix LTS and DTMC. They provide a mathematical framework that models decision making when the outcomes are partly random and partly under the control of a decision maker.

Definition 3 (MDP)**.** A MDP is $\mathscr{M} = \langle \mathscr{S}, \mathscr{A}, T, \mathscr{L} \rangle$ where $\mathscr{S}$ is the set of states, $\mathscr{A}$ is the set of actions, $T : \mathscr{S} \times \mathscr{A} \to \Delta(\mathscr{S}) \cup \{\bot\}$ is the transition function and $\mathscr{L} : \mathscr{S} \to 2^{AP}$ is the labeling function on a set of atomic propositions AP.

We write $s \xrightarrow{a} s'$ when s may perform a and go to s', i.e., $T(s,a)(s') > 0$. We use $s \not\xrightarrow{a}$ to say that a cannot be performed in state s, i.e., $T(s,a) = \bot$.

A *scheduler* of a MDP $\mathscr{M}$ is a function $\eta_{\mathscr{M}} : \mathscr{S} \to \mathscr{A}$ that maps states into actions. We use $Sched^{\mathscr{M}}$ to denote the set of all schedulers over MDP $\mathscr{M}$. We can use a scheduler to solve choices of a MDP. When we apply a scheduler to a MDP in this way, we obtain a DTMC defined as follows

Definition 4 (Induced DTMC)**.** Let $\mathscr{M} = \langle \mathscr{S}, \mathscr{A}, T, \mathscr{L} \rangle$ be a MDP and $\eta \in Sched^{\mathscr{M}}$. The DTMC $\mathscr{M}_\eta$ is given by $\mathscr{M}_\eta = \langle \mathscr{S}, T_\eta \rangle$ where $T_\eta(s) = T(s, \eta(s))$ for every $s \in \mathscr{S}$.

In the approach considered in this paper we use MDP to model the environment where a given agent, i.e. the one we control, operates. Agents and environment will be *composed* to generate a detailed description of the system.

Example 3. To model the environment where the *black* robot of Figure 1 operates, we can use a MDP that describes the movement of the *white robot* in the arena. Each state space in this MDP is a tuple the form $(p_1, \cdots, p_m)$ where each p_i represents the position of $i-$th white robot in the arena. This MDP can perform the same actions of the LTS in Example 1: $\{\mathsf{north}, \mathsf{east}, \mathsf{south}, \mathsf{west}, \mathsf{here}\}$. The probability distribution associated to each of this action in a given state describes how the environment react to the execution of an action by our agent. Here we assume that all the white robots follow a random walk. Let T' be the transition function of the DTMC considered in Example 2, we have that:

$$T((p_1, \cdots, p_m), a)(p'_1, \cdots, p'_m) = \Pi_{i=1}^{m} T'(p_1)(p'_1)$$

Partially observable models

DTMCs and MDP can be used to formally describe a system behaviour. Moreover, many tools have been introduced to support system analysis. Unfortunately, only in rare cases one can have a complete view of the context where they are operating. Often, only a partial view is available. In this section we recall some basic concepts from the literature about HMM [21].

Definition 5 (HMM)**.** An HMM is $\mathscr{H} = \langle \mathscr{S}, T, \mathscr{O}, Z \rangle$ where $\langle \mathscr{S}, T \rangle$ is a DTMC, $\mathscr{O}$ is the set of observations and $Z : \mathscr{S} \to \Delta(\mathscr{O})$ is the observation function.¡

We define $b \in \Delta(\mathscr{S})$ as a *belief state*. We denote $\mathscr{B}_{\mathscr{S}}$ as the *belief space*, we know that $\mathscr{B} \subseteq \Delta(S)$, and $b_0 \in \mathscr{B}_{\mathscr{S}}$ denotes the initial belief.

We can have partial information about the current state also for MDP extending it in the same way we defined HMMs as DTMCs with limited information. The resulting model is a POMDP that include probabilistic behaviour, partial observability and action control, and it is defined as follows

Definition 6 (POMDP)**.** A POMDP is $\mathscr{P} = \langle \mathscr{S}, \mathscr{A}, T, \mathscr{O}, Z \rangle$ where $\langle \mathscr{S}, \mathscr{A}, T \rangle$ is a MDP, $\mathscr{O}$ is the set of observations and $Z : \mathscr{S} \to \Delta(\mathscr{O})$ is the observation function.

We write $s \xrightarrow{a} s'$ when s may perform action a and go into s', i.e., $T(s,a)(s') > 0$. $Paths^{\mathscr{P}}$ contains infinite sequences π of elements $\pi_i = (s, o, a) \in \mathscr{S} \times \mathscr{O} \times \mathscr{A}$ for $i \in [0..]$, such that $s_i \xrightarrow{a_i} s_{i+1}$, $Z(s_i)(o_i) > 0$. We define a finite-memory scheduler over $\mathscr{P}$ as a function $\eta : \mathscr{B}_{\mathscr{S}} \times FPaths^{\mathscr{P}}_{\mathscr{A},\mathscr{O}} \to \mathscr{A}$ such that

$$\eta(b, \pi) = \begin{cases} \eta(b^{\pi_{i,\mathscr{A}}, \pi_{i,\mathscr{O}}}, \pi[1..]) & \text{if } |\pi| > 0 \\ \eta(b) & \text{otherwise} \end{cases}$$

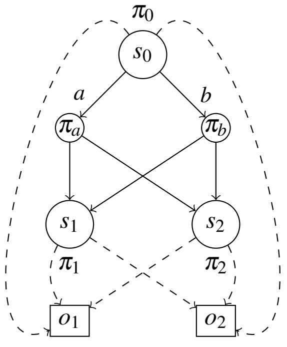

Figure 2: An example of POMDP

where $\eta(b)$ is a memoryless scheduler that maps belief states into choices $\eta : \mathscr{B} \to \mathscr{A}$. We use $Sched^{\mathscr{P}}$ to denote the set of all (memoryless) schedulers over POMDP $\mathscr{P}$. We use $\overline{\mathscr{P}} = \langle \mathscr{S}_{\mathscr{P}}, \mathscr{A}_{\mathscr{P}}, T_{\mathscr{P}} \rangle$ to isolate the hidden MDP from the partially observable model.

We can use a scheduler to solve choices of a POMDP. When we apply a scheduler to a POMDP in this way, we obtain a HMM defined as follows

Definition 7 (Induced HMM)**.** Let $\mathscr{P} = \langle \mathscr{S}, \mathscr{A}, T, \mathscr{O}, Z \rangle$ be a POMDP and $\eta \in Sched(\overline{\mathscr{P}})$. The HMM $\mathscr{P}_\eta$ is given by

$$\mathscr{P}_\eta = \langle \mathscr{S}, T_\eta, \mathscr{O}, Z \rangle$$

where $T_\eta(s) = T(s, \eta(s))$ for every $s \in \mathscr{S}$.

Given a belief state b we can apply an action a and an observation o to compute the next belief state $b' = b^{a,o}$. We can compute such state, even considering the effect of a single action or a single observations, with the following belief update formulae

$$b^{a,o}(s') = \frac{Z(s')(o) \sum_{s \in \mathscr{S}} T(s,a)(s')b(s)}{\sum_{\tilde{s} \in \mathscr{S}} Z(\tilde{s})(o) \sum_{s \in \mathscr{S}} T(s,a)(\tilde{s})b(s)} \tag{1}$$

$$b^a(s') = \sum_{s \in \mathscr{S}} T(s,a)(s')b(s) \tag{2}$$

3 Methodology

In this section we define the necessary models by relying on the preliminary notions introduced in Section 2 and show how these models can be transformed and used. The proposed methodology is depicted in Figure 3 where dependencies between components are shown.

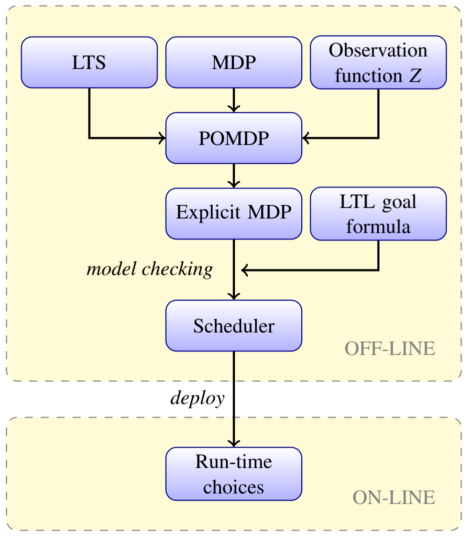

Figure 3: Elements and dependences of the proposed methodology

We define a general framework to model a generic scenario that involves the controller of an adaptive agent and the surrounding environment. The controller performs (output) actions and has neither uncertainties on state transitions nor uncertainties on state observations; it is a fully observable non probabilistic system that can be modeled as a LTS. The agent performs (input) actions with uncertain state transitions, finally the environment is modeled as a MDP. Partial observability is introduced later in this section to model what the adaptive agent perceives from the environment and in what measure.

We compose LTS and MDP together with an observation function that models what the agent perceives from the environment to obtain a POMDP that describe the whole system. We get rid of incomplete information by transforming the partially observable model into another MDP that transfers observations to states. Further more, we show that the properties verified on the new MDP also holds for the previous model and we exploit this result to build a scheduler for the initial LTS. The scheduler is computed in order to maximize the probability of satisfying a given goal, expressed as a Linear Temporal

Logic (LTL) formula that can refer to the agent's perceptions. Finally the scheduler is deployed on the agent that can take decisions at run-time.

Most of the workload is located in the model transformation phase, when the state space grows significantly. However the nature of the adaptation problem allow us to split the method into an off-line and an on-line phase: in the former the scheduler is built and provided to the agent that can use it during the latter phase. Determining the action to perform at run-time can be done in a very short time since the scheduler is a light data structure that maps the current configurations of local states and observations into actions.

3.1 Composing agents models and their environment

Given a state, a LTS describe what decision can be made by the agent. The following definition introduces the notion of *compatibility* between an agent and its environment to guarantee communications without lost messages; the agent cannot do anything that the environment cannot perceive.

Definition 8 (Compatibility)**.** We say that a LTS $\mathscr{L}$ and a MDP $\mathscr{M}$ are *compatible* when $\mathscr{A}_{\mathscr{L}} \subseteq \mathscr{A}_{\mathscr{M}}$

We push further this notion, to express the fact that the environment has to be *responsive* and cannot avoid receiving messages from an agent.

Definition 9 ($\mathscr{A}$-responsiveness)**.** Let $\mathscr{M}$ be a MDP and $\mathscr{A} \subseteq \mathscr{A}_{\mathscr{M}}$. $\mathscr{M}$ is said *$\mathscr{A}$-responsive* if

$$\forall\, a \in \mathscr{A}, s \in \mathscr{S}_{\mathscr{M}}\ \exists\, s' \in \mathscr{S}_{\mathscr{M}}\ .\ T_{\mathscr{M}}(s,a) \neq \bot$$

Responsiveness of the environment to agents actions ensures that communication between them is forced to happen at every step.

Definition 10 (Product POMDP)**.** Let $\mathscr{L}$ be a LTS, $\mathscr{M}$ be a MDP such that $\mathscr{M}$ is *$\mathscr{A}_{\mathscr{L}}$-responsive*, and let $Z : \mathscr{S}_{\mathscr{L}} \times \mathscr{S}_{\mathscr{M}} \to \Delta(\mathscr{O})$ be an observation function with codomain in an observation set $\mathscr{O}$. We call the composition of the system a *product POMDP that is defined as the following POMDP*

$$\mathscr{W}(\mathscr{L},\mathscr{M},Z,\mathscr{O}) = \langle \mathscr{S}_{\mathscr{L}} \times \mathscr{S}_{\mathscr{M}}, \mathscr{A}_{\mathscr{L}}, T_{\mathscr{W}}, \mathscr{O}, Z\rangle$$

where the transition function $T_{\mathscr{W}} : \mathscr{S}_{\mathscr{L}} \times \mathscr{S}_{\mathscr{M}} \times \mathscr{A}_{\mathscr{L}} \to \Delta(\mathscr{S}_{\mathscr{L}} \times \mathscr{S}_{\mathscr{M}})$ is defined as follows

$$T_{\mathscr{W}}(s_l,s_m,a_l)(s'_l,s'_m) = \begin{cases} T_{\mathscr{M}}(s_m,a_l)(s'_m) & \textit{if } s_l \xrightarrow{a_l}_{\mathscr{L}} s'_l \\ 0 & \textit{otherwise} \end{cases}$$

Once the communication between agent and environment is correctly defined, we can safely compose the two elements into a product POMDP that represents their synchronized progress. The state contains now local and environmental information, the former are assumed to be known while the latter are not.

If we know the current state for both the components, say $l \in \mathscr{S}_{\mathscr{L}}$ a state of $\mathscr{L}$ and $m \in \Delta(\mathscr{S}_{\mathscr{M}})$ a belief state of $\mathscr{M}$, we can build the corresponding belief state $b_{\mathscr{W}} \in \mathscr{B}_{\mathscr{W}}$ of the product POMDP as the following function $b_{\mathscr{W}} : \mathscr{S}_{\mathscr{L}} \times \Delta(\mathscr{S}_{\mathscr{M}}) \to \Delta(\mathscr{S}_{\mathscr{L}} \times \mathscr{S}_{\mathscr{M}})$

$$b_{\mathscr{W}}(s,b)(l,m) = \begin{cases} b(m) & \text{if } l = s \\ 0 & \text{otherwise} \end{cases}$$

We can then obtain the initial belief state of $\mathscr{W}$ as $b_{\mathscr{W}}(s_0,b_0)$, where s_0 is the initial state of $\mathscr{L}$ and b_0 is the initial belief state of $\mathscr{M}$.

Example 4. We can merge the controller of Example 1 with the environment of Example 3 to obtain a POMDP. To compute this POMDP we need to define how the main robot perceives the others. We assume that the robot can perceive the presence of at least one robot in each of the four adjacent positions or in its own position, and that this perception is not affected by precision errors. Starting from a set of basic observations $basic = \{north, south, east, west, here\}$ we obtain the set of possible observations as the power set $\mathscr{O} = 2^{basic}$, indeed every basic observation can happen independently from the others. $next(s_0, o)$, where s_0 is the position of the white robot and o is a basic observation, is used to represent the position perceived by sensors. The observation function can be then defined in the following way

$$Z(s_0, s_1, s_2)(O) \quad = \quad \begin{cases} 1 & \text{if } s_1 = next(s_0, o) \vee s_2 = next(s_0, o),\ \forall\, o \in O \\ 0 & \text{otherwise} \end{cases}$$

□

We exploit the construction of the POMDP $\mathscr{P}$ starting from an LTS $\mathscr{L}$ to drive the possible sequence of choices. Let $\eta \in Sched^{\mathscr{L}}$ be a scheduler and let $\sigma = \eta(s_0), \eta(s_1), \eta(s_2), \ldots$ be the sequence of actions induced by η, where $s_i = T_{\mathscr{L}}(s_{i-1}, \eta(s_{i-1}))$ for $i \in \mathbb{N}$. We can then construct a scheduler $\overline{\eta}$ for a product POMDP $\mathscr{W}$ as a function $\overline{\eta} : \mathscr{S}_{\mathscr{L}} \times \Delta(\mathscr{S}_{\mathscr{M}}) \to \mathscr{A}_{\mathscr{L}}$ such that $\overline{\eta}(l, b) \in init(l)$. We denote the set of this kind of schedulers as $Sched^{\mathscr{W}}$.

We are able to show that the set of schedulers on $\mathscr{L}$ and the set of schedulers extended to $\mathscr{W}$ are essentially the same since every scheduler preserves the induced sequence of actions after the extension. This is due to the fact that extended schedulers act only on the projection of $\mathscr{W}$ on $\mathscr{L}$.

Proposition 11. *Let $\mathscr{W}(\mathscr{L}, \mathscr{M}, Z, \mathscr{O})$ be a product POMDP, $\eta \in Sched^{\mathscr{L}}$, then $\forall\, s \in \mathscr{S}_{\mathscr{L}}$ and $\forall\, a \in \mathscr{A}_{\mathscr{L}}$*

$$T_{\mathscr{L}}(s, \eta(s)) = s' \iff \sum_{m \in \mathscr{S}_{\mathscr{M}}} T_{\mathscr{W}}(s, \cdot, \overline{\eta}(s, \cdot))(s', m) = 1$$

The following result says that belief states of $\mathscr{W}(\mathscr{L}, \mathscr{M}, Z, \mathscr{O})$ that give probability mass to a single state in $\mathscr{S}_{\mathscr{L}}$ will transfer all the probability only to states containing the successive state of $\mathscr{L}$. This means that if the local state is known at the current time, it will be known even after an action execution.

Proposition 12. *Let $\mathscr{W}(\mathscr{L}, \mathscr{M}, Z, \mathscr{O})$ be a product POMDP, $l, l' \in \mathscr{S}_{\mathscr{L}}$, $b \in \mathscr{B}_{\mathscr{W}}$ and $\exists\, a \in \mathscr{A}_{\mathscr{L}} : l \xrightarrow{a} l'$, then*

$$\sum_{m \in \mathscr{S}_{\mathscr{M}}} b(l, m) = 1 \Rightarrow \sum_{m \in \mathscr{S}_{\mathscr{M}}} b^{a,o}(l', m) = 1$$

We define now the transformation of a POMDP into a fully observable MDP.The transformation does preserve some information of the probability measure that makes it possible to convert some model checking results on the MDP to valid results on the POMDP.

Definition 13 (Explicit Markov decision process)**.** Let $\mathscr{P} = \langle \mathscr{S}, \mathscr{A}, T, \mathscr{O}, Z \rangle$ be a POMDP, we define the *explicit MDP* $\widehat{\mathscr{P}}$ of $\mathscr{P}$ as the following MDP

$$\widehat{\mathscr{P}} = \langle \mathscr{S} \times \mathscr{O}, \mathscr{A}, \widehat{T}, \mathscr{L} \rangle$$

where

- $\widehat{T}((s, o), a)(s', o') = T(s, a)(s') \cdot Z(s')(o')$
- $\mathscr{L}(s, o) = o$

The POMDP model is transformed into an MDP considering observations as atomic propositions of the new states, indeed we consider perceptions as part of the state and we modify the probability weights on transitions in order to preserve the likelihood of ending up into a state and make a specific observation.

An example of transformation is depicted in Figure 4, big circles are states, small circles are distributions, rectangles are observations, arrows are transitions and dashed arrows are observation distributions. The explicit MDP has two levels of distributions that represents a product distribution between the transition function and the observation function. During the transformation, we lose information about the observation made in the initial state, this loss will be taken into account when constructing the scheduler.

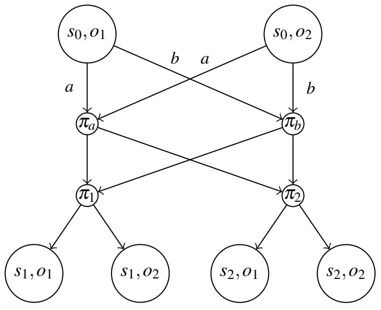

Figure 4: Explicit MDP of the POMDP in Figure 2

3.2 Specifying agents goals

The last ingredient of our approach is the language used to describe the required and expected goals. In this context we will use LTL formulae specialized to handle POMDP, indeed the syntax (and consequently the satisfaction relation) of this logic differs from the classical one only for the presence of an observation as a terminal, instead of a set of atomic propositions.

Definition 14 (Syntax of LTL)**.** A LTL formula φ is defined as follows

$$\varphi ::= tt \mid o \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\, \varphi \mid \varphi_1 U \varphi_2$$

where $o \in \mathscr{O}$.

Let LTL denote the set of all possible LTL formula. We define the satisfaction relation $\models$ with respect to a HMM $\mathscr{H}$ by induction in the following way

Definition 15 (Satisfaction relation)**.** The satisfaction relation $\models\ \subseteq Paths_{\mathscr{O}}^{\mathscr{H}} \times LTL$ is defined by induction as follows:

$$\begin{array}{lll}
\pi \models tt & & \text{always} \\
\pi \models o & \text{iff} & \pi_0 = o \\
\pi \models \varphi_1 \wedge \varphi_2 & \text{iff} & \pi \models \varphi_1 \wedge \pi \models \varphi_2 \\
\pi \models \neg\varphi & \text{iff} & \pi \not\models \varphi \\
\pi \models \bigcirc\varphi & \text{iff} & \pi[1..] \models \varphi \\
\pi \models \varphi_1 U \varphi_2 & \text{iff} & (\exists\, i \in [0..]\ .\ \pi[i] \models \varphi_2 \wedge \forall\, j \in [0..i-1]\ \pi[j] \models \varphi_1) \\
 & & \quad \vee \forall\, k \in \mathbb{N}\ .\ \pi[k] \models \varphi_1
\end{array}$$

This logic is employed to express desired behaviour for the adaptive agent, like liveness, safety or time bonded properties of signals coming from the environment.

Example 5. The problem of avoiding collisions with other robots as addressed with the following LTL formula

$$\varphi_{avoid} = \neg\ collision \wedge \bigcirc\neg\ collision$$

where label $collision$ is defined as $\bigvee_{o \in \mathscr{O}: here \in o} o$.

Let $\mathscr{M}$ be a MDP. For any formula $\varphi \in LTL$ and for any initial state $s \in \mathscr{S}_{\mathscr{M}}$, we define the following probability measure ($\models^*$ is the satisfaction relation defined for the classical LTL, see [1, Definition 5.7])

$$p^{\mathscr{M}}_{min}(s,\varphi) \triangleq \inf_{\eta \in Sched^{\mathscr{M}}} \left(Pr^{\mathscr{M}_\eta}_s \{\pi \in Paths^{\mathscr{M}_\eta} \mid \pi \models^* \varphi\} \right) \tag{3}$$

Let $\mathscr{P}$ be a POMDP. For any formula $\varphi \in LTL$ and for any initial state $s \in \mathscr{S}_{\mathscr{P}}$, we define the following probability measure

$$p^{\mathscr{P}}_{min}(s,\varphi) \triangleq \inf_{\eta \in Sched^{\mathscr{P}}} \left(Pr^{\mathscr{P}_\eta}_s \{\pi \in Paths^{\mathscr{P}_\eta}_{\mathscr{O}} \mid \pi \models \varphi\} \right) \tag{4}$$

The following result allows to compute the minimum probability of satisfying a formula φ on a POMDP by aggregating the minimum probabilities computed on the respective explicit MDP, recovering the initial observation probabilities lost during the model transformation.

Theorem 16. *Let $\mathscr{P}$ be a POMDP, $\widehat{\mathscr{P}}$ the explicit MDP of $\mathscr{P}$, φ an LTL formula with $AP = \mathscr{O}$, then it holds*

$$p^{\mathscr{P}}_{min}(s,\varphi) \leq \sum_{o \in \mathscr{O}} Z_{\mathscr{P}}(s)(o) \cdot p^{\widehat{\mathscr{P}}}_{min}((s,o),\varphi)$$

The converse inequality holds for the maximum probability.

Corollary 17. *Let $\mathscr{P}$ be a POMDP, $s \in \mathscr{S}_{\mathscr{P}}$ and $\varphi \in LTL$, it holds*

$$p^{\mathscr{P}}_{min}(s,\varphi) = \min_{\eta \in Sched^{\mathscr{P}}} \left(Pr^{\mathscr{P}_\eta}_s \{\pi \in Paths^{\mathscr{P}_\eta}_{\mathscr{O}} \mid \pi \models \varphi\} \right)$$

Now we can use Theorem 16 to define the value function that will be used as optimization criterion from the scheduler. Since we need to evaluate the effect of actions executed from a belief state, we need another level of aggregation; we thus choose to maximize the average of the minimum probabilities weighted on belief probabilities.

Definition 18 (Value function)**.** Let $\mathscr{W}(\mathscr{L},\mathscr{M},Z,\mathscr{O})$ be a product POMDP, the value function of $\mathscr{W}$ of a belief state $b \in \mathscr{B}$ and an action $a \in \mathscr{A}_{\mathscr{L}}$ with respect to an LTL formula φ is defined as follows

$$V^{\mathscr{W}}_a(b,\varphi) = \sum_{s \in \mathscr{S}} b^a(s) \cdot \sum_{o \in \mathscr{O}} Z(s)(o) \cdot p^{\widehat{\mathscr{W}}}_{min}((s,o),\varphi)$$

The scheduler $\mathfrak{S} \in Sched^{\mathscr{W}}$ is consequently defined as

$$\mathfrak{S}_\varphi(b) = \operatorname*{arg\,max}_{a \in \mathscr{A}_{\mathscr{L}}} V^{\mathscr{W}}_a(b,\varphi)$$

3.3 Taking decisions

With Algorithm 1 we define the on-line procedure used at every step to select the best action to execute. The algorithm consists in evaluating the value function $V^{\mathscr{W}}_a$ of every action considering the current belief state b. This is implemented by the cycle (lines $1-4$).

At line 2 we generate the belief state after an action a using Equation (2). Equation (2) computes $b^a(s)$ in $O(|\mathscr{S}|^2)$, then we can obtain the entire state b^a in $O(|\mathscr{S}|^3)$. The step at line 3 is the average over the minimum probabilities to satisfy φ weighted by the probability of being in that state according to b^a. Computing $p^{\mathscr{W}}_{min}(s,\varphi)$ costs $O(k|\mathscr{O}||\mathscr{S}|(|\mathscr{S}|+M))$ (see [1, Theorem 10.127]) where k is number of maximal end components of the composition of the MDP with the Deterministic Rabin Automaton

(DRA) representing the LTL formula φ, and M is the number of triples (s,a,s') such that $T(s,a)(s') > 0$. Summing everything up we can compute $V_a^{\mathscr{W}}$ in $O(k|\mathscr{O}||\mathscr{S}|^2(|\mathscr{S}|+M))$ that is polynomial and slightly dominates the complexity of EXPAND(b,a). However we can precompute minimum probabilities in order to access them at execution time in $O(|\mathscr{S}|)$.

The construction of the DRA $\mathscr{A}_\varphi$ is the most expensive operation in terms of computational complexity (2EXPTIME on $|\varphi|$). However this high complexity is not completely a bad news, indeed in linear time model checking we usually have short formulae and we can expect to have the same situation for planning. Moreover φ does not change during the evaluation of every action, then it can be computed off-line once and for all the duration of the execution.

The last step, at line 5, is just a search for the index of a maximum that has cost $O(|\mathscr{A}_\mathscr{L}|)$. Considering precomputed minimum probabilities we have the following result

Proposition 19. *Algorithm 1 has complexity* $O(|\mathscr{A}_\mathscr{L}||\mathscr{S}_\mathscr{W}|^2)$

Algorithm 1: Online decision algorithm to compute $\mathfrak{S}_\varphi^{\mathscr{W}}(b)$

```
  input : $\varphi \in$ LTL, $b \in \mathscr{B}$, $\mathscr{W}$ POMDP
  output: $a_t \in \mathscr{A}_\mathscr{L}$
1 for $a \in \mathscr{A}_\mathscr{L}$ do
2     $b^a \leftarrow$ EXPAND$(b,a)$;
3     $V_a^{\mathscr{W}} \leftarrow \sum_{s\in\mathscr{S}} b^a(s) \cdot p_{min}^{\mathscr{W}}(s,\varphi)$;
4 end
5 $a_t \leftarrow \arg\max_{a\in\mathscr{A}_\mathscr{L}} V_a^{\mathscr{W}}$;
6 return $a$;
```

4 Our approach at work

In this section we discuss the case study gradually introduced in Figure 1 and in Examples 1, 2, 3, 4 and 5 where we described the controller of the black robot, the environment model composed by white robots, the merged model as a POMDP, and a possible target formula φ_{avoid}.

We have run some experiments to compare the criterion we adopted with other possible approaches. The AVG scheduler, the one we propose, chooses the action which maximize the minimum probability averaged over the belief probabilities. With scheduler MAX we consider to maximize the maximum minimum probability weighted again with belief probabilities. The RND scheduler makes the robot move towards a uniformly random direction that is not already taken by another robot. RPL is the repulsive approach: the robot stands still until he perceive another robot, then it starts to move to the opposite direction, without keeping any memory of the past. RPL is defined for each pair state-observation.

We compare all the different approaches in a 5×5 arena with three robots, running 100 simulations for 100 steps for each case. Every run starts from a random initial state. The experiments results are shown in Figure 5 (left-hand side) where we measure the cumulate number of collisions over time, summing up over all the runs.

Apart from some specific situation with the highest density of robots, the AVG scheduler behaves generally better with respect to the other schedulers for collisions avoidance. It is worth noting that, the higher is the arena, the lower is the density of robots inside it; then avoiding collisions becomes easier. The MAX scheduler turns out to be suitable only for some specific situations since it gets good performances with a high density of robots.

Keeping other robots under observation is another possible target that can be expressed with a LTL formula in the following way:

$$\varphi_{track} = \neg\, collision \wedge vision \wedge \bigcirc\, (\neg\, collision \wedge vision)$$

where *vision* is a label that stands for $\neg\, \varnothing$ meaning that the robot is perceiving "nothing". Results from simulations depicted in Figure 5 (right-hand side) are obtained with the same set up of the previous experiment, and show that the synthesized behaviour is such that the number of perceptions is maximized while the number of collisions is minimized. It is worth noting that, by considering closely the simulation it is possible to appreciate some counterintuitive movement pattern such as moving towards other robots: this behaviour is actually the one that maximize our target since the other robot is more likely to move away from its current position making room for the controlled robot and, at the same time, remaining in its sight with probability 1.

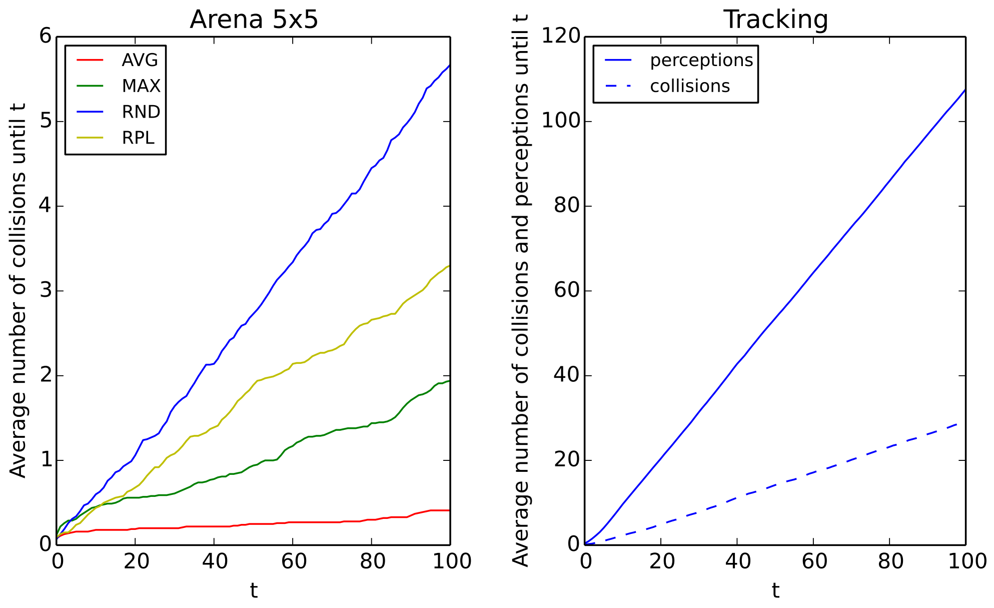

Figure 5: Collisions and perceptions with φ_{track}

Simulations are written in python, code and data can be downloaded from `http://dropproxy.com/f/BF7`[1].

5 Conclusions and related works

We have presented a framework for modeling scenarios involving controllable agents, environmental elements and partial perception, where both probabilistic and deterministic behaviours are viable and can be mixed together. Then, we have provided an automatic procedure to build a scheduler for a controllable agent once a LTL formula describing the target is given. Since we exploit a model checking algorithm to construct the scheduler, the resulting strategy is guaranteed to maximize the probability to satisfy the target formula.

The model checking approach allows to split the algorithm into an off-line and an on-line phase, during the former we can evaluate the model and precompute the adaptive scheduler that will be used in

[1]The actual code is also in a github repository but we do not report the link for the sake of anonymity.

the latter to get better performance. Indeed, the resulting scheduler is just a simple function that maps local states and observations into actions. A possible scenario for using our approach could be to rely on a high-performance computer to find the appropriate schedulers that could then be sent to small adaptive agents with a low computational power that by themselves would not be able to deal with the model generation phase. Even simpler, the high-performance computer could send the action that has to be executed next directly to the agent, whenever these call for support and communicate their observations.

Using LTL to describe goals gives large expressivity to the framework since conditions on perceptions can be described. Formulae describing avoidance and tracking of moving objects have been introduced and tested, individually and mixed together. Simulations show that multiple objectives can be handled at the same time. As future work, we would also like to provide a formal language to define composed partially observable models. In this way the designer would be exempted to explicitly write the models, but he would have at the same time the possibility of describing behaviours and observations. Another possible extension of the work is the formula updating in case of a reached (or missed) target.

The model and scheduler generation suffers from a well known scalability issue, and even if this phase does not directly penalize the run-time performance, it could take a prohibitive amount of time. For this reason we plan also to exploit the abstraction technique presented in [12] that permits computing ranges of minimum and maximum probabilities of a given formula and a partition of the states; we want to use an observation-based partition to unburden the workload. Otherwise we could aim at analysing the model before the transformation phase (where we can have state space explosion) when agents and environment are described individually.

Some related work Partially observable models like HMMs [20] and POMDPs [8] have been successfully employed in many fields like speech-recognition [20] and activity recognition for ambient assisted living [7]. These models cope with the adaptive system's typical lack of information. In particular POMDPs give the possibility to model the partial knowledge in presence of input and output interactions, as sensing and acting procedures. Finding the optimal scheduler is a problem that has been approached in different ways [15] since many theoretical limits have been proven to subsist: building the optimal scheduler function is PSPACE-complete in case of finite horizon [18] and determining its existence is undecidable for infinite horizon [16].

Model checking on HMMs is proposed in [21] together with the extended definitions of branching-time and linear-time logics to include belief states and observation specifications. In this work we adopt a different perspective and we consider observations as label of states. We use classic LTL formulae [19] to express properties with probabilistic observations on POMDPs.

We did employ model checking techniques to solve a planning problem, like in [2], but we further exploit the classical algorithm to solve the same problem at run-time. Also the logic used to formulate agent's objectives differs from the goal language proposed in [14] since we focus on the explicit use of observation signals.

References

[1] Christel Baier & Joost-Pieter Katoen (2008): *Principles of model checking*. MIT Press.

[2] Piergiorgio Bertoli, Alessandro Cimatti, Marco Roveri & Paolo Traverso (2001): *Planning in Nondeterministic Domains under Partial Observability via Symbolic Model Checking*. In: *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI*, pp. 473–478.

[3] Stephen D. Brookes, C. A. R. Hoare & A. W. Roscoe (1984): *A Theory of Communicating Sequential Processes*. *J. ACM* 31(3), pp. 560–599, doi:10.1145/828.833. Available at `http://doi.acm.org/10.1145/828.833`.

[4] Roberto Bruni, Andrea Corradini & Fabio Gadducci (2013): *Adaptable transition systems*. *Recent Trends in Algebraic Development Techniques*, pp. 95–110. Available at `http://link.springer.com/chapter/10.1007/978-3-642-37635-1_6`.

[5] Roberto Bruni, Andrea Corradini, Fabio Gadducci, Alberto Lluch Lafuente & Andrea Vandin (2012): *Modelling and Analyzing Adaptive Self-Assembling Strategies with Maude*. *ASCENS*.

[6] Roberto Bruni, Andrea Corradini, Fabio Gadducci, Alberto Lluch-Lafuente & Andrea Vandin (2012): *A Conceptual Framework for Adaptation*. In: *Fundamental Approaches to Software Engineering - 15th International Conference, FASE 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia. Proceedings*, pp. 240–254, doi:10.1007/978-3-642-28872-2_17. Available at `http://dx.doi.org/10.1007/978-3-642-28872-2_17`.

[7] Laura Carnevali, Christopher Nugent, Fulvio Patara & Enrico Vicario (2015): *A Continuous-Time Model-Based Approach to Activity Recognition for Ambient Assisted Living*. *Quantitative Evaluation of Systems: 12th International Conference, QEST 2015, Madrid, Spain, Proceedings* 9259, p. 38.

[8] Anthony R. Cassandra (1998): *A survey of POMDP applications*. In: *Technical Report MCC-INSL-111-98. Microelectronics and Computer Technology Corporation (MCC). AAAI Fall Symposium*, 1724, Citeseer.

[9] Alessandro Cimatti, Marco Pistore, Marco Roveri & Paolo Traverso (2003): *Weak, strong, and strong cyclic planning via symbolic model checking*. *Artificial Intelligence* 147(1-2), pp. 35–84, doi:10.1016/S0004-3702(02)00374-0. Available at `http://linkinghub.elsevier.com/retrieve/pii/S0004370202003740http://www.sciencedirect.com/science/article/pii/S0004370202003740`.

[10] Yuxin Deng, Rob J. van Glabbeek, Matthew Hennessy & Carroll Morgan (2008): *Characterising Testing Preorders for Finite Probabilistic Processes*. *Logical Methods in Computer Science* 4(4), doi:10.2168/LMCS-4(4:4)2008. Available at `http://dx.doi.org/10.2168/LMCS-4(4:4)2008`.

[11] Edmond Gjondrekaj, Michele Loreti, Rosario Pugliese & Francesco Tiezzi (2011): *A Robot Foraging Scenario in Klaim*.

[12] Mark Kattenbelt, Marta Z. Kwiatkowska, Gethin Norman & David Parker (2010): *A game-based abstraction-refinement framework for Markov decision processes*. *Formal Methods in System Design* 36(3), pp. 246–280, doi:10.1007/s10703-010-0097-6. Available at `http://dx.doi.org/10.1007/s10703-010-0097-6`.

[13] Marta Z. Kwiatkowska, Gethin Norman & David Parker (2011): *PRISM 4.0: Verification of Probabilistic Real-Time Systems*. In: *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA. Proceedings*, pp. 585–591, doi:10.1007/978-3-642-22110-1_47. Available at `http://dx.doi.org/10.1007/978-3-642-22110-1_47`.

[14] Ugo Dal Lago, Marco Pistore & Paolo Traverso (2002): *Planning with a Language for Extended Goals*. In: *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, Edmonton, Alberta, Canada*, pp. 447–454. Available at `http://www.aaai.org/Library/AAAI/2002/aaai02-068.php`.

[15] Michael L. Littman, Anthony R. Cassandra & Leslie Pack Kaelbling (1995): *Learning Policies for Partially Observable Environments: Scaling Up*. In: *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA*, pp. 362–370.

[16] Omid Madani, Steve Hanks & Anne Condon (1999): *On the Undecidability of Probabilistic Planning and Infinite-Horizon Partially Observable Markov Decision Problems*. In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence, Orlando, Florida, USA*, pp. 541–548. Available at `http://www.aaai.org/Library/AAAI/1999/aaai99-077.php`.

[17] omissis (2015): *Partially Observable LTL Planning*. omissis.

[18] Christos Papadimitriou & John N. Tsitsiklis (1987): *The Complexity of Markov Decision Processes*. *Math. Oper. Res.* 12(3), pp. 441–450, doi:10.1287/moor.12.3.441. Available at `http://dx.doi.org/10.1287/moor.12.3.441`.

[19] Amir Pnueli (1977): *The Temporal Logic of Programs*. In: *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA*, pp. 46–57, doi:10.1109/SFCS.1977.32. Available at `http://dx.doi.org/10.1109/SFCS.1977.32`.

[20] Lawrence R. Rabiner (1990): *Readings in Speech Recognition*. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 267–296. Available at `http://dl.acm.org/citation.cfm?id=108235.108253`.

[21] Lijun Zhang, Holger Hermanns & David N. Jansen (2005): *Logic and Model Checking for Hidden Markov Models*. In: *Formal Techniques for Networked and Distributed Systems - FORTE 2005, 25th IFIP WG 6.1 International Conference, Taipei, Taiwan, Proceedings*, pp. 98–112, doi:10.1007/11562436_9. Available at `http://dx.doi.org/10.1007/11562436_9`.

Appendix: Auxiliary notions and proofs

Preliminary concepts

Discrete-time Markov chains The transition function T induces a measure space on the set of *paths* in a DTMC. Given a finite path $\hat{\pi} = s_0 \dots s_n$, the cylinder set of $\hat{\pi} = s_0 \dots s_n \in FPaths^{\mathscr{D}}$ is defined as

$$\mathscr{C}^{\mathscr{D}}(\hat{\pi}) = \{\pi \in Paths^{\mathscr{D}} \mid \hat{\pi} = \pi[..|\hat{\pi}|-1]\}$$

A σ-algebra for the possible paths of a Markov chain $\mathscr{D}$ can be defined starting from the cylinder sets. The probability measure can be finally defined as follows:

$$Pr_s^{\mathscr{D}}(\mathscr{C}^{\mathscr{D}}(s_0 \dots s_n)) = \begin{cases} \prod_{i=1}^{n} T(s_{i-1})(s_i) & \text{if } s = s_0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Hidden Markov models A path π of $\mathscr{H}$ is a sequence $(s_0, o_0), (s_1, o_1) \dots \in (\mathscr{S} \times \mathscr{O})^{\omega}$ where $T(s)(s') > 0$, $Z(s)(o) > 0$ and $i \in \mathbb{N}$. Let $Paths^{\mathscr{H}}$ denote the set of all paths in $\mathscr{H}$. and $\hat{\pi} \in FPaths^{\mathscr{H}}$ be a finite path on $\mathscr{H}$, we define a *basic cylinder set* on $\hat{\pi}$ as follows:

$$\mathscr{C}^{\mathscr{H}}(\hat{\pi}) = \{\pi \in Paths^{\mathscr{H}} \mid \forall\, i \in [0..n].\pi_{i,\mathscr{S}} = \hat{\pi}_{i,\mathscr{S}} \wedge \pi_{i,\mathscr{O}} = \hat{\pi}_{i,\mathscr{O}}\}$$

Then we define a probability measure $Pr_s^{\mathscr{H}} : Cyl^{\mathscr{H}} \to [0,1]$ (see [21]) starting from a state $s \in \mathscr{S}$ as

$$Pr_s^{\mathscr{H}}(\mathscr{C}^{\mathscr{H}}((s_0,o_0)\dots(s_n,o_n))) = Pr_s^{\mathscr{H}}(\mathscr{C}^{\mathscr{H}}((s_0,o_0)\dots(s_{n-1},o_{n-1})))\; T(s_{n-1})(s_n)\; Z(s_n)(o_n)$$

$$Pr_s^{\mathscr{H}}(\mathscr{C}^{\mathscr{H}}((s_0,o_0))) = \begin{cases} Z(s_0)(o_0) & \text{if } s_0 = s \\ 0 & \text{otherwise} \end{cases}$$

By induction on n we obtain:

$$Pr_s^{\mathscr{H}}(\mathscr{C}^{\mathscr{H}}((s_0,o_0)\dots(s_n,o_n))) = \begin{cases} Z(s_0)(o_0) \prod_{i=1}^{n} T(s_{i-1})(s_i)\; Z(s_i)(o_i) & \text{if } s_0 = s \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Proposition 20 ([21]). *Let $\mathscr{H}$ be a HMM and $Cyl^{\mathscr{H}}$ be the set of all the basic cylinder sets over $FPaths^{\mathscr{H}}$, then $(Paths^{\mathscr{H}}, Cyl^{\mathscr{H}})$ is a measurable space. Moreover, let $Pr_s^{\mathscr{H}} : Cyl^{\mathscr{H}} \to [0,1]$ the probability measure defined in Equation (6), then $(Paths^{\mathscr{H}}, Cyl^{\mathscr{H}}, Pr_s^{\mathscr{H}})$ is a probability space.*

Intermediate results

The result that follows describe the relation between the probabilistic measure of POMDPs and MDPs when the initial state is known.

Proposition 21. *Let $\mathscr{P}$ be a POMDP, $s, s_0, \dots, s_n \in \mathscr{S}_{\mathscr{P}}$, $o_0, \dots, o_n \in \mathscr{O}_{\mathscr{P}}$ and $\eta \in Sched^{\mathscr{P}}$, it holds*

$$Pr_s^{\mathscr{P}_\eta}(\mathscr{C}^{\mathscr{P}_\eta}(s_0, o_0 \dots s_n, o_n)) = Z(s_0, o_0) \cdot Pr_{(s,o_0)}^{\widehat{\mathscr{P}}_\eta}(\mathscr{C}^{\widehat{\mathscr{P}}_\eta}((s_0,o_0)\dots(s_n,o_n)))$$

Proposition 22. *Let $\mathscr{P}$ be a POMDP, $\widehat{\mathscr{P}}$ be the explicit MDP of $\mathscr{P}$ and $\overline{\mathscr{P}}$ be the hidden MDP of $\mathscr{P}$, $\widehat{Sched^{\overline{\mathscr{P}}}} \equiv \{\xi : \mathscr{S} \times \mathscr{O} \to \mathscr{A} \mid \exists\, \eta \in Sched^{\overline{\mathscr{P}}} : \forall\, s \in \mathscr{S}, \forall\, o \in \mathscr{O} : \xi(s,o) = \eta(s)\}$, then*

$$\widehat{Sched^{\overline{\mathscr{P}}}} \subseteq Sched^{\widehat{\mathscr{P}}}$$

Proofs

Proposition 11. The straight implication can be proved as follows

$$\sum_{m' \in \mathscr{S}_{\mathscr{M}}} T_{\mathscr{W}}(s,m,\overline{\eta}(s,\cdot))(s',m') = \sum_{m' \in \mathscr{S}_{\mathscr{M}}} T_{\mathscr{M}}(m,\eta(s))(m') = 1$$

The first equivalence is given by definition of $\overline{\eta}$ and $T_{\mathscr{W}}$ and we do not lose any component of the sum by hypothesis. In the second step we can exclude the case in which $T_{\mathscr{M}}(m,\eta(s)) = \perp$ because of the $\mathscr{A}_{\mathscr{L}}$-responsiveness of $\mathscr{M}$.

The converse implication simply derives from the definition of $T_{\mathscr{W}}$: since the sum of transition probabilities is one, it exists at least one transition such that $T_{\mathscr{W}}(s,m,\eta(s))(s',m') > 0$ with positive probability, it follows that $s \xrightarrow{\eta(s)} s'$. □

Proposition 12. By hypothesis b has support only on states having l as a state of $\mathscr{L}$, that means that $init(l)$ are all and only the actions that can be executed from b. By the definition of $\mathscr{W}$ the transition function $T_{\mathscr{W}}$ must have probability zero only when states from $\mathscr{L}$ are not connected by an action, then it holds $\sum_{m \in \mathscr{M}} b(l,m) = 1 \Rightarrow \sum_{m \in \mathscr{M}} b^a(l',m) = 1$. We conclude the proof since it holds $b^a = 0 \Rightarrow b^{a,o}(s) = 0$ for any action a, observation o, belief state b and state s. □

Proposition 21. It trivially follows from equations (5) and (6). □

Proposition 22. We define $\widehat{Sched^{\overline{\mathscr{P}}}}$ as a set of schedulers for explicit MDPs that chose actions only relying on the current state mimicking the behavior of schedulers for the hidden MDP of $\overline{\mathscr{P}}$. This kind of construction considers only schedulers for the explicit MDP that choose the same action from the same state ignoring the observation. We can split schedulers in $Sched^{\widehat{\mathscr{P}}}$ in two partitions, schedulers that always take the same decisions in the same state and schedulers that take at least a different decision from the same state with different observations. We generate the former category excluding the latter, it follows that $\widehat{Sched^{\overline{\mathscr{P}}}} \subseteq Sched^{\widehat{\mathscr{P}}}$ □

Theorem 16. We rely on Proposition 21 for this proof, since it shows that cylinders probabilities in DTMCs and HMMs differ only of a multiplicative factor.

During the passages the notation $psat(\varphi) \in (\mathscr{S}_{\mathscr{P}} \times \mathscr{O}_{\mathscr{P}})^*$ is used to define the set of finite paths of maximal length that satisfy φ.

$$\begin{aligned}
p^{\widehat{\mathscr{P}}}_{min}((s,o),\varphi) &= \inf_{\xi \in Sched^{\widehat{\mathscr{P}}}} Pr^{\widehat{\mathscr{P}}_\xi}_{(s,o)}\{\pi \in Paths^{\widehat{\mathscr{P}}} \mid \pi \models \varphi\} \\
&\quad \{\text{We reformulate the set of paths as the union of cylinder sets obtained by } \varphi\} \\
&= \inf_{\xi \in Sched^{\widehat{\mathscr{P}}}} \sum_{\hat{\pi} \in psat(\varphi)} Pr^{\widehat{\mathscr{P}}_\xi}_{(s,o)}(\mathscr{C}^{\widehat{\mathscr{P}}_\xi}(\hat{\pi})) \\
&\quad \{\text{By the definition of probability measure on Markov chains (5)}\} \\
&= \inf_{\xi \in Sched^{\widehat{\mathscr{P}}}} \sum_{\substack{\hat{\pi} \in psat(\varphi) \wedge \\ \hat{\pi} = s,o,s_1,o_1 \ldots s_n,o_n}} \prod_{i=1}^{n} \widehat{T}((s_{i-1},o_{i-1}),\xi(s_{i-1},o_{i-1}))(s_i,o_i) \\
&\quad \{\text{By Definition 13 of } \widehat{T} \text{ and Proposition 22}\} \\
&\geq \inf_{\eta \in \widehat{Sched^{\mathscr{P}}}} \sum_{\substack{\hat{\pi} \in psat(\varphi) \wedge \\ \hat{\pi} = s,o,s_1,o_1 \ldots s_n,o_n}} \prod_{i=1}^{n} T(s_{i-1},\eta(s_{i-1}))(s_i)Z(s_i)(o_i)
\end{aligned}$$

In the following chain of equations we use the previous result to connect the minimum probability on $\mathscr{P}$ with a weighted sum of minimum probabilities on $\widehat{\mathscr{P}}$.

$$\begin{aligned}
p^{\mathscr{P}}_{min}(s,\varphi) &= \inf_{\eta \in \overline{Sched^{\mathscr{P}}}} Pr^{\mathscr{P}_\eta}_s\{\pi \in Paths^{\mathscr{P}} \mid \pi \models \varphi\} \\
&\quad \{\text{We reformulate the set of paths as the union of cylinder sets obtained by } \varphi\} \\
&= \inf_{\eta \in \overline{Sched^{\mathscr{P}}}} \sum_{\hat{\pi} \in psat(\varphi)} Pr^{\mathscr{P}_\eta}_s(\mathscr{C}^{\mathscr{P}_\eta}(\hat{\pi})) \\
&\quad \{\text{By the definition of probability measure on HMM (6)}\} \\
&= \inf_{\eta \in \overline{Sched^{\mathscr{P}}}} \sum_{\substack{\hat{\pi} \in psat(\varphi) \wedge \\ \hat{\pi} = s,o_0,s_1,o_1 \ldots s_n,o_n}} Z(s)(o_0) \prod_{i=1}^{n} T(s_{i-1},\eta(s_{i-1}))(s_i)Z(s_i)(o_i) \\
&= \inf_{\eta \in \overline{Sched^{\mathscr{P}}}} \sum_{\tilde{o} \in \mathscr{O}} \sum_{\substack{\hat{\pi} \in psat(\varphi) \wedge \\ \hat{\pi} = s,\tilde{o},s_1,o_1 \ldots s_n,o_n}} Z(s)(\tilde{o}) \prod_{i=1}^{n} T(s_{i-1},\eta(s_{i-1}))(s_i)Z(s_i)(o_i) \\
&\quad \{\text{Since the initial state } s \text{ does not change we can swap the sum with the infimum operator}\} \\
&= \sum_{\tilde{o} \in \mathscr{O}} Z(s)(\tilde{o}) \inf_{\eta \in \overline{Sched^{\mathscr{P}}}} \sum_{\substack{\hat{\pi} \in psat(\varphi) \wedge \\ \hat{\pi} = s,\tilde{o},s_1,o_1 \ldots s_n,o_n}} \prod_{i=1}^{n} T(s_{i-1},\eta(s_{i-1}))(s_i)Z(s_i)(o_i) \\
&\quad \{\text{By the result of the previous equation}\} \\
&\leq \sum_{\tilde{o} \in \mathscr{O}} Z(s)(\tilde{o}) \cdot p^{\widehat{\mathscr{P}}}_{min}((s,\tilde{o}),\varphi)
\end{aligned}$$

□

Corollary 17. From [1, Lemma 10.102] we know that, for a MDP $\mathscr{M}$, it always exists a finite-memory scheduler that minimize the probabilities for φ. The result directly follows from it and Theorem 16. □