51st IEEE Conference on Decision and Control
December 10-13, 2012. Maui, Hawaii, USA

Incremental Control Synthesis in Probabilistic Environments with Temporal Logic Constraints

Alphan Ulusoy* Tichakorn Wongpiromsarn† Calin Belta*

Abstract— **In this paper, we present a method for optimal control synthesis of a plant that interacts with a set of agents in a graph-like environment. The control specification is given as a temporal logic statement about some properties that hold at the vertices of the environment. The plant is assumed to be deterministic, while the agents are probabilistic Markov models. The goal is to control the plant such that the probability of satisfying a syntactically co-safe Linear Temporal Logic formula is maximized. We propose a computationally efficient incremental approach based on the fact that temporal logic verification is computationally cheaper than synthesis. We present a case-study where we compare our approach to the classical non-incremental approach in terms of computation time and memory usage.**

I. Introduction

Temporal logics [1], such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL), are traditionally used for verification of non-deterministic and probabilistic systems [2]. Even though temporal logics are suitable for specifying complex missions for control systems, they did not gain popularity in the control community until recently [3], [4], [5].

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 [3], [6], have finitely many states. With few exceptions [7], 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 [4]. If the system is non-deterministic, the control problem can be mapped to the solution of a Rabin game [8], [6], or a simpler Büchi [9] or GR(1) game [10], if the specification is restricted to fragments of LTL. For probabilistic systems, the LTL control synthesis problem reduces to computing a control policy for a Markov Decision Process (MDP) [11], [12], [13].

In this work, we consider mission specifications expressed as syntactically co-safe LTL formulas [14]. We focus on a particular type of a multi-agent system formed by a deterministically controlled plant and a set of independent, probabilistic, uncontrollable agents, operating on a common, graph-like environment. An illustrative example is a car (plant) approaching a pedestrian crossing, while there are some pedestrians (agents) waiting to cross or already crossing the road. As the state space of the system grows exponentially with the number of pedestrians, one may not be able to utilize any of the existing approaches under computational resource constraints when there is a large number of pedestrians.

We partially address this problem by proposing an incremental control synthesis method that exploits the independence between the components of the system, *i.e.*, the plant modeled as a deterministic transition system and the agents, modeled as Markov chains, and the fact that verification is computationally cheaper than synthesis. We aim to synthesize a plant control strategy that maximize the probability of satisfying a mission specification given as a syntactically co-safe LTL formula. Our method initially considers a considerably smaller agent subset and synthesizes a control policy that maximizes the probability of satisfying the mission specification for the subsystem formed by the plant and this subset. This control policy is then verified against the remaining agents. At each iteration, we remove 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 plant is a Markov Decision Process, and we consider a deterministic plant only for simplicity of presentation. We experimentally evaluate the performance of our approach and show that our method clearly outperforms existing non-incremental approaches. Various methods that also use verification during incremental synthesis have been previously proposed in [15], [16]. However, 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.

The rest of the paper is organized as follows: In Sec. II, we give necessary definitions and some preliminaries in formal methods. The control synthesis problem is formally stated in Sec. III and the solution is presented in Sec. IV. Experimental results are included in Sec. V. We conclude with final remarks in Sec. VI. Due to page constraints we omit the proofs of all results. The proofs can be found in an extended version, available online [17].

II. Preliminaries

For a set Σ, we use $|\Sigma|$ and 2^{Σ} to denote its cardinality and power set, respectively. A (finite) word ω over a set Σ is a sequence of symbols $\omega = \omega^0 \dots \omega^l$ such that $\omega^i \in \Sigma \; \forall i = 0, \dots, l$.

Definition II.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;* $\mathcal{A}_{\mathrm{T}}$

This work was supported in part by ONR-MURI N00014-09-1051, ONR MURI N00014-10-10952, NSF CNS-0834260 and NSF CNS-1035588.

* Division of Systems Engineering, Boston University, Boston, MA 02215 (alphan@bu.edu, cbelta@bu.edu)

† Singapore-MIT Alliance for Research and Technology, Singapore 117543 (nok@smart.mit.edu)

978-1-4673-2066-5/12/$31.00 ©2012 IEEE

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;* Π_{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 II.2 (Markov Chain). *A (discrete-time, labelled) Markov chain (MC) is a tuple* $\mathbf{M} := (\mathcal{Q}_{\mathrm{M}}, q^0_{\mathrm{M}}, \delta_{\mathrm{M}}, \Pi_{\mathrm{M}}, \mathcal{L}_{\mathrm{M}})$*, where* $\mathcal{Q}_{\mathrm{M}}$*,* Π_{M}*, and* $\mathcal{L}_{\mathrm{M}}$ *are the set of states, the set of atomic propositions, and the satisfaction map, respectively, as in Def. II.1, and* $q^0_{\mathrm{M}} \in \mathcal{Q}_{\mathrm{M}}$ *is the initial state;* $\delta_{\mathrm{M}} : \mathcal{Q}_{\mathrm{M}} \times \mathcal{Q}_{\mathrm{M}} \to [0, 1]$ *is the transition probability function that satisfies* $\sum_{q' \in \mathcal{Q}_{\mathrm{M}}} \delta(q, q') = 1 \; \forall q \in \mathcal{Q}_{\mathrm{M}}$.

In this paper, we are interested in temporal logic missions over a finite time horizon and we use syntactically co-safe LTL formulas [18] to specify them. Informally, a syntactically co-safe LTL formula over the set Π of atomic propositions comprises boolean operators $\neg$ (negation), $\vee$ (disjunction) and $\wedge$ (conjunction), and temporal operators $\mathbf{X}$ (next), $\mathcal{U}$ (until) and $\mathbf{F}$ (eventually). Any syntactically co-safe LTL formula can be written in positive normal form, where the negation operator $\neg$ occurs only in front of atomic propositions. For instance, $\mathbf{X}\,\mathtt{p}$ states that at the next position of the word, proposition $\mathtt{p}$ is true. The formula $\mathtt{p}_1 \,\mathcal{U}\, \mathtt{p}_2$ states that there is a future position of the word when proposition $\mathtt{p}_2$ is true, and proposition $\mathtt{p}_1$ is true at least until $\mathtt{p}_2$ is true. For any syntactically co-safe LTL formula ϕ over a set Π, one can construct a FSA with input alphabet 2^{Π} accepting all and only finite words over 2^{Π} that satisfy ϕ, which is defined next.

Definition II.3 (Finite State Automaton). *A (deterministic) finite state automaton (FSA) is a tuple* $\mathbf{F} := (\mathcal{Q}_{\mathrm{F}}, q^0_{\mathrm{F}}, \Sigma_{\mathrm{F}}, \delta_{\mathrm{F}}, \mathcal{F}_{\mathrm{F}})$*, where* $\mathcal{Q}_{\mathrm{F}}$ *is a finite set of states;* $q^0_{\mathrm{F}} \in \mathcal{Q}_{\mathrm{F}}$ *is the initial state;* Σ_{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 $q^0 = q^0_{\mathrm{F}}$. An FSA $\mathbf{F}$ accepts a word over Σ_{F} if and only the corresponding run ends in some $q \in \mathcal{F}_{\mathrm{F}}$.

Definition II.4 (Markov Decision Process). *A Markov decision process (MDP) is a tuple* $\mathbf{P} := (\mathcal{Q}_{\mathrm{P}}, q^0_{\mathrm{P}}, \mathcal{A}_{\mathrm{P}}, \alpha_{\mathrm{P}}, \delta_{\mathrm{P}}, \Pi_{\mathrm{P}}, \mathcal{L}_{\mathrm{P}})$*, where* $\mathcal{Q}_{\mathrm{P}}$ *is a finite set of states;* $q^0_{\mathrm{P}} \in \mathcal{Q}_{\mathrm{P}}$ *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* $\sum_{q' \in \mathcal{Q}_{\mathrm{P}}} \delta(q, a, q') = 1 \; \forall q \in \mathcal{Q}_{\mathrm{P}}, a \in \alpha_{\mathrm{P}}(q)$ *and* $\sum_{q' \in \mathcal{Q}_{\mathrm{P}}} \delta(q, a, q') = 0 \; \forall q \in \mathcal{Q}_{\mathrm{P}}, a \notin \alpha_{\mathrm{P}}(q).\Pi_{\mathrm{P}}$ *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 in 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 $\mu(q)$ at each $q \in \mathcal{Q}_{\mathrm{P}}$. A path of $\mathbf{P}$ under policy μ_{P} is a finite sequence of states $r^{\mu_{\mathrm{P}}}_{\mathrm{P}} = q^0 q^1 \ldots q^l$ such that $l \geq 0$, $q^0 = q^0_{\mathrm{P}}$ and $\delta_{\mathrm{P}}(q^{k-1}, \mu_{\mathrm{P}}(q^{k-1}), q^k) > 0 \; \forall k \in [1, l]$. A path $r^{\mu_{\mathrm{P}}}_{\mathrm{P}}$ generates a finite word $\mathcal{L}_{\mathrm{P}}(r^{\mu_{\mathrm{P}}}_{\mathrm{P}}) = \mathcal{L}_{\mathrm{P}}(q^0)\mathcal{L}_{\mathrm{P}}(q^1) \ldots \mathcal{L}_{\mathrm{P}}(q^l)$ where $\mathcal{L}_{\mathrm{P}}(q^k)$ is the set of atomic propositions satisfied at state q^k. Next, we use $Paths^{\mu_{\mathrm{P}}}_{\mathrm{P}}$ to denote the set of all paths of $\mathbf{P}$ under a policy μ_{P}. Finally, we define $P^{\mu_{\mathrm{P}}}(\phi)$ as the probability of satisfying ϕ under policy μ_{P}.

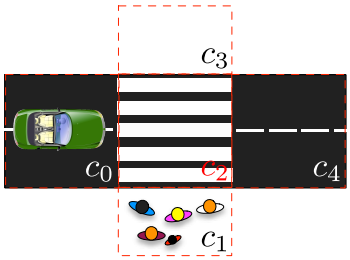

Fig. 1: A partitioned road environment, where a car (plant) is required to reach c_4 without colliding with any of the pedestrians (agents).

Remark II.5. *Syntactically co-safe LTL formulas have infinite time semantics, thus they are actually interpreted over infinite words [18]. Measurability of languages satisfying LTL formulas is also defined for infinite words generated by infinite paths [2]. However, one can determine whether a given infinite word satisfies a syntactically co-safe LTL formula by considering only a finite prefix of it. It can be easily shown that our above definition of* $Paths^{\mu_{\mathrm{P}}}_{\mathrm{P}}$ *inherits the same measurability property given in [2].*

III. Problem Formulation and Approach

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

A. System Model

Consider a system consisting of a deterministic plant that we can control (*e.g.*, a robot) 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 Figs. 1, 2). Agent i is modeled as an MC $\mathbf{M}_i = (\mathcal{Q}_i, q^0_i, \delta_i, \Pi_i, \mathcal{L}_i)$, with $\mathcal{Q}_i \subseteq V$ and $\delta_i \subseteq \rightarrow_{\mathcal{E}}$, $i = 1, \ldots, n$. The plant is assumed to be a deterministic transition system TS $\mathbf{T} = (\mathcal{Q}_{\mathrm{T}}, q^0_{\mathrm{T}}, \mathcal{A}_{\mathrm{T}}, \alpha_{\mathrm{T}}, \delta_{\mathrm{T}}, \Pi_{\mathrm{T}}, \mathcal{L}_{\mathrm{T}})$, where $\mathcal{Q}_{\mathrm{T}} \subseteq V$ and $\delta_t \subseteq \rightarrow_{\mathcal{E}}$. We assume that all components of the system (the plant 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 plant 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. 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 plant, 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 Π of propositions as $\Pi = \Pi_{\mathrm{T}} \cup \Pi_i \cup \ldots \cup \Pi_n \subseteq \{(i, p) | i = \{\mathrm{T}, 0, \ldots, n\}, p \in \Pi_{\mathcal{E}}\}$.

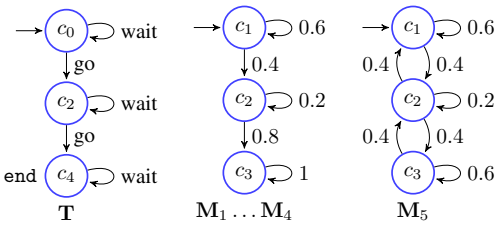

Fig. 2: TS $\mathbf{T}$ and MCs $\mathbf{M}_1 \dots \mathbf{M}_5$ that model the car and the pedestrians.

B. Problem Formulation

As it will become clear in Sec. IV-D, the joint behavior of the plant 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 Def. II.4). Given a syntactically co-safe LTL formula ϕ over Π, 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 ϕ is either maximized or above a given threshold. Since we assume perfect state information, the plant 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 plant and a control policy for the system, and we will refer to it simply as *control policy*. We can now formulate the main problem considered in this paper:

Problem III.1. *Given a system described by a plant $\mathbf{T}$ and a set of agents $\mathbf{M}_1, \dots, \mathbf{M}_n$ operating on a graph $\mathcal{E}$, and given a specification in the form of a syntactically co-safe LTL formula ϕ over Π, synthesize a control policy $\mu^\star$ that satisfies the following objective: (a) If a probability threshold p_{thr} is given, the probability that the system satisfies ϕ under $\mu^\star$ exceeds p_{thr}. (b) Otherwise, $\mu^\star$ maximizes the probability that the system satisfies ϕ. If no such policy exists, report failure.*

As will be shown in Sec. IV-A, 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 plant is a Markov Decision Process. We consider a deterministic plant only for simplicity of presentation.

Example III.2. *Fig. 1 illustrates a car in a 5-cell environment with 5 pedestrians, where $\mathcal{L}_\mathcal{E}(v) = v$ for $v \in \{c_0, \dots, c_4\}$. Fig. 2 illustrates the TS $\mathbf{T}$ and the MCs $\mathbf{M}_1, \dots, \mathbf{M}_5$ that model the car and the pedestrians. The car is required to reach the end of the crossing (c_4) without colliding with any of the pedestrians. To enforce this behavior, we write our specification as*

$$\phi := \left(\neg \bigvee_{i=1\dots5, j=0\dots4} ((\mathrm{T}, c_j) \wedge (i, c_j)) \right) \mathcal{U} (\mathrm{T}, c_4). \quad (1)$$

The deterministic FSA that corresponds to ϕ is given in Fig. 3, where $\texttt{col} = \bigvee_{i=1\dots5, j=0\dots4}((\mathrm{T}, c_j) \wedge (i, c_j))$ and $\texttt{end} = (\mathrm{T}, c_4)$.

C. Solution Outline

One can directly solve Prob. III.1 by reducing it to a Maximal Reachability Probability (MRP) problem on the MDP modeling the overall system [19]. This approach, however, is very resource demanding as it scales exponentially with the number 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 method that exploits the independence between the system components and the fact that verification is less demanding than synthesis. At each iteration i, our method will involve the following steps: synthesis of an optimal control policy considering only some of the agents (Sec. IV-D), verification of this control policy with respect to the complete system (Sec. IV-E) and minimization of the system model under the guidance of this policy (Sec. IV-F).

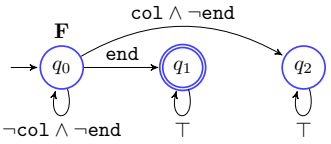

Fig. 3: Deterministic FSA $\mathbf{F}$ that corresponds to $\phi = \neg\texttt{col}\,\mathcal{U}\,\texttt{end}$. q_0 and q_1 are initial and final states, respectively.

IV. Problem Solution

Our solution to Prob. III.1 is given in the form of Alg. 1. In the rest of this section, we explain each of its steps in detail.

A. 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 $\mathbf{T}$ and agents in $\mathcal{M}_i = \{\mathbf{M}_{i1}, \dots, \mathbf{M}_{ij}\}$ for different types of $\mathbf{T}$ as follows.

If $\mathbf{T}$ 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 $\mathcal{Q}_\mathrm{A} \subseteq \mathcal{Q}_\mathrm{T} \times \mathcal{Q}_{i1} \times \dots \times \mathcal{Q}_{ij}$ 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_T is the element of q that corresponds to the state of $\mathbf{T}$; $\Pi_\mathrm{A} = \Pi_\mathrm{T} \cup \Pi_{i1} \cup \dots \cup \Pi_{ij}$; $\mathcal{L}_\mathrm{A}(q) = \mathcal{L}_\mathrm{T}(q_\mathrm{T}) \cup \mathcal{L}_{i1}(q_{i1}) \cup \dots \cup \mathcal{L}_{ij}(q_{ij})$; $\delta_\mathrm{A}(q = (q_\mathrm{T}, q_{i1}, \dots, q_{ij}), a, q' = (q'_\mathrm{T}, q'_{i1}, \dots, q'_{ij})) = 1\{(q_\mathrm{T}, a, q'_\mathrm{T}) \in \delta_T\} \times \delta(q_{i1}, q'_{i1}) \times \dots \times \delta(q_{ij}, q'_{ij})$, 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 $\mathbf{A} = (\mathcal{Q}_\mathrm{A}, q^0_\mathrm{A}, \mathcal{A}_\mathrm{A}, \alpha_\mathrm{A}, \delta_\mathrm{A}, \Pi_\mathrm{A}, \mathcal{L}_\mathrm{A}) = \mathbf{T} \otimes \mathcal{M}_i$, such that $\mathcal{Q}_\mathrm{A}$, q^0_A, $\mathcal{A}_\mathrm{A}$, α_A, Π_A, and $\mathcal{L}_\mathrm{A}$ are as given in the case where $\mathbf{T}$ is a TS and $\delta_\mathrm{A}(q = (q_\mathrm{T}, q_{i1}, \dots, q_{ij}), a, q' = (q'_\mathrm{T}, q'_{i1}, \dots, q'_{ij})) = \delta_\mathrm{T}(q_\mathrm{T}, a, q'_\mathrm{T}) \times \delta_{i1}(q_{i1}, q'_{i1}) \times \dots \times \delta_{ij}(q_{ij}, q'_{ij})$.

Finally if $\mathbf{T}$ is an MC, then we define $\mathbf{T} \otimes \mathcal{M}_i$ as the MC $\mathbf{A} = (\mathcal{Q}_\mathrm{A}, q^0_\mathrm{A}, \delta_\mathrm{A}, \Pi_\mathrm{A}, \mathcal{L}_\mathrm{A}) = \mathbf{T} \otimes \mathcal{M}_i$ where $\mathcal{Q}_\mathrm{A}$, q^0_A, Π_A, $\mathcal{L}_\mathrm{A}$ are as given in the case where $\mathbf{T}$ is a TS and $\delta_\mathrm{A}(q = (q_\mathrm{T}, q_{i1}, \dots, q_{ij}), q' = (q'_\mathrm{T}, q'_{i1}, \dots, q'_{ij})) = \delta_\mathrm{T}(q_\mathrm{T}, q'_\mathrm{T}) \times \delta_{i1}(q_{i1}, q'_{i1}) \times \dots \times \delta_{ij}(q_{ij}, q'_{ij})$.

B. Product MDP and Product MC

Given the deterministic FSA $\mathbf{F}$ that recognizes all and only the finite words that satisfy ϕ, 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 $\mathbf{P} = (\mathcal{Q}_\mathrm{P}, q^0_\mathrm{P}, \mathcal{A}_\mathrm{P}, \alpha_\mathrm{P}, \delta_\mathrm{P}, \Pi_\mathrm{P}, \mathcal{L}_\mathrm{P}) = \mathbf{M} \otimes \mathbf{F}$, 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

Algorithm 1: INCREMENTAL-CONTROL-SYNTHESIS

Input: $\mathbf{T}, \mathbf{M}_1, \ldots, \mathbf{M}_n, \phi, (p_{thr})$.
Output: $\mu^\star$ s.t. $P^{\mu^\star}(\phi) \geq P^{\mu}(\phi)\, \forall \mu$ if p_{thr} is not given, otherwise $P^{\mu^\star}(\phi) > p_{thr}$.

1 $\mathcal{M} \leftarrow \{\mathbf{M}_1, \ldots, \mathbf{M}_n\}$.
2 Construct FSA $\mathbf{F}$ corresponding to ϕ.
3 $\mu^\star \leftarrow \emptyset$, $P^{\mu^\star}_{\mathcal{M}}(\phi) \leftarrow 0$, $\mathcal{M}_0 \leftarrow \emptyset$, $\mathbf{A}_0 \leftarrow \mathbf{T}$, $i \leftarrow 1$.
4 Process ϕ to form $\mathcal{M}_i^{new}$.
5 **while** *True* **do**
6 $\quad \mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \cup \mathcal{M}_i^{new}$.
7 $\quad \mathbf{A}_i \leftarrow \mathbf{A}_{i-1} \otimes \mathcal{M}_i^{new}$.
8 $\quad \mathbf{P}_i \leftarrow \mathbf{A}_i \otimes \mathbf{F}$.
9 $\quad$ Synthesize μ_i that maximizes $P^{\mu_i}_{\mathcal{M}_i}(\phi)$ using $\mathbf{P}_i$.
10 $\quad$ **if** p_{thr} *given* **then**
11 $\quad\quad$ **if** $P^{\mu_i}_{\mathcal{M}_i}(\phi) < p_{thr}$ **then**
12 $\quad\quad\quad$ Fail: $\nexists \mu$ such that $P^{\mu}(\phi) \geq p_{thr}$.
13 $\quad\quad$ **else if** $\mathcal{M}_i = \mathcal{M}$ **then**
14 $\quad\quad\quad$ Success: $\mu^\star \leftarrow \mu_i$, Return $\mu^\star$.
15 $\quad\quad$ **else**
16 $\quad\quad\quad$ Continue with verification on line 20.
17 $\quad$ **else if** $\mathcal{M}_i = \mathcal{M}$ **then**
18 $\quad\quad$ Success: $\mu^\star \leftarrow \mu_i$, Return $\mu^\star$.
19 $\quad$ **else**
20 $\quad\quad$ Obtain the MC $\mathbf{M}^{\mu_i}_{\mathcal{M}_i}$ induced on $\mathbf{P}_i$ by μ_i.
21 $\quad\quad \overline{\mathcal{M}_i} \leftarrow \mathcal{M} \setminus \mathcal{M}_i$.
22 $\quad\quad \mathbf{M}^{\mu_i}_{\mathcal{M}} \leftarrow \mathbf{M}^{\mu_i}_{\mathcal{M}_i} \otimes \overline{\mathcal{M}_i}$.
23 $\quad\quad \mathbf{V}_i \leftarrow \mathbf{M}^{\mu_i}_{\mathcal{M}} \otimes \mathbf{F}$.
24 $\quad\quad$ Compute $P^{\mu_i}_{\mathcal{M}}(\phi)$ using $\mathbf{V}_i$.
25 $\quad\quad$ **if** $P^{\mu_i}_{\mathcal{M}}(\phi) > P^{\mu^\star}_{\mathcal{M}}(\phi)$ **then**
26 $\quad\quad\quad \mu^\star \leftarrow \mu_i$, $P^{\mu^\star}_{\mathcal{M}}(\phi) \leftarrow P^{\mu_i}_{\mathcal{M}}(\phi)$.
27 $\quad\quad$ **if** p_{thr} *given and* $P^{\mu^\star}_{\mathcal{M}}(\phi) > p_{thr}$ **then**
28 $\quad\quad\quad$ Success: Return $\mu^\star$.
29 $\quad\quad$ **else**
30 $\quad\quad\quad$ Set $\mathcal{M}_{i+1}^{new}$ to some agent $\mathbf{M}_j \in \overline{\mathcal{M}_i}$.
31 $\quad\quad\quad$ Minimize $\mathbf{A}_i$.
32 $\quad\quad\quad$ Increment i.

initial states; $q^0_{\mathrm{P}} = (q^0_{\mathrm{M}}, q_{\mathrm{F}})$ such that $(q^0_{\mathrm{F}}, \mathcal{L}_{\mathrm{M}}(q_{\mathrm{M}}), q_{\mathrm{F}}) \in \delta_{\mathrm{F}}$; $\mathcal{A}_{\mathrm{P}} = \mathcal{A}_{\mathrm{M}}$; $\alpha_{\mathrm{P}}((q_{\mathrm{M}}, q_{\mathrm{F}})) = \alpha_{\mathrm{M}}(q_{\mathrm{M}})$; $\Pi_{\mathrm{P}} = \Pi_{\mathrm{M}}$; $\mathcal{L}_{\mathrm{P}}((q_{\mathrm{M}}, q_{\mathrm{F}})) = \mathcal{L}_{\mathrm{M}}(q_{\mathrm{M}})$; $\delta_{\mathrm{P}}((q_{\mathrm{M}}, q_{\mathrm{F}}), a, (q'_{\mathrm{M}}, q'_{\mathrm{F}})) = 1\{(q_{\mathrm{F}}, \mathcal{L}_{\mathrm{M}}(q'_{\mathrm{M}}), q'_{\mathrm{F}}) \in \delta_{\mathrm{F}}\} \times \delta_{\mathrm{M}}(q_{\mathrm{M}}, a, q'_{\mathrm{M}})$, 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 $\mathbf{P} = (\mathcal{Q}_{\mathrm{P}}, q^0_{\mathrm{P}}, \delta_{\mathrm{P}}, \Pi_{\mathrm{P}}, \mathcal{L}_{\mathrm{P}}) = \mathbf{M} \otimes \mathbf{F}$ where $\mathcal{Q}_{\mathrm{P}}$, q^0_{P}, Π_{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.

C. Initialization

Lines 1 to 4 of Alg. 1 correspond to the initialization procedure of our algorithm. First, we form the set $\mathcal{M} = \{\mathbf{M}_1, \ldots, \mathbf{M}_n\}$ of all agents and construct the FSA $\mathbf{F}$ that corresponds to ϕ. Such $\mathbf{F}$ can be automatically constructed using existing tools, *e.g.*, [20]. Since we have not synthesized any control policies so far, we reset the variable $\mu^\star$ that holds the best policy at any given iteration and set the probability $P^{\mu^\star}_{\mathcal{M}}(\phi)$ of satisfying ϕ under policy $\mu^\star$ in the presence of agents in $\mathcal{M}$ to 0. As we have not considered any agents so far, we set the subset $\mathcal{M}_0$ to be an empty set. We then set $\mathbf{A}_0$, which stands for the parallel composition of the plant $\mathbf{T}$ and the agents in $\mathcal{M}_0$, to $\mathbf{T}$. We also initialize the iteration counter i to 1.

Line 4 of Alg. 1 initializes the set $\mathcal{M}_1^{new}$ of agents that will be considered in the synthesis step of the first iteration of our algorithm. In order to be able to guarantee completeness, we require this set to be the maximal set of agents that satisfy the mission, *i.e.*, the agent subset that can satisfy ϕ but not strictly needed to satisfy ϕ. To form $\mathcal{M}_1^{new}$, we first rewrite ϕ in positive normal form to obtain ϕ_{pnf}, where the negation operator $\neg$ occurs only in front of atomic propositions. Conversion of ϕ to ϕ_{pnf} can be performed automatically using De Morgan's laws and equivalences for temporal operators as given in [2]. Then, using this fact, we include an agent $\mathbf{M}_i \in \mathcal{M}$ in $\mathcal{M}_1^{new}$ if any of its corresponding propositions of the form $(i, p), p \in \Pi_i$ appears non-negated in ϕ_{pnf}. For instance, given $\phi := \neg((3, \mathtt{p3}) \wedge (\mathrm{T}, \mathtt{p3}))\, \mathcal{U}\, ((1, \mathtt{p1}) \vee (2, \mathtt{p2}))$, either one of agents $\mathbf{M}_1$ and $\mathbf{M}_2$ can satisfy the formula, whereas agent $\mathbf{M}_3$ can only violate it. Therefore, for this example we set $\mathcal{M}_1^{new} = \{\mathbf{M}_1, \mathbf{M}_2\}$. In case $\mathcal{M}_1^{new} = \emptyset$ after this procedure, we form $\mathcal{M}_1^{new}$ arbitrarily by including some agents from $\mathcal{M}$ and proceed with the synthesis step of our approach.

D. Synthesis

Lines 6 to 19 of Alg. 1 correspond to the synthesis step of our algorithm. At the i^{th} iteration, the agent subset that we consider is given by $\mathcal{M}_i = \mathcal{M}_{i-1} \cup \mathcal{M}_i^{new}$ where $\mathcal{M}_i^{new}$ contains the agents that will be newly considered as provided by the previous iteration's verification stage or by the initialization procedure given in Sec. IV-C if i is 1. First, we construct the parallel composition $\mathbf{A}_i = \mathbf{A}_{i-1} \otimes \mathcal{M}_i^{new}$ of our plant and the agents in $\mathcal{M}_i$ as described in Sec. IV-A. Notice that, we use $\mathbf{A}_{i-1}$ to save from computation time and memory as $\mathbf{A}_{i-1} \otimes \mathcal{M}_i^{new}$ is typically smaller than $\mathbf{T} \otimes \mathcal{M}_i$ due to the minimization procedure explained in Sec. IV-F. Next, we construct the product MDP $\mathbf{P}_i = \mathbf{A}_i \otimes \mathbf{F}$ as explained in Sec. IV-B. 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 $\mathcal{F}_{\mathrm{P}}$ from the initial state q^0_{P} [19], after which the corresponding optimal control policy μ_i can be recovered as given in [2], [13]. Consequently, at line 9 of Alg. 1 we solve the MRP problem on $\mathbf{P}_i$ using value iteration to obtain optimal policy μ_i that maximizes the probability of satisfaction of ϕ in the presence of the agents in $\mathcal{M}_i$. We denote this probability by $P^{\mu_i}_{\mathcal{M}_i}(\phi)$, whereas $P^{\mu}(\phi)$ stands for the probability that the complete system satisfies ϕ under policy μ.

The steps that we take at the end of the synthesis, *i.e.*, lines 10 to 19 of Alg. 1, depends on whether p_{thr} is given or not. At any iteration i, if p_{thr} is given and $P^{\mu_i}_{\mathcal{M}_i}(\phi) < p_{thr}$, we terminate by reporting that there exists no control policy $\mu : P^{\mu}(\phi) \geq p_{thr}$ which is a direct consequence of Prop. IV.1. If

p_{thr} is given and $P^{\mu_i}_{\mathcal{M}_i}(\phi) \geq p_{thr}$, we consider the following cases. If $\mathcal{M}_i = \mathcal{M}$, we set $\mu^\star$ to μ_i and return $\mu^\star$ as it satisfies the probability threshold. Otherwise, we proceed with the verification of μ_i as there are remaining agents that were not considered during synthesis and can potentially violate ϕ. For the case where p_{thr} is not given we consider the current agent subset $\mathcal{M}_i$. If $\mathcal{M}_i = \mathcal{M}$ we terminate and return $\mu^\star$ as there are no agents left to consider. Otherwise, we proceed with the verification stage.

Proposition IV.1. *The sequence* $\{P^{\mu_i}_{\mathcal{M}_i}(\phi)\}$ *is non-increasing.*

Corollary IV.2. *If at any iteration* $P^{\mu_i}_{\mathcal{M}_i}(\phi) < p_{thr}$*, then there does not exist a policy* $\mu : P^{\mu}(\phi) \geq p_{thr}$*, where* μ_i *is an optimal control policy that we compute at the synthesis stage of the* i^{th} *iteration considering only the agents in* $\mathcal{M}_i$*.*

E. Verification and Selection of $\mathcal{M}^{new}_{i+1}$

Lines 20 to 30 of Alg. 1 correspond to the verification stage of our algorithm. In the verification stage, we verify the policy μ_i that we have just synthesized considering the entire system and accordingly update the best policy so far, which we denote by $\mu^\star$.

Note that μ_i maximizes the probability of satisfying ϕ in the presence of agents in $\mathcal{M}_i$ and induces an MC by resolving all non-deterministic choices in $\mathbf{P}_i$. Thus, we first obtain the induced Markov Chain $\mathbf{M}^{\mu_i}_{\mathcal{M}_i}$ that captures the joint behavior of the plant and the agents in $\mathcal{M}_i$ under policy μ_i. Then, we proceed by considering the agents that were not considered during synthesis of μ_i, *i.e.*, 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 systems and construct the MC $\mathbf{M}^{\mu_i}_{\mathcal{M}} = \mathbf{M}^{\mu_i}_{\mathcal{M}_i} \otimes \overline{\mathcal{M}_i}$ in line 22. In lines 23 and 24 of Alg. 1, we construct the product MC $\mathbf{V}_i = \mathbf{M}^{\mu_i}_{\mathcal{M}} \otimes \mathbf{F}$ and compute the probability $P^{\mu_i}_{\mathcal{M}}(\phi)$ of satisfying ϕ in the presence of all agents in $\mathcal{M}$ by computing the probability of reaching $\mathbf{V}_i$'s final states from its initial state using value iteration. Finally, in lines 25 and 26 we update $\mu^\star$ so that $\mu^\star = \mu_i$ if $P^{\mu_i}_{\mathcal{M}}(\phi) > P^{\mu^\star}_{\mathcal{M}}(\phi)$, *i.e.*, 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 Alg. 1 an anytime algorithm, *i.e.*, the algorithm can be terminated as soon as some $\mu^\star$ is obtained.

At the end of the verification stage, if p_{thr} is given and $P^{\mu^\star}_{\mathcal{M}}(\phi) \geq p_{thr}$ we terminate and return $\mu^\star$, as it satisfies the given probability threshold. Otherwise in line 30 of Alg. 1, we 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. Note that, one can also choose to pick the smallest $\mathbf{M}_j$ in terms of state and transition count to minimize the overall computation time, which we call the *smallest agent first (SAF)* rule.

Proposition IV.3. *The sequence* $\{P^{\mu^\star}_{\mathcal{M}}(\phi)\}$ *is a non-decreasing sequence.*

F. Minimization

The minimization stage of our approach (line 31 in Alg. 1) 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. We first set the minimization threshold p_{min} to p_{thr} if given, otherwise we set it to $P^{\mu^\star}_{\mathcal{M}}(\phi)$. Next,

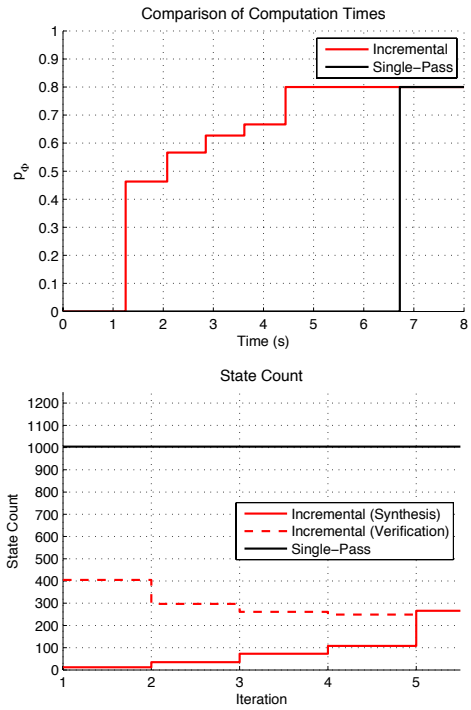

Fig. 4: Comparison of the classical single-pass and proposed incremental algorithms. The top plot shows the running times of the algorithms and the probabilities of satisfying ϕ under synthesized policies. The bottom plot compares the state counts of the product MDPs on which the MRP problem was solved 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).

we iterate over the states of $\mathbf{P}_i$ and check the maximum probability of satisfying the mission under each available action. Note that, the value iteration that we perform in the synthesis step already provides us with the maximum probability of satisfying ϕ from any state in $\mathbf{P}_i$. Then, we remove an action a from state q_{A} in $\mathbf{A}_i$ if for all $q_{\mathrm{F}} \in Q_{\mathrm{F}}$, the maximum probability of satisfying the mission by taking action a at $(q_{\mathrm{A}}, q_{\mathrm{F}})$ in $\mathbf{P}_i$ is below p_{min}. After removing the transitions corresponding to all such actions, we also prune any orphan states in $\mathbf{A}_i$, *i.e.*, states that are not reachable from the initial state. Then, we proceed with the synthesis stage of the next iteration.

Proposition IV.4. *Minimization phase does not affect the correctness and the completeness of our approach.*

We finally show that Alg. 1 correctly solves Prob. III.1.

Proposition IV.5. *Alg. 1 solves Prob. III.1.*

V. Experimental Results

In this section we return to the pedestrian crossing problem given in Example III.2 and illustrated in Figs. 1, 2. The mission specification ϕ for this example is given in Eq. (1). In the following, we compare the performance of our incremental algorithm with the performance of the classical method that attempts to solve this problem in a single pass using value iteration as in [19].

In our experiments we used an iMac i5 quad-core desktop computer and considered C++ implementations of both approaches. During the experiments, our algorithm picked the new agent $\mathcal{M}^{new}_i$ 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 Sec. IV-E.

When no p_{thr} was given, optimal control policies synthesized by both of the algorithms satisfied ϕ with a probability of 0.8. The classical approach solved the control synthesis problem in 6.75 seconds, and the product MDP on which the MRP problem was solved had 1004 states and 26898 transitions. In comparison, our incremental approach solved the same problem in 4.44 seconds, thanks to the minimization 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 405 states and 6125 transitions. The probabilities of satisfying ϕ under policy μ_i obtained at each iteration of our algorithm were $P^{\mu_1}_{\mathcal{M}}(\phi) = 0.463$, $P^{\mu_2}_{\mathcal{M}}(\phi) = 0.566$, $P^{\mu_3}_{\mathcal{M}}(\phi) = 0.627$, $P^{\mu_4}_{\mathcal{M}}(\phi) = 0.667$, and $P^{\mu_5}_{\mathcal{M}}(\phi) = 0.8$. When p_{thr} was given as 0.65, our approach finished in 3.63 seconds and terminated after the fourth iteration returning a sub-optimal control policy with a 0.667 probability of satisfying ϕ. In this case, the largest product MDP on which the MRP problem was solved had only 99 states and 680 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 (1.25 seconds). Fig. 4 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. It must also be noted that, $|\mathcal{M}_1|$, *i.e.*, cardinality of the initial agent subset, is an important factor for the performance of our algorithm. As discussed in this section, for $|\mathcal{M}_1| << |\mathcal{M}|$ our algorithm outperforms the classical method both in terms of running time and memory usage. However, for $|\mathcal{M}_1| \sim |\mathcal{M}|$ we expect the resource usage of our algorithm to be close to that of the classical approach, as in this case almost all of the agents will be considered in the synthesis stage of the first iteration. We plan to address this issue in future work. Nevertheless, most typical finite horizon *safety* missions, where the plant is expected to reach a goal while avoiding a majority or all of the agents, already satisfy the condition $|\mathcal{M}_1| << |\mathcal{M}|$.

VI. Conclusions

In this paper we presented a highly efficient incremental method for automatic synthesis of optimal control policies for a system comprising a plant and multiple independent agents, where the plant is expected to satisfy a co-safe LTL formula in the presence of the agents. We considered independent agents modeled as Markov chains and assumed that the plant was modeled as a deterministic transition system. However, our approach is general enough to accommodate plants modeled as Markov Decision Processes. If a probability threshold is given, our method exploits this knowledge to terminate earlier and returns a sub-optimal control policy. 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 safe control policy must be synthesized under resource constraints. For future work, we plan to extend our approach to mission specifications expressed in full LTL as opposed to a subset of it.

References

[1] E. A. Emerson, "Temporal and modal logic," in *Handbook of Theoretical Computer Science: Formal Models and Semantics*, J. van Leeuwen, Ed. North-Holland Pub. Co./MIT Press, 1990, vol. B, pp. 995–1072.

[2] C. Baier and J.-P. Katoen, *Principles of Model Checking*. The MIT Press, 2008.

[3] P. Tabuada and G. J. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

[4] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic specifications," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 48–61, 2010.

[5] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in *Hybrid systems: Computation and Control*, Stockholm, Sweden, 2010, pp. 101–110.

[6] J. Tumova, B. Yordanov, C. Belta, I. Cerna, and J. Barnat, "A symbolic approach to controlling piecewise affine systems," in *IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, 2010.

[7] D. Berwanger, K. Chatterjee, M. De Wulf, L. Doyen, and T. A. Henzinger, "Strategy construction for parity games with imperfect information," *Inf. Comput.*, vol. 208, pp. 1206–1220, October 2010.

[8] W. Thomas, "Infinite games and verification," in *CAV*, 2002, pp. 58–64.

[9] M.Kloetzer and C. Belta, "Dealing with non-determinism in symbolic control," in *Hybrid Systems: Computation and Control: 11th International Workshop*, ser. Lecture Notes in Computer Science, M. Egerstedt and B. Mishra, Eds. Springer Berlin / Heidelberg, 2008, pp. 287–300.

[10] H. Kress-Gazit, G. Fainekos, and G. J. Pappas, "Where's waldo? sensor-based temporal logic motion planning," in *IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 3116–3121.

[11] A. Bianco and L. de Alfaro, "Model checking of probabilistic and nondeterministic systems," in *Foundations of Software Technology and Theoretical Computer Science*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1995, pp. 499–513.

[12] M. Kwiatkowska, G. Norman, and D. Parker, "Probabilistic symbolic model checking with prism: A hybrid approach," in *International Journal on Software Tools for Technology Transfer*. Springer, 2002, pp. 52–66.

[13] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, "MDP optimal control under temporal logic constraints," in *2011 IEEE Conference on Decision and Control (CDC 2011)*, Orlando, FL, 2011.

[14] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, pp. 291–314, October 2001.

[15] S. Jha, S. Gulwani, S. Seshia, and A. Tiwari, "Synthesizing switching logic for safety and dwell-time requirements," in *International Conference on Cyber-Physical Systems*, 2010, pp. 22–31.

[16] S. Gulwani, S. Jha, A. Tiwari, and R. Venkatesan, "Synthesis of loop-free programs," in *32nd ACM SIGPLAN conference on Programming Language Design and Implementation*, 2011, pp. 62–73.

[17] A. Ulusoy, T. Wongpiromsarn, and C. Belta, "Incremental control synthesis in probabilistic environments with temporal logic constraints," 2012, available at http://arxiv.org/abs/1209.0136.

[18] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Motion planning with hybrid dynamics and temporal goals," in *IEEE Conf. on Decision and Control*, 2010, pp. 1108–1115.

[19] L. Alfaro, *Formal verification of probabilistic systems*. Standford University, 1997, Ph.D. dissertation.

[20] T. Latvala, "Efficient model checking of safety properties," in *Model Checking Software. 10th International SPIN Workshop*. Springer, 2003, pp. 74–88.