2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 7-12, 2012. Vilamoura, Algarve, Portugal

Incremental Temporal Logic Synthesis of Control Policies for Robots Interacting with Dynamic Agents

Tichakorn Wongpiromsarn, Alphan Ulusoy, Calin Belta, Emilio Frazzoli and Daniela Rus

***Abstract*— We consider the synthesis of control policies from temporal logic specifications for robots that interact with multiple dynamic environment agents. Each environment agent is modeled by a Markov chain whereas the robot is modeled by a finite transition system (in the deterministic case) or Markov decision process (in the stochastic case). Existing results in probabilistic verification are adapted to solve the synthesis problem. To partially address the state explosion issue, we propose an incremental approach where only a small subset of environment agents is incorporated in the synthesis procedure initially and more agents are successively added until we hit the constraints on computational resources. Our algorithm runs in an anytime fashion where the probability that the robot satisfies its specification increases as the algorithm progresses.**

I. Introduction

Temporal logics [1]–[3] have been recently employed to precisely express complex behaviors of robots. In particular, given a robot specification expressed as a formula in a temporal logic, control policies that ensure or maximize the probability that the robot satisfies the specification can be automatically synthesized based on exhaustive exploration of the state space [4]–[12]. Hence, the main limitation of existing approaches for synthesizing control policies from temporal logic specifications is almost invariably due to a combinatorial blow up of the state space, commonly known as the state explosion problem.

In many applications, robots need to interact with external, potentially dynamic agents, including human and other robots. As a result, the control policy synthesis problem becomes more computationally complex as it takes into account more external (i.e., environment) agents. Consider, as an example, the problem where an autonomous vehicle needs to go through a pedestrian crossing while there are multiple pedestrians who are already at or approaching the crossing. The state space of the complete system (i.e., the vehicle and all the pedestrians) grows exponentially with the number of the pedestrians. Hence, given a limited budget of computational resources, solving the control policy synthesis problem with respect to temporal logic specifications may not be feasible when there are a large number of pedestrians.

In this paper, we partially address the aforementioned issue and propose an algorithm for computing a robot control policy in an anytime manner. Our algorithm progressively computes a sequence of control policies, taking into account only a small subset of the environment agents initially and successively adds more agents to the synthesis procedure in each iteration until the computational resource constraints are exceeded. As opposed to existing incremental synthesis approaches that handle temporal logic specifications where representative robot states are incrementally added to the synthesis procedure [8], we consider incrementally adding representative environment agents instead. Since the size of the state space grows linearly with the number of states of the robot but grows exponentially with the number of the environment agents, our approach potentially better handles the case with a large number of environment agents.

The main contribution of this paper is twofold. First, we propose an anytime algorithm for synthesizing a control policy for a robot interacting with multiple environment agents with the objective of maximizing the probability for the robot to satisfy a given temporal logic specification. Deciding if the resulting probability is acceptable, however, is application specific and is not in the scope of this paper. Second, we propose an incremental construction of various objects needed to be computed during the synthesis procedure. Such an incremental construction makes our anytime algorithm more efficient by avoiding unnecessary computation and exploiting the objects computed in the previous iteration. Experimental results show that not only we obtain a reasonable solution much faster, but we are also able to obtain an optimal solution faster than existing approaches.

II. Preliminaries

We consider systems that comprise multiple (possibly stochastic) components. In this section, we define the formalisms used in this paper to describe such systems and their desired properties. Throughout the paper, we let X^*, X^ω and X^+ denote the set of finite, infinite and nonempty finite strings, respectively, of a set X.

A. Automata

Definition 1: A *deterministic finite automaton* (DFA) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_{init}, F)$ where (a) Q is a finite set of states, (b) Σ is a finite set called alphabet, (c) $\delta : Q \times \Sigma \to Q$ is a transition function, (d) $q_{init} \in Q$ is the initial state, and (e) $F \subseteq Q$ is a set of final states. We use the relation notation, $q \xrightarrow{w} q'$ to denote $\delta(q, w) = q'$.

Consider a finite string $\sigma = \sigma_1\sigma_2 \dots \sigma_n \in \Sigma^*$. A *run* for σ in a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_{init}, F)$ is a finite sequence of states $q_0 q_1 \dots q_n$ such that $q_0 = q_{init}$ and $q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2 \xrightarrow{\sigma_3} \dots \xrightarrow{\sigma_n} q_n$. A run is *accepting* if $q_n \in F$. A string $\sigma \in \Sigma^*$ is *accepted* by $\mathcal{A}$ if there is an accepting run of σ in $\mathcal{A}$. The language *accepted* by $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of all accepted strings of $\mathcal{A}$.

T. Wongpiromsarn is with the Singapore-MIT Alliance for Research and Technology, Singapore 117543, Singapore. nok@smart.mit.edu

A. Ulusoy and C. Belta are with Boston University, Boston, MA, USA alphan@bu.edu, cbelta@bu.edu

E. Frazzoli and D. Rus are with the Massachusetts Institute of Technology, Cambridge, MA, USA frazzoli@mit.edu, rus@csail.mit.edu

B. Linear Temporal Logic

A linear temporal logic (LTL) formula is built up from a set Π of atomic propositions, the logic connectives $\neg$, $\vee$, $\wedge$ and $\Longrightarrow$ and the temporal modal operators $\bigcirc$ ("next"), $\Box$ ("always"), $\Diamond$ ("eventually") and $\mathcal{U}$ ("until") and can be used to reason about a time line. An LTL formula over a set Π of atomic propositions is inductively defined as

$$\varphi := \mathit{True} \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc \varphi \mid \varphi \,\mathcal{U}\, \varphi$$

where $p \in \Pi$. Other operators can be defined as follows: $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$, $\varphi \Longrightarrow \psi = \neg\varphi \vee \psi$, $\Diamond\varphi = \mathit{True}\,\mathcal{U}\,\varphi$, and $\Box\varphi = \neg\Diamond\neg\varphi$.

Semantics of LTL: LTL formulas are interpreted on infinite strings over 2^{Π}. Let $\sigma = \sigma_0\sigma_1\sigma_2\ldots$ where $\sigma_i \in 2^{\Pi}$ for all $i \geq 0$. The satisfaction relation $\models$ is defined inductively on LTL formulas as follows: (a) $\sigma \models \mathit{True}$, (b) for an atomic proposition $p \in \Pi$, $\sigma \models p$ if and only if $p \in \sigma_0$, (c) $\sigma \models \neg\varphi$ if and only if $\sigma \not\models \varphi$, (d) $\sigma \models \varphi_1 \wedge \varphi_2$ if and only if $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$, (e) $\sigma \models \bigcirc\varphi$ if and only if $\sigma_1\sigma_2\ldots \models \varphi$, and (f) $\sigma \models \varphi_1\,\mathcal{U}\,\varphi_2$ if and only if there exists $j \geq 0$ such that $\sigma_j\sigma_{j+1}\ldots \models \varphi_2$ and for all i such all $0 \leq i < j$, $\sigma_i\sigma_{i+1}\ldots \models \varphi_1$.

More details on LTL can be found, e.g., in [1]–[3].

In this paper, we are particularly interested in a class of LTL known as co-safety formulas. An important property of a co-safety formula is that any word satisfying the formula has a finite good prefix, i.e., a finite prefix that cannot be extended to violate the formula. Specifically, given an alphabet Σ, a language $L \subseteq \Sigma^{\omega}$ is co-safety if and only if every $w \in L$ has a good prefix $x \in \Sigma^{*}$ such that for all $y \in \Sigma^{\omega}$, we have $x \cdot y \in L$. In general, the problem of determining whether an LTL formula is co-safety is PSPACE-complete [13]. However, there is a class of co-safety formulas, known as *syntactically co-safe* LTL formulas, which can be easily characterized. A *syntactically co-safe* LTL formula over Π is an LTL formula over Π whose only temporal operators are $\bigcirc$, $\Diamond$ and $\mathcal{U}$ when written in positive normal form where the negation operator $\neg$ occurs only in front of atomic propositions [3], [13]. For example, properties such as avoiding obstacles before reaching a goal state can be specified using a syntactically co-safe LTL formula $\neg q\,\mathcal{U}\,p$ where p labels goal states and q labels states with an obstacle. It can be shown that for any syntactically co-safe formula φ, there exists a DFA $\mathcal{A}_{\varphi}$ that accepts all and only words in $\mathit{pref}(\varphi)$, i.e., $\mathcal{L}(\mathcal{A}_{\varphi}) = \mathit{pref}(\varphi)$, where $\mathit{pref}(\varphi)$ denote the set of all good prefixes for φ [9].

C. Systems and Control Policies

Definition 2: A *deterministic finite transition system (DFTS)* is a tuple $\mathcal{T} = (S, \mathit{Act}, \longrightarrow, s_{\mathit{init}}, \Pi, L)$ where (a) S is a finite set of states, (b) Act is a finite set of actions, (c) $\longrightarrow \subseteq S \times \mathit{Act} \times S$ is a transition relation such that for all $s \in S$ and $\alpha \in \mathit{Act}$, $|\mathit{Post}(s,\alpha)| \leq 1$ where $\mathit{Post}(s,\alpha) = \{s' \in S \mid (s,\alpha,s') \in \longrightarrow\}$, (d) $s_{\mathit{init}} \in S$ is the initial state, (e) Π is a set of atomic propositions, and (f) $L : S \to 2^{\Pi}$ is a labeling function. $(s,\alpha,s') \in \longrightarrow$ is denoted by $s \xrightarrow{\alpha} s'$. An action α is *enabled* in state s if and only if there exists s' such that $s \xrightarrow{\alpha} s'$.

Definition 3: A *(discrete-time) Markov chain* (MC) is a tuple $\mathcal{M} = (S, \mathbf{P}, \iota_{\mathit{init}}, \Pi, L)$ where (a) S, Π and L are defined as in DFTS, (b) $\mathbf{P} : S \times S \to [0,1]$ is the transition probability function such that for any state $s \in S$, $\sum_{s' \in S} \mathbf{P}(s,s') = 1$, and (c) $\iota_{\mathit{init}} : S \to [0,1]$ is the initial state distribution satisfying $\sum_{s \in S} \iota_{\mathit{init}}(s) = 1$.

Definition 4: A *Markov decision process* (MDP) is a tuple $\mathcal{M} = (S, \mathit{Act}, \mathbf{P}, \iota_{\mathit{init}}, \Pi, L)$ where S, Act, ι_{init}, Π and L are defined as in DFTS and MC and $\mathbf{P} : S \times \mathit{Act} \times S \to [0,1]$ is the transition probability function such that for any $s \in S$ and $\alpha \in \mathit{Act}$, $\sum_{s' \in S} \mathbf{P}(s,\alpha,s') \in \{0,1\}$. An action α is *enabled* in state s if and only if $\sum_{s' \in S} \mathbf{P}(s,\alpha,s') = 1$. Let $\mathit{Act}(s)$ denote the set of enabled actions in s.

Given a complete system as the composition of all its components, we are interested in computing a control policy for the system that optimizes certain objectives. We define a control policy for a system modeled by an MDP as follows.

Definition 5: Let $\mathcal{M} = (S, \mathit{Act}, \mathbf{P}, \iota_{\mathit{init}}, \Pi, L)$ be a Markov decision process. A *control policy* for $\mathcal{M}$ is a function $\mathcal{C} : S^{+} \to \mathit{Act}$ such that $\mathcal{C}(s_0 s_1 \ldots s_n) \in \mathit{Act}(s_n)$ for all $s_0 s_1 \ldots s_n \in S^{+}$.

Let $\mathcal{M} = (S, \mathit{Act}, \mathbf{P}, \iota_{\mathit{init}}, \Pi, L)$ be an MDP and $\mathcal{C} : S^{+} \to \mathit{Act}$ be a control policy for $\mathcal{M}$. Given an initial state s_0 of $\mathcal{M}$ such that $\iota_{\mathit{init}}(s_0) > 0$, an infinite sequence $r^{\mathcal{C}}_{\mathcal{M}} = s_0 s_1 \ldots$ on $\mathcal{M}$ generated under policy $\mathcal{C}$ is called a *path* on $\mathcal{M}$ if $\mathbf{P}(s_i, \mathcal{C}(s_0 s_1 \ldots s_i), s_{i+1}) > 0$ for all i. The subsequence $s_0 s_1 \ldots s_n$ where $n \geq 0$ is the *prefix* of length n of $r^{\mathcal{C}}_{\mathcal{M}}$. We define $\mathit{Paths}^{\mathcal{C}}_{\mathcal{M}}$ and $\mathit{FPaths}^{\mathcal{C}}_{\mathcal{M}}$ as the set of all infinite paths of $\mathcal{M}$ under policy $\mathcal{C}$ and their finite prefixes, respectively, starting from any state s_0 with $\iota_{\mathit{init}}(s_0) > 0$. For $s_0 s_1 \ldots s_n \in \mathit{FPaths}^{\mathcal{C}}_{\mathcal{M}}$, we let $\mathit{Paths}^{\mathcal{C}}_{\mathcal{M}}(s_0 s_1 \ldots s_n)$ denote the set of all paths in $\mathit{Paths}^{\mathcal{C}}_{\mathcal{M}}$ with the prefix $s_0 s_1 \ldots s_n$.

We refer the reader to [3] for a detailed discussion and to the full version of the paper [14] for a summary of the probability measure $\mathit{Pr}^{\mathcal{C}}_{\mathcal{M}}$ on the $\sigma-$algebra associated with $\mathcal{M}$. Roughly, given an LTL formula φ, one can show that the set $\{s_0 s_1 \ldots \in \mathit{Paths}^{\mathcal{C}}_{\mathcal{M}} \mid L(s_0)L(s_1)\ldots \models \varphi\}$ is measurable. The probability for $\mathcal{M}$ to satisfy φ under policy $\mathcal{C}$ is then defined as

$$\Pr^{\mathcal{C}}_{\mathcal{M}}(\varphi) = \Pr^{\mathcal{C}}_{\mathcal{M}}\{s_0 s_1 \ldots \in \mathit{Paths}^{\mathcal{C}}_{\mathcal{M}} \mid L(s_0)L(s_1)\ldots \models \varphi\},$$

where for any $s_0 s_1 \ldots s_n \in \mathit{FPaths}^{\mathcal{C}}_{\mathcal{M}}$,

$$\Pr^{\mathcal{C}}_{\mathcal{M}}\{\mathit{Paths}^{\mathcal{C}}_{\mathcal{M}}(s_0 s_1 \ldots s_n)\} = \iota_{\mathit{init}}(s_0) \textstyle\prod_{0 \leq i < n} \mathbf{P}(s_i, \mathcal{C}(s_0 s_1 \ldots s_i), s_{i+1}).$$

For a given (possibly noninitial) state $s \in S$, we define $\Pr^{\mathcal{C}}_{\mathcal{M}}(s \models \varphi) = \Pr^{\mathcal{C}}_{\mathcal{M}^s}(\varphi)$ as the probability for $\mathcal{M}$ to satisfy φ under policy $\mathcal{C}$, starting from s. Here, $\mathcal{M}^s = (S, \mathit{Act}, \mathbf{P}, \iota^s_{\mathit{init}}, \Pi, L)$ where $\iota^s_{\mathit{init}}(t) = 1$ if $s = t$ and $\iota^s_{\mathit{init}}(t) = 0$ otherwise.

A control policy essentially resolves all nondeterministic choices in an MDP and induces a Markov chain $\mathcal{M}_{\mathcal{C}}$ that formalizes the behavior of $\mathcal{M}$ under control policy $\mathcal{C}$ [3]. In general, $\mathcal{M}_{\mathcal{C}}$ contains all the states in S^{+} and hence may not be finite even though $\mathcal{M}$ is finite. However, for a special case where $\mathcal{C}$ is memoryless, it can be shown that $\mathcal{M}_{\mathcal{C}}$ can be identified with a finite MC.

Definition 6: Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, \Pi, L)$ be a Markov decision process. A control policy $\mathcal{C}$ on $\mathcal{M}$ is *memoryless* if and only if for each sequence $s_0 s_1 \dots s_n$ and $t_0 t_1 \dots t_m \in S^+$ with $s_n = t_m$, $\mathcal{C}(s_0 s_1 \dots s_n) = \mathcal{C}(t_0 t_1 \dots t_m)$. A memoryless control policy $\mathcal{C}$ can be described by a function $\mathcal{C} : S \to Act$.

III. Problem Formulation

Consider a system that comprises the plant (e.g., the robot) and N independent environment agents. We assume that at any time instance, the state of the system, which incorporates the state of the plant and the environment agents, can be precisely observed. The system can regulate the state of the plant but has no control over the state of the environment agents. Hence, we do not distinguish between a control policy *for the system* and a control policy *for the plant* and refer to them as a control policy in general, as there is no confusion that in both cases, only the state of the plant can be regulated and both the system and the plant can precisely observe the current state of the complete system. Hence, even though a control policy may be implemented on the plant, it may be defined over the state of the complete system.

We assume that each environment agent can be modeled by a finite Markov chain. Let $\mathcal{M}_i = (S_i, \mathbf{P}_i, \iota_{init,i}, \Pi_i, L_i)$ be the model of the ith environment agent. The plant is modeled either by a deterministic finite transition system or by a finite Markov decision process, depending on whether each control action leads to a deterministic state transition. We use $\mathcal{T}$ to denote the model of the plant and let $\mathcal{T} = (S_0, Act, \longrightarrow, s_{init,0}, \Pi_0, L_0)$ for the case where $\mathcal{T}$ is a DFTS and $\mathcal{T} = (S_0, Act, \mathbf{P}_0, \iota_{init,0}, \Pi_0, L_0)$ for the case where $\mathcal{T}$ is an MDP. For the simplicity of the presentation, we assume that for all $s \in S_0$, $Act(s) \neq \emptyset$. In addition, we assume that all the components $\mathcal{T}, \mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_N$ in the system make a transition simultaneously, i.e., each of them makes a transition at every time step.

Example 1: Consider a problem where an autonomous vehicle (plant) needs to go through a pedestrian crossing while there are N pedestrians (agents) who are already at or approaching the crossing. Suppose the road is discretized into a finite number of cells $c_0, c_2, \dots, c_M$. The vehicle is modeled by either a DFTS $\mathcal{T} = (S_0, Act, \longrightarrow, s_{init,0}, \Pi_0, L_0)$ or an MDP $\mathcal{T} = (S_0, Act, \mathbf{P}_0, \iota_{init,0}, \Pi_0, L_0)$ whose state $s \in S_0$ describes the cell occupied by the vehicle and whose action $\alpha \in Act$ corresponds to a motion primitive of the vehicle (e.g., stop, accelerate, decelerate). If each motion primitive leads to a deterministic change in the vehicle's state, then $\mathcal{T}$ is a DFTS. Otherwise, $\mathcal{T}$ is an MDP. The motion of the ith pedestrian is modeled by an MC $\mathcal{M}_i = (S_i, \mathbf{P}_i, \iota_{init,i}, \Pi_i, L_i)$ whose state $s \in S_i$ describes the cell occupied by the ith pedestrian. The labeling function L_i, $i \in \{0, \dots, N\}$ essentially maps each cell to its label, indexed by the agent ID, i.e., $L_i(c_j) = c_j^i$ for all $j \in \{0, \dots M\}$.

Control Policy Synthesis Problem: Given a system model described by $\mathcal{T}, \mathcal{M}_1, \dots, \mathcal{M}_N$ and a syntactically co-safe LTL formula φ over $\Pi_0 \cup \Pi_1 \cup \dots \cup \Pi_N$, we want to automatically synthesize a control policy that maximizes the probability for the system to satisfy φ.

Example 2: Consider the autonomous vehicle problem described in Example 1 and the desired property stating that the vehicle does not collide with any pedestrian until it reaches cell c_M (e.g., the other side of the pedestrian crossing). In this case, the specification φ is given by $\varphi = \left(\neg \bigvee_{i \geq 1, j \geq 0} (c_j^0 \wedge c_j^i)\right) \mathcal{U}\ c_M^0$. Using simple logic manipulation, it can be checked that φ is a co-safe LTL formula.

IV. Control Policy Synthesis

We employ existing results in probabilistic verification and consider the following 3 main steps to solve the control policy synthesis problem defined in Section III: 1) Compute the composition of all the system components to obtain the complete system. 2) Construct the product MDP. 3) Extract an optimal control policy for the product MDP.

In this section, we describe these steps in more detail and discuss their connection to our control policy synthesis problem described in Section III.

A. Parallel Composition of System Components

Assuming that all the components of the system make a transition simultaneously, we first construct the synchronous parallel composition of all the components to obtain the complete system. Synchronous parallel composition of different types of components is defined as follows.

Definition 7: Let $\mathcal{M}_1 = (S_1, \mathbf{P}_1, \iota_{init,1}, \Pi_1, L_1)$ and $\mathcal{M}_2 = (S_2, \mathbf{P}_2, \iota_{init,2}, \Pi_2, L_2)$ be Markov chains. Their synchronous parallel composition, denoted by $\mathcal{M}_1 || \mathcal{M}_2$, is the MC $\mathcal{M} = (S_1 \times S_2, \mathbf{P}, \iota_{init}, \Pi_1 \cup \Pi_2, L)$ where for each $s_1, s_1' \in S_1$ and $s_2, s_2' \in S_2$,

- $\mathbf{P}(\langle s_1, s_2 \rangle, \langle s_1', s_2' \rangle) = \mathbf{P}_1(s_1, s_1') \mathbf{P}_2(s_2, s_2')$,
- $\iota_{init}(\langle s_1, s_2 \rangle) = \iota_{init,1}(s_1) \iota_{init,2}(s_2)$, and
- $L(\langle s_1, s_2 \rangle) = L(s_1) \cup L(s_2)$.

Definition 8: Let $\mathcal{T}_1 = (S_1, Act, \longrightarrow, s_{init}, \Pi_1, L_1)$ be a deterministic finite transition system and $\mathcal{M}_2 = (S_2, \mathbf{P}_2, \iota_{init,2}, \Pi_2, L_2)$ be a Markov chain. Their synchronous parallel composition, denoted by $\mathcal{T}_1 || \mathcal{M}_2$, is the MDP $\mathcal{M} = (S_1 \times S_2, Act, \mathbf{P}, \iota_{init}, \Pi_1 \cup \Pi_2, L)$ where for each $s_1, s_1' \in S_1$, $s_2, s_2' \in S_2$ and $\alpha \in Act$,

- $\mathbf{P}(\langle s_1, s_2 \rangle, \alpha, \langle s_1', s_2' \rangle) = \mathbf{P}_2(s_2, s_2')$ if $s_1 \xrightarrow{\alpha} s_1'$ and $\mathbf{P}(\langle s_1, s_2 \rangle, \alpha, \langle s_1', s_2' \rangle) = 0$ otherwise,
- $\iota_{init}(\langle s_{init}, s_2 \rangle) = \iota_{init,2}(s_2)$ and $\iota_{init}(\langle s_1, s_2 \rangle) = 0$ for all $s_1 \in S \setminus \{s_{init}\}$, and
- $L(\langle s_1, s_2 \rangle) = L(s_1) \cup L(s_2)$.

Definition 9: Let $\mathcal{M}_1 = (S_1, Act, \mathbf{P}_1, \iota_{init,1}, \Pi_1, L_1)$ be a Markov decision process and $\mathcal{M}_2 = (S_2, \mathbf{P}_2, \iota_{init,2}, \Pi_2, L_2)$ be a Markov chain. Their synchronous parallel composition, denoted by $\mathcal{M}_1 || \mathcal{M}_2$, is the MDP $\mathcal{M} = (S_1 \times S_2, Act, \mathbf{P}, \iota_{init}, \Pi_1 \cup \Pi_2, L)$ where for each $s_1, s_1' \in S_1$, $s_2, s_2' \in S_2$ and $\alpha \in Act$,

- $\mathbf{P}(\langle s_1, s_2 \rangle, \alpha, \langle s_1', s_2' \rangle) = \mathbf{P}_1(s_1, \alpha, s_1') \mathbf{P}_2(s_2, s_2')$,
- $\iota_{init}(\langle s_1, s_2 \rangle) = \iota_{init,1}(s_1) \iota_{init,2}(s_2)$, and
- $L(\langle s_1, s_2 \rangle) = L(s_1) \cup L(s_2)$.

From the above definitions, our complete system can be modeled by the MDP $\mathcal{T} || \mathcal{M}_1 || \dots || \mathcal{M}_N$, regardless of whether $\mathcal{T}$ is a DFTS or an MDP. We denote this MDP by $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, \Pi, L)$.

B. Construction of Product MDP

Let $\mathcal{A}_\varphi = (Q, 2^\Pi, \delta, q_{init}, F)$ be a DFA that recognizes the good prefixes of φ. Such $\mathcal{A}_\varphi$ can be automatically constructed using existing tools [15]. Our next step is to obtain a finite MDP $\mathcal{M}_p = (S_p, Act_p, \mathbf{P}_p, \iota_{p,init}, Q, L_p)$ as the product of $\mathcal{M}$ and $\mathcal{A}_\varphi$, defined as follows.

Definition 10: Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, \Pi, L)$ be an MDP and let $\mathcal{A} = (Q, 2^\Pi, \delta, q_{init}, F)$ be a DFA. The product of $\mathcal{M}$ and $\mathcal{A}$ is the MDP $\mathcal{M}_p = \mathcal{M} \otimes \mathcal{A}$ defined by $\mathcal{M}_p = (S_p, Act, \mathbf{P}_p, \iota_{p,init}, \Pi, L_p)$ where $S_p = S \times Q$ and $L_p(\langle s, q\rangle) = L(s)$. $\mathbf{P}_p$ is defined as

$$\mathbf{P}_p(\langle s,q\rangle, \alpha, \langle s',q'\rangle) = \begin{cases} \tilde{\mathbf{P}}_p(\langle s,q\rangle, \alpha, \langle s',q'\rangle) & \\ \qquad \text{if } q' = \delta(q, L(s')) & \\ 0 \qquad \text{otherwise} & \end{cases}, \tag{1}$$

where $\tilde{\mathbf{P}}_p(\langle s,q\rangle, \alpha, \langle s',q'\rangle) = \mathbf{P}(s, \alpha, s')$. For the rest of the paper, we refer to $\tilde{\mathbf{P}}_p : S_p \times Act \times S_p \to [0,1]$ as the *intermediate transition probability function* for $\mathcal{M}_p$. Finally,

$$\iota_{p,init}(\langle s,q\rangle) = \begin{cases} \tilde{\iota}_{p,init}(\langle s,q\rangle) & \text{if } q = \delta(q_{init}, L(s)) \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

where $\tilde{\iota}_{p,init}(\langle s,q\rangle) = \iota_{init}(s)$. For the rest of the paper, we refer to $\tilde{\iota}_{p,init} : S_p \to [0,1]$ as the *intermediate initial state distribution* for $\mathcal{M}_p$.

Stepping through the above definition shows that given a path $r^{\mathcal{C}_p}_{\mathcal{M}_p} = \langle s_0, q_0\rangle\langle s_1, q_1\rangle \dots$ on $\mathcal{M}_p$ generated under some control policy $\mathcal{C}_p$, the corresponding path $s_0 s_1 \dots$ on $\mathcal{M}$ generates a word $L(s_0)L(s_1)\dots$ that satisfies φ if and only if there exists $n \geq 0$ such that $q_n \in F$ (and hence $q_0 q_1 \dots q_n$ is an accepting run on $\mathcal{A}_\varphi$), in which case we say that $r^{\mathcal{C}_p}_{\mathcal{M}_p}$ is accepting. Therefore, each accepting path of $\mathcal{M}_p$ uniquely corresponds to a path of $\mathcal{M}$ whose word satisfies φ. In addition, a control policy $\mathcal{C}_p$ on $\mathcal{M}_p$ induces the corresponding control policy $\mathcal{C}$ on $\mathcal{M}$. The details for generating $\mathcal{C}$ from $\mathcal{C}_p$ can be found, e.g. in [3], [10].

Based on this argument, our control policy synthesis problem defined in Section III can be reduced to computing a control policy for $\mathcal{M}_p$ that maximizes the probability of reaching a state in $B_p = \{\langle s,q\rangle \in S_p \mid q \in F\}$.

C. Control Policy Synthesis for Product MDP

For each $s \in S_p$, let x_s denote the maximum probability of reaching a state in B_p, starting from s. Formally, $x_s = \sup_{\mathcal{C}} \Pr^{\mathcal{C}}_{\mathcal{M}_p}(s \models \Diamond B_p)$, where, with an abuse of notation, B_p in $\Diamond B_p$ is a proposition that is satisfied by all states in B_p. There are two main techniques for computing the probability x_s for each $s \in S_p$: linear programming (LP) and value iteration. LP-based techniques yield an exact solution but it typically does not scale as well as value iteration. On the other hand, value iteration is an iterative numerical technique. This method works by successively computing the probability vector $(x^{(k)}_s)_{s\in S_p}$ for increasing $k \geq 0$ such that $\lim_{k\to\infty} x^{(k)}_s = x_s$ for all $s \in S_p$. Initially, we set $x^{(0)}_s = 1$ if $s \in B_p$ and $x^{(0)}_s = 0$ otherwise. In the $(k+1)$th iteration where $k \geq 0$, we set

$$x^{(k+1)}_s = \begin{cases} 1 & \text{if } s \in B_p \\ \max\limits_{\alpha \in Act_p(s)} \sum\limits_{t \in S_p} \mathbf{P}_p(s, \alpha, t) x^{(k)}_t & \text{otherwise.} \end{cases} \tag{3}$$

In practice, we terminate the computation and say that $x^{(k)}_s$ converges when a termination criterion such as $\max_{s\in S_p} |x^{(k+1)}_s - x^{(k)}_s| < \epsilon$ is satisfied for some fixed (typically very small) threshold ϵ.

As discussed in [16], [17], decomposition of $\mathcal{M}_p$ into *strongly connected components* (SCC) can help speed up value iteration. $C \subseteq S_p$ is an SCC of $\mathcal{M}_p$ if there is a path in $\mathcal{M}_p$ between any two states in C and C is maximal (i.e., there does not exist any $\tilde{C} \subseteq S_p$ such that $C \subset \tilde{C}$ and $\tilde{C}$ is an SCC). The algorithm proposed in [18] allows us to identify all the SCCs of $\mathcal{M}_p$ with time and space complexity that is linear in the size of $\mathcal{M}_p$.

The SCC-based value iteration works as follows. First, we set $x^{(0)}_s = 1$ if $s \in B_p$ and $x^{(0)}_s = 0$ otherwise. Next, we identify all the SCCs $C^{\mathcal{M}_p}_1, \dots, C^{\mathcal{M}_p}_m$ of $\mathcal{M}_p$. From the definition of SCC, we get that $C^{\mathcal{M}_p}_i \cap C^{\mathcal{M}_p}_j = \emptyset, \forall i \neq j$ and $\bigcup_i C^{\mathcal{M}_p}_i = S_p$. For each SCC $C^{\mathcal{M}_p}_i$, we define $Succ(C^{\mathcal{M}_p}_i) \subseteq S_p \setminus C^{\mathcal{M}_p}_i$ to be the set of all the immediate successors of states in $C^{\mathcal{M}_p}_i$ that are not in $C^{\mathcal{M}_p}_i$. A (strict) partial order, $\prec_{\mathcal{M}_p}$, among $C^{\mathcal{M}_p}_1, \dots, C^{\mathcal{M}_p}_m$ can be defined such that $C^{\mathcal{M}_p}_j \prec_{\mathcal{M}_p} C^{\mathcal{M}_p}_i$ if $Succ(C^{\mathcal{M}_p}_i) \cap C^{\mathcal{M}_p}_j \neq \emptyset$. (Note that from the definition of SCC and $Succ$, there cannot be cyclic dependency among SCCs; hence, such a partial order can always be defined.)

An important property of SCCs and their partial order that we will exploit in the computation of the probability vector $(x_s)_{s\in S_p}$ is that the probability values of states in $C^{\mathcal{M}_p}_i$ can be affected only by the probability values of states in $C^{\mathcal{M}_p}_i$ and all $C^{\mathcal{M}_p}_j \prec_{\mathcal{M}_p} C^{\mathcal{M}_p}_i$. Thus, our next step is to generate an order $\mathbb{O}^{\mathcal{M}_p}$ among $C^{\mathcal{M}_p}_1, \dots, C^{\mathcal{M}_p}_m$ such that $C^{\mathcal{M}_p}_i$ appears before $C^{\mathcal{M}_p}_j$ in $\mathbb{O}^{\mathcal{M}_p}$ if $C^{\mathcal{M}_p}_i \prec_{\mathcal{M}_p} C^{\mathcal{M}_p}_j$. We can then process each SCC separately, according to the order in $\mathbb{O}^{\mathcal{M}_p}$, since the probability values of states in $C^{\mathcal{M}_p}_j$ that appears after $C^{\mathcal{M}_p}_i$ in $\mathbb{O}^{\mathcal{M}_p}$ cannot affect the probability values of states in $C^{\mathcal{M}_p}_i$. Processing of SCC $C^{\mathcal{M}_p}_i$ terminates at the kth iteration where all $x^{(k)}_s$, $s \in C^{\mathcal{M}_p}_i$ converges. Let x_s be the value to which $x^{(k)}_s$ converges. When processing $C^{\mathcal{M}_p}_i$, we exploit the order in $\mathbb{O}^{\mathcal{M}_p}$ and existing values of x_t for all $t \in Succ(C^{\mathcal{M}_p}_i)$ to determine the set of $s \in C^{\mathcal{M}_p}_i$ where $x^{(k+1)}_s$ needs to be updated from $x^{(k)}_s$. The formula in (3) with $x^{(k)}_t$ replaced by x_t for all $t \in Succ(C^{\mathcal{M}_p}_i)$ can be used to update those $x^{(k+1)}_s$. We refer the reader to [16], [17] for more details.

Note that computation of an order $\mathbb{O}^{\mathcal{M}_p}$ requires $O(|S_p|^2)$ time. Thus, the pre-computation required by the SCC-based value iteration can be computationally expensive, unless all the SCCs of $\mathcal{M}_p$ and an order $\mathbb{O}^{\mathcal{M}_p}$ are provided a-priori. As a result, the SCC-based value iteration may require more computation time than the normal value iteration, if the pre-computation time is also taken into account.

Once the vector $(x_s)_{s\in S_p}$ is computed, a memoryless control policy $\mathcal{C}$ such that for any $s \in S_p$, $\Pr^{\mathcal{C}}_{\mathcal{M}}(s \models \Diamond B_p) = x_s$ can be constructed as follows. For each state $s \in S_p$, let $Act^{max}_p(s) \subseteq Act_p(s)$ be the set of actions such that

for all $\alpha \in Act_p^{max}(s)$, $x_s = \sum_{t \in S_p} \mathbf{P}(s,\alpha,t)x_t$. For each $s \in S_p$ with $x_s > 0$, let $\|s\|$ be the length of a shortest path from s to a state in B_p, using only actions in Act_p^{max}. $\mathcal{C}(s) \in Act_p^{max}(s)$ for a state $s \in S_p \setminus B_p$ with $x_s > 0$ is then chosen such that $\mathbf{P}_p(s,\mathcal{C}(s),t) > 0$ for some $t \in S_p$ with $\|t\| = \|s\| - 1$. For a state $s \in S_p$ with $x_s = 0$ or a state $s \in B_p$, $\mathcal{C}(s) \in Act_p(s)$ can be chosen arbitrarily.

V. Incremental Computation of Control Policies

Automatic synthesis described in the previous section suffers from the state explosion problem since it requires constructing the composition $\mathcal{T}||\mathcal{M}_1||\ldots||\mathcal{M}_N$. In this section, we propose an incremental synthesis approach where we progressively compute a sequence of control policies, taking into account a small subset of the environment agents initially and successively add more agents to the synthesis procedure in each iteration until we hit the computational constraints. Hence, even though the complete synthesis problem cannot be solved due to the computational resource limitation, we can still obtain a reasonable control policy that takes into account a certain set of environment agents.

A. Overview of Incremental Computation of Control Policies

Initially, we consider a small subset $\mathbf{M}_0 \subset \{\mathcal{M}_1,\ldots,\mathcal{M}_N\}$ of the environment agents. For each $\mathcal{M}_i = (S_i,\mathbf{P}_i,\iota_{init,i},\Pi_i,L_i) \notin \mathbf{M}_0$, we consider a simplified model $\tilde{\mathcal{M}}_i$ that essentially assumes that the ith environment agent is stationary (i.e., we take into account their presence but do not consider their full model). Formally, $\tilde{\mathcal{M}}_i = (\{s_i\},\tilde{\mathbf{P}}_i,\tilde{\iota}_{init,i},\Pi_i,\tilde{L}_i)$ where $s_i \in S_i$ can be chosen arbitrarily, $\tilde{\mathbf{P}}_i(s_i,s_i) = 1$, $\tilde{\iota}_{init,i}(s_i) = 1$ and $\tilde{L}_i(s_i) = L_i(s_i)$. Note that the choice of $s_i \in S_i$ may affect the performance of our incremental synthesis algorithm; hence, it should be chosen such that it is the most likely state of $\mathcal{M}_i$. We let $\tilde{\mathbf{M}}_0 = \{\tilde{\mathcal{M}}_i \mid \mathcal{M}_i \in \{\mathcal{M}_1,\ldots,\mathcal{M}_N\} \setminus \mathbf{M}_0\}$.

The composition of $\mathcal{T}$, all $\mathcal{M}_i \in \mathbf{M}_0$ and all $\tilde{\mathcal{M}}_j \in \tilde{\mathbf{M}}_0$ is then constructed. We let $\mathcal{M}^{\mathbf{M}_0}$ be the MDP that represents such composition. Note that since $\tilde{\mathcal{M}}_i$ is typically smaller than $\mathcal{M}_i$, $\mathcal{M}^{\mathbf{M}_0}$ is typically much smaller than the composition of $\mathcal{T},\mathcal{M}_1,\ldots,\mathcal{M}_N$. We identify all the SCCs of $\mathcal{M}^{\mathbf{M}_0}$ and their partial order. Following the steps for synthesizing a control policy described in Section IV, we construct $\mathcal{M}_p^{\mathbf{M}_0} = \mathcal{M}^{\mathbf{M}_0} \otimes \mathcal{A}_\varphi$ where $\mathcal{A}_\varphi = (Q,2^\Pi,\delta,q_{init},F)$ is a DFA that recognizes the good prefixes of φ. We also store the intermediate transition probability function and the intermediate initial state distribution for $\mathcal{M}_p^{\mathbf{M}_0}$ and denote these functions by $\tilde{\mathbf{P}}_p^{\mathbf{M}_0}$ and $\tilde{\iota}_{p,init}^{\mathbf{M}_0}$, respectively.

At the end of the initialization period (i.e., the 0th iteration), we obtain a control policy $\mathcal{C}^{\mathbf{M}_0}$ that maximizes the probability for $\mathcal{M}^{\mathbf{M}_0}$ to satisfy φ. $\mathcal{C}^{\mathbf{M}_0}$ resolves all nondeterministic choices in $\mathcal{M}^{\mathbf{M}_0}$ and induces a Markov chain, which we denote by $\mathcal{M}_{\mathcal{C}^{\mathbf{M}_0}}^{\mathbf{M}_0}$.

Our algorithm then successively adds more full models of the rest of the environment agents to the synthesis procedure at each iteration. In the $(k+1)$th iteration where $k \geq 0$, we consider $\mathbf{M}_{k+1} = \mathbf{M}_k \cup \{\mathcal{M}_l\}$ for some $\mathcal{M}_l \in \{\mathcal{M}_1,\ldots,\mathcal{M}_N\} \setminus \mathbf{M}_k$. Such $\mathcal{M}_l$ may be picked such that the probability for $\mathcal{M}_{\mathcal{C}^{\mathbf{M}_0}}^{\mathbf{M}_0}||\mathcal{M}_l$ to satisfy φ is the minimum among all $\mathcal{M}_i \in \{\mathcal{M}_1,\ldots,\mathcal{M}_N\} \setminus \mathbf{M}_k$. This probability can be efficiently computed using probabilistic verification [3]. Other criteria for picking $\mathcal{M}_l$, including random selection, can also be used and will not affect the correctness of the algorithm but may affect the computation time. We let $\tilde{\mathbf{M}}_{k+1} = \tilde{\mathbf{M}}_k \setminus \{\tilde{\mathcal{M}}_l\}$ and let $\mathcal{M}^{\mathbf{M}_{k+1}}$ be the MDP that represents the composition of $\mathcal{T}$, all $\mathcal{M}_i \in \mathbf{M}_{k+1}$ and all $\tilde{\mathcal{M}}_j \in \tilde{\mathbf{M}}_{k+1}$. Next, we construct $\mathcal{M}_p^{\mathbf{M}_{k+1}} = \mathcal{M}^{\mathbf{M}_{k+1}} \otimes \mathcal{A}_\varphi$ and obtain a control policy $\mathcal{C}^{\mathbf{M}_{k+1}}$ that maximizes the probability for $\mathcal{M}^{\mathbf{M}_{k+1}}$ to satisfy φ. Similar to the initialization step, during the construction of $\mathcal{M}_p^{\mathbf{M}_{k+1}}$, we store the intermediate transition probability function and the intermediate initial state distribution for $\mathcal{M}_p^{\mathbf{M}_{k+1}}$ and denote these functions by $\tilde{\mathbf{P}}_p^{\mathbf{M}_{k+1}}$ and $\tilde{\iota}_{p,init}^{\mathbf{M}_{k+1}}$, respectively.

The process outlined in the previous paragraph terminates at the Kth iteration where $\mathbf{M}_K = \{\mathcal{M}_1,\ldots,\mathcal{M}_N\}$ or when the computational resource constraints are exceeded. To make this process more efficient, we avoid unnecessary computation and exploit the objects computed in the previous iteration. Consider an arbitrary iteration $k \geq 0$. In Section V-B, we show how $\mathcal{M}_p^{\mathbf{M}_{k+1}}$, $\tilde{\mathbf{P}}_p^{\mathbf{M}_{k+1}}$, and $\tilde{\iota}_{p,init}^{\mathbf{M}_{k+1}}$ can be incrementally constructed from $\mathcal{M}_p^{\mathbf{M}_k}$, $\tilde{\mathbf{P}}_p^{\mathbf{M}_k}$ and $\tilde{\iota}_{p,init}^{\mathbf{M}_k}$. Hence, we can avoid computing $\mathcal{M}^{\mathbf{M}_{k+1}}$. In addition, as previously discussed in Section IV-C, generating an order of SCCs can be computationally expensive. Hence, we only compute the SCCs and their order for $\mathcal{M}^{\mathbf{M}_0}$ and all $\mathcal{M}_j \in \{\mathcal{M}_1,\ldots,\mathcal{M}_N\} \setminus \mathbf{M}_0$, which are typically small. Incremental construction of SCCs of $\mathcal{M}^{\mathbf{M}_{k+1}}$ and their order from those of $\mathcal{M}^{\mathbf{M}_k}$ is considered in Section V-C. (Note that we do not compute $\mathcal{M}^{\mathbf{M}_k}$ but only maintain its SCCs and their order, which are incrementally constructed using the results from the previous iteration.) Finally, Section V-D describes computation of $\mathcal{C}^{\mathbf{M}_k}$, using a method adapted from SCC-based value iteration where we avoid having to identify the SCCs of $\mathcal{M}_p^{\mathbf{M}_k}$ and their order. Instead, we exploit the SCCs of $\mathcal{M}^{\mathbf{M}_k}$ and their order, which can be incrementally constructed using the approach described in Section V-C.

Due to space constraints, we omit the proof here. The detailed proof can be found in a technical report [14].

B. Incremental Construction of Product MDP

For an iteration $k \geq 0$, let $\mathbf{M}_{k+1} = \mathbf{M}_k \cup \{\mathcal{M}_l\}$ for some $\mathcal{M}_l \in \{\mathcal{M}_1,\ldots,\mathcal{M}_N\} \setminus \mathbf{M}_k$. In general, one can construct $\mathcal{M}_p^{\mathbf{M}_{k+1}}$ by first computing $\mathcal{M}^{\mathbf{M}_{k+1}}$, which requires taking the composition of a DFTS or an MDP with N MCs, and then constructing $\mathcal{M}^{\mathbf{M}_{k+1}} \otimes A_\varphi$. To accelerate the process of computing $\mathcal{M}_p^{\mathbf{M}_{k+1}}$, we exploit the presence of $\mathcal{M}_p^{\mathbf{M}_k}$, its intermediate transition probability function $\tilde{\mathbf{P}}_p^{\mathbf{M}_k}$ and intermediate initial state distribution $\tilde{\iota}_{p,init}^{\mathbf{M}_k}$, which are computed in the previous iteration.

First, note that a state s_p of $\mathcal{M}_p^{\mathbf{M}_k}$ is of the form $s_p = \langle s,q\rangle$ where $s = \langle s_0,s_1,\ldots,s_N\rangle \in S_0 \times S_1 \times \ldots \times S_N$ and $q \in Q$. For $s = \langle s_0,s_1,\ldots,s_N\rangle \in S_0 \times S_1 \times \ldots \times S_N$, $i \in \{0,\ldots,N\}$ and $r \in S_i$, we define $s|_{i \leftarrow r} = \langle s_0,\ldots,s_{i-1},r,s_{i+1},\ldots,s_N\rangle$, i.e., $s|_{i\leftarrow r}$ is obtained by replacing the ith element of s by r.

Lemma 1: Consider an arbitrary iteration $k \geq 0$. Let $\mathbf{M}_{k+1} = \mathbf{M}_k \cup \{\mathcal{M}_l\}$ where $\mathcal{M}_l \in \{\mathcal{M}_1, \ldots, \mathcal{M}_N\} \setminus \mathbf{M}_k$. Suppose $\mathcal{M}_p^{\mathbf{M}_k} = (S_p^{\mathbf{M}_k}, Act_p^{\mathbf{M}_k}, \mathbf{P}_p^{\mathbf{M}_k}, \iota_{p,init}^{\mathbf{M}_k}, \Pi_p^{\mathbf{M}_k}, L_p^{\mathbf{M}_k})$ and $\mathcal{M}_l = (S_l, \mathbf{P}_l, \iota_{init,l}, \Pi_l, L_l)$. Assuming that for any $i, j \in \{0, \ldots, N\}$, $\Pi_i \cap \Pi_j = \emptyset$, then $\mathcal{M}_p^{\mathbf{M}_{k+1}} = (S_p^{\mathbf{M}_{k+1}}, Act_p^{\mathbf{M}_{k+1}}, \mathbf{P}_p^{\mathbf{M}_{k+1}}, \iota_{p,init}^{\mathbf{M}_{k+1}}, \Pi_p^{\mathbf{M}_k}, L_p^{\mathbf{M}_{k+1}})$ where $S_p^{\mathbf{M}_{k+1}} = \{\langle s|_{l \leftarrow r}, q\rangle \mid \langle s, q\rangle \in S_p^{\mathbf{M}_k} \text{ and } r \in S_l\}$, $Act_p^{\mathbf{M}_{k+1}} = Act_p^{\mathbf{M}_k}$, $\Pi_p^{\mathbf{M}_{k+1}} = \Pi_p^{\mathbf{M}_k}$, and for any $s = \langle s_0, \ldots, s_N\rangle, s' = \langle s'_0, \ldots, s'_N\rangle \in S_0 \times \ldots S_N$ and $q, q' \in Q$,

- $\mathbf{P}_p^{\mathbf{M}_{k+1}}(\langle s, q\rangle, \alpha, \langle s', q'\rangle) = 0$ if $q' \neq \delta(q, L_p^{\mathbf{M}_{k+1}}(\langle s', q'\rangle))$. Otherwise, $\mathbf{P}_p^{\mathbf{M}_{k+1}}(\langle s, q\rangle, \alpha, \langle s', q'\rangle) = \tilde{\mathbf{P}}_p^{\mathbf{M}_{k+1}}(\langle s, q\rangle, \alpha, \langle s', q'\rangle)$ where the intermediate transition probability function is given by
$$\tilde{\mathbf{P}}_p^{\mathbf{M}_{k+1}}(\langle s, q\rangle, \alpha, \langle s', q'\rangle) = \mathbf{P}_l(s_l, s'_l)\tilde{\mathbf{P}}_p^{\mathbf{M}_k}(\langle \tilde{s}, q\rangle, \alpha, \langle \tilde{s}', q'\rangle) \quad (4)$$
for any $\langle \tilde{s}, q\rangle, \langle \tilde{s}', q'\rangle \in S_p^{\mathbf{M}_k}$ such that $\tilde{s}|_{l \leftarrow s_l} = s$ and $\tilde{s}'|_{l \leftarrow s'_l} = s'$.
- $\iota_{p,init}^{\mathbf{M}_{k+1}}(\langle s, q\rangle) = 0$ if $q \neq \delta(q_{init}, L_p^{\mathbf{M}_{k+1}}(\langle s, q\rangle))$. Otherwise, $\iota_{p,init}^{\mathbf{M}_{k+1}}(\langle s, q\rangle) = \tilde{\iota}_{p,init}^{\mathbf{M}_{k+1}}(\langle s, q\rangle)$ where the intermediate initial state distribution is given by
$$\tilde{\iota}_{p,init}^{\mathbf{M}_{k+1}}(\langle s, q\rangle) = \iota_{init,l}(s_l)\tilde{\iota}_{p,init}^{\mathbf{M}_k}(\langle \tilde{s}, q\rangle) \quad (5)$$
for any $\langle \tilde{s}, q\rangle \in S_p^{\mathbf{M}_k}$ such that $\tilde{s}|_{l \leftarrow s_l} = s$.
- $L_p^{\mathbf{M}_{k+1}}(\langle s, q\rangle) = \big(L_p^{\mathbf{M}_k}(\langle \tilde{s}, q\rangle) \setminus L_l(\tilde{s}_l)\big) \cup L_l(s_l)$ for any $\langle \tilde{s}, q\rangle \in S_p^{\mathbf{M}_k}$ such that $\tilde{s}|_{l \leftarrow s_l} = s$.

C. Incremental Construction of SCCs

Consider an arbitrary iteration $k \geq 0$. Let l be the index of the environment agent such that $\mathbf{M}_{k+1} = \mathbf{M}_k \cup \{\mathcal{M}_l\}$. In this section, we first provide a way to incrementally identify all the SCCs of $\mathcal{M}^{\mathbf{M}_{k+1}}$ from all the SCCs of $\mathcal{M}^{\mathbf{M}_k}$ and $\mathcal{M}_l$. We conclude the section with incremental construction of the partial order over the SCCs of $\mathcal{M}^{\mathbf{M}_{k+1}}$ from the partial order defined over the SCCs of $\mathcal{M}^{\mathbf{M}_k}$ and $\mathcal{M}_l$.

Lemma 2: Let $C^{\mathbf{M}_k}$ be an SCC of $\mathcal{M}^{\mathbf{M}_k}$ and C^l be an SCC of $\mathcal{M}_l$ where $\mathbf{M}_{k+1} = \mathbf{M}_k \cup \{\mathcal{M}_l\}$. Suppose either of the following conditions holds:

Cond 1: $|C^{\mathbf{M}_k}| = 1$ and the state in $C^{\mathbf{M}_k}$ does not have a self-loop in $\mathcal{M}^{\mathbf{M}_k}$.

Cond 2: $|C^l| = 1$ and the state in C^l does not have a self-loop in $\mathcal{M}_l$.

Then, for any $s \in C^{\mathbf{M}_k}$ and $r \in C^l$, $\{s|_{l \leftarrow r}\}$ is an SCC of $\mathcal{M}^{\mathbf{M}_{k+1}}$. Otherwise, $\{s|_{l \leftarrow r} \mid s \in C^{\mathbf{M}_k}, r \in C^l\}$ is an SCC of $\mathcal{M}^{\mathbf{M}_{k+1}}$.

We say that an SCC $C^{\mathbf{M}_{k+1}}$ of $\mathcal{M}^{\mathbf{M}_{k+1}}$ is *derived* from $\langle C^{\mathbf{M}_k}, C^l\rangle$, where $C^{\mathbf{M}_k}$ is an SCC of $\mathcal{M}^{\mathbf{M}_k}$ and C^l is an SCC of $\mathcal{M}_l$, if $C^{\mathbf{M}_{k+1}}$ is constructed from $C^{\mathbf{M}_k}$ and C^l according to Lemma 2, i.e., $C^{\mathbf{M}_{k+1}} = \{s|_{l \leftarrow r}\}$ for some $s \in C^{\mathbf{M}_k}$ and $r \in C^l$ if Cond 1 or Cond 2 in Lemma 2 holds; otherwise, $C^{\mathbf{M}_{k+1}} = \{s|_{l \leftarrow r} \mid s \in C^{\mathbf{M}_k}, r \in C^l\}$.

Lemma 3: For each SCC $C^{\mathbf{M}_{k+1}}$ of $\mathcal{M}^{\mathbf{M}_{k+1}}$, there exists a unique $\langle C^{\mathbf{M}_k}, C^l\rangle$ from which $C^{\mathbf{M}_{k+1}}$ is derived.

Lemma 2 and Lemma 3 provide a way to generate all the SCCs of $\mathcal{M}^{\mathbf{M}_{k+1}}$ from all the SCCs of $\mathcal{M}^{\mathbf{M}_k}$ and $\mathcal{M}_l$ as formally stated below.

Corollary 1: The set of all the SCCs of $\mathcal{M}^{\mathbf{M}_{k+1}}$ is given by $\big\{C^{\mathbf{M}_{k+1}}$ derived from $\langle C^{\mathbf{M}_k}, C^l\rangle \mid C^{\mathbf{M}_k}$ is an SCC of $\mathcal{M}^{\mathbf{M}_k}$ and C^l is an SCC of $\mathcal{M}_l\big\}$.

Finally, in the following lemma, we provide a necessary condition, based on the partial order over the SCCs of $\mathcal{M}^{\mathbf{M}_k}$ and $\mathcal{M}_l$, for the existence of the partial order between two SCCs of $\mathcal{M}^{\mathbf{M}_{k+1}}$.

Lemma 4: Let $C_1^{\mathbf{M}_{k+1}}$ and $C_2^{\mathbf{M}_{k+1}}$ be SCCs of $\mathcal{M}^{\mathbf{M}_{k+1}}$. Suppose $C_1^{\mathbf{M}_{k+1}}$ is derived from $\langle C_1^{\mathbf{M}_k}, C_1^l\rangle$ and $C_2^{\mathbf{M}_{k+1}}$ is derived from $\langle C_2^{\mathbf{M}_k}, C_2^l\rangle$ where $C_1^{\mathbf{M}_k}$ and $C_2^{\mathbf{M}_k}$ are SCCs of $\mathcal{M}^{\mathbf{M}_k}$ and C_1^l and C_2^l are SCCs of $\mathcal{M}_l$. Then, $C_1^{\mathbf{M}_{k+1}} \prec_{\mathcal{M}^{\mathbf{M}_{k+1}}} C_2^{\mathbf{M}_{k+1}}$ only if $C_1^{\mathbf{M}_k} \prec_{\mathcal{M}^{\mathbf{M}_k}} C_2^{\mathbf{M}_k}$ and $C_1^l \prec_{\mathcal{M}_l} C_2^l$.

D. Computation of Probability Vector and Control Policy for $\mathcal{M}_p^{\mathbf{M}_k}$ from SCCs of $\mathcal{M}^{\mathbf{M}_k}$

Consider an arbitrary iteration $k \geq 0$ and the associated product MDP $\mathcal{M}_p^{\mathbf{M}_k} = (S_p^{\mathbf{M}_k}, Act_p^{\mathbf{M}_k}, \mathbf{P}_p^{\mathbf{M}_k}, \iota_{p,init}^{\mathbf{M}_k}, \Pi_p^{\mathbf{M}_k}, L_p^{\mathbf{M}_k})$. Similar to the SCC-based value iteration, we want to generate a partition $\{D_{p,1}^{\mathbf{M}_k}, \ldots, D_{p,m_k}^{\mathbf{M}_k}\}$ of $S_p^{\mathbf{M}_k}$ with a partial order $\prec_{\mathcal{M}_p^{\mathbf{M}_k}}$ such that $D_{p,j}^{\mathbf{M}_k} \prec_{\mathcal{M}_p^{\mathbf{M}_k}} D_{p,i}^{\mathbf{M}_{k+1}}$ if $Succ(D_{p,i}^{\mathbf{M}_k}) \cap D_{p,j}^{\mathbf{M}_k} \neq \emptyset$. However, we relax the condition that each $D_{p,i}^{\mathbf{M}_k}, i \in \{1, \ldots, m_k\}$ is an SCC of $\mathcal{M}_p^{\mathbf{M}_k}$ and only require that if $D_{p,i}^{\mathbf{M}_k}$ contains a state in an SCC $C_p^{\mathbf{M}_k}$ of $\mathcal{M}_p^{\mathbf{M}_k}$, then it has to contain all the states in $C_p^{\mathbf{M}_k}$. Hence, $D_i^{\mathbf{M}_k}$ may include all the states in multiple SCCs of $\mathcal{M}_p^{\mathbf{M}_k}$. The following lemmas provide a method for constructing $\{D_1^{\mathbf{M}_k}, \ldots, D_{m_k}^{\mathbf{M}_k}\}$ and their partial order from SCCs of $\mathcal{M}^{\mathbf{M}_k}$ and their partial order, which can be incrementally constructed as described in Section V-C.

Lemma 5: Let $C_p^{\mathbf{M}_k}$ be an SCC of $\mathcal{M}_p^{\mathbf{M}_k}$. Then, there exists a unique SCC $C^{\mathbf{M}_k}$ of $\mathcal{M}^{\mathbf{M}_k}$ such that $C_p^{\mathbf{M}_k} \subseteq C^{\mathbf{M}_k} \times Q$.

Lemma 6: Let $C_p^{\mathbf{M}_k}$ and $\tilde{C}_p^{\mathbf{M}_k}$ be SCCs of $\mathcal{M}_p^{\mathbf{M}_k}$. Suppose $C^{\mathbf{M}_k}$ and $\tilde{C}^{\mathbf{M}_k}$ are unique SCCs of $\mathcal{M}^{\mathbf{M}_k}$ such that $C_p^{\mathbf{M}_k} \subseteq C^{\mathbf{M}_k} \times Q$ and $\tilde{C}_p^{\mathbf{M}_k} \subseteq \tilde{C}^{\mathbf{M}_k} \times Q$. Then, $C_p^{\mathbf{M}_k} \prec_{\mathcal{M}_p^{\mathbf{M}_k}} \tilde{C}_p^{\mathbf{M}_k}$ only if $C^{\mathbf{M}_k} \prec_{\mathcal{M}^{\mathbf{M}_k}} \tilde{C}^{\mathbf{M}_k}$.

Lemma 7: Let $C_1^{\mathbf{M}_k}, \ldots, C_{m_k}^{\mathbf{M}_k}$ be all the SCCs of $\mathcal{M}^{\mathbf{M}_k}$ and for each $i \in \{1, \ldots, m_k\}$, let $D_{p,1}^{\mathbf{M}_k} = C_i^{\mathbf{M}_k} \times Q$. Then, $\{D_{p,1}^{\mathbf{M}_k}, \ldots, D_{p,m_k}^{\mathbf{M}_k}\}$ is a partition of $S_p^{\mathbf{M}_k}$. In addition, the following statements hold for all $i, j \in \{1, \ldots, m_k\}$.

- If $D_{p,i}^{\mathbf{M}_k}$ contains a state in an SCC $C_p^{\mathbf{M}_k}$ of $\mathcal{M}_p^{\mathbf{M}_k}$, then it contains all the states in $C_p^{\mathbf{M}_k}$.
- $Succ(D_{p,i}^{\mathbf{M}_k}) \cap D_{p,j}^{\mathbf{M}_k} \neq \emptyset$ only if $C_j^{\mathbf{M}_k} \prec_{\mathcal{M}^{\mathbf{M}_k}} C_i^{\mathbf{M}_k}$.

Applying Lemma 7, we generate a partition $\{D_{p,1}^{\mathbf{M}_k}, \ldots, D_{p,m_k}^{\mathbf{M}_k}\}$ of $S_p^{\mathbf{M}_k}$ where for each $i \in \{1, \ldots, m_k\}$, $D_{p,1}^{\mathbf{M}_k} = C_i^{\mathbf{M}_k} \times Q$ and $C_1^{\mathbf{M}_k}, \ldots, C_{m_k}^{\mathbf{M}_k}$ are all the SCCs of $\mathcal{M}^{\mathbf{M}_k}$. A partial order $\prec_{\mathcal{M}_p^{\mathbf{M}_k}}$ over this partition is defined such that $D_{p,j}^{\mathbf{M}_k} \prec_{\mathcal{M}_p^{\mathbf{M}_k}} D_{p,i}^{\mathbf{M}_{k+1}}$ if $C_j^{\mathbf{M}_k} \prec_{\mathcal{M}^{\mathbf{M}_k}} C_i^{\mathbf{M}_k}$. An order $\mathbb{O}_p^{\mathbf{M}_k}$ among $D_{p,1}^{\mathbf{M}_k}, \ldots, D_{p,m_k}^{\mathbf{M}_k}$ can then be derived from the order of $C_1^{\mathbf{M}_k}, \ldots, C_{m_k}^{\mathbf{M}_k}$, which can be incrementally constructed based on Lemma 4. This order $\mathbb{O}_p^{\mathbf{M}_k}$ has the property that the probability values

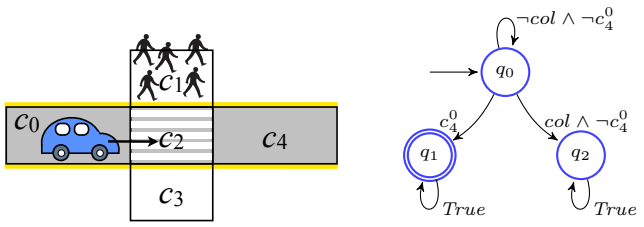

Fig. 1. (left) The road and its partition used in the autonomous vehicle example. (right) A DFA $\mathcal{A}_\varphi$ that recognizes the prefixes of $\varphi = \neg col \; \mathcal{U} \; c_4^0$ where col is defined as $col = \bigvee_{i\geq 1, j\geq 0}(c_j^0 \wedge c_j^i)$. q_1 is the accepting state.

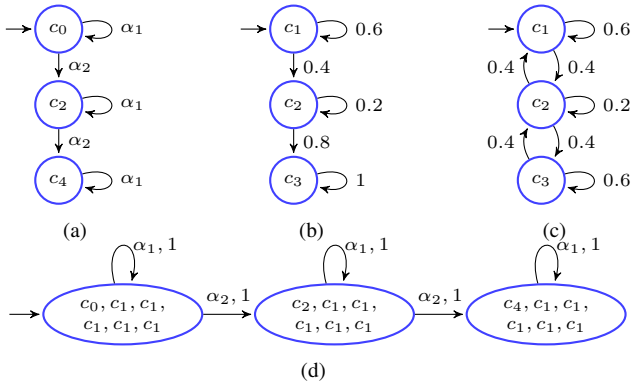

Fig. 2. (a) The vehicle model $\mathcal{T}$. (b) The pedestrian models $\mathcal{M}_1, \ldots, \mathcal{M}_4$. (c) The pedestrian model $\mathcal{M}_5$. (d) $\mathcal{M}^{\mathbf{M}_0}$.

of states in $D_{p,j}^{\mathbf{M}_k}$ that appears after $D_{p,i}^{\mathbf{M}_k}$ in $\mathbb{O}_p^{\mathbf{M}_k}$ cannot affect the probability values of states in $D_{p,i}^{\mathbf{M}_k}$. Hence, we can follow the SCC-based value iteration and process each $D_{p,i}^{\mathbf{M}_k}$ separately, according to the order in $\mathbb{O}_p^{\mathbf{M}_k}$ to compute the probability x_s for all $s \in D_{p,i}^{\mathbf{M}_k}$. Finally, we generate a memoryless control policy $\mathcal{C}^{\mathbf{M}_k}$ from the probability vector $(x_s)_{s\in S_p^{\mathbf{M}_k}}$ as described at the end of Section IV.

VI. Experimental Results

Consider, once again, the autonomous vehicle problem described in Example 1 and Example 2. Suppose the road is discretized into 5 cells $c_0, \ldots, c_4$ where c_2 is the pedestrian crossing area as shown in Figure 1. The vehicle starts in cell c_0 and has to reach cell c_4. There are 5 pedestrians, modeled by MCs $\mathcal{M}_1, \ldots, \mathcal{M}_5$, initially at cell c_1. The models of the vehicle and the pedestrians are shown in Figure 2. A DFA $\mathcal{A}_\varphi$ that accepts all and only words in $pref(\varphi)$ where $\varphi = \left(\neg \bigvee_{i\geq 1, j\geq 0}(c_j^0 \wedge c_j^i)\right) \; \mathcal{U} \; c_4^0$ is shown in Figure 1.

First, we apply the LP-based, value iteration and SCC-based value iteration techniques described in Section IV to synthesize a control policy that maximizes the probability that the complete system $\mathcal{M} = \mathcal{T}||\mathcal{M}_1||\mathcal{M}_2||\ldots||\mathcal{M}_5$ satisfies φ. The time required for each step of computation is summarized in Table I. All the approaches yield the probability of 0.8 that $\mathcal{M}$ satisfies φ under the synthesized control policy. The comparison of the total computation time required for these different approaches is shown in Figure 3. As discussed in Section IV-C, although the SCC-based value iteration itself takes significantly less computation time than the LP-based technique or value iteration, the time spent in identifying SCCs and their order renders the total computation time of the SCC-based value iteration more than the other two approaches.

Technique	Product MDP $\mathcal{M}_p$	SCCs & order of $\mathcal{M}_p$	Prob vector	Control policy	Total
LP	156.3	-	8.8	6.8	171.9
Value iteration	156.3	-	31.3	6.8	194.4
SCC-based	156.3	71.1	1.9	6.8	236.1

TABLE I

Time required (in seconds) for computing various objects when the full models of all the agents are considered.

Next, we apply the incremental technique where we progressively compute a sequence of control policies as more agents are added to the synthesis procedure in each iteration as described in Section V. We let $\mathbf{M}_0 = \emptyset$, $\mathbf{M}_1 = \{\mathcal{M}_1\}$, $\mathbf{M}_2 = \{\mathcal{M}_1, \mathcal{M}_2\}, \ldots$, $\mathbf{M}_6 = \{\mathcal{M}_1, \ldots, \mathcal{M}_5\}$, i.e., we successively add pedestrian $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_5$, respectively, in each iteration. The composition $\mathcal{M}^{\mathbf{M}_0}$ is shown in Figure 2(d). Clearly, $\mathcal{M}^{\mathbf{M}_0}$ has 3 SCCs, each of which contains exactly 1 state and the partial order among these SCCs can be simply defined as $\{(c_3, c_1, c_1, c_1, c_1, c_1)\} \prec_{\mathcal{M}^{\mathbf{M}_0}} \{(c_2, c_1, c_1, c_1, c_1, c_1)\} \prec_{\mathcal{M}^{\mathbf{M}_0}} \{(c_1, c_1, c_1, c_1, c_1, c_1)\}$. $\mathcal{M}^{\mathbf{M}_1}$ then contains 9 states since the state of the first pedestrian can now be anything in $\{c_1, c_2, c_3\}$ (as opposed to $\mathcal{M}^{\mathbf{M}_0}$ where the state of the first pedestrian can only be c_1). Applying Lemma 2, we get that $\mathcal{M}^{\mathbf{M}_1}$ has 9 SCCs, each of which contains exactly 1 state. The partial order among these SCCs can be determined using Lemma 4. For $2 \leq k \leq 6$, $\mathcal{M}^{\mathbf{M}_k}$ and its SCCs can be determined in a similar way.

We consider 2 cases: (1) no incremental construction of various objects is employed (i.e., when $\mathcal{M}^{\mathbf{M}_{k+1}}$ and $\mathcal{M}_p^{\mathbf{M}_{k+1}}$, $k \geq 0$ are computed from scratch in every iteration), and (2) incremental construction of various objects as described in Section V-B–V-D is applied. For the first case, we apply the LP-based technique to compute the probability vector as it has been shown to be the fastest technique when applied to this problem, taking into account the required pre-computation, which needs to be done in every iteration. For both cases, 6 control policies $\mathcal{C}^{\mathbf{M}_0}, \ldots, \mathcal{C}^{\mathbf{M}_5}$ are generated for $\mathcal{M}^{\mathbf{M}_0}, \ldots, \mathcal{M}^{\mathbf{M}_5}$, respectively. For each policy $\mathcal{C}^{\mathbf{M}_k}$, we compute the probability $\Pr_{\mathcal{M}}^{\mathcal{C}^{\mathbf{M}_k}}(\varphi)$ that the complete system $\mathcal{M}$ satisfies φ under policy $\mathcal{C}^{\mathbf{M}_k}$. (Note that $\mathcal{C}^{\mathbf{M}_k}$, when applied to $\mathcal{M}$, is only a function of states of $\mathcal{M}_i \in \mathbf{M}_k$ since it assumes that the other agents $\mathcal{M}_j \notin \mathbf{M}_k$ are stationary.) These probabilities are given by $\Pr_{\mathcal{M}}^{\mathcal{C}^{\mathbf{M}_0}}(\varphi) = 0.08$, $\Pr_{\mathcal{M}}^{\mathcal{C}^{\mathbf{M}_1}}(\varphi) = 0.46$, $\Pr_{\mathcal{M}}^{\mathcal{C}^{\mathbf{M}_2}}(\varphi) = 0.57$, $\Pr_{\mathcal{M}}^{\mathcal{C}^{\mathbf{M}_3}}(\varphi) = 0.63$, $\Pr_{\mathcal{M}}^{\mathcal{C}^{\mathbf{M}_4}}(\varphi) = 0.67$ and $\Pr_{\mathcal{M}}^{\mathcal{C}^{\mathbf{M}_5}}(\varphi) = 0.8$.

The comparison of the cases where the incremental construction of various objects is not and is employed is shown in Figure 3. A jump in the probability occurs each time a new control policy is computed. The time spent during each step of computation is summarized in Table II for both cases. Notice that the time required for identifying the SCCs and their order when the incremental approach is applied is significantly less than when the full model of all the pedestrians is considered in one shot since $\mathcal{M}^{\mathbf{M}_0}$, $\mathcal{M}_1, \ldots, \mathcal{M}_5$, each of which contains 3 states, are much smaller than $\mathcal{M}_p$, which contains 2187 states.

From Figure 3, our incremental approach is able to obtain an optimal control policy faster than any other techniques.

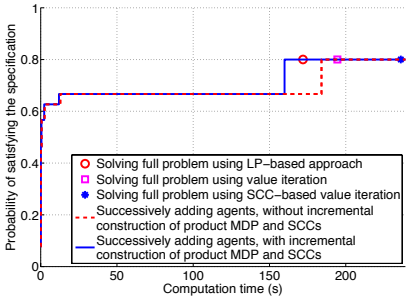

Fig. 3. Comparison of the computation time and the probability for the system to satisfy the specification computed using different techniques.

Iteration	MDP $\mathcal{M}^{\mathbf{M}_k}$	Product MDP $\mathcal{M}_p^{\mathbf{M}_k}$	Prob vector	Control policy	Total
0	0.0064	0.0185	0.0464	0.0084	0.08
1	0.0123	0.0762	0.0203	0.0104	0.12
2	0.0154	0.3383	0.0231	0.0296	0.41
3	0.0357	1.7055	0.0542	0.1503	1.95
4	0.1393	9.1950	0.2155	0.7975	10.35
5	3.1836	152.86	8.2302	6.8938	171.17

Iteration	MDP $\mathcal{M}^{\mathbf{M}_0}$	SCCs & order	Product MDP, partition&order	Prob vector	Control policy	Total
0	0.0055	0.0043	0.0203	0.0112	0.0036	0.04
1	-	-	0.0726	0.0102	0.0087	0.09
2	-	-	0.3239	0.0193	0.0282	0.37
3	-	-	1.6036	0.0567	0.1424	1.80
4	-	-	8.6955	0.1876	0.7755	9.66
5	-	-	139.27	1.6122	7.0125	147.89

TABLE II

TIME REQUIRED (IN SECONDS) FOR COMPUTING VARIOUS OBJECTS IN EACH ITERATION WHEN INCREMENTAL CONSTRUCTION (TOP) IS NOT AND (BOTTOM) IS EMPLOYED.

This is mainly due to the efficiency of our incremental construction of SCCs and their order. In addition, we are able to obtain a reasonable solution, with 0.67 probability of satisfying φ, within 12 seconds while the maximum probability of satisfying φ is 0.8, which requires 160 seconds of computation (or 171.9 seconds without employing the incremental approach).

The advantages provided by the anytime and incremental nature of our approach may be best evaluated through a larger example. For example, with 6 pedestrians, the maximum probability of success is 0.8. Non-incremental approaches take approximately 2 hours to provide a solution, whereas our incremental approach takes approximately 1 hour and 20 minutes to reach the optimal solution. However, exploiting the anytimeness of our approach, we can obtain a solution within 80 seconds with a 0.67 probability of success. Furthermore, if the 6^{th} pedestrian is detected (or appears) toward the end of the computations, the non-incremental approach needs to start from scratch and run for 2 more hours in addition to the time it spent for the first 5 pedestrians. Our incremental approach, on the other hand, can efficiently take the 6^{th} pedestrian into account by exploiting the computation for the first 5 pedestrians.

VII. CONCLUSIONS

An anytime algorithm for synthesizing a control policy for a robot interacting with multiple environment agents with the objective of maximizing the probability for the robot to satisfy a given temporal logic specification was proposed. Each environment agent is modeled by a Markov chain. The robot is modeled by a finite transition system (in the deterministic case) or Markov decision process (in the stochastic case). The proposed algorithm progressively computes a sequence of control policies, taking into account only a small subset of the environment agents initially and successively adding more agents to the synthesis procedure in each iteration until we hit the constraints on computational resources. Experimental results show that our method produces a good solution faster than existing approaches. We are also able to obtain an optimal solution faster than existing approaches.

REFERENCES

[1] E. A. Emerson, "Temporal and modal logic," *Handbook of Theoretical Computer Science (Vol. B): Formal Models and Semantics*, pp. 995–1072, 1990.

[2] Z. Manna and A. Pnueli, *The temporal logic of reactive and concurrent systems*. Springer-Verlag, 1992.

[3] C. Baier and J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[4] G. Fainekos, H. Kress-Gazit, and G. Pappas, "Temporal logic motion planning for mobile robots," in *IEEE International Conference on Robotics and Automation*, pp. 2020–2025, 2005.

[5] H. Kress-Gazit, G. Fainekos, and G. Pappas, "Where's Waldo? Sensor-based temporal logic motion planning," in *IEEE International Conference on Robotics and Automation*, pp. 3116–3121, 2007.

[6] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. Pappas, "Symbolic planning and control of robot motion [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 61–70, 2007.

[7] D. Conner, H. Kress-Gazit, H. Choset, A. Rizzi, and G. Pappas, "Valet parking without a valet," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007*, pp. 572–577, 2007.

[8] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *Proc. of IEEE Conference on Decision and Control*, 2009.

[9] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2689–2696, 2010.

[10] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, "LTL control in uncertain environments with probabilistic satisfaction guarantees," in *IFAC World Congress*, 2011.

[11] A. I. Medina Ayala, S. B. Andersson, and C. Belta, "Temporal logic control in dynamic environments with probabilistic satisfaction guarantees," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007*, pp. 3108–3113, 2011.

[12] H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu, "Correct, reactive robot control from abstraction and temporal logic specifications," *Special Issue of the IEEE Robotics & Automation Magazine on Formal Methods for Robotics and Automation*, vol. 18, pp. 65–74, 2011.

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

[14] T. Wongpiromsarn, A. Ulusoy, C. Belta, E. Frazzoli, and D. Rus, "Incremental temporal logic synthesis of control policies for robots interacting with dynamic agents," tech. rep., 2012. Available at `http://arxiv.org/abs/1203.1180`.

[15] T. Latvala, "Efficient model checking of safety properties," in *Proceedings of the 10th international conference on Model checking software*, SPIN'03, (Berlin, Heidelberg), pp. 74–88, Springer-Verlag, 2003.

[16] F. Ciesinski, C. Baier, M. Größer, and J. Klein, "Reduction techniques for model checking markov decision processes," in *Proceedings of the 2008 Fifth International Conference on Quantitative Evaluation of Systems*, pp. 45–54, 2008.

[17] M. Kwiatkowska, D. Parker, and H. Qu, "Incremental quantitative verification for markov decision processes," in *IEEE/IFIP International Conference on Dependable Systems & Networks*, pp. 359–370, 2011.

[18] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, pp. 146–160, 1972.