The International Journal of Robotics Research
2014, Vol. 33(8) 1130–1144
© The Author(s) 2014
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364913519000
ijr.sagepub.com

# Incremental controller synthesis in probabilistic environments with temporal logic constraints

**Alphan Ulusoy[1], Tichakorn Wongpiromsarn[2] and Calin Belta[1]**

**Abstract**
*In this paper, we consider automatic computation of optimal control strategies for a robot interacting with a set of independent uncontrollable agents in a graph-like environment. The mission specification is given as a syntactically co-safe Linear Temporal Logic formula over some properties that hold at the vertices of the environment. The robot is assumed to be a deterministic transition system, while the agents are probabilistic Markov models. The goal is to control the robot such that the probability of satisfying the mission specification is maximized. We propose a computationally efficient incremental algorithm based on the fact that temporal logic verification is computationally cheaper than synthesis. We present several case studies where we compare our approach to the classical non-incremental approach in terms of computation time and memory usage.*

**Keywords**
Control policy synthesis, incremental synthesis, probabilistic verification, temporal logic, formal methods

## 1. Introduction

Temporal logics (Emerson, 1990), such as linear temporal logic (LTL) and computation tree logic (CTL), are traditionally used for verification of non-deterministic and probabilistic systems (Baier and Katoen, 2008). Even though temporal logics are suitable for specifying complex missions for control systems, they did not gain popularity in the control community until recently (Tabuada and Pappas, 2006; Kloetzer and Belta, 2010; Wongpiromsarn et al., 2010).

The existing works on control synthesis focus on specifications given in linear time temporal logic. The systems, which sometimes are obtained through an additional abstraction process (Tabuada and Pappas, 2006; Tůmová et al., 2010), have finitely many states. With few exceptions (Berwanger et al., 2010), their states are fully observable. For such systems, control strategies can be synthesized through exhaustive search of the state space. If the system is deterministic, model checking tools can be easily adapted to generate control strategies (Kloetzer and Belta, 2010). If the system is non-deterministic, the control problem can be mapped to the solution of a Rabin game (Thomas, 2002; Tůmová et al., 2010), or a simpler Büchi (Kloetzer and Belta, 2008) or GR(1) game (Kress-Gazit et al., 2007), if the specification is restricted to fragments of LTL. For probabilistic systems, the control synthesis problem reduces to computing a control policy for a Markov decision process (MDP) (Bianco and De Alfaro, 1995; Kwiatkowska et al., 2002; Ding et al., 2011; Lahijanian et al., 2012).

The goal of this paper is to efficiently compute control policies for a robot operating in the presence of numerous independent agents so that the probability of satisfying a temporal logic mission specification is maximized. More specifically, we focus on a particular type of a multi-agent system formed by a deterministically controlled robot and a set of independent, probabilistic, uncontrollable agents, operating on a common, graph-like environment. The mission specifications are expressed as syntactically co-safe LTL (scLTL) formulas (Kupferman and Vardi, 2001) over some properties satisfied at the vertices of the graph. Using scLTL, one can express rich correctness specifications for the behavior of the complete multi-agent system. An illustrative example is a car (robot) approaching a pedestrian crossing, while there are some pedestrians (agents) waiting to cross or already crossing the road. The robot is required to pass the crossing without colliding with any of the pedestrians. The classical method to solve this problem consists of two steps (Baier and Katoen, 2008; Bianco

[1]Division of Systems Engineering, Boston University, Boston, MA, USA
[2]Singapore–MIT Alliance for Research and Technology, Singapore

**Corresponding author:**
Alphan Ulusoy, Division of Systems Engineering, Boston University, Boston, MA 02215, USA.
Email: alphan@bu.edu

and De Alfaro, 1995; Alfaro, 1997): first, a relatively large Markov model that captures the robot, the independent agents, and the mission specification is constructed. Then, a maximal reachability problem (MRP) is solved on this model to obtain the optimal control policy that maximizes the probability of satisfying the specification. Nevertheless, as the state space of the system grows exponentially with the number of pedestrians, one may not be able to solve this problem under computational resource constraints when there is a large number of pedestrians. To alleviate this state explosion problem, various approaches based on distributed synthesis of control policies have been proposed (Chen et al., 2011; Ozay et al., 2011) where a global specification is decomposed into smaller local specifications which are then used to synthesize control policies for the individual controllable robots in the system. These methods, however, are not suitable for the problem that we consider in this paper where there is a single robot and a large number of independent uncontrollable agents. On the other hand, Bhatia et al. (2010) proposed a geometry-based discrete abstraction for mobile robots with nonlinear hybrid dynamics and an iterative motion planning algorithm that exploits the benefits of using such an abstraction to reduce the size of the search space. However, their approach assumes a deterministic environment with only static obstacles. Johnson and Kress-Gazit (2012) also looked at temporal logic specifications in probabilistic environments, but they considered only the verification of a previously synthesized controller in a probabilistic environment as opposed to the control synthesis problem that we address in this paper.

The contribution of this paper is to provide an incremental algorithm for computing robot control policies that maximize the probability of satisfying a mission specification given as an scLTL formula. Our algorithm exploits the independence between the components of the system, i.e. the robot modeled as a deterministic transition system and the agents, modeled as Markov chains, and the fact that verification is computationally cheaper than synthesis to reduce the overall resource usage. Each iteration of our algorithm consists of three operations: synthesis, verification, and reduction. In the synthesis step, we construct a subsystem formed by the robot and a subset of agents such that this subsystem is an optimistic simplification of the overall problem (due to agents that are not considered yet). Next, we synthesize a control policy that maximizes the probability of satisfying the mission specification for this subsystem. In the verification step, we compute the actual probability of satisfying the mission specification under this control policy by considering the agents not included in the synthesis step. Then, in the reduction step, we use the probability values obtained in the synthesis and verification steps to remove the transitions and states that are not needed in subsequent iterations. This leads to a significant reduction in computation time and memory usage. It is important to note that our method does not need to run to completion. A sub-optimal control policy can be obtained by forcing termination at a given iteration if the computation is performed under stringent resource constraints. It must also be noted that our framework easily extends to the case when the robot is a Markov decision process, and we consider a deterministic robot only for simplicity of presentation.

Various methods that also use verification during incremental synthesis have been previously proposed in Jha et al. (2010); Gulwani et al. (2011). Nevertheless, the approach that we present in this paper is, to the best of our knowledge, the first use of verification guided incremental synthesis in the context of probabilistic systems. We considered the same problem also in Wongpiromsarn et al. (2012, 2013), where we proposed various methods that exploit the structure of the system for efficient incremental synthesis of optimal satisfying control policies. More specifically, by keeping track of the strongly connected components of the system model from one iteration to the next, we were able to significantly reduce the synthesis effort so that the control policy computed at some iteration of the algorithm would facilitate the solution of the synthesis problems in the subsequent iterations. By contrast, in this paper, we take a completely different approach where verification performed at an iteration simplifies the synthesis problems of the subsequent iterations by reducing the size of the system model. Thus, the approach that we present in this paper may potentially perform better in cases where the system model lacks the structure explained above. A preliminary version of our approach appeared in Ulusoy et al. (2012). However, the algorithm given in Ulusoy et al. (2012) performs best when the agents can only violate the specification, e.g. when the robot must avoid them, and its performance degrades as the number of agents that can satisfy the specification increases, e.g. when the robot needs to reach or catch them. Here, we extend this preliminary work by presenting an algorithm that can handle both of these cases equivalently well. We also provide full proofs and new case studies.

The organization of the paper is as follows. In Section 2, we give necessary definitions and some preliminaries in formal methods. In Section 3, we formally state the control synthesis problem and give an overview of our approach. We present our solution in Section 4 and discuss the details of our implementation and various case studies in Section 5. We conclude with final remarks in Section 6.

## 2. Preliminaries

In this section, we introduce the notation that we use in the rest of the paper and give some definitions. We refer the reader to Baier and Katoen (2008), Clarke et al. (1999), Hopcroft et al. (2007), and references therein for a more complete and rigorous treatment of these topics. For a set $\Sigma$, we use $|\Sigma|$ and $2^{\Sigma}$ to denote its cardinality and power set, respectively. A (finite) word $\omega$ over a set $\Sigma$ is a sequence of symbols $\omega = \omega^0, \ldots, \omega^l$ such that $\omega^i \in \Sigma$ $\forall i = 0, \ldots, l$.

**Definition 2.1** (**Transition system).** *A transition system (TS) is a tuple* $\mathbf{T} := (\mathcal{Q}_{\mathrm{T}}, q_{\mathrm{T}}^0, \mathcal{A}_{\mathrm{T}}, \alpha_{\mathrm{T}}, \delta_{\mathrm{T}}, \Pi_{\mathrm{T}}, \mathcal{L}_{\mathrm{T}})$*, where*

- $\mathcal{Q}_{\mathrm{T}}$ *is a finite set of states;*
- $q_{\mathrm{T}}^0 \in \mathcal{Q}_{\mathrm{T}}$ *is the initial state;*
- $\mathcal{A}_{\mathrm{T}}$ *is a finite set of actions;*
- $\alpha_{\mathrm{T}} : \mathcal{Q}_{\mathrm{T}} \to 2^{\mathcal{A}_{\mathrm{T}}}$ *is a map giving the set of actions available at a state;*
- $\delta_{\mathrm{T}} \subseteq \mathcal{Q}_{\mathrm{T}} \times \mathcal{A}_{\mathrm{T}} \times \mathcal{Q}_{\mathrm{T}}$ *is the transition relation;*
- $\Pi_{\mathrm{T}}$ *is a finite set of atomic propositions;*
- $\mathcal{L}_{\mathrm{T}} : \mathcal{Q}_{\mathrm{T}} \to 2^{\Pi_{\mathrm{T}}}$ *is a satisfaction map giving the set of atomic propositions satisfied at a state.*

**Definition 2.2** (**Markov chain).** *A (discrete-time, labeled) Markov chain (MC) is a tuple* $\mathbf{M} := (\mathcal{Q}_{\mathrm{M}}, q_{\mathrm{M}}^0, \delta_{\mathrm{M}}, \Pi_{\mathrm{M}}, \mathcal{L}_{\mathrm{M}})$*, where* $\mathcal{Q}_{\mathrm{M}}$*,* $q_{\mathrm{M}}^0$*,* $\Pi_{\mathrm{M}}$*, and* $\mathcal{L}_{\mathrm{M}}$ *are the set of states, the initial state, the set of atomic propositions, and the satisfaction map, respectively, as in Definition 2.1, and*

- $\delta_{\mathrm{M}} : \mathcal{Q}_{\mathrm{M}} \times \mathcal{Q}_{\mathrm{M}} \to [0,1]$ *is the transition probability function that satisfies* $\sum_{q' \in \mathcal{Q}_{\mathrm{M}}} \delta_{\mathrm{M}}(q, q') = 1 \; \forall \, q \in \mathcal{Q}_{\mathrm{M}}$.

In this paper, we are interested in temporal logic missions over a finite time horizon and we use scLTL formulas (Kupferman and Vardi, 2001) to specify them.

**Definition 2.3** (**Syntactically co-safe LTL formula).** *An scLTL formula* $\phi$ *over the set* $\Pi$ *of atomic propositions is defined by the following grammar:*

$$\phi := \top \mid \bot \mid \mathrm{p} \mid \neg\mathrm{p} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathbf{F}\phi \mid \mathbf{X}\phi \mid \phi\, \mathcal{U}\, \phi$$

*where* $\mathrm{p} \in \Pi$ *is an atomic proposition,* $\neg$ *(negation),* $\vee$ *(disjunction), and* $\wedge$ *(conjunction) are boolean operators, and* $\mathbf{F}$ *(eventually),* $\mathbf{X}$ *(next), and* $\mathcal{U}$ *(until) are temporal operators.*

For instance, $\mathbf{X}\,\mathrm{p}$ states that at the next position of the word, proposition p is true and $\mathbf{F}\,\mathrm{p}$ states that there exists $0 \le k$ such that $\omega^k$ satisfies p, where $\omega^k$ is the symbol at the $k$th position of the word. The formula $\mathrm{p}_1 \mathcal{U}\, \mathrm{p}_2$ states that there exists $0 \le k$ such that $\omega^k$ satisfies $\mathrm{p}_2$ and $\omega^j$ satisfies $\mathrm{p}_1$ for all $0 \le j < k$. Any scLTL formula can be written in positive normal form, where the negation operator $\neg$ occurs only in front of atomic propositions. For any scLTL formula $\phi$ over a set $\Pi$, one can construct a finite state automaton with input alphabet $2^{\Pi}$ accepting all and only finite words over $2^{\Pi}$ that satisfy $\phi$, which is defined next.

**Definition 2.4** (**Finite state automaton).** *A (deterministic) finite state automaton (FSA) is a tuple* $\mathbf{F} := (\mathcal{Q}_{\mathrm{F}}, q_{\mathrm{F}}^0, \Sigma_{\mathrm{F}}, \delta_{\mathrm{F}}, \mathcal{F}_{\mathrm{F}})$*, where*

- $\mathcal{Q}_{\mathrm{F}}$ *is a finite set of states;*
- $q_{\mathrm{F}}^0 \in \mathcal{Q}_{\mathrm{F}}$ *is the initial state;*
- $\Sigma_{\mathrm{F}}$ *is an input alphabet;*
- $\delta_{\mathrm{F}} : \mathcal{Q}_{\mathrm{F}} \times \Sigma_{\mathrm{F}} \times \mathcal{Q}_{\mathrm{F}}$ *is a deterministic transition relation;*
- $\mathcal{F}_{\mathrm{F}} \subseteq \mathcal{Q}_{\mathrm{F}}$ *is a set of accepting (final) states.*

A *run* of $\mathbf{F}$ over an input word $\omega = \omega^0, \omega^1, \ldots \omega^l$ where $\omega^i \in \Sigma_{\mathrm{F}} \;\forall\; i = 0, \ldots, l$ is a sequence $r_{\mathrm{F}} = q^0, q^1, \ldots, q^l, q^{l+1}$, such that $(q^i, \omega^i, q^{i+1}) \in \delta_{\mathrm{F}} \;\forall\; i = 0, \ldots, l$ and $q^0 = q_{\mathrm{F}}^0$. An FSA $\mathbf{F}$ accepts a word over $\Sigma_{\mathrm{F}}$ if and only if the corresponding run ends in some $q \in \mathcal{F}_{\mathrm{F}}$.

**Definition 2.5** (**Markov decision process).** *A Markov decision process (MDP) is a tuple* $\mathbf{P} := (\mathcal{Q}_{\mathrm{P}}, q_{\mathrm{P}}^0, \mathcal{A}_{\mathrm{P}}, \alpha_{\mathrm{P}}, \delta_{\mathrm{P}}, \Pi_{\mathrm{P}}, \mathcal{L}_{\mathrm{P}})$*, where*

- $\mathcal{Q}_{\mathrm{P}}$ *is a finite set of states;*
- $q_{\mathrm{P}}^0 \in \mathcal{Q}_{\mathrm{P}}$ *is the initial state;*
- $\mathcal{A}_{\mathrm{P}}$ *is a finite set of actions;*
- $\alpha_{\mathrm{P}} : \mathcal{Q}_{\mathrm{P}} \to 2^{\mathcal{A}_{\mathrm{P}}}$ *is a map giving the set of actions available at a state;*
- $\delta_{\mathrm{P}} : \mathcal{Q}_{\mathrm{P}} \times \mathcal{A}_{\mathrm{P}} \times \mathcal{Q}_{\mathrm{P}} \to [0,1]$ *is the transition probability function that satisfies* $\sum_{q' \in \mathcal{Q}_{\mathrm{P}}} \delta_{\mathrm{P}}(q, a, q') = 1 \;\forall\; q \in \mathcal{Q}_{\mathrm{P}}, a \in \alpha_{\mathrm{P}}(q)$ *and* $\sum_{q' \in \mathcal{Q}_{\mathrm{P}}} \delta_{\mathrm{P}}(q, a, q') = 0 \;\forall\; q \in \mathcal{Q}_{\mathrm{P}}, a \notin \alpha_{\mathrm{P}}(q)$*;*
- $\Pi_{\mathrm{P}}$ *is a finite set of atomic propositions;*
- $\mathcal{L}_{\mathrm{P}} : \mathcal{Q}_{\mathrm{P}} \to 2^{\Pi_{\mathrm{P}}}$ *is a map giving the set of atomic propositions satisfied at a state.*

For an MDP $\mathbf{P}$, we define a stationary policy $\mu_{\mathrm{P}} : \mathcal{Q}_{\mathrm{P}} \to \mathcal{A}_{\mathrm{P}}$ such that for a state $q \in \mathcal{Q}_{\mathrm{P}}$, $\mu_{\mathrm{P}}(q) \in \alpha_{\mathrm{P}}(q)$. This stationary policy can then be used to resolve all nondeterministic choices in $\mathbf{P}$ by applying action $\mu_{\mathrm{P}}(q)$ at each $q \in \mathcal{Q}_{\mathrm{P}}$, essentially inducing an MC $\mathbf{P}^{\mu_{\mathrm{P}}}$ on $\mathbf{P}$ that models the behavior of $\mathbf{P}$ under control policy $\mu_{\mathrm{P}}$ (Baier and Katoen, 2008). A path of $\mathbf{P}$ under policy $\mu_{\mathrm{P}}$ is a finite sequence of states $r = q^0, q^1, \ldots, q^l$ such that $l \ge 0$, $q^0 = q_{\mathrm{P}}^0$ and $\delta_{\mathrm{P}}(q^{k-1}, \mu_{\mathrm{P}}(q^{k-1}), q^k) > 0 \;\forall\; 0 < k \le l$. A finite path $r$ of $\mathbf{P}$ generates a finite word $\mathcal{L}_{\mathrm{P}}(r) = \mathcal{L}_{\mathrm{P}}(q^0), \mathcal{L}_{\mathrm{P}}(q^1), \ldots, \mathcal{L}_{\mathrm{P}}(q^l)$ where $\mathcal{L}_{\mathrm{P}}(q^k)$ is the set of atomic propositions satisfied at state $q^k$. We use $\mathrm{Paths}_{\mathrm{P}}^{\mu_{\mathrm{P}}}$ to denote the set of all finite paths of $\mathbf{P}$ under a policy $\mu_{\mathrm{P}}$. Finally, we define $\Pr^{\mu_{\mathrm{P}}}(\phi)$ as the probability of satisfying $\phi$ under policy $\mu_{\mathrm{P}}$.

**Remark 1.** *scLTL formulas have infinite time semantics, thus they are actually interpreted over infinite words (Kupferman and Vardi, 2001). Measurability of languages satisfying LTL formulas is also defined for infinite words generated by infinite paths (Baier and Katoen, 2008). However, one can determine whether a given infinite word satisfies an scLTL formula by considering only a finite prefix of it. It can be easily shown that our above definition of* $\mathrm{Paths}_{\mathrm{P}}^{\mu_{\mathrm{P}}}$ *inherits the same measurability property given in Baier and Katoen (2008).*

**Remark 2** (**Complexity of probabilistic synthesis and verification for scLTL formulas).** *Given an MDP* $\mathbf{M}$ *and an scLTL formula* $\phi$*, an optimal stationary control policy* $\mu^\star$ *that maximizes the probability of satisfying* $\phi$ *can be computed in time polynomial in the total number of nondeterministic choices in the product MDP* $\mathbf{P} := \mathbf{M} \otimes \mathbf{F}$ *(see Section 4.2), where* $\mathbf{F}$ *is the deterministic FSA (Definition 2.4) corresponding to* $\phi$ *(Baier and Katoen, 2008).*

*Given the product MDP* **P** *and a control policy $\mu$, the probability of satisfying $\phi$ under $\mu$ can also be computed in time polynomial in the total number of non-deterministic choices in the MC* $\mathbf{P}^{\mu}$ *induced on* **P** *by $\mu$. Let $n$ be the number of states of* **P** *and $m$ be the number of available actions, i.e. non-deterministic choices, at each state of* **P**. *Since* $\mathbf{P}^{\mu}$ *is an MC, it has at most one non-deterministic choice at each state and at most n states. Then, synthesis can be done in time polynomial in $m \times n$, whereas verification can be done in time polynomial in n. Thus, depending on the total number of non-deterministic choices, one can solve a larger verification problem using the same resources required by a smaller synthesis problem.*

## 3. Problem formulation and approach

In this section we introduce the control synthesis problem with temporal logic constraints for a system that models a robot operating in the presence of independent, probabilistic, uncontrollable agents.

### 3.1. System model

Consider a system consisting of a deterministic robot that we can control (e.g. a car) and $n$ agents operating in an environment modeled by a graph

$$\mathcal{E} = (V, \rightarrow_{\mathcal{E}}, \mathcal{L}_{\mathcal{E}}, \Pi_{\mathcal{E}})$$

where $V$ is the set of vertices, $\rightarrow_{\mathcal{E}} \subseteq V \times V$ is the set of edges, and $\mathcal{L}_{\mathcal{E}}$ is the labeling function that maps each vertex to a proposition in $\Pi_{\mathcal{E}}$. For example, $\mathcal{E}$ can be the quotient graph of a partitioned environment, where $V$ is a set of labels for the regions in the partition and $\rightarrow_{\mathcal{E}}$ is the corresponding adjacency relation (see Figure 1). Agent $i$ is modeled as an MC $\mathbf{M}_i = (\mathcal{Q}_i, q_i^0, \delta_i, \Pi_i, \mathcal{L}_i)$, with $\mathcal{Q}_i \subseteq V$ and $\delta_i : \mathcal{Q}_i \times \mathcal{Q}_i \rightarrow [0,1]$, $i = 1, \ldots, n$, while the robot is assumed to be a deterministic transition system TS $\mathbf{T} = (\mathcal{Q}_{\mathrm{T}}, q_{\mathrm{T}}^0, \mathcal{A}_{\mathrm{T}}, \alpha_{\mathrm{T}}, \delta_{\mathrm{T}}, \Pi_{\mathrm{T}}, \mathcal{L}_{\mathrm{T}})$, where $\mathcal{Q}_{\mathrm{T}} \subseteq V$ and $\delta_t \subseteq \mathcal{Q}_{\mathrm{T}} \times \mathcal{A}_{\mathrm{T}} \times \mathcal{Q}_{\mathrm{T}}$ (see Figure 2). We assume that all components of the system (the robot and the agents) make transitions synchronously by picking edges of the graph. We also assume that the state of the system is perfectly known at any given instant and we can control the robot but we have no control over the agents. We define the sets of propositions and labeling functions of the individual components of the system such that they inherit the propositions of their current vertex from the graph while preserving their own identities. We slightly abuse notation and use T and $1, 2, \ldots, n$ as the *identities* of the robot and the independent agents, respectively. Formally, we have $\Pi_{\mathrm{T}} = \{(\mathrm{T}, \mathcal{L}_{\mathcal{E}}(q)) \,|\, q \in \mathcal{Q}_{\mathrm{T}}\}$ and $\mathcal{L}_{\mathrm{T}}(q) = (\mathrm{T}, \mathcal{L}_{\mathcal{E}}(q))$ for the robot, and $\Pi_i = \{(i, \mathcal{L}_{\mathcal{E}}(q)) \,|\, q \in \mathcal{Q}_i\}$ and $\mathcal{L}_i(q) = (i, \mathcal{L}_{\mathcal{E}}(q))$ for agent $i$. Finally, we define the set $\Pi$ of propositions as $\Pi = \Pi_{\mathrm{T}} \cup \Pi_1 \cup \cdots \cup \Pi_n \subseteq \{(i,p) \,|\, i \in \{\mathrm{T}, 1, \ldots, n\}, p \in \Pi_{\mathcal{E}}\}$.

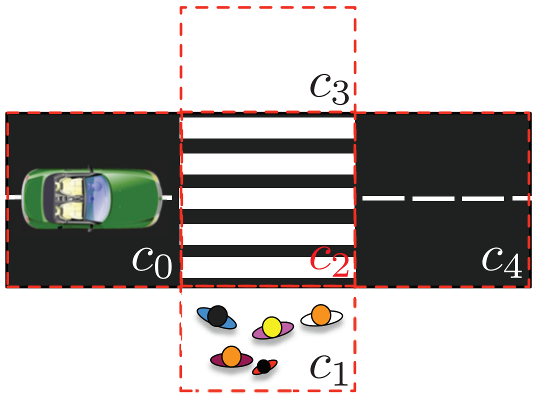


**Fig. 1.** A partitioned road environment, where a car (robot) is required to reach $c_4$ without colliding with any of the pedestrians (agents).

### 3.2. Problem formulation

As it will become clear in Section 4.4, the joint behavior of the robot and agents in the graph environment can be modeled by the parallel composition of the TS and MC models described above, which takes the form of an MDP (see Definition 2.5). Given a scLTL formula $\phi$ over $\Pi$, our goal is to synthesize a policy for this MDP, which we will simply refer to as the *system*, such that the probability of satisfying $\phi$ is either maximized or above a given threshold. Since we assume perfect state information, the robot can implement a control policy computed for the system, i.e. based on its state and the state of all the other agents. As a result, we will not distinguish between a control policy for the robot and a control policy for the system, and we will refer to it simply as the *control policy*. We can now formulate the main problem considered in this paper:

**Problem 3.1.** *Given a system described by a robot* **T** *and a set of agents* $\mathbf{M}_1, \ldots, \mathbf{M}_n$ *operating on a graph* $\mathcal{E}$, *and given a specification expressed as a scLTL formula* $\phi$ *over* $\Pi$, *synthesize a control policy* $\mu^{\star}$ *that satisfies the following:*

1. *If a probability threshold* $p_{thr}$ *is given, the probability that the system satisfies* $\phi$ *under* $\mu^{\star}$ *is at least* $p_{thr}$.
2. *Otherwise,* $\mu^{\star}$ *maximizes the probability that the system satisfies* $\phi$.

*If no such policy exists, report failure.*

As will be shown in Section 4.1, the parallel composition of MDP and MC models also takes the form of an MDP. Hence, our approach can easily accommodate the case where the robot is a Markov decision process. We consider a deterministic robot only for simplicity of presentation.

**Example 3.2.** *Figure 1 illustrates a car in a five-cell environment with five pedestrians, where* $\mathcal{L}_{\mathcal{E}}(v) = v$ *for* $v \in \{c_0, \ldots, c_4\}$. *Figure 2 illustrates the TS* **T** *and the MCs* $\mathbf{M}_1, \ldots, \mathbf{M}_5$ *that model the car and the pedestrians. The car is required to reach the end of the crossing (cell* $c_4$*)*

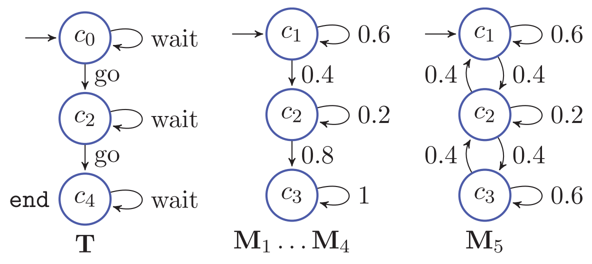


**Fig. 2.** TS **T** and MCs $\mathbf{M}_1, \dots, \mathbf{M}_5$ that model the car and the pedestrians. Note that the model of the fifth pedestrian $\mathbf{M}_5$ is different than the models of the remaining pedestrians.

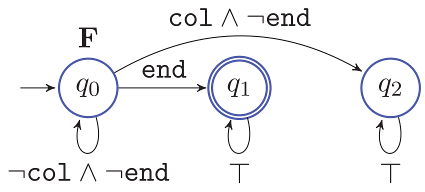


**Fig. 3.** Deterministic FSA **F** that corresponds to $\phi = \neg \texttt{col}\ \mathcal{U}\ \texttt{end}$ where $\texttt{col} = \bigvee_{i=1,\dots,5, j=0,\dots,4}((\mathrm{T}, c_j) \wedge (i, c_j))$ and $\texttt{end} = (\mathrm{T}, c_4)$. $q_0$ and $q_1$ are the initial and final states, respectively.

*without colliding with any of the pedestrians. To enforce this behavior, we write our specification as*

$$\phi := \left( \neg \bigvee_{i=1,\dots,5,\ j=0,\dots,4} ((\mathrm{T}, c_j) \wedge (i, c_j)) \right) \mathcal{U}\, (\mathrm{T}, c_4) \quad (1)$$

*The deterministic FSA that corresponds to $\phi$ is given in Figure 3, where* $\texttt{col} = \bigvee_{i=1,\dots,5,\ j=0,\dots,4}((\mathrm{T}, c_j) \wedge (i, c_j))$ *and* $\texttt{end} = (\mathrm{T}, c_4)$.

**Remark 3.** *Note that since the specification $\phi$ is an scLTL formula over $\Pi$, the alphabet of the corresponding FSA can be potentially large due to the large number of propositions used to define the properties of interest. Instead, one can trivially rewrite $\phi$ by replacing the Boolean conjunctions and disjunctions of the propositions in $\Pi$ with new atomic propositions resulting in an equivalent but more succinct formula which translates to an FSA with a considerably smaller alphabet as we show in Example 3.2. These new atomic propositions are then satisfied at those states of the system where their respective subformulas are satisfied.*

### *3.3. Solution outline*

One can directly solve Problem 3.1 by reducing it to a maximal reachability probability (MRP) problem on the MDP modeling the overall system as given in Alfaro (1997); Baier and Katoen (2008). This approach, however, is very resource demanding as it scales exponentially with the number of agents. As a result, the environment size and the number of agents that can be handled in a reasonable time frame and with limited memory are small. To address this issue, we propose a highly efficient incremental control synthesis algorithm that exploits the independence between the system components and the fact that verification is less demanding than synthesis. Figure 4 gives an overview of our approach, where a cycle of the flow chart corresponds to an iteration of our algorithm. At each iteration $i$, our method will involve the following steps: synthesis of an optimal control policy considering only some of the agents (Section 4.4), verification of this control policy with respect to the complete system (Section 4.5), and reduction of the size of the system model using the values obtained in the synthesis and verification steps (Section 4.6).

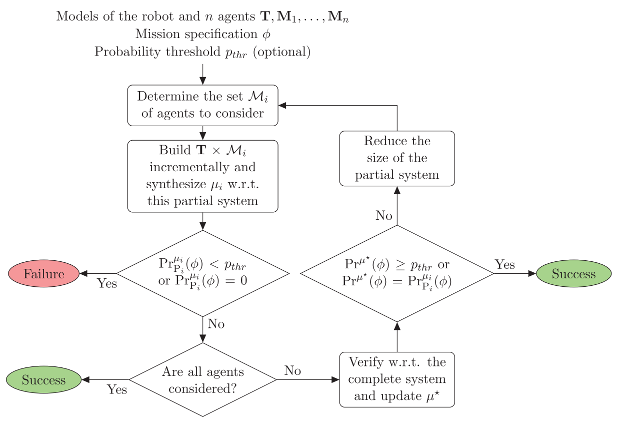

Models of the robot and $n$ agents $\mathbf{T}, \mathbf{M}_1, \dots, \mathbf{M}_n$
Mission specification $\phi$
Probability threshold $p_{thr}$ (optional)
Determine the set $\mathcal{M}_i$ of agents to consider
Build $\mathbf{T} \times \mathcal{M}_i$ incrementally and synthesize $\mu_i$ w.r.t. this partial system
Reduce the size of the partial system
No
$\mathrm{Pr}^{\mu_i}_{\mathrm{P}_i}(\phi) < p_{thr}$ or $\mathrm{Pr}^{\mu_i}_{\mathrm{P}_i}(\phi) = 0$
Failure
Yes
$\mathrm{Pr}^{\mu^\star}(\phi) \geq p_{thr}$ or $\mathrm{Pr}^{\mu^\star}(\phi) = \mathrm{Pr}^{\mu_i}_{\mathrm{P}_i}(\phi)$
Yes
Success
No
Are all agents considered?
Success
Yes
No
Verify w.r.t. the complete system and update $\mu^\star$

**Fig. 4.** An overview of our approach. $\mathrm{Pr}^{\mu_i}_{\mathrm{P}_i}(\phi)$ denotes the maximum probability of satisfying $\phi$ in the product MDP $\mathbf{P}_i$ computed at the $i$th iteration and $\mathrm{Pr}^{\mu^\star}(\phi)$ denotes the probability of satisfying $\phi$ under the best policy computed so far.

## 4. Problem solution

Our solution to Problem 3.1 is given in the form of Algorithm 1. In the rest of this section, we explain each of its steps in detail.

### *4.1. Parallel composition of system components*

Given the set $\mathcal{M} = \{\mathbf{M}_1, \dots, \mathbf{M}_n\}$ of all agents, we use $\mathcal{M}_i \subseteq \mathcal{M}$ to denote its subset used at iteration $i$. Then, we define the synchronous parallel composition $\mathbf{T} \otimes \mathcal{M}_i$ of **T** and agents in $\mathcal{M}_i = \{\mathbf{M}_{i1}, \dots, \mathbf{M}_{ij}\}$ for different types of **T** as follows.

If **T** is a TS, then we define $\mathbf{T} \otimes \mathcal{M}_i$ as the MDP $\mathbf{A} = (\mathcal{Q}_\mathrm{A}, q^0_\mathrm{A}, \mathcal{A}_\mathrm{A}, \alpha_\mathrm{A}, \delta_\mathrm{A}, \Pi_\mathrm{A}, \mathcal{L}_\mathrm{A}) = \mathbf{T} \otimes \mathcal{M}_i$, such that

- $\mathcal{Q}_\mathrm{A} \subseteq \mathcal{Q}_\mathrm{T} \times \mathcal{Q}_{i1} \times \cdots \times \mathcal{Q}_{ij}$ such that a state $q = (q_\mathrm{T}, q_{i1}, \dots, q_{ij})$ exists iff it is reachable from the initial states;
- $q^0_\mathrm{A} = (q^0_\mathrm{T}, q^0_{i1}, \dots, q^0_{ij})$;
- $\mathcal{A}_\mathrm{A} = \mathcal{A}_\mathrm{T}$;
- $\alpha_\mathrm{A}(q) = \alpha_\mathrm{T}(q_\mathrm{T})$, where $q_\mathrm{T}$ is the element of $q$ that corresponds to the state of **T**;
- $\Pi_\mathrm{A} = \Pi_\mathrm{T} \cup \Pi_{i1} \cup \cdots \cup \Pi_{ij}$;

**Algorithm 1:** INCREMENTAL-CONTROL-SYNTHESIS

```
   Input: $\mathbf{T}, \mathbf{M}_1, \ldots, \mathbf{M}_n, \phi$, (optional: $p_{thr}$).
   Output: $\mu^\star$ s.t. $\Pr^{\mu^\star}(\phi) \geq \Pr^{\mu}(\phi)\ \forall \mu$ if $p_{thr}$ is not given,
           otherwise $\Pr^{\mu^\star}(\phi) \geq p_{thr}$.
1  $\mathcal{M} = \{\mathbf{M}_j \mid (j,p) \in \phi, j \in \{1, \ldots, n\}\}$, $i \leftarrow 0$.
2  Construct FSA $\mathbf{F}$ corresponding to $\phi$.
3  $\mu^\star \leftarrow \emptyset$, $\Pr^{\mu^\star}(\phi) \leftarrow 0$, $\mathcal{M}_i \leftarrow \emptyset$, $\mathbf{A}_i \leftarrow \mathbf{T}$, $i \leftarrow 1$.
4  Process $\phi$ to form $\mathcal{M}^+$ and $\mathcal{M}^-$.
5  if $|\mathcal{M}^+| \leq |\mathcal{M}^-|$ then mode $\leftarrow$ avoid and $\mathcal{M}_i^{\text{new}} \leftarrow \mathcal{M}^+$.
6  else mode $\leftarrow$ reach and $\mathcal{M}_i^{\text{new}} \leftarrow \mathcal{M}^-$.
7  if $\mathcal{M}_i^{\text{new}} = \emptyset$ then Add an arbitrary element to
   $\mathcal{M}_i^{\text{new}}$ from $\mathcal{M}$.
8  while True do
9      $\mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \cup \mathcal{M}_i^{\text{new}}$, $\overline{\mathcal{M}_i} \leftarrow \mathcal{M} \setminus \mathcal{M}_i$.
10     $\mathbf{A}_i \leftarrow \mathbf{A}_{i-1} \otimes \mathcal{M}_i^{\text{new}}$.
11     if mode = reach then
12         $\mathcal{L}_{\mathrm{A}_i}(q) = \mathcal{L}_{\mathrm{A}_i}(q) \cup \{(j,p) \mid (j,p) \in \phi, \mathbf{M}_j \in \overline{\mathcal{M}_i}\}\ \forall q \in$
           $\mathcal{Q}_{\mathrm{A}_i}$, and $\Pi_{\mathrm{A}_i} = \Pi_{\mathrm{A}_i} \cup (\bigcup_{\mathbf{M}_j \in \overline{\mathcal{M}_i}} \Pi_{\mathrm{M}_j})$.
13     $\mathbf{P}_i \leftarrow \mathbf{A}_i \otimes \mathbf{F}$.
14     Synthesize $\mu_i$ that maximizes $\Pr_{\mathrm{P}_i}^{\mu_i}(\phi)$ using $\mathbf{P}_i$.
15     if $p_{thr}$ given and $\Pr_{\mathrm{P}_i}^{\mu_i}(\phi) < p_{thr}$ then
16         Fail: $\nexists\, \mu$ such that $\Pr^{\mu}(\phi) \geq p_{thr}$.
17     else if $\Pr_{\mathrm{P}_i}^{\mu_i}(\phi) = 0$ then Fail: $\phi$ cannot be satisfied.
18     else if $\mathcal{M}_i = \mathcal{M}$ then Success: Return $\mu_i$.
19     else
20         if mode = reach then
21             $\mathcal{L}_{\mathrm{A}_i}(q) = \mathcal{L}_{\mathrm{A}_i}(q) \setminus \{(j,p) \mid (j,p) \in \phi, \mathbf{M}_j \in$
               $\overline{\mathcal{M}_i}\}\ \forall q \in \mathcal{Q}_{\mathrm{A}_i}$, and $\Pi_{\mathrm{A}_i} = \Pi_{\mathrm{A}_i} \setminus (\bigcup_{\mathbf{M}_j \in \overline{\mathcal{M}_i}} \Pi_{\mathrm{M}_j})$.
               Perform the same operation on $\mathbf{P}_i$.
22         Obtain the MC $\mathbf{M}_{\mathrm{P}_i}^{\mu_i}$ induced on $\mathbf{P}_i$ by $\mu_i$.
23         $\mathbf{M}_{\mathcal{M}}^{\mu_i} \leftarrow \mathbf{M}_{\mathrm{P}_i}^{\mu_i} \otimes \overline{\mathcal{M}_i}$.
24         $\mathbf{R}_i \leftarrow \mathbf{M}_{\mathcal{M}}^{\mu_i} \otimes \mathbf{F}$.
25         Compute $\Pr^{\mu_i}(\phi)$ using $\mathbf{R}_i$.
26         if $\Pr^{\mu_i}(\phi) > \Pr^{\mu^\star}(\phi)$ then
           $\mu^\star \leftarrow \mu_i$, $\Pr^{\mu^\star}(\phi) \leftarrow \Pr^{\mu_i}(\phi)$.
27         if $\Pr_{\mathrm{P}_i}^{\mu_i}(\phi) = \Pr^{\mu^\star}(\phi)$ then
28             Success: Return $\mu^\star$.
29         if $p_{thr}$ given and $\Pr^{\mu^\star}(\phi) \geq p_{thr}$ then
30             Success: Return $\mu^\star$.
31         else
32             $\mathcal{M}_{i+1}^{\text{new}} \leftarrow \{\mathbf{M}_j\}$, where $\mathbf{M}_j$ is the smallest agent in
               $\overline{\mathcal{M}_i}$, reduce the size of $\mathbf{A}_i$, $i \leftarrow i+1$.
```

- $\mathcal{L}_{\mathrm{A}}(q) = \mathcal{L}_{\mathrm{T}}(q_{\mathrm{T}}) \cup \mathcal{L}_{i1}(q_{i1}) \cup \cdots \cup \mathcal{L}_{ij}(q_{ij})$;
- $\delta_{\mathrm{A}}(q = (q_{\mathrm{T}}, q_{i1}, \ldots, q_{ij}), a, q' = (q'_{\mathrm{T}}, q'_{i1}, \ldots, q'_{ij})) = 1\{(q_{\mathrm{T}}, a, q'_{\mathrm{T}}) \in \delta_T\} \times \delta(q_{i1}, q'_{i1}) \times \cdots \times \delta(q_{ij}, q'_{ij})$,

where $1\{\cdot\}$ is the indicator function.

If $\mathbf{T}$ is an MDP, then we define $\mathbf{T} \otimes \mathcal{M}_i$ as the MDP $\mathbf{A} = (\mathcal{Q}_{\mathrm{A}}, q_{\mathrm{A}}^0, \mathcal{A}_{\mathrm{A}}, \alpha_{\mathrm{A}}, \delta_{\mathrm{A}}, \Pi_{\mathrm{A}}, \mathcal{L}_{\mathrm{A}}) = \mathbf{T} \otimes \mathcal{M}_i$, such that $\mathcal{Q}_{\mathrm{A}}, q_{\mathrm{A}}^0, \mathcal{A}_{\mathrm{A}}, \alpha_{\mathrm{A}}, \Pi_{\mathrm{A}}$, and $\mathcal{L}_{\mathrm{A}}$ are as given in the case where $\mathbf{T}$ is a TS and

- $\delta_{\mathrm{A}}(q = (q_{\mathrm{T}}, q_{i1}, \ldots, q_{ij}), a, q' = (q'_{\mathrm{T}}, q'_{i1}, \ldots, q'_{ij})) = \delta_{\mathrm{T}}(q_{\mathrm{T}}, a, q'_{\mathrm{T}}) \times \delta_{i1}(q_{i1}, q'_{i1}) \times \cdots \times \delta_{ij}(q_{ij}, q'_{ij})$.

Finally if $\mathbf{T}$ is an MC, then we define $\mathbf{T} \otimes \mathcal{M}_i$ as the MC $\mathbf{A} = (\mathcal{Q}_{\mathrm{A}}, q_{\mathrm{A}}^0, \delta_{\mathrm{A}}, \Pi_{\mathrm{A}}, \mathcal{L}_{\mathrm{A}}) = \mathbf{T} \otimes \mathcal{M}_i$ where $\mathcal{Q}_{\mathrm{A}}, q_{\mathrm{A}}^0, \Pi_{\mathrm{A}}, \mathcal{L}_{\mathrm{A}}$ are as given in the case where $\mathbf{T}$ is a TS and

- $\delta_{\mathrm{A}}(q = (q_{\mathrm{T}}, q_{i1}, \ldots, q_{ij}), q' = (q'_{\mathrm{T}}, q'_{i1}, \ldots, q'_{ij})) = \delta_{\mathrm{T}}(q_{\mathrm{T}}, q'_{\mathrm{T}}) \times \delta_{i1}(q_{i1}, q'_{i1}) \times \cdots \times \delta_{ij}(q_{ij}, q'_{ij})$.

## 4.2. *Product MDP and product MC*

Given the deterministic FSA $\mathbf{F}$ that recognizes all and only the finite words that satisfy $\phi$, we define the product of $\mathbf{M} \otimes \mathbf{F}$ for different types of $\mathbf{M}$ as follows.

If $\mathbf{M}$ is an MDP, we define $\mathbf{M} \otimes \mathbf{F}$ as the product MDP $\mathbf{P} = (\mathcal{Q}_{\mathrm{P}}, q_{\mathrm{P}}^0, \mathcal{A}_{\mathrm{P}}, \alpha_{\mathrm{P}}, \delta_{\mathrm{P}}, \Pi_{\mathrm{P}}, \mathcal{L}_{\mathrm{P}}) = \mathbf{M} \otimes \mathbf{F}$, where

- $\mathcal{Q}_{\mathrm{P}} \subseteq \mathcal{Q}_{\mathrm{M}} \times \mathcal{Q}_{\mathrm{F}}$ such that a state $q$ exists iff it is reachable from the initial states;
- $q_{\mathrm{P}}^0 = (q_{\mathrm{M}}^0, q_{\mathrm{F}})$ such that $(q_{\mathrm{F}}^0, \mathcal{L}_{\mathrm{M}}(q_{\mathrm{M}}^0), q_{\mathrm{F}}) \in \delta_{\mathrm{F}}$;
- $\mathcal{A}_{\mathrm{P}} = \mathcal{A}_{\mathrm{M}}$;
- $\alpha_{\mathrm{P}}((q_{\mathrm{M}}, q_{\mathrm{F}})) = \alpha_{\mathrm{M}}(q_{\mathrm{M}})$;
- $\Pi_{\mathrm{P}} = \Pi_{\mathrm{M}}$;
- $\mathcal{L}_{\mathrm{P}}((q_{\mathrm{M}}, q_{\mathrm{F}})) = \mathcal{L}_{\mathrm{M}}(q_{\mathrm{M}})$;
- $\delta_{\mathrm{P}}((q_{\mathrm{M}}, q_{\mathrm{F}}), a, (q'_{\mathrm{M}}, q'_{\mathrm{F}})) = 1\{(q_{\mathrm{F}}, \mathcal{L}_{\mathrm{M}}(q'_{\mathrm{M}}), q'_{\mathrm{F}}) \in \delta_{\mathrm{F}}\} \times \delta_{\mathrm{M}}(q_{\mathrm{M}}, a, q'_{\mathrm{M}})$,

where $1\{\cdot\}$ is the indicator function. In this product MDP, we also define the set $\mathcal{F}_{\mathrm{P}}$ of final states such that a state $q = (q_{\mathrm{M}}, q_{\mathrm{F}}) \in \mathcal{F}_{\mathrm{P}}$ iff $q_{\mathrm{F}} \in \mathcal{F}_{\mathrm{F}}$, where $\mathcal{F}_{\mathrm{F}}$ is the set of final states of $\mathbf{F}$.

If $\mathbf{M}$ is an MC, we define $\mathbf{M} \otimes \mathbf{F}$ as the product MC $\mathbf{P} = (\mathcal{Q}_{\mathrm{P}}, q_{\mathrm{P}}^0, \delta_{\mathrm{P}}, \Pi_{\mathrm{P}}, \mathcal{L}_{\mathrm{P}}) = \mathbf{M} \otimes \mathbf{F}$ where $\mathcal{Q}_{\mathrm{P}}, q_{\mathrm{P}}^0, \Pi_{\mathrm{P}}, \mathcal{L}_{\mathrm{P}}$ are as given in the case where $\mathbf{M}$ is an MDP and

- $\delta_{\mathrm{P}}((q_{\mathrm{M}}, q_{\mathrm{F}}), (q'_{\mathrm{M}}, q'_{\mathrm{F}})) = 1\{(q_{\mathrm{F}}, \mathcal{L}_{\mathrm{M}}(q'_{\mathrm{M}}), q'_{\mathrm{F}}) \in \delta_{\mathrm{F}}\} \times \delta_{\mathrm{M}}(q_{\mathrm{M}}, q'_{\mathrm{M}})$.

In this product MC, we also define the set $\mathcal{F}_{\mathrm{P}}$ of final states as given above.

## 4.3. *Initialization*

Lines 1 to 3 of Algorithm 1 correspond to the first part of the initialization of our algorithm. First, we form the set $\mathcal{M} = \{\mathbf{M}_j \mid (j,p) \in \phi, j \in \{1, \ldots, n\}\}$ of all agents that appear in the mission specification, where $(j,p) \in \phi$ means that proposition $(j,p)$ appears in $\phi$. Note that due to our assumption of independence between the components of the system, we can safely ignore any agent that is not a part of the mission specification $\phi$. Then, we construct the FSA $\mathbf{F}$ that corresponds to $\phi$, which can be automatically done using existing tools such as *scheck* from Latvala (2003). We use $\mathcal{M}_i \subseteq \mathcal{M}$ to denote the subset of independent agents considered in the synthesis step of the $i$th iteration of our algorithm. At any given iteration, the variables $\mu^\star$ and $\Pr^{\mu^\star}(\phi)$ hold the best control policy and the probability of satisfying $\phi$ under this policy in the presence of all agents, respectively. Since we have not synthesized any control policies so far, we reset $\mathcal{M}_i$ and $\mu^\star$, and set $\Pr^{\mu^\star}(\phi)$ to 0. Then, we set $\mathbf{A}_i$, which stands for the parallel composition of the robot $\mathbf{T}$ and the agents in $\mathcal{M}_i$, to $\mathbf{T}$, and set the iteration counter to 1.

For the sake of efficiency, our algorithm can operate in two modes: *avoid* and *reach*. The *avoid* mode applies to the cases where the robot has to avoid the majority of the agents in order to satisfy $\phi$, whereas the *reach* mode applies to the cases where the robot has to reach the majority of the agents in order to satisfy $\phi$. In the *avoid* mode, we start by considering all the agents that can satisfy $\phi$, denoted by the set $\mathcal{M}^+ \subseteq \mathcal{M}$, whereas in the *reach* mode, we start by considering all the agents that can violate $\phi$, denoted by the set $\mathcal{M}^- \subseteq \mathcal{M}$. In either mode, our algorithm starts by building a partial model composed of the robot and a subset of agents, either $\mathcal{M}^+$ or $\mathcal{M}^-$ depending on the mode of the algorithm, and incrementally adds the remaining agents until a termination condition is satisfied. This allows us to incrementally solve the synthesis problem while guaranteeing completeness as discussed in greater detail in Section 4.4 and Section 4.5. Next, we discuss how we form the sets $\mathcal{M}^+$ and $\mathcal{M}^-$ and choose the operation mode of the algorithm.

Lines 4 to 7 of Algorithm 1 choose the mode in which the algorithm will operate in and initializes the set $\mathcal{M}_1^{\text{new}}$ of agents that will be considered in the synthesis step of the first iteration. We first rewrite $\phi$ in positive normal form to obtain $\phi_{pnf}$, where the negation operator $\neg$ occurs only in front of atomic propositions. Conversion of $\phi$ to $\phi_{pnf}$ can be performed automatically using De Morgan's laws and equivalences for temporal operators as given in Baier and Katoen (2008). Then, we define the set $\mathcal{M}^+$ to be the set of agents that can satisfy the specification and include an agent $\mathbf{M}_i \in \mathcal{M}$ in $\mathcal{M}^+$ if any of its corresponding propositions of the form $(i,p) \in \Pi_i$ appears non-negated in $\phi_{pnf}$. Similarly, we define the set $\mathcal{M}^-$ to be the set of agents that can violate the specification and include an agent $\mathbf{M}_i \in \mathcal{M}$ in $\mathcal{M}^-$ if any of its corresponding propositions of the form $(i,p) \in \Pi_i$ appears negated in $\phi_{pnf}$. Note that the sets $\mathcal{M}^+$ and $\mathcal{M}^-$ are not necessarily mutually exclusive, i.e. there may be an agent $i$ with some proposition $(i,p) \in \Pi_i$ that appears both negated and non-negated in $\phi_{pnf}$. Next, we set $\mathcal{M}_1^{\text{new}}$ to the smaller of these two sets and set the mode to *avoid* if $\mathcal{M}_1^{\text{new}} = \mathcal{M}^+$ and to *reach* otherwise. Note that this comparison can also be performed in terms of the total size of the models in $\mathcal{M}^+$ and $\mathcal{M}^-$ if the model sizes of the individual agents differ a lot from each other. In case $\mathcal{M}_1^{\text{new}} = \emptyset$ after this procedure, we form $\mathcal{M}_1^{\text{new}}$ arbitrarily by including an agent from $\mathcal{M}$ and proceed with the synthesis step of our approach.

**Example 3.2 Revisited.** *For this example we have $\mathcal{M}^+ = \emptyset$, $\mathcal{M}^- = \{\mathbf{M}_1, \dots, \mathbf{M}_5\}$, and the algorithm operates in avoid mode. Since $\mathcal{M}_1^{\text{new}} = \emptyset$ after the procedure, we set $\mathcal{M}_1^{\text{new}} = \{\mathbf{M}_1\}$ and proceed with the synthesis step of our approach.*

## *4.4. Synthesis*

Lines 9 to 19 of Algorithm 1 correspond to the synthesis step of our approach. In the synthesis stage, we incrementally build a partial model of the complete system and synthesize an optimal control policy $\mu_i$ considering only the subset $\mathcal{M}_i \subseteq \mathcal{M}$ of agents.

At the *i*th iteration, the agent subset that we consider is given by $\mathcal{M}_i = \mathcal{M}_{i-1} \cup \mathcal{M}_i^{\text{new}}$, where $\mathcal{M}_i^{\text{new}}$ contains the agents that will be newly considered as selected at the end of the previous iteration or by the initialization procedure given in Section 4.3 if $i$ is 1. First, we construct the parallel composition $\mathbf{A}_i = \mathbf{A}_{i-1} \otimes \mathcal{M}_i^{\text{new}}$ of our robot and the agents in $\mathcal{M}_i$ as described in Section 4.1. Notice that, we use $\mathbf{A}_{i-1}$ to save from computation time and memory as $\mathbf{A}_{i-1} \otimes \mathcal{M}_i^{\text{new}}$ is typically smaller than $\mathbf{T} \otimes \mathcal{M}_i$ due to the reduction procedure explained in Section 4.6. Then, we handle the propositions of the agents that are not considered in the partial system model $\mathbf{A}_i$, denoted by $\overline{\mathcal{M}_i}$, so that the resulting synthesis problem is an optimistic simplification of the original problem, i.e. the maximum probability of satisfying $\phi$ in the partial system $\mathbf{A}_i$ is higher than or equal to the maximum probability of satisfying $\phi$ in the complete system $\mathbf{T} \otimes \mathbf{M}_1 \otimes \dots \mathbf{M}_n$. If the algorithm is in *avoid* mode, this is by construction as the partial system model $\mathbf{A}_i$ that we obtain at line 10 excludes the agents in $\overline{\mathcal{M}_i} = \mathcal{M}^- \setminus \mathcal{M}_i$ whose propositions may violate $\phi$. If the algorithm is in *reach* mode, we add the propositions of the agents in $\overline{\mathcal{M}_i} = \mathcal{M}^+ \setminus \mathcal{M}_i$ that may be needed to satisfy $\phi$ to the propositions of all states of the partial model $\mathbf{A}_i$ at line 12. Following this, we construct the product MDP $\mathbf{P}_i = \mathbf{A}_i \otimes \mathbf{F}$ as explained in Section 4.2. Then, our control synthesis problem can be solved by solving a maximal reachability probability (MRP) problem on $\mathbf{P}_i$, where one computes the maximum probability of reaching the set $\mathcal{F}_{\mathrm{P}_i}$ from the initial state $q^0_{\mathrm{P}_i}$ (Alfaro (1997)), after which the corresponding optimal control policy $\mu_i$ can be recovered as given in Baier and Katoen (2008) and Ding et al. (2011). Consequently, at line 14 of Algorithm 1 we solve the MRP problem on $\mathbf{P}_i$ using value iteration to obtain optimal policy $\mu_i$ that maximizes the probability of satisfaction of $\phi$ in the presence of the agents in $\mathcal{M}_i$. We denote this probability by $\Pr^{\mu_i}_{\mathrm{P}_i}(\phi)$, whereas $\Pr^{\mu_i}(\phi)$ stands for the probability that the complete system satisfies $\phi$ under policy $\mu_i$. Note that $\Pr^{\mu_i}_{\mathrm{P}_i}(\phi)$ is higher than or equal to the actual probability of satisfying $\phi$ in the presence of all agents which we will compute in the verification step of our algorithm. Next, we prove that $\{\Pr^{\mu_i}_{\mathrm{P}_i}(\phi)\}$ is a non-increasing sequence and is always greater than or equal to the maximum probability of the complete system satisfying $\phi$.

**Proposition 4.1.** *The sequence $\{\Pr^{\mu_i}_{\mathrm{P}_i}(\phi)\}$ is non-increasing and $\Pr^{\mu_i}_{\mathrm{P}_i}(\phi) \geq \Pr^{\mu'}(\phi)$, where $\mu'$ is an optimal control policy for the complete system.*

*Proof.* The proof follows from the fact that for any iteration $i$, the synthesis problem for the partial system model is an optimistic simplification of the original synthesis problem. Let MC $\mathbf{M}_j$ of agent $j$ be such that $\mathbf{M}_j \notin \mathcal{M}_1$. Let $r = q^0, q^1, \ldots, q^l$ be a path of the system after including $\mathbf{M}_j$ at iteration $i > 1$ and $\omega = \mathcal{L}(r) = \omega^0, \omega^1, \ldots, \omega^l$ be the word generated by $r$. We first consider the *avoid* mode, where $\mathbf{M}_j \notin \mathcal{M}_1 \Rightarrow \mathbf{M}_j \notin \mathcal{M}^+$ (due to line 5 of Algorithm 1). This means that the propositions of agent $j$ cannot appear non-negated in $\phi_{pnf}$ and therefore are not responsible for satisfaction of $\phi$. Consequently, in the *avoid* mode, if $\omega$ satisfies $\phi$, then $\tilde{\omega} = \tilde{\omega}^0, \tilde{\omega}^1, \ldots, \tilde{\omega}^l$ also satisfies $\phi$, where $\tilde{\omega}^k = \omega^k \setminus \mathcal{L}_j(q_j^k)$ for each $k \in \{0, \ldots, l\}$, $q_j^k$ is the state of $\mathbf{M}_j$ in $q^k$, and $\mathcal{L}_j(q_j^k)$ is the proposition satisfied at state $q_j^k$ of $\mathbf{M}_j$. For the case when the algorithm is in *reach* mode, we let $\tilde{\omega}^k = \omega^k \cup \{(j,p) \,|\, (j,p) \in \phi\}$ (due to line 12 of Algorithm 1) and see that if $\omega$ satisfies $\phi$, $\tilde{\omega}$ also satisfies $\phi$. This follows from the fact that in the *reach* mode, $\mathbf{M}_j \notin \mathcal{M}_1 \Rightarrow \mathbf{M}_j \notin \mathcal{M}^-$ (line 6 of Algorithm 1), meaning that the propositions of agent $j$ cannot appear negated in $\phi_{pnf}$ and they are not responsible for violation of $\phi$. Thus, we conclude that the probability of satisfying $\phi$ cannot increase after we add agent $\mathbf{M}_j \in \mathcal{M} \setminus \mathcal{M}_1$, and the sequence $\{P_{\mathrm{P}_i}^{\mu_i}(\phi)\}$ is non-increasing such that it attains its maximum value $P_{\mathrm{P}_1}^{\mu_1}(\phi)$ at the first iteration and does not increase as more agents from $\mathcal{M} \setminus \mathcal{M}_1$ are considered in the following iterations. When Algorithm 1 terminates at line 18 after considering all the independent agents at some iteration $k$ (assuming the failure conditions given at lines 15 and 17 are never satisfied), the control policy returned is an optimal control policy for the complete system under which the probability of satisfying $\phi$ equals the minimum of the sequence $\{\Pr_{\mathrm{P}_i}^{\mu_i}(\phi)\}$. Thus, $\Pr_{\mathrm{P}_i}^{\mu_i}(\phi) \geq \Pr^{\mu'}(\phi), i = 1, \ldots, k$, where $\mu'$ is an optimal control policy for the complete system. ■

**Corollary 4.2.** *If at any iteration* $\Pr_{\mathrm{P}_i}^{\mu_i}(\phi) < p_{thr}$, *then there does not exist a policy* $\mu : \Pr^{\mu}(\phi) \geq p_{thr}$, *where* $\mu_i$ *is an optimal control policy that we compute at the synthesis stage of the* i*th iteration considering only the agents in* $\mathcal{M}_i$.

The steps that we take at the end of the synthesis, i.e. lines 15 to 19 of Algorithm 1 are as follows. If $p_{thr}$ is given and $\Pr_{\mathrm{P}_i}^{\mu_i}(\phi) < p_{thr}$, we terminate by reporting that there exists no control policy $\mu : \Pr^{\mu}(\phi) \geq p_{thr}$, which is a direct consequence of Proposition 4.1. Else, if $\Pr_{\mathrm{P}_i}^{\mu_i}(\phi) = 0$, we terminate by reporting that the specification $\phi$ cannot be satisfied, which again follows from Proposition 4.1. Else, if $\mathcal{M}_i = \mathcal{M}$, we return $\mu_i$ as we have considered all of the agents (and $\mu_i$ satisfies the probability threshold $p_{thr}$ if it is given). Otherwise, we proceed with the verification of $\mu_i$ to compute the probability that the complete system satisfies $\phi$ under policy $\mu_i$, which we denote by $\Pr^{\mu_i}(\phi)$.

## 4.5. Verification and selection of $\mathcal{M}_{i+1}^{\mathrm{new}}$

Lines 20 to 32 of Algorithm 1 correspond to the verification stage of our algorithm. In the verification stage, we verify the policy $\mu_i$ that we have just synthesized considering the complete system, obtain the actual probability of satisfaction $\Pr^{\mu_i}(\phi)$, and update the best policy so far, which we denote by $\mu^\star$, as required.

Note that $\mu_i$ maximizes the probability of satisfying $\phi$ in the presence of agents in $\mathcal{M}_i$ and induces an MC by resolving all non-deterministic choices in $\mathbf{P}_i$. For the *reach* case, $\mu_i$ is synthesized assuming that the propositions of the remaining agents are satisfied at all states of $\mathbf{A}_i$ (line 12). Thus, if the algorithm is in *reach* mode, we first remove these propositions from the states of $\mathbf{A}_i$ and $\mathbf{P}_i$ (lines 20–21). Then, we obtain the induced Markov chain $\mathbf{M}_{\mathrm{P}_i}^{\mu_i}$ that captures the joint behavior of the robot and the agents in $\mathcal{M}_i$ under policy $\mu_i$, and we proceed by considering the agents that were not considered during synthesis of $\mu_i$, i.e. the agents in $\overline{\mathcal{M}_i} = \mathcal{M} \setminus \mathcal{M}_i$. In order to account for the existence of the agents that we newly consider, we exploit the independence between the components of the system and construct the MC $\mathbf{M}_{\mathcal{M}}^{\mu_i} = \mathbf{M}_{\mathrm{P}_i}^{\mu_i} \otimes \overline{\mathcal{M}_i}$ in line 23. In lines 24–25 of Algorithm 1, we construct the product MC $\mathbf{R}_i = \mathbf{M}_{\mathcal{M}}^{\mu_i} \otimes \mathbf{F}$ and compute the actual probability $\Pr^{\mu_i}(\phi)$ of satisfying $\phi$ for the complete system by computing the probability of reaching $\mathbf{R}_i$'s final states from its initial state using value iteration. Finally, in line 26 we update $\mu^\star$ if we have a policy that is better than the best we have found so far. Notice that keeping track of the best policy $\mu^\star$ makes Algorithm 1 an anytime algorithm since the algorithm can be terminated as soon as some $\mu^\star$ is obtained.

At the end of the verification stage, we first check if the probability that the partial system considered in the synthesis stage satisfies $\phi$ under policy $\mu_i$ equals the probability that the complete system satisfies $\phi$ under the best policy $\mu^\star$ found so far, i.e. $\Pr_{\mathrm{P}_i}^{\mu_i}(\phi) = \Pr^{\mu^\star}(\phi)$. If so, we terminate and return $\mu^\star$ as this indicates that the remaining agents do not affect the satisfaction of $\phi$ and $\mu^\star$ is therefore an optimal policy for the complete system (see Proposition 4.4). Else, if $p_{thr}$ is given and $\Pr^{\mu^\star}(\phi) \geq p_{thr}$ we terminate and return $\mu^\star$, as it satisfies the given probability threshold. Otherwise in line 32 of Algorithm 1, we pick the smallest $\mathbf{M}_j$ in terms of state and transition count, which we call the *smallest agent first (SAF)* rule. Note that, one can also choose to pick an arbitrary $\mathbf{M}_j \in \overline{\mathcal{M}_i}$ to be included in $\mathcal{M}_{i+1}$, which we call the *random agent first (RAF)* rule.

**Proposition 4.3.** *The sequence* $\{\Pr^{\mu^\star}(\phi)\}$ *is a non-decreasing sequence.*

*Proof.* The result directly follows from the fact that $\mu^\star$ is set to $\mu_i$ if and only if $\Pr^{\mu_i}(\phi) > P^{\mu^\star}(\phi)$. ■

**Proposition 4.4.** *If* $\Pr_{\mathrm{P}_k}^{\mu_k}(\phi) = \Pr^{\mu^\star}(\phi)$ *at some iteration k, then* $\Pr_{\mathrm{P}_k}^{\mu_k}(\phi) = \Pr^{\mu'}(\phi)$, *where* $\mu'$ *is an optimal control policy for the complete system and* $\mu^\star$ *is the best policy found until iteration k.*

*Proof.* Note that $\Pr_{\mathrm{P}_i}^{\mu_i}(\phi) \geq \Pr^{\mu'}$ and $\{\Pr_{\mathrm{P}_i}^{\mu_i}(\phi)\}$ is non-increasing for all $i$ (Proposition 4.1). Also note that $\Pr^{\mu'}(\phi) \geq \Pr^{\mu^\star}(\phi)$ and $\{\Pr^{\mu^\star}(\phi)\}$ is non-decreasing for all

$i$ (Proposition 4.3). Thus, if we have $\Pr^{\mu_k}_{\mathrm{P}_k}(\phi) = \Pr^{\mu^\star}(\phi)$ at some iteration $k$, then we can conclude that $\Pr^{\mu_k}_{\mathrm{P}_k}(\phi) = \Pr^{\mu'}(\phi)$ and $\Pr^{\mu_j}_{\mathrm{P}_j}(\phi) = \Pr^{\mu'}(\phi)$ for all $j \geq k$. ■

## 4.6. Size reduction

The reduction stage of our approach (line 32) aims to reduce the overall resource usage by removing those transitions and states of $\mathbf{A}_i$ that are not needed in the subsequent iterations. The main idea is to use the probability values obtained in the synthesis and verifications stages as over and under approximations for the probability of satisfying $\phi$ by taking some action at some state of $\mathbf{A}_i$ and removing those actions that will not be used in the following iterations.

Let $\mathrm{S}_i(q, a)$ denote the probability of satisfying $\phi$ after taking action $a$ at state $q$ of $\mathbf{P}_i$ under policy $\mu_i$ as given by the synthesis step of the $i$th iteration, where $a \in \mathcal{A}_{\mathrm{P}_i}(q)$. Also, let $\mathrm{V}_i(q)$ denote the probability of satisfying $\phi$ under policy $\mu_i$ from state $q$ of $\mathbf{R}_i$ as given by the verification step of the $i$th iteration. Since we are interested in reducing the size of $\mathbf{A}_i$ before it is used again in the next iteration, we first define two new operators $\overline{\mathrm{S}_i}(q, a)$ and $\underline{\mathrm{V}_i}(q)$ based on $\mathrm{S}_i$ and $\mathrm{V}_i$:

$$\overline{\mathrm{S}_i}(q, a) = \max_{q_{\mathrm{F}} \in \mathcal{Q}_{\mathrm{F}}} \mathrm{S}_i(q \oplus q_{\mathrm{F}}, a) \;\; \forall\, q \in \mathcal{Q}_{\mathrm{A}_i},\;\; a \in \mathcal{A}_{\mathrm{A}_i}(q)$$

and

$$\underline{\mathrm{V}_i}(q) = \min_{q' \text{ s.t. } q \oplus q' \in \mathcal{Q}_{\mathrm{R}_i}} \mathrm{V}_i(q \oplus q') \;\; \forall\, q \in \mathcal{Q}_{\mathrm{A}_i}$$

where $\oplus$ is the concatenation operator defined as follows. If both $q$ and $q'$ are tuples such that $q = (q_1, \ldots, q_n)$ and $q' = (q'_1, \ldots, q'_m)$, then $q \oplus q' = (q_1, \ldots, q_n, q'_1, \ldots, q'_m)$. If $q'$ is not a tuple then $q \oplus q' = (q_1, \ldots, q_n, q')$. Then, we remove an action $a$ from state $q$ of $\mathbf{A}_i$ if $\overline{\mathrm{S}_i}(q, a) < \underline{\mathrm{V}_i}(q)$ or $\overline{\mathrm{S}_i}(q, a) = 0$ and prune any states of $\mathbf{A}_i$ that are not reachable from the initial state. Next, we proceed with the synthesis step of the next iteration.

**Proposition 4.5.** *The reduction step does not affect the correctness and completeness of our approach.*

*Proof.* Since the reduction step does not add any new transitions and does not change any transition probabilities, it does not affect the correctness of our approach. Thus, it remains to show that reduction does not affect the completeness of our approach, which we prove by contradiction. Suppose that according to an optimal control policy $\mu'$ for the complete system, the optimal action to take at state $(q_{\mathrm{T}}, q_1, \ldots, q_n, q_{\mathrm{F}})$ of $\mathbf{T} \otimes \mathbf{M}_1 \otimes \cdots \otimes \mathbf{M}_n \otimes \mathbf{F}$, which results in a non-zero probability of satisfaction, is $a$. Suppose further that, w.l.o.g., at the first iteration of Algorithm 1 this action is removed from state $(q_{\mathrm{T}}, q_1)$ of the partial system model $\mathbf{A}_1 = \mathbf{T} \otimes \mathbf{M}_1$, causing the algorithm to fail to return a control policy or to return a suboptimal policy. For this to happen, we must have either $\overline{\mathrm{S}_1}((q_{\mathrm{T}}, q_1), a) < \underline{\mathrm{V}_1}((q_{\mathrm{T}}, q_1))$ or $\overline{\mathrm{S}_1}((q_{\mathrm{T}}, q_1), a) = 0$. Note that since $\Pr^{\mu_1}_{\mathrm{P}_1}(\phi) \geq \Pr^{\mu'}(\phi)$ (Proposition 4.1), $\overline{\mathrm{S}_1}((q_{\mathrm{T}}, q_1), a)$ is also greater than or equal to the maximum probability of satisfying $\phi$ by taking action $a$ at any state of the form $(q_{\mathrm{T}}, q_1, \ldots)$ of the complete system. Note also that, $\underline{\mathrm{V}_i}((q_{\mathrm{T}}, q_1))$ is less than or equal to the maximum probability of satisfying $\phi$ from any state of the complete system of the form $(q_{\mathrm{T}}, \underline{q_1}, \ldots)$. So, we must have $\overline{\mathrm{S}_1}((q_{\mathrm{T}}, q_1), a) \geq \underline{\mathrm{V}_1}((q_{\mathrm{T}}, q_1))$ and $\overline{\mathrm{S}_1}((q_{\mathrm{T}}, q_1), a) > 0$ which contradicts with our initial assumption. Thus, we conclude that the reduction step does not remove any actions that can be part of an optimal control policy for the complete system and therefore does not affect the completeness of our approach. ■

We finally show that Algorithm 1 correctly solves Problem 3.1.

**Proposition 4.6.** *Algorithm 1 solves Problem 3.1.*

*Proof.* Algorithm 1 combines all the steps given in this section and synthesizes a control policy $\mu^\star$ that either ensures $\Pr^{\mu^\star}(\phi) \geq p_{thr}$ if $p_{thr}$ is given, or maximizes $\Pr^{\mu^\star}(\phi)$. If Algorithm 1 terminates at line 15 or 17, completeness is guaranteed by the fact that $\Pr^{\mu_i}_{\mathrm{P}_i}$ is a non-increasing sequence as given in Proposition 4.1. Also, as given in Proposition 4.5, the reduction stage does not affect the correctness and completeness of the approach. Thus, Algorithm 1 solves Problem 3.1. ■

**Remark 4** (Asymptotic time and space complexity). *We must note that the worst-case (asymptotic) time and space complexity of our approach is the same as the classical approach where one solves the synthesis problem over a relatively large system model that captures all components of the system and the mission specification. This follows from the fact that, in the worst case, a probability threshold may not be given and the reduction step of our approach may not remove any transitions or states causing the algorithm to run until all agents are considered in the synthesis step. However, as discussed in the following section, in practice our algorithm may perform much better than the classical approach due to the reduction step as well as its ability to terminate early if a low enough probability threshold is given.*

## 5. Implementation and case studies

We implemented Algorithm 1 in Python as the LTL Optimal Multi-Agent Planner (LOMAP) package, which is publicly available online.[1] LOMAP uses the NetworkX graph package described in Hagberg et al. (2008) to represent various models in our implementation and the scheck software described in Latvala (2003) to convert scLTL specifications to deterministic finite state automata. A typical usage of our package is as follows:

1. The user defines the transition system $\mathbf{T}$ that models the robot and the Markov chains $\{\mathbf{M}_1, \ldots, \mathbf{M}_n\}$ that model the agents in individual plain text files using LOMAP's format.

2. The user writes a short python script that defines the mission specification $\phi$ expressed in scLTL, the probability threshold $p_{thr}$ (if desired), and calls the appropriate LOMAP function.
3. If $p_{thr}$ is not given, our implementation returns a control policy $\mu^\star$ that maximizes $\Pr^{\mu^\star}(\phi)$. If $p_{thr}$ is given, our implementation returns a control policy such that $\Pr^{\mu^\star}(\phi) \geq p_{thr}$. If no such policy exists, our implementation shows an error message and quits.

In the following, we compare the performance of our incremental approach given in Algorithm 1 with the performance of the classical method that attempts to solve this problem in a single pass using value iteration as given in Baier and Katoen (2008), Bianco and De Alfaro (1995), and Alfaro (1997). In our experiments we used an iMac i5 quad-core desktop computer and considered Python implementations of both approaches.

**Case study 1.** We return to the pedestrian crossing problem given in Example 3.2 and illustrated in Figures 1 and 2. The mission specification given in equation (1) (which we repeat here for convenience) is

$$\phi_1 := \left(\neg \bigvee_{i=1,\ldots,5, j=0,\ldots,4} ((\mathrm{T}, c_j) \wedge (i, c_j))\right) \mathcal{U} (\mathrm{T}, c_4)$$

and the corresponding deterministic FSA is given in Figure 3, where `col` $= \bigvee_{i=1,\ldots,5,j=0,\ldots,4} ((\mathrm{T}, c_j) \wedge (i, c_j))$ and `end` $= (\mathrm{T}, c_4)$. During the experiments, our algorithm ran in avoid mode and picked the new agent $\mathcal{M}_i^{\mathrm{new}}$ to be considered at the next iteration in the following order: $\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3, \mathbf{M}_4, \mathbf{M}_5$, i.e. according to the smallest agent first rule given in Section 4.5. When no $p_{thr}$ was given, optimal control policies synthesized by both of the algorithms satisfied $\phi$ with a probability of 0.8. The classical approach solved the control synthesis problem in 4.11 s, and the product MDP on which the MRP problem was solved had 1004 states and 26,898 transitions. In comparison, our incremental approach solved the same problem in 3.53 s, thanks to the reduction stage of our approach, which reduced the size of the problem at every iteration by pruning unneeded actions and states. The largest product MDP on which the MRP problem was solved in the synthesis stage of our approach had 266 states and 4474 transitions. The largest product MC that was considered in the verification stage of our approach had 486 states and 7000 transitions. The probabilities of satisfying $\phi$ under policy $\mu_i$ obtained at each iteration of our algorithm were $\Pr^{\mu_1}(\phi) = 0.463$, $\Pr^{\mu_2}(\phi) = 0.566$, $\Pr^{\mu_3}(\phi) = 0.627$, $\Pr^{\mu_4}(\phi) = 0.667$, and $\Pr^{\mu_5}(\phi) = 0.8$. When $p_{thr}$ was given as 0.6, our approach finished in 2 s and terminated after the third iteration returning a sub-optimal control policy with a 0.627 probability of satisfying $\phi$. In this case, the largest product MDP on which the MRP problem was solved in the synthesis stage had only 73 states and 322 transitions. Furthermore, since our algorithm runs in an anytime manner, it could be terminated as soon as a control policy was available, i.e. at the end of the first iteration (0.8 s). Figure 5 compares the classical single-pass approach with our incremental algorithm in terms of running time and state counts of the product MDPs and MCs. It is interesting to note that state count of the product MDP considered in the synthesis stage of our algorithm increases as more agents are considered, whereas state count of the product MC considered in the verification stage of our algorithm decreases as the minimization stage removes unneeded states and transitions after each iteration.

**Case study 2.** Next, we consider the setup given in the previous case study under a different scenario, e.g. a rescue/extraction operation, where friendlies $\mathbf{M}_1, \ldots, \mathbf{M}_4$ must be saved from adversary $\mathbf{M}_5$ by the car. More specifically, the car is expected to pick-up agents $\mathbf{M}_1, \ldots, \mathbf{M}_4$ while avoiding agent $\mathbf{M}_5$. We can express this mission specification in scLTL as

$$\phi_2 := \left(\bigwedge_{i=1,\ldots,4} \mathbf{F}\left(\bigvee_{j=0,\ldots,4} ((\mathrm{T}, c_j) \wedge (i, c_j))\right)\right)$$
$$\wedge \left(\left(\neg \bigvee_{j=0,\ldots,4} ((\mathrm{T}, c_j) \wedge (5, c_j))\right) \mathcal{U} (\mathrm{T}, c_4)\right).$$

If we let `reach_i` $= \bigvee_{j=0,\ldots,4} ((\mathrm{T}, c_j) \wedge (i, c_j))$, `col_5` $= \bigvee_{j=0,\ldots,4} ((\mathrm{T}, c_j) \wedge (5, c_j))$, and `end` $= (\mathrm{T}, c_4)$, $\phi_2$ can be rewritten simply as

$$\phi_2 := \left(\bigwedge_{i=1,\ldots,4} \mathbf{F}\,\texttt{reach\_i}\right) \wedge (\neg \texttt{col\_5}\ \mathcal{U}\ \texttt{end}).$$

The FSA that corresponds to $\phi_2$ has 33 states and 260 transitions. In this case, our algorithm ran in reach mode and added agents to $\mathcal{M}_i$ in the following order: $\mathbf{M}_5, \mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3, \mathbf{M}_4$. Optimal control policies synthesized by both the classical and the proposed algorithm satisfied $\phi$ with a probability of 0.157. The classical approach solved the control synthesis problem in 7.46 s and the product MDP on which the MRP problem was solved had 3116 states and 60,915 transitions, whereas our incremental approach solved the same problem in 5.94 s. The largest product MDP on which the MRP problem was solved in the synthesis stage of our approach had 371 states and 3264 transitions. The largest product MC that was considered in the verification stage of our approach had 2523 states and 40,547 transitions. The probabilities of satisfying $\phi$ under policy $\mu_i$ obtained at each iteration of our algorithm were $\Pr^{\mu_1}(\phi) = 0.004$, $\Pr^{\mu_2}(\phi) = 0.066$, $\Pr^{\mu_3}(\phi) = 0.104$, $\Pr^{\mu_4}(\phi) = 0.133$, and $\Pr^{\mu_5}(\phi) = 0.157$. Figure 6 compares the classical single-pass approach with our incremental algorithm in terms of running time and state counts of the product MDPs and MCs. It is interesting to note that,

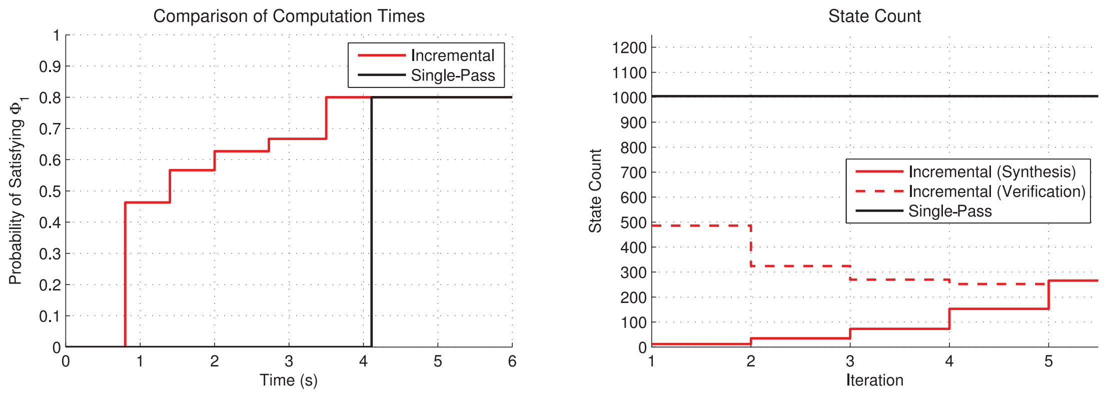


**Fig. 5.** Comparison of the classical single-pass and proposed incremental algorithms for case study 1. The left plot shows the running times of the algorithms and the probabilities of satisfying $\phi_1$ under synthesized policies. The right plot compares the state counts of the product MDPs for which synthesis was performed in both approaches (black and red lines) and shows the state count of the product MC considered in the verification stage of our incremental algorithm (red dashed line).

the number of states considered in the synthesis stage of the fifth iteration is less than the number of states considered at the synthesis stage of the fourth iteration, which is due to the large number of edges and states removed at the reduction stage of the fourth iteration (438 edges and 12 states). Note also that low probability of satisfaction of $\phi_2$ is due to the requirement to save all friendlies. If we relax the mission specification such that *at least one* friendly must be saved as opposed to all of them, i.e. $(\bigvee_{i=1,\ldots,4} \mathbf{F}\ \mathtt{reach\_i}) \wedge (\neg \mathtt{col\_5}\ \mathcal{U}\ \mathtt{end})$, optimal control policies computed by both approaches satisfy the specification with a probability of 0.606. In this case, the largest product MDP on which the MRP problem was solved in the synthesis stage of our approach had 985 states and 20,644 transitions whereas the classical approach had to construct a product MDP with 1404 states and 36,033 transitions to solve the control synthesis problem. If we further let the car to reverse from $c_2$ to $c_0$, the maximum probability of satisfaction increases by 0.000594. The reason for this small increase is because going back from $c_2$ to $c_0$ is optimal only when there are no pedestrians at $c_2$, which occurs only with a relatively small probability. It is also interesting to note that the number of states considered in the verification step of our approach can be less than the number of states of the partial system considered in the synthesis step, as in the fourth iteration of this case study. This can occur when the number of states of the Markov chain induced by the control policy computed in the synthesis step is small enough such that when the remaining components are added to this model, the resulting model is still smaller than the partial system model considered in the synthesis step.

**Case study 3.** Now, we consider a larger example where a robot has to safely drive out of a room with six traps. Figure 7(a) shows the room partitioned into 23 cells where the black cells correspond to the walls, the blue square at cell $c_0$ corresponds to the robot, and the cell highlighted in green ($c_{22}$) corresponds to the exit of the room (the target of the robot). The transition system $\mathbf{T}$ that models the motion of the robot in the room is shown in Figure 7(b), where we omit the action labels at the edges for ease of presentation. In Figure 7(a), the red and yellow circles represent the different kinds of traps located at various cells in the room, and Figures 7(c) and 7(d) show their respective Markov models. The traps have two states, *trig* (short for triggered) and *safe*. We assume that the robot can detect the current states of the traps at all times and becomes inoperative if it is in the same cell with a triggered trap. Note that our approach can easily capture such *logical* states of the agents by modeling them as cells in the environment which are not reachable by the robot. In this case, the set of cells in the environment would be $\{c_0, \ldots, c_{22}, \mathrm{trig}, \mathrm{safe}\}$, where the robot can only move between $c_0, \ldots, c_{22}$ (Figure 7(b)) and the traps move between trig and safe (Figures 7(c), 7(d)). The mission specification for this example can be expressed as the scLTL formula:

$$\phi_3 := \neg\, \mathtt{unsafe}\, \mathcal{U}\, \mathtt{end}$$

where $\mathtt{unsafe} = ((\mathrm{T}, c_9) \wedge (1, \mathrm{trig})) \vee ((\mathrm{T}, c_{17}) \wedge (2, \mathrm{trig})) \vee ((\mathrm{T}, c_{19}) \wedge (3, \mathrm{trig})) \vee ((\mathrm{T}, c_2) \wedge (4, \mathrm{trig})) \vee ((\mathrm{T}, c_{11}) \wedge (5, \mathrm{trig})) \vee ((\mathrm{T}, c_8) \wedge (6, \mathrm{trig}))$, and $\mathtt{end} = (\mathrm{T}, c_{22})$. In this case, our algorithm ran in avoid mode and terminated at the end of the fourth iteration after adding agents (traps) to $\mathcal{M}_i$ in the following order: $\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3, \mathbf{M}_4$. Note that our algorithm terminated before considering $\mathbf{M}_5$ and $\mathbf{M}_6$ in the synthesis as it already found an optimal solution for the complete system at the fourth iteration. The optimal control policies synthesized by both the classical approach and the proposed algorithm satisfied $\phi$ with a probability of 0.512. The classical approach solved the control synthesis problem in 60 s and the product MDP on which the MRP problem was solved had 2752 states and 540,672 transitions, whereas our incremental approach solved the same problem in 20 s. The largest product MDP on which

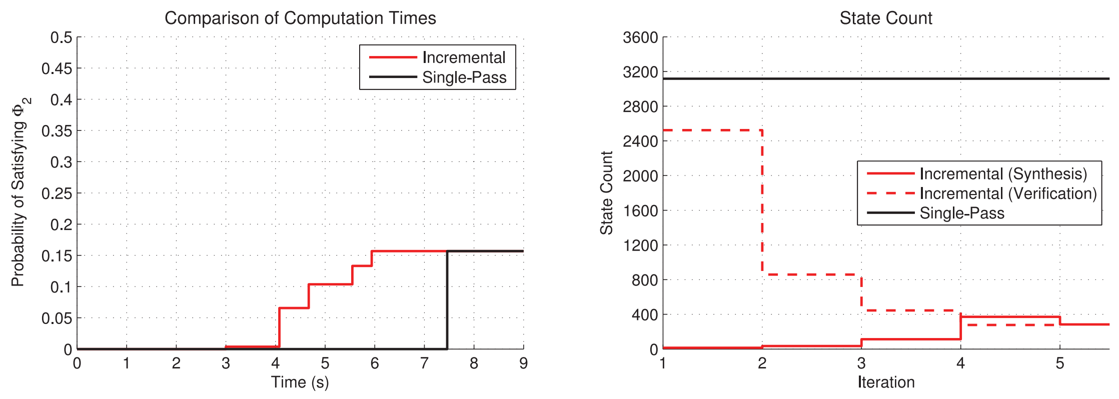


**Fig. 6.** Comparison of the classical single-pass and proposed incremental algorithms for case study 2. The left plot shows the running times of the algorithms and the probabilities of satisfying $\phi_2$ under synthesized policies. The right plot compares the state counts of the product MDPs for which synthesis was performed in both approaches (black and red lines) and shows the state count of the product MC considered in the verification stage of our incremental algorithm (red dashed line).

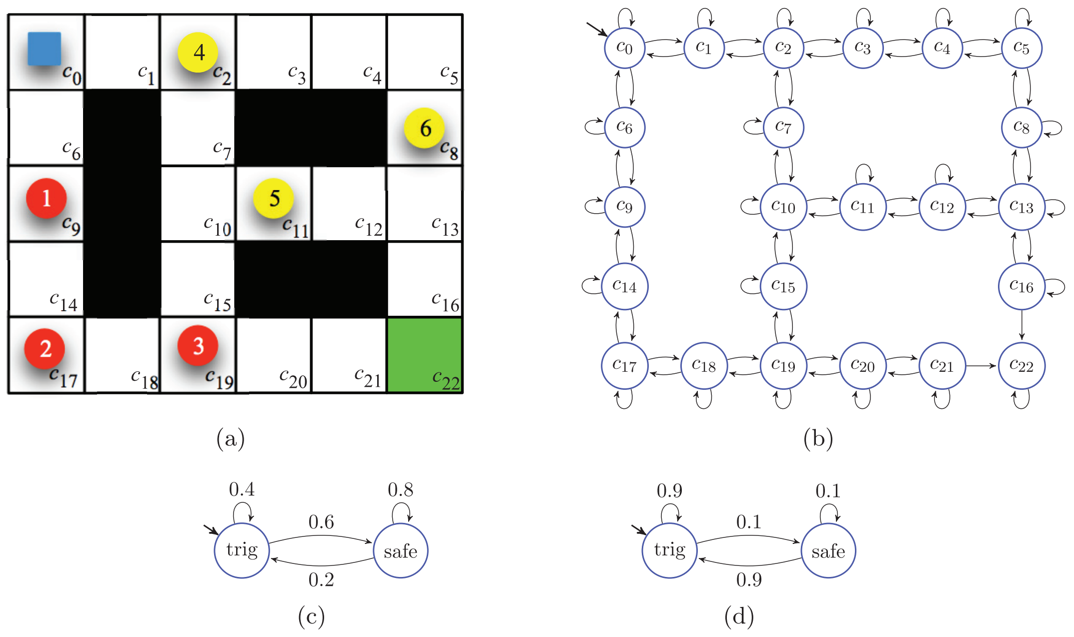
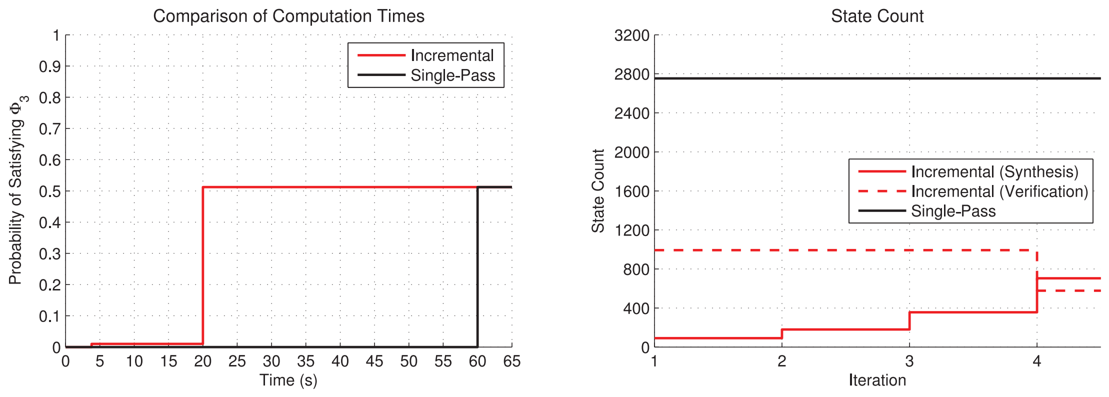

$c_0$ $c_1$ $c_2$ $c_3$ $c_4$ $c_5$ $c_6$ $c_7$ $c_8$ $c_9$ $c_{10}$ $c_{11}$ $c_{12}$ $c_{13}$ $c_{14}$ $c_{15}$ $c_{16}$ $c_{17}$ $c_{18}$ $c_{19}$ $c_{20}$ $c_{21}$ $c_{22}$
1
2
3
4
5
6
(a)
(b)
0.4
0.6
0.8
0.2
trig
safe
(c)
0.9
0.1
0.1
0.9
trig
safe
(d)

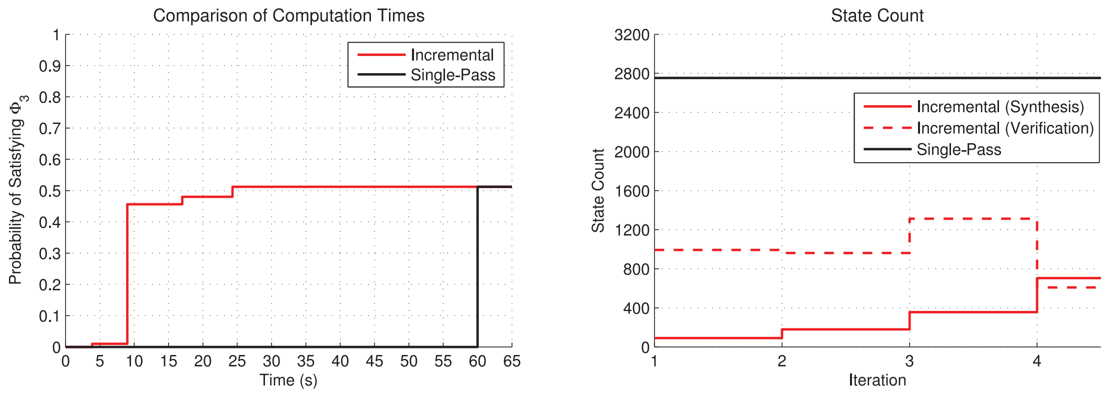
**Fig. 7.** (a) A partitioned room where the robot (the blue square at $c_0$) is required to safely reach cell $c_{22}$ (highlighted in green) by avoiding the traps (the red and yellow circles) when they are in the triggered state. (b) The transition system **T** that models the motion of the robot in the room. (c) and (d) The Markov models of red ($\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3$) and yellow traps ($\mathbf{M}_4, \mathbf{M}_5, \mathbf{M}_6$), respectively.

the MRP problem was solved in the synthesis stage of our approach had 704 states and 33,280 transitions. The largest product MC that was considered in the verification stage of our approach had 993 states and 63,552 transitions. The probabilities of satisfying $\phi$ under policy $\mu_i$ obtained at each iteration of our algorithm were $\Pr^{\mu_1}(\phi) = 0.01$, $\Pr^{\mu_2}(\phi) = 0.01$, $\Pr^{\mu_3}(\phi) = 0.01$, and $\Pr^{\mu_4}(\phi) = 0.512$. Figure 8(a) compares the classical single-pass approach with our incremental algorithm in terms of running time and state counts of the product MDPs and MCs. After simulating the optimal control policy returned by our approach $10^6$ times, the ratio of the number of satisfying trajectories to the total number of trajectories was found to be 0.5119. Multimedia Extension 1 shows a satisfying trajectory and two violating trajectories resulting from the execution of this control policy by the robot.

We must also note that the order in which our algorithm considers the agents can affect the running time, resource usage, and intermediate outputs of our algorithm. When our algorithm considered the agents in the order $\mathbf{M}_3, \mathbf{M}_4, \mathbf{M}_1, \mathbf{M}_2$, it again terminated after the fourth iteration returning an optimal control policy with 0.512 probability of satisfying $\phi_3$. Total computation took 24.32 s (slightly longer than the previous case), the largest product



(a)



(b)

**Fig. 8.** Comparison of the classical single-pass and proposed incremental algorithms for case study 3. (a) Incremental approach adds agents to $\mathcal{M}_i$ in the order $\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3, \mathbf{M}_4$ and (b) incremental approach adds agents to $\mathcal{M}_i$ in the order $\mathbf{M}_3, \mathbf{M}_4, \mathbf{M}_1, \mathbf{M}_2$. The left plots show the running times of the algorithms and the probabilities of satisfying $\phi_3$ under synthesized policies. The right plots compare the state counts of the product MDPs for which synthesis was performed in both approaches (black and red lines) and show the state count of the product MC considered in the verification stage of our incremental algorithm (red dashed line).

MDP on which the MRP problem was solved had 704 states and 33,152 transitions, and the largest product MC used in the verification stage had 1313 states and 79,936 transitions. This time, however, the probabilities of satisfying $\phi$ under policy $\mu_i$ obtained at each iteration were $\Pr^{\mu_1}(\phi) = 0.01$, $\Pr^{\mu_2}(\phi) = 0.456$, $\Pr^{\mu_3}(\phi) = 0.480$, $\Pr^{\mu_4}(\phi) = 0.512$, and our algorithm was able to compute a relatively good control policy in as little as 9 s. Figure 8(b) shows the results for this case. Even though our approach clearly outperforms the classical approach in resource usage, these results suggest that its performance can be improved even further if the next agent to be included in the synthesis stage is selected according to some metric or heuristic tuned to the specific problem at hand.

## 6. Conclusions

We present a highly efficient incremental method for automatically synthesizing optimal control policies for a system comprising a robot and multiple independent agents. The robot, which can be a deterministic transition system or a Markov decision process, is expected to satisfy a high-level mission specification in the presence of the agents which are modeled as Markov chains. For mission specifications, we consider scLTL formulas over a set of propositions that are satisfied by the robot and the agents. If a probability threshold is given, our method terminates as soon as a control policy with probability of satisfaction greater than or equal to this threshold is found. Otherwise, our method synthesizes an optimal control policy that maximizes the probability of satisfying the mission. Since our method does not need to run to completion, it has practical value in applications where a control policy must be synthesized under resource constraints.

## Funding

This work was supported in part by the Office of Naval Research (grant numbers MURI N00014-09-1051, MURI N00014-10-10952) and by the National Science Foundation (grant numbers CNS-0834260, CNS-1035588).

## Note

1. LTL Optimal Multi-Agent Planner (LOMAP) Python Package is available at http://hyness.bu.edu/lomap/.

## References

Alfaro L (1997) *Formal verification of probabilistic systems*. PhD dissertation, Stanford University.

Baier C and Katoen JP (2008) *Principles of Model Checking*. Cambridge, MA: MIT Press.

Berwanger D, Chatterjee K, Wulf MD, et al. (2010) Strategy construction for parity games with imperfect information. *Information Computing* 208: 1206–1220.

Bhatia A, Kavraki LE and Vardi MY (2010) Motion planning with hybrid dynamics and temporal goals. In: *Proceedings of IEEE conference on decision and control (CDC 2010)*, Atlanta, GA, 15–17 December 2010, pp. 1108–1115.

Bianco A and De Alfaro L (1995) Model checking of probabilistic and nondeterministic systems. In: *Foundations of Software Technology and Theoretical Computer Science* (Lecture Notes in Computer Science). Berlin; Heidelberg: Springer, pp.499–513.

Chen Y, Ding XC and Belta C (2011) Synthesis of distributed control and communication schemes from global LTL specifications. In: *Proceedings of IEEE conference on decision and control (CDC 2011)*. Orlando, FL, 12–15 December 2011, pp. 2718–2723.

Clarke EM, Peled D and Grumberg O (1999) *Model Checking*. Cambridge, MA: MIT Press.

Ding XC, Smith SL, Belta C, et al. (2011) MDP optimal control under temporal logic constraints. In: *Proceedings of IEEE conference on decision and control (CDC 2011)*, Orlando, FL, 12–15 December 2011, pp. 532–538.

Emerson EA (1990) Temporal and Modal Logic. In: Van Leeuwen J (ed) *Handbook of Theoretical Computer Science: Formal Models and Semantics*. Amsterdam; New York; MIT Press: North-Holland Publishing Co./MIT Press, Vol. B, pp. 995–1072.

Gulwani S, Jha S, Tiwari A, et al. (2011) Synthesis of loop-free programs. In: *Proceedings of 32nd ACM SIGPLAN conference on programming language design and implementation (PLDI 2011)*, San Jose, CA, 4–8 June 2011, pp. 62–73.

Hagberg AA, Schult DA and Swart PJ (2008) Exploring network structure, dynamics, and function using NetworkX. In: *Proceedings of the 7th Python in science conference (SciPy 2008)*, Pasadena, CA, 19–24 August 2008, pp. 11–15.

Hopcroft JE, Motwani R and Ullman JD (2007) *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison Wesley.

Jha S, Gulwani S, Seshia S, et al. (2010) Synthesizing switching logic for safety and dwell-time requirements. In: *Proceedings of international conference on cyber-physical systems*, Stockholm, Sweden, 12–14 April 2010, pp. 22–31.

Johnson B and Kress-Gazit H (2012) Probabilistic guarantees for high-level robot behavior in the presence of sensor error. *Autonomous Robots* 33(3): 309–321.

Kloetzer M and Belta C (2008) Dealing with non-determinism in symbolic control. In: Egerstedt M and Mishra B (eds) *Hybrid Systems: Computation and Control: 11th International Workshop* (Lecture Notes in Computer Science). Berlin; Heidelberg: Springer, pp. 287–300.

Kloetzer M and Belta C (2010) Automatic deployment of distributed teams of robots from temporal logic specifications. *IEEE Transactions on Robotics* 26(1): 48–61.

Kress-Gazit H, Fainekos GE and Pappas GJ (2007) Where's Waldo? sensor-based temporal logic motion planning. In: *Proceedings of IEEE International conference on robotics and automation (ICRA 2007)*, Rome, 10-14 April 2007, pp. 3116–3121.

Kupferman O and Vardi MY (2001) Model checking of safety properties. *Formal Methods in System Design* 19: 291–314.

Kwiatkowska M, Norman G and Parker D (2002) Probabilistic symbolic model checking with PRISM: A hybrid approach. In: *Proceedings of 8th International conference TACAS 2002 held as part of the joint European conferences on theory and practice of software (ETAPS 2002)*, Grenoble, France, 8–12 April 2002, pp. 52–66.

Lahijanian M, Andersson SB and Belta C (2012) Temporal logic motion planning and control with probabilistic satisfaction guarantees. *IEEE Transactions on Robotics* 28(2): 396–409.

Latvala T (2003) Efficient model checking of safety properties. In: *Proceedings of 10th International SPIN workshop on model checking software*, Portland, OR, 9–10 May 2002, pp. 74–88.

Ozay N, Topcu U, Wongpiromsarn T and Murray RM (2011) Distributed synthesis of control protocols for smart camera networks. In: *Proceedings of IEEE/ACM International conference on cyber-physical systems (ICCPS 2011)*, Chicago, IL, 12–14 April 2011, pp. 45–54.

Tabuada P and Pappas GJ (2006) Linear time logic control of discrete-time linear systems. *IEEE Transactions on Automatic Control* 51(12): 1862–1877.

Thomas W (2002) Infinite games and verification. In: *Proceedings of conference on computer-aided verification (CAV)*, Copenhagen, Denmark, 27–31 July 2002, pp. 58–64.

Tůmová J, Yordanov B, Belta C, et al. (2010) A symbolic approach to controlling piecewise affine systems. In: *Proceedings of IEEE conference on decision and control (CDC 2010)*, Atlanta, GA, 15–17 December 2010, pp. 4230–4235.

Ulusoy A, Wongpiromsarn T and Belta C (2012) Incremental control synthesis in probabilistic environments with temporal logic constraints. In: *Proceedings of IEEE conference on decision and control (CDC 2012)*, Maui, HI, 10–13 December 2012, pp. 7658–7663.

Wongpiromsarn T, Topcu U and Murray RM (2010) Receding horizon control for temporal logic specifications. In: *Proceedings of Hybrid systems: Computation and control*, Stockholm, Sweden, 12–16 April 2010, pp. 101–110.

Wongpiromsarn T, Ulusoy A, Belta C, et al. (2012) Incremental temporal logic synthesis of control policies for robots interacting with dynamic agents. In: *Proceedings of IEEE/RSJ International conference on intelligent robots and systems (IROS 2012)*, Vilamoura, Portugal, 7–12 October 2012, pp. 229–236.

Wongpiromsarn T, Ulusoy A, Belta C, et al. (2013) Incremental synthesis of control policies for heterogeneous multi-agent systems with linear temporal logic specifications. In: *Proceedings of IEEE International conference on robotics and automation*, Karlsruhe, Germany, 6–10 May 2013, pp. 5011–5018.

## Appendix: Index to Multimedia Extension

The multimedia extension page is found at http://www.ijrr.org

**Table of Multimedia Extension**

| Extension | Media type | Description |
|---|---|---|
| 1 | Video | Execution of the control policy in case study 3. |