A Compositional Approach to Reactive Games under Temporal Logic Specifications

Abhishek Ninad Kulkarni and Jie Fu

***Abstract*— We study the problem of compositional synthesis of controllers for reactive games with linear temporal logic (LTL) specifications. A reactive game is an abstraction of the interaction between a controllable system and its uncontrolled and dynamic environment. A centralized control design for such systems under complex specifications can be computationally expensive. Instead, a compositional approach aims to synthesize a controller for a complex specification by composing the solutions for its component sub-specifications. This mitigates the issue of scalability and has the advantages of being modular and flexible. This paper solves the problem of reactive game synthesis using the compositional approach in two steps. First, we use the notion of randomized permissive strategy to reduce the strategy synthesis problem to that of identifying only the winning region for the controlled agent against the uncontrolled environment. Then, we exploit the inherent compositional nature of LTL formulas to compose the independently computed winning regions of two sub-games into a superset of the composed-game winning region. We make use of elementary set operations to construct this superset. Finally, we introduce an iterative algorithm to extract the exact winning region from the superset. We prove the correctness of our proposed method and illustrate the solution using a toy-problem and a robot motion planning example.**

I. Introduction

The interactions between a robot and its dynamic, uncontrolled environment can be captured as a two-player, zero-sum game [1]. For such a game, the reactive synthesis aims to compute a winning strategy, or equivalently a controller, for the robot with respect to a given temporal logic specification against all admissible behaviors of its environment. Recently, reactive synthesis has been investigated intensively for systems that interact with open, dynamic, and uncontrollable environments. The synthesis of such controllers has been explored for various applications such as hardware design [2], control of autonomous robotic systems [3–5], and under various classes of temporal logic constraints such as linear temporal logic, μ-calculus, and signal temporal logic [6]. In addition, several software tools [7–9] have been developed to facilitate the design process.

In spite of the recent advances, several challenges remain open in the current framework. On one hand, for complex temporal logic constraints, reactive synthesis suffers from the problem of scalability because the size of the game grows doubly-exponential in the size of the specification [1]. On the other hand, most practical systems are required to undergo several iterations of specification revision during the design process. When a task specification is composed of several sub-specifications and the synthesis cannot find a controller, it is often expensive to identify the conflicting specifications leading to the unrealizability of the overall specification.

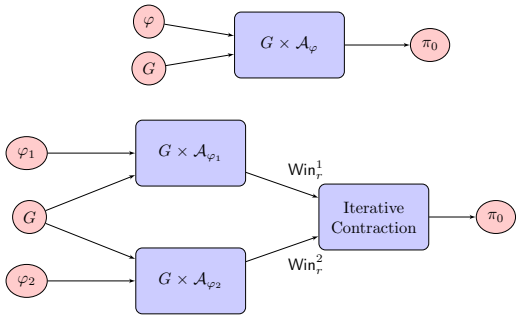

Fig. 1. Centralized Solution vs. Compositional Solution for $\varphi = \varphi_1 \wedge \varphi_2$

To address these challenges, we investigate the compositional synthesis approach in this work. For a global task specification composed of several sub-specifications, $\varphi = \varphi_1 \wedge .. \wedge \varphi_n$, the solution by compositional synthesis first solves for the controllers of the individual sub-specifications $\varphi_i,\ i = 1..n$ and then composes them into a single controller that satisfies φ. We propose a method of compositional synthesis that only uses elementary set-theoretic operations. First, we consider a class of Linear Temporal Logic (LTL) specification represented as a conjunction of sub-specifications, which can be equivalently translated to a deterministic finite-state automaton (DFA). Then, for each sub-specification, a reactive controller is synthesized by constructing a *two-player, zero-sum game*, which is a product of the specification automaton and the transition system [10]. Then, with the insight from the synthesized controllers and the intersection product of sub-specification automata, we introduce a method to construct the winning region of the global reactive game by composing the winning regions of the games constructed from sub-specifications. Finally, we show how to extract a controller from the constructed winning region for the global specification using randomized permissive strategies. Figure 1 pictorially compares the proposed solution with the centralized solution.

In the literature, compositional synthesis has been investigated in [11], where the authors propose a compositional synthesis for multi-agent decoupled systems with local spec-

J. Fu is with Faculty of Electrical and Computer Engineering, Robotics Engineering Program, Worcester Polytechnic Institute, Worcester, MA. 01609, USA jfu2@wpi.edu

A. Kulkarni is with Robotics Engineering Program, Worcester Polytechnic Institute, Worcester, MA. 01609, USA ankulkarni@wpi.edu

ifications. However, our problem formulation is different. We assume that the global specification is represented as a conjunction of sub-specification (or alternatively in its conjunctive normal form) and show that the centralized solution can be composed of the solutions of this system with respect to sub-specifications. In [12], the authors propose a compositional framework for treating multiple linear-time objectives inductively and synthesis in a cascade manner: Namely, given $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_n$, a controller is first synthesized for the system with task φ_1, and then φ_2 is enforced onto the system that satisfies φ_1, and so on. While this approach manages to reduce the required number of computations, it does not fully address the problem of scalability. This is because the composition operation is still sequential. Close to our formulation and solution approach is the reference [13] that provides a compositional algorithm that approximates the specification using k-co-Büchi Automata, from which the strategy can be synthesized by solving for the corresponding safety games that have a partial order structure. Comparing with the aforementioned approaches, the proposed method in this paper can be used to compose any two sub-specifications with each other independently and in any arbitrary order. This allows the composition to be done in a completely parallel manner, thus providing a significant computational advantage.

The rest of the paper is organized as follows: Section II provides the necessary preliminaries in the centralized reactive synthesis using game theoretic solutions. Section III presents the main algorithm for the compositional reactive synthesis for an LTL formula with two sub-specifications and its proof of correctness. Section IV validates the correctness of our method with a toy example and a robot motion planning example. Section V concludes the paper and discusses our future work.

II. Preliminaries

Given two sets S and A, a set-valued function $f : S \to 2^A$ is defined for $s \in S$ if $f(s) \neq \emptyset$.

Reactive system as a game: A deterministic *game arena* capturing the interaction between a robot and its dynamic environment is

$$GTS = \langle S, A, T, s_0, \mathcal{AP}, L \rangle$$

where the components are defined as follows.

- $S = S_r \cup S_e$ is the set of states. At each state in S_r, the robot takes an action and at each state in S_e, the environment takes an action.
- $A = A_r \cup A_e$ is the set of actions. For a given state $s \in S$, the set $A(s)$ represents the set of the *enabled* actions from state s.
- $T : S \times A \to S$ is the deterministic transition function.
- $\mathcal{AP}$ is a set of atomic propositions.
- $s_0 \in S$ is the initial state of the game.
- $L : S \to 2^{\mathcal{AP}}$ is the labeling function.

A game arena is called *turn-based* if the two players always take turns in playing the game. By the inclusion of empty action λ, a turn-based game arena can capture the case when one player performs multiple actions before the other player acts. A game arena is said to be reachable if any state $s \in S$ is reachable with a sequence of actions from the initial state. In this work, we consider only the turn-based and reachable game arenas. Although concurrent actions are not considered in the scope of this work, it turns out a large class of reactive systems can be captured as turn-based games, such as reactive robot motion planning [3], nonlinear systems with exogenous disturbance [4].

Game arena is also referred to as a game transition system or a reactive system [14]. A run in GTS is a finite sequence $\rho = s_0 s_1 \ldots s_n \in Q^*$ of states, where $*$ represents the Kleene star, or an infinite sequence $\rho = s_0 s_1 ... \in S^\omega$, such that s_0 is the initial state and for all $i \geq 0$ there exists $a_i \in A$, $T(s_i, a_i) = s_{i+1}$. As mentioned earlier, this paper considers the specifications that can be represented by a DFA, which accepts finite runs. The prefix u of a run ρ is a finite sequence of states such that there exists a run w and $\rho = uw$. The set of prefixes of all runs in the game arena G is denoted $\mathsf{Pref}(G)$.

Strategy: For both the players, the robot and the environment, a *deterministic* strategy for the player is a function $\pi_i : S^* \to A_i \cup \{\perp\}$, where $i \in \{r,\ e\}$ and $\perp$ means undefined. For example, for any run $\rho = s_0 s_1 \ldots s_n$ that ends up with environment's state $s_n \in S_e$, the strategy $\pi_r(\rho) = \perp$. A *randomized* strategy for a player is a function $\pi_i : S^* \times (A_i \cup \{\perp\}) \to [0, 1]$. That is, given the run ρ for which the strategy π_i is defined, the strategy π_i outputs a probability distribution over action set A_i.

Specification: We consider system specifications in the form of LTL formulas. Formally, the set of LTL formulas over a finite set $\mathcal{AP}$ of atomic propositions can be defined inductively as follows:

- Any atomic proposition $p \in \mathcal{AP}$ is an LTL formula.
- If ϕ and ψ are LTL formulas, so are $\neg\phi$, $\phi \wedge \psi$, $\bigcirc \phi$ and $\phi \,\mathsf{U}\, \psi$ where $\bigcirc$ and U are temporal modal operators for "next" and "until".

Additional temporal logic operators include $\Diamond$ (eventually) and $\Box$ (always), defined by $\Diamond \phi := \mathsf{true}\,\mathsf{U}\,\phi$ and $\phi := \neg\Diamond\neg\phi$.

Particularly, we consider a subset of LTL formulas φ that can be equivalently translated into a DFA $\mathcal{A}_\varphi = \langle Q, 2^{\mathcal{AP}}, \delta, I, F \rangle$ where Q is a finite state set, $2^{\mathcal{AP}}$ is the alphabet and $\mathcal{AP}$ is a set of atomic propositions, $I \in Q$ is the initial state, and $\delta : Q \times 2^{\mathcal{AP}} \to Q$ the transition function. The acceptance condition F is a set of final states. The run for an finite word $w = w[0]w[1] \ldots \in (2^{\mathcal{AP}})^*$ is a finite sequence of states $q_0 q_1 \ldots q_n$ where $q_0 = I$ and $q_{i+1} = \delta(q_i, w[i])$. A run $\rho = q_0 q_1 \ldots$ is accepting in $\mathcal{A}_\varphi$ if $\mathsf{Occ}(\rho) \cap F \neq \emptyset$ where $\mathsf{Occ}(\rho)$ is the set of states that occurs in the run ρ. With this winning condition, we can make all the final states F as sink states. That is, for any $q \in F$ and for any $\sigma \in 2^{\mathcal{AP}}$, $\delta(q, \sigma) = q$.

The product game: Given a game transition system GTS and a DFA, $\mathcal{A}_\varphi$, as described above, the product game is given by $G = GTS \times \mathcal{A}_\varphi = \langle \tilde{S}, A, \Delta, \tilde{s}_0, \tilde{F} \rangle$ with components defined as follows:

- $\tilde{S} = S \times Q$ is the set of states, partitioned into the set $\tilde{S}_r = S_r \times Q$ of the robot's states, and the set $\tilde{S}_e = S_e \times Q$ of the environment's states.
- The map $\Delta : \tilde{S} \times A \to \tilde{S}$ is the transition function, defined as $\Delta((s,q),a) = (s',q')$ with $s' = T(s,a)$ and $q' = \delta(q, L(s'))$.
- The initial state is $\tilde{s}_0 = (s_0, q_0)$ where $s_0 = \Delta(I, L(s_0))$.
- The set $\tilde{F} = \{(s,q) \mid q \in F\}$ specifies the winning condition.

A run $\rho \in \tilde{S}^*$ is winning for the robot if $\mathsf{Occ}(\rho) \cap \tilde{F} \neq \emptyset$. A run $\rho \in \tilde{S}^\omega$ is winning for the environment if $\mathsf{Occ}(\rho) \cap \tilde{F} = \emptyset$.

Given the product game, a *sure winning* strategy for the robot is a *memory-less, deterministic* strategy $\pi_r : \mathsf{Win}_r \to A_r$ where $\mathsf{Win}_r \subseteq \tilde{S}_r$ is the set of states from which the robot can force a win. This winning set is computed using the attractor set computation defined as a fix-point construction as

- $\mathsf{Win}_r^0 = \tilde{F}$
- $\mathsf{Win}_r^k = \mathsf{Win}_r^{k-1} \cup$
  $\{s \in \tilde{S}_r \mid \exists \alpha \in A(s) : \Delta(s,\alpha) \in \mathsf{Win}_r^{k-1}\} \cup$
  $\{s \in \tilde{S}_e \mid \forall \alpha \in A(s) : \Delta(s,\alpha) \in \mathsf{Win}_r^{k-1}\}$

The complement of Win_r with respect to $\tilde{S}$ is the set Win_e of winning states for the environment. For any state $\tilde{s} \in \mathsf{Win}_r$, by following the strategy π_r, the robot is ensured to win the game G.

A. Problem statement

For a specification represented as a conjunction of sub-specifications, $\varphi = \varphi_1 \wedge .. \wedge \varphi_n$, if n is small, then the centralized solution may be sufficient to decide the winning strategies and winning regions for players. However, for large n, the doubly-exponential blow-up of the state-space makes the centralized solution to be impractical to used in robotics applications. To mitigate this issue, we propose to use a compositional approach. Namely, if the specification is decomposed into a conjunction of sub-specifications, then we obtain the solutions independently for each sub-game constructed from the game arena and a sub-specification, and then compose the winning strategies and regions in a meaningful way so as to ensure that the overall specification is satisfied. Formally, our problem is stated as follows:

Problem 1. *Let G be a reactive game and let $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_n$ be an LTL specification represented as a conjunction of $n \geq 2$ formulas. Let Win_r^k, π_r^k represent the winning region and winning strategy for the robot in k-th sub-game. Similarly, let Win_e^k, π_e^k represent the corresponding winning region and strategy for the environment. If Win_r, π_r represent the robot's winning strategy and strategy in the game for global specification φ, then how to represent Win_r, π_r in terms of Win_r^k, π_r^k, Win_e^k and π_e^k for $k = 1 \ldots n$?*

Henceforth in this paper, we shall refer to the games corresponding to the sub-specifications, φ_k, as the sub-games of the global game, which corresponds to the global specification, φ.

III. Parallel synthesis under LTL constraints

In this section, we present a solution to Problem 1. In our solution, we first show how to use the *randomized permissive* strategies to reduce the problem of determining the winning region and strategy to the problem of only computing the winning region. Then we show how to construct a superset of the global winning region using the winning regions and strategies for two sub-games. Finally, we introduce a fix-point method to extract the global winning region from this superset without having to construct the global game.

A. Computing permissive strategies through randomization

We recall the synthesis algorithm for a reactive game as given in [14]: For a game $G = \langle \tilde{S}, A, \Delta, \tilde{s}_0, \tilde{F} \rangle$, the winning region can be partitioned into the level-sets as $\mathsf{Win}_r = \bigcup_{i=0}^{m} X_i$ and a corresponding deterministic winning strategy is given by $\pi_r : \mathsf{Win}_r \to A_r$. Given a state $\tilde{s} \in \mathsf{Win}_r$, there exists a unique ordinal i such that $\tilde{s} \in X_i$. If $\tilde{s} \in \tilde{S}_r \cap X_i$ for some $i > 0$, then after applying $\pi_r(\tilde{s})$, the robot reaches a state in X_{i-1}. If $\tilde{s} \in \tilde{S}_e \cap X_i$, regardless of the action the environment takes, the next state is in X_{i-1}. The game terminates when $\tilde{s} \in X_0$, which equals $\tilde{F}$.

Theorem 1. *Given a game, $G = \langle \tilde{S}, A, \Delta, \tilde{s}_0, \tilde{F} \rangle$, let $\mathsf{Win}_r \subseteq \tilde{S}$ be the winning region for the robot. Given a randomized strategy $\pi_r : \mathsf{Win}_r \times A_r \to [0,1]$, satisfying the following conditions: For all $\alpha \in A_r$, $\pi_r(\tilde{s},a) > 0$ if and only if $\Delta(\tilde{s},a) \in \mathsf{Win}_r$. then, by following the strategy π_r, the robot is ensured to win the game with probability **one**.*

Proof. By following this randomized strategy π_r, the robot is ensured to stay within Win_r, regardless of how the environment acts. Moreover, in each state $\tilde{s} \in \tilde{S}_r \cap X_i$, there is a strictly positive probability to reach X_{i-1}. Thus, the probability of reaching X_0 as the number of steps approaches infinity is 1. Because, for any number n, the probability that the state has not reached X_j for $j < i$ is not more than $(1-\gamma)^n < 1$ as $\gamma = \min_{(\tilde{s},a) \in \tilde{S} \times A} \pi_r(\tilde{s},\alpha) \in (0,1)$. Therefore, the probability of eventually visiting X_0, which equals $\tilde{F}$, is ensured to be 1. □

This theorem enables us to obtain a winning strategy only with the knowledge of the winning region and the game transition function. Thus, the problem of synthesizing a strategy that satisfies a conjunction of temporal logic constraints can be reduced to the problem of deciding the winning region for the robot under these constraints. This understanding is fundamental to our proposed solution to compositional synthesis. As we will show that the winning region for the robot given a conjunction $\varphi = \varphi_1 \wedge \varphi_2$ of LTL sub-specifications can be computed in a compositional manner, by solving for the winning regions of the sub-games G_1 and G_2.

B. Composition of the winning regions and strategies

Given $\varphi = \varphi_1 \wedge \varphi_2$, we construct the sub-game $G_i = GTS \times \mathcal{A}_{\varphi_i} = (\tilde{S}^i, A, \Delta^i, \tilde{s}_0^i, \tilde{F}^i)$ for $i = 1, 2$. To solve for a meaningful composition of the winning regions of sub-games G_1 and G_2, we analyze the set inclusion of the winning regions. We note that the game corresponding to global specification is equal to the game defined by computing the product of game transition system and the intersection product of sub-specification automata, i.e. $GTS \times A_\varphi = GTS \times (A_{\varphi_1} \otimes A_{\varphi_2})$, where $\otimes$ represents the intersection product operation of two DFAs [15].

Note that we assume all the specification automata in the consideration to be *complete*. That is, for any state and action pair, the transition is defined. An *incomplete* automaton can be completed by introducing a *sink* state and directing all undefined transitions to the sink state [15].

Next, we will use elementary set operations on the sub-game winning regions and strategies to construct a superset of the global winning region. Let Win_r^1, Win_r^2 denote the robot's winning regions in sub-games G_1 and G_2. Similarly, Win_e^1, Win_e^2 denote the corresponding environment's winning regions. Let Win_r, Win_e denote the winning regions for the robot and the environment in the global game. Then we define the following two sets,

$$\begin{aligned}\mathsf{Win}_r^\wedge &= \{(s, q_1, q_2) \in \tilde{S}_r \mid (s, q_1) \in \mathsf{Win}_r^1 \wedge (s, q_2) \in \mathsf{Win}_r^2\} \\ \mathsf{Win}_e^\vee &= \{(s, q_1, q_2) \in \tilde{S}_e \mid (s, q_1) \in \mathsf{Win}_e^1 \vee (s, q_2) \in \mathsf{Win}_e^2\}\end{aligned} \tag{1}$$

Intuitively, the $\mathsf{Win}_r^\wedge$ captures all the robot's states in the global game such that the corresponding sub-game states are both winning for the robot. Similarly, $\mathsf{Win}_e^\vee$ captures all the environment's states in the global game from which the environment can force a win in either of the sub-games. define a set Win_r^{12} as follows:

$$\mathsf{Win}_r^{12} = \mathsf{Win}_r^\wedge \cup (\tilde{S}_e \setminus \mathsf{Win}_e^\vee)$$

Lemma 1. *Given* $\mathsf{Win}_r^\wedge$ *and* $\mathsf{Win}_e^\vee$ *defined in* (1)*, let the set* Win_r^{12} *be defined as,*

$$\mathsf{Win}_r^{12} = \mathsf{Win}_r^\wedge \cup (\tilde{S}_e \setminus \mathsf{Win}_e^\vee)$$

Then, the set Win_r^{12} *is a superset of the winning region* Win_r*.*

$$\mathsf{Win}_r^{12} \supseteq \mathsf{Win}_r$$

Proof. For a state $\tilde{s} = (s, q_1, q_2) \in \mathsf{Win}_r^\wedge$, at which the robot makes a move, it is ensured that there exist two winning strategies π_r^1 and π_r^2 for the robot in the games G_1 and G_2, respectively. When $(s, q_2) \xrightarrow{\pi_r^1(s,q_1)} (s', q_2') \notin \mathsf{Win}_r^2$, or, $(s, q_1) \xrightarrow{\pi_r^2(s,q_2)} (s', q_1') \notin \mathsf{Win}_r^1$, then we cannot ensure that $\tilde{s} \in \mathsf{Win}_r$. In other words, it is possible that the winning strategy in one sub-game makes a losing move for other.

On the other hand, for a state $\tilde{s} = (s, q_1, q_2) \in \mathsf{Win}_r \cap \tilde{S}_r$, a winning strategy π_r exists to ensure winning in both the games G_1 and G_2. Thus, witnessed by the winning strategy π_r, $(s, q_1) \in \mathsf{Win}_r^1$ and $(s, q_2) \in \mathsf{Win}_r^2$. Therefore $\tilde{s} \in \mathsf{Win}_r^\wedge$. Thus, we have $\mathsf{Win}_r \cap \tilde{S}_r \subseteq \mathsf{Win}_r^\wedge$.

For a state $\tilde{s} = (s, q_1, q_2) \in \mathsf{Win}_e^\vee$, which is a state when the environment makes a move, it is ensured that there exists at least one winning strategy π_e^i to ensure the specification $i \in \{1, 2\}$ is not satisfied. For $\tilde{s} \in \tilde{S} \setminus \mathsf{Win}_e^\vee$, the robot could ensure to win either sub-game G_1 or G_2. However, $\tilde{s}$ may not be in Win_r because there may not exist a strategy to ensure all specifications are satisfied. On the other hand, taking a state $\tilde{s} \in \mathsf{Win}_r \cap \tilde{S}_e$, it is ensured that no matter how the environment behaves, there exists a strategy for the robot to satisfy $\varphi_1 \wedge \varphi_2$ and hence $\tilde{s} \in \tilde{S}_e \setminus \mathsf{Win}_e^\vee$. □

C. Iterative construction of superset to global winning region

In this section, we introduce an iterative method to eliminate the spurious states that are contained in Win_r^{12} but not in Win_r. Let π_r^1, π_r^2 represent the deterministic sub-game winning strategies for the robot. We define the progress set as the set of all the one-step reachable states from a given state (s, q_1, q_2), such that the robot makes progress in at least one sub-game while staying inside the winning region for the other. The progress set for a sub-game i can be represented as,

$$\begin{aligned}&\mathsf{prog}^i(s, q_1, q_2; V) = \{(s, q_1, q_2) \in \tilde{S}_r \mid \\ &\exists \alpha \in A(s),\ (s, q_1, q_2) \xrightarrow{\alpha} (s', q_1', q_2') \in V \wedge \\ &((s', q_j') \in \mathsf{Win}_r^j \wedge \alpha \in \pi_r^i(s, q_i))\}\end{aligned}$$

where $V \subseteq \mathsf{Win}_r^{12}$, and $(i, j) \in \{(1, 2), (2, 1)\}$. We define

$$\begin{aligned}\mathsf{prog}(s, q_1, q_2; V) = \mathsf{prog}^1(s, q_1, q_2; V) \cup \\ \mathsf{prog}^2(s, q_1, q_2; V)\end{aligned}$$

Using this definition, we define the iterative construction of Win_r^{12} as follows

1) $V^0 = \mathsf{Win}_r^{12}$
2) $\bar{V} = \{(s, q_1, q_2) \in V^k \mid s \in S_r \wedge \mathsf{prog}(s, q_1, q_2, V^k) = \emptyset\} \cup \{(s, q_1, q_2) \in V^k \mid s \in S_e \wedge \exists \alpha \in A(s)\ \texttt{s.t.}\ \Delta((s, q_1, q_2), \alpha) \notin V^k\}$
3) $V^{k+1} = V^k \setminus \bar{V}$
4) Repeat (2), (3) until the fix-point $V^{m+1} = V^m$ is reached.

Informally, the algorithm starts with the set Win_r^{12}, which is a superset of the winning region Win_r. Then, it iteratively removes the states where the winning action for one game means losing for another sub-game or the states from where the environment can take the transition to a state outside of the winning region computed at the iteration k. We note that during each iteration, V^k is a superset of Win_r. Therefore, a state outside of V^k is certainly losing for the robot. Next, we prove this formally.

Lemma 2. *For each iteration* k*,* $V^k \supseteq \mathsf{Win}_r$*.*

Proof. By induction.

Base case: For $k = 0$, by using Lemma 1 we have $V^0 = \mathsf{Win}_r^{12} \supseteq \mathsf{Win}_r$.

Induction step: Assume for iteration k, $V^k \supseteq \mathsf{Win}_r$. We need to show that, at the iteration $k+1$, $V^{k+1} \supseteq \mathsf{Win}_r$, or equivalently show that $\bar{V} \cap \mathsf{Win}_r = \emptyset$.

By contradiction, suppose there exists $\tilde{s} = (s, q_1, q_2) \in \bar{V} \cap \mathsf{Win}_r$. Then we have two cases,

Case a: $\exists \alpha \in A_e, \Delta(\tilde{s}, \alpha) \notin V^k$*:* Here, as $V^k \supseteq \mathsf{Win}_r$, the action α of the environment leads to a state outside of the winning region Win_r. Thus, $\tilde{s} \notin \mathsf{Win}_r$ — a contradiction.

Case b: $\mathsf{prog}(\tilde{s}; V^k) = \emptyset$*:* The progress set can be empty if and only if for all $\alpha \in A(s)$, if $(s, q_1, q_2) \xrightarrow{\alpha} (s', q_1', q_2') \in V$, then both $((s', q_1') \notin \mathsf{Win}_r^1 \vee \alpha \notin \pi_r^2(s, q_2))$ and $(\alpha \notin \pi_r^1(s, q_1) \vee (s', q_2') \notin \mathsf{Win}_r^2)$ are true. For $(s', q_1') \notin \mathsf{Win}_r^1 \vee \alpha \notin \pi_r^2(s, q_2)$ to hold true, either the robot is losing in first game if action $\pi_r^2(s, q_2) = \alpha$ is applied, or the environment can indefinitely block the robot for making any progress in the second game. Similar reasoning can be done for the progress set of sub game 2. Therefore, we require that $\tilde{s} \notin \mathsf{Win}_r$ — a contradiction. □

Theorem 2. *The fixed point $V = V^{m+1} = V^m$ satisfies*

$$V = \mathsf{Win}_r.$$

Proof. The proof is by construction. First, we denote $\{X_i^k \mid i = 1, \ldots, m^k\}$ the level sets of sub-game k with respect to the winning region Win_r^k, for $k = 1, 2$ and m^k is the total number of level sets.

We consider a (class of) randomized strategies $\pi^* : (V \cap \tilde{S}_r) \times A_r \to [0, 1]$ defined as follows: $\pi^*(\tilde{s}, \alpha) > 0$ for all $\alpha \in \mathsf{prog}(\tilde{s}; V)$ and $\pi^*(\tilde{s}, \alpha) = 0$ if $\alpha \notin \mathsf{prog}((\tilde{s}; V)$.

Given a state $\tilde{s} \in \tilde{S}_r$ where the robot makes a move, let $(s, q_1) \in X_i^1$ and $(s, q_2) \in X_j^2$, for some $0 < i \leq m^1$ and $0 < j \leq m^2$. It is clear that every move made by the robot ensures that for the next state $\tilde{s}' = \Delta(\tilde{s}, \alpha)$ with $\alpha \in \mathsf{prog}(\tilde{s}; V)$, either $(s', q_1') \in X_{i-1}^1$ or $(s', q_2') \in X_{j-1}^2$. In other words, the robot will make progress in at least one of the sub-games. Suppose that $\alpha = \pi_r^1(s, q_1)$, then the game arrives at (s', q_1'). By definition of the winning strategy for the robot, it is guaranteed that for any action β of the environment, the next state $(s'', q_1'') = \Delta^1((s', q_1'), \beta) \in X_{i-2}^1$, when $i > 2$ or $(s'', q_1'') \in S \times F_1$ when the robot wins the sub-game 1. A similar argument can be derived for the sub-game 2.

Now, let's consider the case when during a finite run $\rho = \tilde{s}_0 \ldots \tilde{s}_n \in \tilde{S}^*$, for any $\tilde{s}_i \in S_r \times F_1 \times Q_2$ where $i = 0, \ldots, n$, it holds that $\mathsf{prog}^1(\tilde{s}_i; V) \neq \emptyset$ and $\mathsf{prog}^2(\tilde{s}_i; V) = \emptyset$. Thus, by the above reasoning, the robot is guaranteed to reach one of the states in $S \times F_1 \times Q_2$ for some finite step $n \leq m^1$ and win the sub-game 1. After reaching $\tilde{s}_n = (s, q_1, q_2)$ with $q_1 \in F_1$, the state (s, q_2) is ensured to be within Win_r^2 for sub-game 2 by definition of Win_r^i and prog^i, for $i = 1, 2$. Thus, by exercising the winning strategy in sub-game 2, the robot is then ensured to visit $S \times F_1 \times F_2$ in the game $G \times \mathcal{A}_\varphi$ because all states in F_1 are sink states in the automaton. The above argument shows that it is impossible to only win one sub-game but not the other. That is to say, any state $\tilde{s} \in (S_r \times Q_1 \times Q_2) \cap V$ has to be in Win_r. For the environment's state $\tilde{s} \in (S_e \times Q_1 \times Q_2) \cap V$, it is ensured that the next state stays within V and is a state in $S_r \times Q_1 \times Q_2$. Since we have show for all states in $S_r \times Q_1 \times Q_2 \cap V$ is winning for the robot, the state $\tilde{s} \in V$ is winning for the robot too. Given the state $\tilde{s}$ is arbitrarily picked, we have all states in $S_e \times Q_1 \times Q_2 \cap V$ is winning for the robot, which means $V \subseteq \mathsf{Win}_r$. Together with Lemma 2, it holds that $V = \mathsf{Win}_r$. □

Remark: So far, we have presented the method and proof of pairwise composition. To realize the composition of n controllers for n subspecifications, we can use pairwise composition between n subspecifications, and then employ a tree structure between subspecifications to synthesize a controller for the global policy. For instance, given $\varphi = \varphi_1 \wedge \varphi_2 \wedge \varphi_3$, a pairwise composition will provide controllers for $\varphi_{12} \coloneqq \varphi_1 \wedge \varphi_2$ and $\varphi_{23} \coloneqq \varphi_2 \wedge \varphi_3$. The winning region for the global specification is composed again with pairwise construction from the winning regions of subgames with φ_{12} and φ_{23}. Another approach is to modify the progress set computation so as to capture safe actions in multiple subgames instead of one. In this manner, the composition can be done in one shot. However, this one shot composition may not be desirable if there are possible conflicting subspecifications, which could have been detected if we use the pairwise composition approach.

IV. Experiments

We illustrate the working of our proposed method using a toy problem and a robot motion planning problem below. The first problem is a handcrafted game with eight states. We demonstrate the construction of Win_r^{12} from sub-game winning regions and strategies and show that it is a superset of the global winning region Win_r. Then, we explain the iterative construction process step-by-step to show how it removes all the spurious states.

In the second problem, we solve a robot motion planning problem in the presence of an adversarial robot using the proposed method and the centralized method. The implementation is scripted using Python on a laptop with Intel(R) Core(TM) i7 processor and 16GB of RAM. The specification automata are generated using Spot 2 tool [16].

A. Toy problem

Consider the game arena as shown in Figure 2. The game is a two player game with the robot states drawn in circles and the environment states drawn as rectangles. The action associated with each transition is shown as edge label in Figure 2. Therefore, the actions for the robot and the environment are given by $A_r = \{01, 02, 30, 35, 40, 41, 46, 70, 77\}$, $A_e = \{13, 15, 24, 51, 53, 57, 67\}$ respectively.

Next, we consider the following objective for the robot, *visit state 7* and *visit state 5* in any sequence. This is represented as compositional LTL formula $\varphi = \varphi_1 \wedge \varphi_2$, where $\varphi_1 = \Diamond(s = 7)$ and $\varphi_2 = \Diamond(s = 5)$ with equivalent automaton as shown in Fig. 3. The automaton for φ is shown in Fig. 4.

By solving for the individual sub-games and the global game we have the sets as shown below. We note that for every

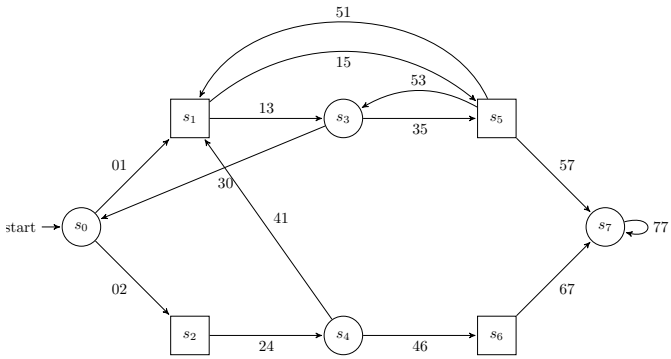

Fig. 2. 8-State Game Graph

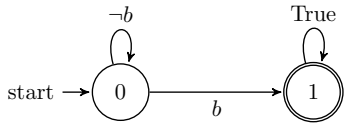

Fig. 3. Automata for LTL Formula $\Diamond a$

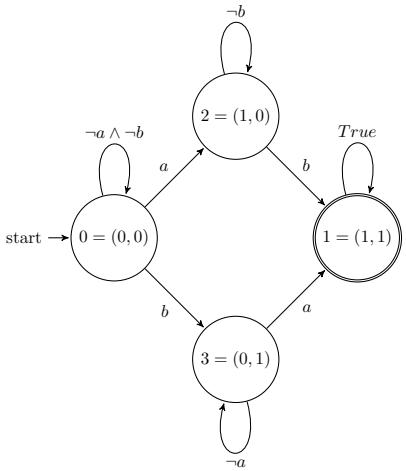

Fig. 4. Automata for LTL Formula $\Diamond a \wedge \Diamond b$

winning state in Win_r, we can find a unique state in each of Win_r^1 and Win_r^2, whose composition defines the winning state. For example, consider the state $(2,1,0) \in \mathsf{Win}_r$, we have $(2,1) \in \mathsf{Win}_r^1$ and $(2,0) \in \mathsf{Win}_r^2$.

$$\begin{aligned}\mathsf{Win}_r^1 = \{&(0,1),(1,1),(2,1),(3,1),(4,1),(5,1),(6,1),\\ &(7,1),(0,0),(2,0),(3,0),(4,0),(6,0)\}\end{aligned}$$

$$\begin{aligned}\mathsf{Win}_e^1 &= \{(1,0),(5,0)\}\\ \mathsf{Win}_r^2 &= \{(0,1),(1,1),(2,1),(3,1),(4,1),\\ &\quad (5,1),(6,1),(7,1),(0,0),(1,0),\\ &\quad (2,0),(3,0),(4,0),(5,0),(6,0)\}\\ \mathsf{Win}_e^2 &= \emptyset\\ \mathsf{Win}_r &= \{(0,0,0),(2,0,0),(3,0,0),(4,0,0),(6,0,0),\\ &\quad (0,1,0),(1,1,0),(2,1,0),(3,1,0),(4,1,0),\\ &\quad (6,1,0),(7,1,0),(0,0,1),(2,0,1),(3,0,1),\\ &\quad (4,0,1),(6,0,1),(0,1,1),(1,1,1),(2,1,1),\\ &\quad (3,1,1),(4,1,1),(5,1,1),(6,1,1),(7,1,1)\}\end{aligned}$$

Next, we compute the $\mathsf{Win}_r^\wedge$ and $\mathsf{Win}_e^\vee$ sets as

$$\begin{aligned}\mathsf{Win}_r^\wedge = \{&(0,1,0),(3,1,1),(4,0,0),(7,1,0),(0,1,1),\\ &(4,1,0),(7,1,1),(3,0,1),(4,1,1),(0,0,1),\\ &(3,0,0),(0,0,0),(4,0,1),(3,1,0)\}\\ \mathsf{Win}_e^\vee = \{&(1,0,1),(5,0,1),(1,0,0)\}\end{aligned}$$

On constructing the set Win_r^{12}, we observe that it has 4 spurious states that are not present in Win_r, marked in boldface.

$$\begin{aligned}\mathsf{Win}_r^{12} = \{&(7,1,1),(4,0,1),(3,0,1),(0,0,1),(2,0,1),\\ &(6,0,1),(\mathbf{0},\mathbf{0},\mathbf{0}),(\mathbf{2},\mathbf{0},\mathbf{0}),(\mathbf{3},\mathbf{0},\mathbf{0}),(\mathbf{4},\mathbf{0},\mathbf{0})\} \quad (2)\\ \mathsf{Win}_r = \{&(7,1,1),(4,0,1),(3,0,1),(0,0,1),(2,0,1),\\ &(6,0,1)\} \quad (3)\end{aligned}$$

Now let us apply the iterative solution to contract Win_r^{12} into Win_r. Consider the first iteration, $k = 1$, for the state $(s, q_1, q_2) = (4,0,0)$ with $V^0 = \mathsf{Win}_r^{12}$. We see that $\pi_r^1(4,0) = \{46\}$ and $\pi_r^2(4,0) = \{41\}$. Therefore, $\mathsf{prog}(4,0,0;V^0) = \emptyset$ and $V^1 = V^0 \setminus \{(4,0,0)\}$.

In the next iteration, $k = 2$, we observe that for the environment, we have $(2,0,0) \xrightarrow{24} (4,0,0) \notin V^1$. This results in $(2,0,0) \in \bar{V}$ and thus getting eliminated. Similarly, we observe that the robot states $(0,0,0)$ and $(3,0,0)$ get eliminated at $k = 3,4$ respectively, because the actions 02 and 30 that are in non-empty progress set, can no longer keep the robot inside V. Finally, we reach the fix point at $k = 5$ with $V^5 = V^4 = \mathsf{Win}_r$.

B. Robot motion planning

Consider a 5×5 grid world with a controlled robot, an uncontrolled robot, and static obstacles. We refer to the static obstacles and the uncontrolled robot as the environment of the controlled robot, henceforth referred to as the robot. In a given turn, the robot can move into any of the 8-connected adjacent cells. The environment can only make a move into its 4-connected neighbors, horizontally or vertically. The Fig. 5 shows the grid world with regions of interest in green and obstacle regions in red.

The Fig. 5 shows four regions marked as R_1, R_2, R_3, R_4. The problem requires the robot to synthesize a controller to satisfy the objective $\varphi = \varphi_1 \wedge \varphi_2$, where $\varphi_1 = \Diamond(R_1 \vee R_2) \wedge \Box(\neg obs)$ and $\varphi_2 = \Diamond(R_3 \wedge \Diamond R_4) \wedge \Box(\neg obs)$. The first sub-specification requires the robot to visit the regions R_1 *or* R_2 without colliding with any obstacles, while the second sub-specification requires the robot to visit the region R_3 *first and then the region* R_4.

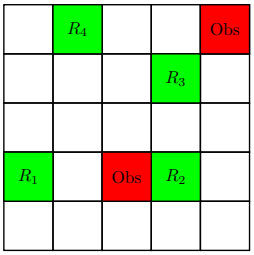

Fig. 5. Grid-world Environment

	Centralized		Compositional			
	φ		φ_1		φ_2	
	Size (states)	Time (sec)	Size (states)	Time (sec)	Size (states)	Time (sec)
Product Game	7500	7.206	2500	1.032	3750	1.865
Attractor Set	4825	0.423	2450	0.106	2450	0.207
Iterative Construction	-	-	0.821 sec			

TABLE I
COMPARISON BETWEEN CENTRALIZED AND COMPOSITIONAL

We start by computing the product sub-games $G_1 = GTS \times \mathcal{A}_{\varphi_1}$ and $G_2 = GTS \times \mathcal{A}_{\varphi_2}$ and the global game $G = GTS \times \mathcal{A}_{\varphi}$. It is straightforward to see that the GTS has $5 \cdot 5 \cdot 5 \cdot 5 \cdot 2 = 1250$ states. Correspondingly, $|G_1| = 1250 \cdot 2 = 2500$ and $|G_2| = 1250 \cdot 3 = 3750$. For the global game, the size of state space is $|G| = 1250 \cdot 6 = 7500$.

With the naïve implementation of the algorithms, the centralized synthesis solution took 7.629 seconds to compute the strategy for φ while the proposed method took 4.031 seconds. Furthermore, it is possible to solve for the sub-game strategies in parallel, which might reduce the computation time further to 2.893 seconds. The Table I shows the size and time required for each step of the centralized solution and the proposed method. The iterative construction process requires two iterations to eliminate all the spurious states from the Win_r^{12} set. The winning regions of the composed and global game have the equal size.

V. DISCUSSION AND CONCLUSION

The proposed algorithm for composing the strategies of two sub-games runs in a polynomial time in the size of sub-games, and linear in the number of sub-specifications. Furthermore, the iterative construction is linear in the size of the constructed superset. However, a major advantage is that the approach does not construct the global game, and enables a modular, compositional, and flexible reactive synthesis approach. A modification in a sub-specification does not necessitate reconstruction of the entire game from scratch and then solving it again. We can reuse the previously computed solutions of the other sub-games and combine them with the solution of the new sub-game to obtain the solution of the reactive system with its updated specification. This enables us to develop an efficient verification and synthesis toolbox for several practical systems, for example, mobile robots, whose specifications are not complete and constantly revised given the newly discovered constraints and ad-hoc tasks.

Furthermore, the extension to specifications given by Büchi automata is non-trivial as the intersection of Büchi automata needs to include auxiliary information in the states, instead of a direct Cartesian product of the state spaces of the automata. This extension and the development of a toolbox are our ongoing work.

REFERENCES

[1] A. Pnueli and R. Rosner, "On the synthesis of a reactive module," in *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM, 1989, pp. 179–190.

[2] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive (1) designs," *Journal of Computer and System Sciences*, vol. 78, no. 3, pp. 911–938, 2012.

[3] H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu, "Correct, reactive, high-level robot control," *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, pp. 65–74, 2011.

[4] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, "Synthesis of reactive switching protocols from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 58, no. 7, pp. 1771–1785, 2013.

[5] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[6] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, 2015, pp. 239–248.

[7] C. Finucane, G. Jing, and H. Kress-Gazit, "Designing reactive robot controllers with ltlmop," in *Proceedings of the 9th AAAI Conference on Automated Action Planning for Autonomous Mobile Robots*. AAAI Press, 2011, pp. 70–71.

[8] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "Tulip: a software toolbox for receding horizon temporal logic planning," in *Proceedings of the 14th international conference on Hybrid systems: computation and control*. ACM, 2011, pp. 313–314.

[9] R. Ehlers and V. Raman, "Slugs: Extensible gr (1) synthesis," in *International Conference on Computer Aided Verification*. Springer, 2016, pp. 333–339.

[10] E. Filiot, N. Jin, and J.-F. Raskin, "Compositional algorithms for ltl synthesis," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2010, pp. 112–127.

[11] R. Alur, S. Moarref, and U. Topcu, "Compositional synthesis of reactive controllers for multi-agent systems," in *International Conference on Computer Aided Verification*. Springer, 2016, pp. 251–269.

[12] C. Baier, J. Klein, and S. Klüppelholz, "A compositional framework for controller synthesis," in *International Conference on Concurrency Theory*. Springer, 2011, pp. 512–527.

[13] E. Filiot, N. Jin, and J.-F. Raskin, "Antichains and compositional algorithms for ltl synthesis," *Formal Methods in System Design*, vol. 39, no. 3, pp. 261–296, 2011.

[14] W. Thomas *et al.*, *Automata, logics, and infinite games: a guide to current research*. Springer Science & Business Media, 2002, vol. 2500.

[15] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of model checking*. MIT press, 2008.

[16] A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, and L. Xu, "Spot 2.0a framework for ltl and \omega -automata manipulation," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2016, pp. 122–129.