Submitted, 2017 Conference on Decision and Controlhttp://www

Decomposing GR(1) Games with Singleton Liveness Guarantees for Efficient Synthesis

Sumanth Dathathri Richard M. Murray

***Abstract*— Temporal logic based synthesis approaches are often used to find trajectories that are correct-by-construction in systems–eg. synchronization for multi-agent hybrid systems, reactive motion planning for robots. However, the scalability of such approaches is of concern and at times a bottleneck when transitioning from theory to practice. In this paper, we identify a class of problems in the GR(1) fragment of linear-time temporal logic (LTL) where the synthesis problem allows for a decomposition that enables easy parallelization. This decomposition also reduces the alternation depth, resulting in more efficient synthesis. A multi-agent robot gridworld example with coordination tasks is presented to demonstrate the application of the developed ideas and also to perform empirical analysis for benchmarking the decomposition-based synthesis approach.**

I. Introduction

Robot motion planning has traditionally focused on generating trajectories from a given initial state to a final goal position. Recently, there has been increased attention towards generating trajectories from high-level temporal logic specifications to ensure correct behavior in terms of synchronization of processes, safety and scheduling. The case of reactive robot motion planning from logic specifications involves considering complex behaviors for an adversarial environment and reasoning about all admissible behaviors for the environment to generate a plan for the robot. Temporal logic is often employed in this setting for reactive motion planning when we wish to generate motion plans that may exhibit complex behavior but are provably correct.

In particular, we focus on LTL [15], [16], a modal temporal logic often used as the mathematical language to formally specify the desired behavior for the robot [2], [8]. Synthesizing finite-memory strategies from LTL specifications for the general case is doubly exponential in the length of the formula [17], but for *Generalized Reactivity (1)* (GR (1))–a rich, expressive fragment of LTL, the synthesis can be done in polynomial time in the number of states and the number of liveness guarantees for the system and the number of liveness assumptions for the adversary [11]. GR(1) specifications model a game where the system and its adversary infinitely often satisfy a set of liveness constraints while making moves that satisfy certain safety constraints. This fragment in particular has received considerable attention since its conception because of the computational tractability associated with it. The GR(1) fragment is also particularly attractive because of the symbolic nature of the synthesis algorithms, that enables scaling to large finite-transition systems.

The complexity of synthesis for this class of temporal logic scales is cubic or quadratic [3] depending on the algorithms used for computing the fixed point. In our work, we identify a special subclass of GR(1) where we can provably decompose the synthesis problem into several smaller independent synthesis problems. During the decomposition procedure, we also eliminate one of the nested fixed points to reduce the *alternation depth*, thereby improving performance. In the context of motion planning, this class corresponds to finding paths that visit a set of nodes in a graph infinitely often, where each of the nodes in the graph satisfies a separate liveness condition. When an adversary is present, the relevant reactive motion planning problem is that of performing coordinated tasks for multi-agent systems [5]. Informally, the motion planning problem is to find a path for the robot such that when the environment is in pose $\mathcal{X}$, the robot has to be in pose $\mathcal{Y}$ where $(\mathcal{X}, \mathcal{Y})$ completes the coordination task. Figure 1 depicts an instance of the problem when the robot has to find a plan for its motion such that when the agent it is coordinating with has a pose E, it has to attain pose C. When the environment is at pose D and pose B, the robot must be at pose C and pose F respectively. Assuming some behavior of the agent, the robot must (if feasible) find a reactive trajectory such that the agent-robot system visits the poses (E, C), (D, C) and (B, F) infinitely often. For example,

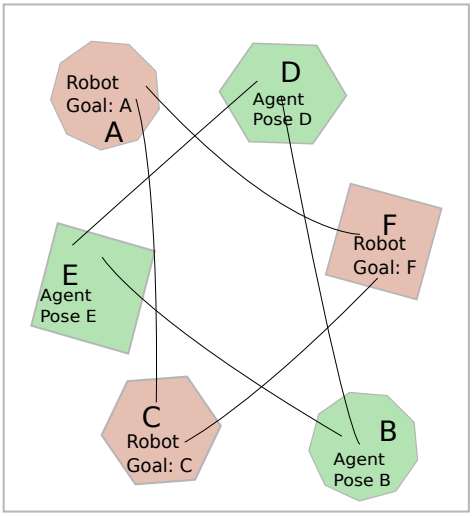

Fig. 1: Robot motion planning

in [10], the authors propose an intermittent communication framework for mobile robot networks where the intermittent communication requirement is captured through an LTL formula that enforces the robots to meet infinitely often at certain rendezvous points.

The issue of scalability for synthesis algorithms has re-

ceived attention in the past. In [1], the specification is compositionally decomposed to allow for scalable synthesis, with refinements being made to the decomposed specifications based on counter-strategies and strategies for the system and the environment. In [14], the GR(1) synthesis problem is decomposed into checking a feasibility problem and an online short-horizon strategy generation problem to improve scalability. In [18], the authors improve the scalability of controller synthesis by restricting themselves to linear systems and safety constraints, thereby enabling a reduction of the problem to the computation of control invariant sets. The main contribution here is to identify a class of GR(1) synthesis problems where the problem can be decomposed into smaller *reachability games*. The decomposed subgames are independent, allowing for the parallelization of the synthesis process.

II. Preliminaries

In this section, we introduce game structures and μ-calculus ([12]) over game structures as in [3] before presenting our contribution.

A. Game Structures and μ-calculus

A *game structure* $G = \langle \mathrm{AP}, \mathrm{AP}_{\mathrm{env}}, \mathrm{AP}_{\mathrm{sys}}\, \theta^{\mathrm{env}}, \theta^{\mathrm{sys}}, \rho^{\mathrm{sys}}, \rho^{\mathrm{env}}, \varphi \rangle$ represents a two player game where

- $\mathrm{AP} := \{v_1, \ldots, v_n\}$: A finite set of atomic propositions,
- $\mathrm{AP}_{\mathrm{env}} \subseteq \mathrm{AP}$: Set of propositions controlled by the environment,
- $\mathrm{AP}_{\mathrm{sys}} \subseteq \mathrm{AP}$: Set of atomic propositions controlled by the system. Also, $\mathrm{AP}_{\mathrm{env}} \cap \mathrm{AP}_{\mathrm{sys}} = \emptyset$.
- $\theta^{\mathrm{sys}}, \theta^{\mathrm{env}}$: Boolean formulas in $\mathrm{AP}_{\mathrm{sys}}$ and $\mathrm{AP}_{\mathrm{env}}$ characterizing the system and environment initial states,
- $\Sigma := 2^{\mathrm{AP}}$,
- ρ^{env} is a formula in the propositions $\mathrm{AP}, \mathrm{AP}_{\mathrm{env}}$. An input $x \subseteq \mathrm{AP}_{\mathrm{env}}$ is a valid input at a state $s \in \Sigma$ if $(s, x) \models \rho^{\mathrm{env}}$,
- ρ^{sys} is a formula in the propositions $\mathrm{AP}, \mathrm{AP}_{\mathrm{env}}, \mathrm{AP}_{\mathrm{sys}}$ that specifies the system transition rules. A system output/action $y \subseteq \mathrm{AP}_{\mathrm{sys}}$ is possible if $(s, x, y) \models \rho^{\mathrm{sys}}$.

To simplify notation, we extend the semantics of LTL for finite strings. For a finite string γ in Σ^*, we define:

$$\gamma \models \rho \Leftrightarrow \gamma\alpha \models \rho \text{ for any } \alpha \in \Sigma^{\omega}. \tag{1}$$

This allows for reasoning about states that might deadlock and playing games whose winning conditions allow for finite strings. For example, consider the formula $\Diamond\rho$. In accordance with the semantics of LTL, the language of this formula consists of strings in Σ^{ω} for which $\sigma_k \models \rho$ for some finite k. But, if we wanted to express the behavior for a motion planning problem where we are interested in only reaching a goal and not the behavior beyond, the extended semantics allow for that. According to the extended semantics of LTL for finite strings, the language of the above formula consists of strings in $\Sigma^{\omega} \cup \Sigma^*$ such that for σ in the language of the above formula, there exists a finite k such that $\sigma_k \models \rho$.

For a Boolean formula ξ, denote by $[[\xi]] \subseteq \Sigma$ the set of states that satisfy ξ. Additionally, given a game-structure G and a Boolean formula ψ, we define the relevant μ-calculus operator $\textcircled{\Diamond}$ as:

$$[[\textcircled{\Diamond}\psi]] = \{s \in \Sigma | \forall x \in \mathcal{P}(\mathrm{AP}_{\mathrm{env}}), (s,x) \models \rho^{\mathrm{env}} \rightarrow \exists y \in \mathcal{P}(\mathrm{AP}_{\mathrm{sys}}).(s,x,y) \models \rho^{\mathrm{sys}} \wedge (x,y) \models \psi\}.$$

where $\mathcal{P}(\mathrm{AP}_{\mathrm{env}})$ denotes the power set of $\mathrm{AP}_{\mathrm{env}}$. $[[\textcircled{\Diamond}\psi]]$ characterises the states from which the system can force the next state to satisfy ψ for any valid input. In this paper, we use the subscript notation, e.g., $\sigma_0\sigma_1\sigma_2\cdots\sigma_n \in \Sigma^*$, noting that infinite strings can also be regarded as functions of the natural numbers $\mathbb{N}$ into Σ. For a finite string γ, by $\gamma_{:,r}$ we refer to the the string $\gamma_0\gamma_1\ldots\gamma_{r-1}$. By γ_{-1} we refer to the last element of γ. In other words, $\gamma_{-1} = \gamma_{|\gamma|-1}$.

B. Definitions

Let M be a finite set of memory values with $m^i \in M$ set to mark the initial memory value. A partial function $f : M \times \Sigma \times \mathcal{P}(\mathrm{AP}_{\mathrm{env}}) \rightarrow M \times \mathcal{P}(\mathrm{AP}_{\mathrm{sys}})$ is a finite-memory *strategy* for the game G if for all (w, s, x) for which $f(w, s, x)$ is defined, the condition $(s, x) \models \rho^{\mathrm{env}} \rightarrow (s, x, y) \models \rho^{\mathrm{sys}}$ holds.

A *play* $\sigma \in \Sigma^{\omega} \cup \Sigma^*$ for a strategy f is the maximimal sequence of states such that $\exists m \in M^{\omega}$ with $m_0 = m^i$ such that $(m_{k+1}, \sigma_{k+1} \cap \mathrm{AP}_{\mathrm{sys}}) = f(m_k, \sigma_k, \sigma_{k+1} \cap \mathrm{AP}_{\mathrm{env}})$ and $\sigma_k\sigma_{k+1} \models \rho^{\mathrm{env}}$ for $|\sigma| \geq k - 1 \geq 0$ if $\sigma \in \Sigma^*$ and $k \geq 0$ if $\sigma \in \Sigma^{\omega}$. By maximimal sequence, we imply that the play terminates when we reach a state where the strategy is not defined for a valid input or the environment deadlocks i.e. there is no valid input. We denote by $\mathrm{Plays}(f)$ the set of all plays generated by f. Also, define the set $\mathrm{Pref}(f) \in \Sigma^*$ as:

$$\mathrm{Pref}(f) := \{\sigma \in \Sigma^* | \exists \sigma\gamma \in \mathrm{Plays}(f).\gamma \in \Sigma^{\omega} \cup \Sigma^*\}.$$

Let us denote by $m^{\sigma,f}$ the sequence of memory values generate by f corresponding to $\sigma \in \mathrm{Plays}(f) \cup \mathrm{Pref}(f)$ with $m_0^{\sigma,f} = m^i$. Given a strategy $f : M \times \Sigma \times \mathcal{P}(\mathrm{AP}_{\mathrm{env}}) \rightarrow M \times \mathcal{P}(\mathrm{AP}_{\mathrm{sys}})$, we define the set of reachable state memory pairs from an initial condition $\theta = \theta^{\mathrm{sys}} \wedge \theta^{\mathrm{env}}$ and an initial memory value m^i:

$$\{(w,s) | \exists j : (w,s) = (m_j^{\sigma,f}, \sigma_j) :\ \sigma \in \mathrm{Plays}(f), \sigma_0 \models \theta, m_0^{\sigma,f} = m^i\}.$$

For a state s in Σ, a strategy f is *winning* if

$$\forall \sigma \in \ \mathrm{Plays}(f).(\sigma_0 = s \rightarrow \sigma \models \varphi), \tag{2}$$

$$f(m_{-1}^{\sigma,f}, \sigma_{-1}, x) \text{ is defined } \forall \sigma \in \mathrm{Pref}(f) \text{ with } \sigma_0 = s, \forall x \in \mathrm{AP}_{\mathrm{env}} \text{ with } \sigma_{-1}x \models \rho^{\mathrm{env}}. \tag{3}$$

with φ being the winning condition. Note that from the above definition the system always has a well-defined action according to the winning strategy f on starting from a winning state, as long as the environment does not violate the safety assumption. Dually, we define a winning state $s \in \Sigma$. A state s is a winning state against a condition φ if there exists a strategy f that is winning from that state. The winning set is the maximal set of winning states from which there exists a strategy f that is winning. We use W_{φ} to denote the set of winning states associated with a formula φ.

C. Generalized Reactivity(1)

A game structure G with a winning condition of the form

$$\varphi := \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}} \rightarrow \bigwedge_{j=1}^{n} \square\diamond\psi_j^{\text{sys}} \tag{4}$$

is an instance of a *Generalized Reactivity(1)* game where ψ_i^{env} and ψ_j^{sys} are Boolean formula in AP.

Problem 1: For a given game structure G, the *GR(1) synthesis* problem is to find a finite memory strategy f that is winning for the set of states satisfying the initial condition θ, against the condition φ (as in equation (4)).

The worst case complexity for GR(1) synthesis in general scales as $\mathcal{O}\left((nm|\Sigma|)^3\right)$ [11]. Using the approach in [4], a GR(1) game can be solved with $\mathcal{O}(nm|\Sigma|^2)$ *next step* computations. However, computing the *next steps* takes more time than simply $\mathcal{O}(1)$.

For the case when all the liveness formulae ψ_i^{sys} are such that the sets $[[\psi_i^{\text{sys}}]]$ are singletons i.e there exists exactly one $s \in \Sigma$ for each ψ_i^{sys} such that $s \models \psi_i^{\text{sys}}$, we propose an approach to decompose the original GR(1) games into $n+1$ independent smaller subgames. Solving each smaller subgame involves solving a μ-calculus formula with a smaller alternation depth of 2. The *alternation depth* of a formula is the number of alternations in the nesting of least and greatest fixpoints.

III. Decomposition for Singleton Liveness Guarantees

A. Reachability Games

A *reachability* game is a game G with a winning condition of the form

$$\varphi_{\text{rg}} := \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}} \rightarrow \diamond\psi^{\text{sys}}. \tag{5}$$

$L(\varphi_{rg}) = \{\sigma : \sigma \in \Sigma^* \cup \Sigma^\omega, \exists$ finite k such that $\sigma_k \models \psi^{\text{sys}}$ OR $\sigma \in \Sigma^\omega$ such that $\sigma \models \bigvee_{i=1}^{m} \diamond\square\neg\psi_i^{\text{env}}\}$.

Remark 1: For a game structure G, the winning states for a reachability game can be computed by solving a μ-calculus formula with an alternation depth of 2.
This holds as a direct consequence of Lemma 9 from [11]. Consider the μ-calculus formula μ_{rg} defined as:

$$\mu_{\text{rg}} := \mu Y \left(\bigvee_{j=1}^{m} \nu X \left(\left((\psi^{\text{sys}} \vee \bigcirc Y) \vee \neg\psi_j^{\text{env}} \right) \wedge \bigcirc X \right) \right). \tag{6}$$

Here, ν is the greatest fixpoint operator and μ is the least fixpoint operator (See [19] for detailed definitions of these operators). The alternation depth for this formula is 2. Intuitively, the fixed point in X characterizes the set of states from which the system can force the play to stay indefinitely in $[[\neg\psi_j^{\text{env}}]]$ for some j or in a finite number of steps reach a state satisfying $\psi^{\text{sys}} \vee \bigcirc Y$. Staying in $[[\neg\psi_j^{\text{env}}]]$ indefinitely implies blocking the environment from satisfying one of its liveness assumptions. The outer least fixed point in Y makes sure that the phase of play represented by $\bigcirc Y$ eventually ends in $[[\psi^{\text{sys}}]]$. This way either $\diamond\psi^{\text{sys}}$ is satisfied or $\bigvee_{i=1}^{m} \diamond\square\neg\psi_i^{\text{env}}$ is satisfied. These fixed points can be computed with complexity $\mathcal{O}\left((m|\Sigma|)^2\right)$[7]. The approach in [4] results in $\mathcal{O}(m|\Sigma|^2)$ *next step* computations to solve for the fixed points.

B. Generalized Reactivity (1) Games

In this section, we identify a special class of $GR(1)$ synthesis problems where the winning strategy can be computed by solving $n+1$ reachability games instead of solving the cyclic μ-calculus formula with an alternation depth of 3.

For the case with two liveness guarantees, the μ-calculus formula in [11] can be written using the *vector notation* as:

$$\mu_\varphi = \nu \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \begin{bmatrix} \mu Y \left(\bigvee_{j=1}^{m} \nu X \left(\left((\psi_1^{\text{sys}} \wedge \bigcirc Z_2) \vee \bigcirc Y \vee \neg\psi_j^{\text{env}} \right) \wedge \bigcirc X \right) \right) \\ \mu Y \left(\bigvee_{j=1}^{m} \nu X \left(\left((\psi_2^{\text{sys}} \wedge \bigcirc Z_1) \vee \bigcirc Y \vee \neg\psi_j^{\text{env}} \right) \wedge \bigcirc X \right) \right) \end{bmatrix} \tag{7}$$

The μ-calculus formula in equation (7) has an alternating depth of 3. We propose and prove an algorithm to solve for the winning states and extract a strategy by solving independent reachability-games, each involving solving a μ-calculus formula with an alternation depth of 2.

A GR(1) game with n liveness guarantees is decomposed into $n+1$ reachability games when $|[[\psi_i^{\text{sys}}]]| = 1$ for each i. The reachability games are independent, unlike the cyclic dependency in equation (7) between the various liveness guarantees. The outermost fixed point computation in Z_i can be avoided here as the liveness guarantees correspond to singleton sets and this allows for the separation of the subgames (we prove this later). Here, for example, the GR(1) game can be split into three reachability games:

$$\begin{aligned} &\bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}} \rightarrow \diamond\psi_1^{\text{sys}}, \\ &\bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}} \rightarrow \diamond\psi_2^{\text{sys}}, \\ &\bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}} \rightarrow \diamond \texttt{False}, \end{aligned} \tag{8}$$

with the initial conditions $\theta_1 = \psi_2^{\text{sys}}$, $\theta_2 = \psi_1^{\text{sys}} \vee \theta$ and $\theta_0 = \theta$ for the reachability games respectively (recall θ is the initial condition for the original synthesis problem). In general, for a problem with n liveness constraints, the reachability games can be set-up as for $j \in \{1, 2, \ldots, n\}$:

$$\bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}} \rightarrow \diamond\psi_{j\oplus1}^{\text{sys}} \tag{9}$$

with the initial conditions being $\theta_j = \psi_j^{\text{sys}}$ for $j \neq n$ and $\theta_j = \psi_j^{\text{sys}} \vee \theta$ for $j = n$. We shall refer to the winning condition $\bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}} \rightarrow \diamond\psi_k^{\text{sys}}$ as φ_k^{reach}. Additionally,

define φ_0^{reach} as

$$\bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}} \rightarrow \diamond \texttt{False} \tag{10}$$

and the initial condition for this game is θ. Note that for any given state, a strategy that is winning against this condition can only do so by forcing the play to block the environment from satisfying its assumptions.

Define $\bar{\varphi}$ as the following formula:

$$\bar{\varphi} := \bigwedge_{i=1}^{m} \psi_i^{\text{env}} \rightarrow \diamond\psi_1^{\text{sys}} \wedge \left(\bigwedge_{i=1}^{n} \diamond \left(\psi_i^{\text{sys}} \rightarrow \diamond\psi_{i\oplus 1}^{\text{sys}} \right) \right). \tag{11}$$

Claim 2: $W_\varphi = W_{\bar{\varphi}}$ if $|[[\psi_i^{\text{sys}}]]| = 1 \, \forall i \in \{1, 2 \ldots n\}$.

Proof: First we show $W_\varphi \subseteq W_{\bar{\varphi}}$. Let f_G^{φ} be a winning strategy for the condition φ for the set W_φ. We show that f_G^{φ} is winning for the condition $\bar{\varphi}$ for the set W_φ, thereby proving that $W_\varphi \subseteq W_{\bar{\varphi}}$.

Consider $\sigma \in \text{Plays}(f_G^{\varphi})$ such that $\sigma_0 \in W_\varphi$. By definition,

$$\sigma \models \left(\bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}} \rightarrow \bigwedge_{j=1}^{n} \square\diamond\psi_j^{\text{sys}} \right).$$

If $\sigma \models \bigvee_{i=1}^{m} \diamond\square\neg\psi_i^{\text{env}}$, then $\sigma \models \bar{\varphi}$ directly. In the other case, $\sigma \models \left(\bigwedge_{j=1}^{n} \square\diamond\psi_j^{\text{sys}} \right)$ has to hold. For this case, the semantics of the $\diamond$ operator imply that there exists $k_1, k_2, \ldots, k_n$ that are finite such that $\sigma_{k_i} \models \psi_i^{\text{sys}}$ ($\square\diamond\psi_i^{\text{sys}}$ implies $\diamond\psi_i^{\text{sys}}$ has to hold at every step along a run). For each such finite k_i, $\forall j \in \{1, 2, \ldots, n\} \exists h_{ij} > k_i . \sigma_{h_{ij}} \models \psi_j^{\text{sys}}$. This implies $\diamond(\psi_i^{\text{sys}} \wedge \diamond\psi_{i\oplus 1}^{\text{sys}})$ holds for each i. Therefore, $\sigma_0 \in W_{\bar{\varphi}}$ and hence $W_\varphi \subseteq W_{\bar{\varphi}}$.

Now, let us show $W_{\bar{\varphi}} \subseteq W_\varphi$. $W_{\bar{\varphi}}$ is the winning set for $\bar{\varphi}$. By definition of winning sets there exists a winning strategy $f_G^{\bar{\varphi}}$ that is winning against $\bar{\varphi}$ for every element of $W_{\bar{\varphi}}$. Also, $W_{\bar{\varphi}}$ is not an empty set if the system can win for $\bar{\varphi}$ from any state. If $W_{\bar{\varphi}}$ is an empty set, $W_{\bar{\varphi}} \subseteq W_\varphi$ is trivially true.

To prove that $W_{\bar{\varphi}} \subseteq W_\varphi$, we construct a new strategy $\bar{f}$ that is winning against the condition φ for all states in $W_{\bar{\varphi}}$. This way we show that every state in $W_{\bar{\varphi}}$ is winning against φ and hence in W_φ.

Consider some play $\bar{\sigma}$ of $f_G^{\bar{\varphi}}$ such that $\bar{\sigma} \models \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}}$ and $\bar{\sigma}_0 \in W_{\bar{\varphi}}$. If no such play exists, then for all plays of $f_G^{\bar{\varphi}}$, the condition $\neg \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}}$ holds and the strategy $f_G^{\bar{\varphi}}$ is winning for φ because all plays of $f_G^{\bar{\varphi}}$ satisfy φ.

Consider the case when such a play exists. $\diamond\psi_1^{\text{sys}}$ holds implying that at some finite k, $\bar{\sigma}_{k:} \models \psi_1^{\text{sys}} \wedge \diamond\psi_{1\oplus 1}^{\text{sys}}$ holds. Denote the smallest k at which $\bar{\sigma}_k \models \psi_1^{\text{sys}} \wedge \diamond\psi_{1\oplus 1}^{\text{sys}}$ as k_1. By a similar reasoning, we can go on to define $k_1, k_2, \ldots, k_n$. Next, we introduce a variable $\mathcal{Z}_n$ that can take values in $\{1, 2, \ldots, n\}$ and tracks which of the liveness guarantees have been satisfied. $\mathcal{Z}_n$ is initialized to 1. The strategy

$$\bar{f} : (M \times \{1, 2, \ldots, n\}) \times \Sigma \times \mathcal{P}(\text{AP}_{\text{env}}) \rightarrow (M \times \{1, 2, \ldots, n\}) \times \mathcal{P}(\text{AP}_{\text{sys}})$$

is constructed as

$$\bar{f}((w, \mathcal{Z}_n), s, s' \cap \text{AP}_{\text{env}}) = ((w', \mathcal{Z}_n'), s' \cap \text{AP}_{\text{sys}}),$$

where if $s \models \psi_{\mathcal{Z}_n}^{\text{sys}}$,

$$(w', s' \cap \text{AP}_{\text{env}}) = f_G^{\bar{\varphi}}(m_{k_{\mathcal{Z}_n}}^{\bar{\sigma}, f_G^{\bar{\varphi}}}, s, s' \cap \text{AP}_{\text{env}}),$$
$$\mathcal{Z}_n' = \mathcal{Z}_n \oplus 1,$$

and if $s \not\models \psi_{\mathcal{Z}_n}^{\text{sys}}$,

$$(w', s' \cap \text{AP}_{\text{env}}) = f_G^{\bar{\varphi}}(w, s, s' \cap \text{AP}_{\text{env}}),$$
$$\mathcal{Z}_n' = \mathcal{Z}_n.$$

a) Showing well-definedness for all relevant inputs: For any reachable state-memory pair (s, w) of $f_G^{\bar{\varphi}}$ and any input $x \in \mathcal{P}(\text{AP}_{\text{env}})$, $f_G^{\bar{\varphi}}(w, s, x)$ is defined if $(s, x) \models \rho^{\text{env}}$ (since $f_G^{\bar{\varphi}}$ is winning for $\bar{\varphi}$). For the case when (s, w) is reachable, then $f(w, s, x)$ is also reachable if $sx \models \rho^{\text{env}}$. This implies that when $s \not\models \psi_{\mathcal{Z}_n}^{\text{sys}}$ if $ss' \models \rho^{\text{env}}$ and (s, w) is reachable, then $(w', s') = \bar{f}(w, s, s' \cap \text{AP}_{\text{env}})$ is reachable.

Consider a state s such that $s \models \psi_{\mathcal{Z}_n}^{\text{sys}}$, $s = \bar{\sigma}_{k_{\mathcal{Z}_n}}$ because $\bar{\sigma}_{k_{\mathcal{Z}_n}} \models \psi_{\mathcal{Z}_n}^{\text{sys}}$ and $[[\psi_{\mathcal{Z}_n}^{\text{sys}}]]$ is a singleton. If $s = \bar{\sigma}_{k_{\mathcal{Z}_n}}$, then $f_G^{\bar{\varphi}}(m_{k_{\mathcal{Z}_n}}^{\bar{\sigma}, f_G^{\bar{\varphi}}}, s, x)$ is defined $\forall x \in \mathcal{P}(\text{AP}_{\text{env}}) . sx \models \rho^{\text{env}}$. This is because $(s, m_{k_{\mathcal{Z}_n}}^{\bar{\sigma}, f_G^{\bar{\varphi}}})$ is reached during the execution $\bar{\sigma} \in Plays(f_G^{\bar{\varphi}})$. Therefore, for any $s \models \psi_{\mathcal{Z}_n}^{\text{sys}}$, $\bar{f}(m_{k_{\mathcal{Z}_n}}^{\bar{\sigma}, f_G^{\bar{\varphi}}}, s, (.))$ is well-defined for all valid environmental inputs and $(s, m_{k_{\mathcal{Z}_n}}^{\bar{\sigma}, f_G^{\bar{\varphi}}})$ is reachable for $f_G^{\bar{\varphi}}$.

Additionally, we begin execution for the first input at an initial memory value $m^i \in M$. For a valid initial state $s \in W_{\bar{\varphi}}$ and the initial memory value m^i, (s, m^i) is reachable for $f_G^{\bar{\varphi}}$. To summarize, we start at a reachable state-memory pair for $f_G^{\bar{\varphi}}$.

We showed that for any reachable state-memory pair (s, w) of $f_G^{\bar{\varphi}}$, $\bar{f}$ is well-defined for all valid environmental inputs. We also showed that the output for this case is a reachable state-memory pair (for $f_G^{\bar{\varphi}}$) if the environmental input is valid. Additionally, we also start at a reachable state-memory pair. Therefore, for any $\sigma \in \text{Pref}(\bar{f})$, at $(\sigma_{-1}, m_{-1}^{\sigma, f})$, $\bar{f}$ is well-defined for all valid inputs if $\sigma_r \sigma_{r+1} \models \rho^{\text{env}} \; \forall r < |\sigma| - 1$, $\sigma_0 \in W_{\bar{\varphi}}$, and execution starts with the initial memory value m^i.

b) Proving properties about the strategy $\bar{f}$: We argued that $\bar{f}$ satisfies the condition in equation (3). This implies that for a state, $\bar{f}$ is well-defined for any valid environmental input when the environment assumption has not been violated in the past while getting to that state. Now all that remains is to show that the plays of $\bar{f}$ satisfy the specification φ.

Consider any $\sigma \in \text{Plays}(\bar{f})$ with $\sigma_0 \in W_{\bar{\varphi}}$. Note that the state sequence $\bar{\sigma}$ used for the construction of the strategy $\bar{f}_G^{\varphi}$ is independent of the sequence of inputs corresponding to

σ. Also, note that the strategy $\bar{f}$ and $\text{Plays}(\bar{f})$ have already been defined. Here we only prove properties about elements of the set $\text{Plays}(\bar{f})$, specifically that they satisfy φ. Consider the case when $\sigma \models \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}}$, because for the other case φ holds directly.

Execution begins at a valid initial state m^i and $\sigma_0 \in W_{\bar{\varphi}}$. If $\sigma \not\models \diamond\psi_1^{\text{sys}}$, it implies that execution continued in accordance with $f_G^{\bar{\varphi}}$ without any memory resets (from the definition of $\bar{f}$). This implies that $\sigma \in \text{Plays}(f_G^{\bar{\varphi}})$, but this leads to a contradiction since $\sigma \models \diamond\psi_1^{\text{sys}} \wedge \bigwedge_{i=1}^{n} \diamond(\psi_i^{\text{sys}} \rightarrow \diamond\psi_{i\oplus 1}^{\text{sys}})$, implying $\sigma \models \diamond\psi_1^{\text{sys}}$. This is because $\sigma \models \bar{\varphi}$ and we are looking at the case when $\sigma \models \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}}$. So, let l_1 be the smallest value at which $\sigma_{l_1} \models \psi_1^{\text{sys}}$ holds.

Now consider $\bar{\sigma}_{:k_1}$, the path to $\bar{\sigma}_{k_1}$. Let us look at the sequence $\sigma_{l_1:}$. If $\sigma_{l_1:} \not\models \diamond\psi_2^{\text{sys}}$, then the sequence $\bar{\sigma}_{:k_1}\sigma_{l_1:} \in \text{Plays}(f_G^{\bar{\varphi}})$. This is because $|[[\psi_1^{\text{sys}}]]|$=1, $\bar{\sigma}_{k_1} = \sigma_{l_1}$. And by construction,

$$\bar{f}((m^{\sigma_{l_1}, \bar{f}}, 1), \sigma_{l_1}, \sigma_{l_1+1} \cap \text{AP}_{\text{env}}) = f(m_{k_1}^{\bar{\sigma}, f_G^{\bar{\varphi}}}, \bar{\sigma}_{k_1}, \sigma_{l_1+1} \cap \text{AP}_{\text{env}}).$$

Therefore, $\bar{\sigma}_{:k_1}\sigma_{l_1:} \in \text{Plays}(f_G^{\bar{\varphi}})$ and $(\bar{\sigma}_{:k_1}\sigma_{l_1:})_0 \in W_{\bar{\varphi}}$. This implies that

$$\bar{\sigma}_{:k_1}\sigma_{l_1:} \models \psi_1^{\text{sys}} \wedge \diamond(\psi_1^{\text{sys}} \rightarrow \diamond\psi_2^{\text{sys}})$$

from the definition of $\bar{\varphi}$ and $m_{k_1}^{\bar{\sigma}, f_G^{\bar{\varphi}}}$ – leading to a conttradiction to our assumption $\sigma_{l_1:} \not\models \diamond\psi_2^{\text{sys}}$. Thus, there exists a finite $l_2 \geq l_1$ at which $\sigma_{l_2} \models \psi_2^{\text{sys}}$. The inequality $l_2 \geq l_1$ can be made strict i.e $l_2 > l_1$ by identifying any i, j for which $[[\psi_j^{\text{sys}}]] = [[\psi_i^{\text{sys}}]]$, and combining them into one progress condition. This means that the same state will not satisfy any two distinct progress conditions, hence $l_2 > l_1$ from the condition $\diamond(\psi_1^{\text{sys}} \rightarrow \diamond\psi_2^{\text{sys}})$.

Repeating the argument for any i, we get $\exists l_{i\oplus 1} > l_i$ such that l_{i+1} is finite and $\sigma_{l_{i\oplus 1}} \models \psi_{i\oplus 1}^{\text{sys}}$ with $\sigma_{l_i} \models \psi_i^{\text{sys}}$. This way we showed that there exists a sequence of integers such that $l_1^1 < l_2^1 < \ldots l_n^1 < l_1^2 \ldots < l_j^k \ \forall j \leq n, \forall k$ with $\sigma_{l_k^i} \models \psi_i^{\text{sys}}$. Given any $j \leq n$ and $r \in \mathbb{N}$ we can find a k such that $r < (k-1)n$, $\sigma_{l_j^k} \models \psi_j^{\text{sys}}$. Therefore, $\sigma_{r:} \models \diamond\psi_j^{\text{sys}}$. This holds true for all r and for all $j \leq n$, hence $\sigma \models \bigwedge_{i=1}^{n} \square\diamond\psi_i^{\text{sys}}$. Therefore, $\bar{f}$ is winning against φ and $\sigma_0 \in W_{\bar{\varphi}} \rightarrow \sigma_0 \in W_{\varphi}$. Therefore, $W_{\bar{\varphi}} \subseteq W_{\varphi}$. And from before $W_{\varphi} \subseteq W_{\bar{\varphi}}$, hence $W_{\bar{\varphi}} = W_{\varphi}$. ∎

Lemma 3: A game structure G with a GR(1) winning condition of the form $\varphi = \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}} \rightarrow \bigwedge_{i=1}^{n} \square\diamond\psi_i^{\text{sys}}$ can be solved by solving $n+1$ independent reachability games if $|[[\psi_i^{\text{sys}}]]| = 1 \ \forall i \in \{1, 2, \ldots, n\}$.

Proof: We first show that for a game with $\bar{\varphi}$ as the winning condition, we can compute the winning strategy from solving $n+1$ reachability games. Then we use the result from Claim 2.

Consider a state s that is winning for $\bar{\varphi}$. Let $\bar{f}_G^{\bar{\varphi}}$ be the winning strategy for $\bar{\varphi}$ from s. Consider $\sigma \in Plays(\bar{f}_G^{\bar{\varphi}})$, then $\sigma \models \psi_1^{\text{sys}} \wedge \bigwedge_{j=1}^{n} \diamond(\psi_j^{\text{sys}} \rightarrow \diamond\psi_{j\oplus 1}^{\text{sys}})$ or $\sigma \models \bigvee_{i=1}^{m} \diamond\square\neg\psi_i^{\text{env}}$. If all plays of $\bar{f}_G^{\bar{\varphi}}$ with the initial state as s satisfy $\bigvee_{i=1}^{m} \diamond\square\neg\psi_i^{\text{env}}$, then $\bar{f}_G^{\bar{\varphi}}$ is winning for φ_0^{reach} as well from s. Therefore, by solving the reachability game with φ_0^{reach} as the winning condition, we can obtain a strategy winning for φ.

Consider the case when $\exists\sigma \in \text{Plays}(\bar{f}_G^{\bar{\varphi}})$ such that $\sigma \models \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}}$ and $\sigma_0 = s$. We observe that for this case, for each $i \in \{1, 2, \ldots, n\}$, the reachability game with condition φ_i^{reach} is winnable from $[[\psi_i^{\text{sys}}]]$. To do this, first note

$$\sigma \models \psi_1^{\text{sys}} \wedge \bigwedge_{j=1}^{n} \diamond(\psi_j^{\text{sys}} \rightarrow \diamond\psi_{j\oplus 1}^{\text{sys}})$$

since $\sigma \models \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}}$ and $\sigma \models \bar{\varphi}$. Define r_i to be the smallest instance such that $\sigma_{r_i} \models \psi_i^{\text{sys}}$ (we know that such an r_i exists from the arguments in the proof of Claim 2).

Next, we prove that the strategy $\bar{f}_G^{\bar{\varphi}}$ with the initial memory value as $m_{r_i}^{\sigma, \bar{f}_G^{\bar{\varphi}}}$ is winning for game with condition φ_i^{reach}. To see this, consider any $\underline{\sigma} \in \text{Plays}(\bar{f}_G^{\bar{\varphi}})$ with $\underline{\sigma}_0 = q$ such that $q \models \psi_i^{\text{sys}}$ i.e execute according to the strategy starting at the state q and memory $m_{r_i}^{\sigma, \bar{f}_G^{\bar{\varphi}}}$. Note that since $[[\psi_i^{\text{sys}}]]$ is a singleton, $q = \sigma_{r_i} = \underline{\sigma}_0$. And, at the state-memory value pair $(q, m_{r_i}^{\sigma, \bar{f}_G^{\bar{\varphi}}})$, $\bar{f}_G^{\bar{\varphi}}$ is well-defined since this state-memory value pair is reachable for $\bar{f}_G^{\bar{\varphi}}$ (recall we selected this state and memory value from a execution in $Plays(\bar{f}_G^{\bar{\varphi}})$ starting from the winning set for $\bar{\varphi}$).

Case 1: $\underline{\sigma} \models \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}}$

For this case, $\sigma_{:r_i}\underline{\sigma} \in Plays(\bar{f}_G^{\bar{\varphi}})$ since $\forall k < r_i, (m_{k+1}^{\sigma}, \sigma_{k+1} \cap \text{AP}_{\text{sys}}) = \bar{f}_G^{\bar{\varphi}}(m_k^{\sigma}\sigma_k, \sigma_{k+1} \cap \text{AP}_{\text{env}})$ and since $\sigma_{r_i} = \underline{\sigma}_0$. We continue execution from $(\sigma_{r_i}, m_{r_i}^{\sigma, \bar{f}_G^{\bar{\varphi}}})$ in accordance with $\bar{f}_G^{\bar{\varphi}}$, so the entire sequence was generated in accordance with this strategy. Using the fact that this strategy is winning from σ_0 for $\bar{\varphi}$, and that the semantics of LTL imply

$$\underline{\sigma} \models \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}} \rightarrow \sigma_{:r_i}\underline{\sigma} \models \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}},$$

we arrive at the conclusion that

$$\sigma_{:r_i}\underline{\sigma} \models \psi_1^{\text{sys}} \wedge \bigwedge_{j=1}^{n} \diamond(\psi_j^{\text{sys}} \rightarrow \diamond\psi_{j\oplus 1}^{\text{sys}}).$$

Therefore, $\sigma_{:r_i}\underline{\sigma} \models (\psi_i^{\text{sys}} \wedge \diamond\psi_{i\oplus 1}^{\text{sys}})$ and r_i was the smallest instance at which ψ_i^{sys} holds. It follows that $\underline{\sigma} \models (\psi_i^{\text{sys}} \wedge \diamond\psi_{i\oplus 1}^{\text{sys}})$. Therefore, $\underline{\sigma} \models \varphi_i^{\text{reach}}$.

Case 2: $\underline{\sigma} \models \neg \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}}$

$\underline{\sigma} \models \neg \bigwedge_{i=1}^{m} \square\diamond\psi_i^{\text{env}} \rightarrow \underline{\sigma} \models \varphi_i^{\text{reach}}$.

This implies that all plays of $f_G^{\bar{\varphi}}$ starting with $s \models \psi_i^{\text{sys}}$ and initial memory value $m_{r_i}^{\sigma, \bar{f}_G^{\bar{\varphi}}}$ are winning against φ_i^{reach}. Hence, $f_G^{\bar{\varphi}}$ is winning against φ_i^{reach} for the state $s \models \psi_i^{\text{sys}}$.

For the case with $s \models \theta$, by the definition of $\bar{\varphi}$, the set of states $[[\theta]]$ is winning for φ_n^{reach}.

Now, we have shown that for the case when the reachability game φ_0^{reach} is not winnable, if $\bar{\varphi}$ is winnable, we can find a winning strategy for each of the φ_j^{reach} games with their respective initial conditions as described in Section III. Let the winning strategy for each such reachability game be $f_G^{\text{reach}_j} : M^j \times \Sigma \times \mathcal{P}(\text{AP}_{\text{env}}) \to M^j \times \mathcal{P}(\text{AP}_{\text{sys}})$ with m_0^i as the initial memory. Without loss of generality, assume that the for any i, j with $i \neq j$, $M^i \cap M^j = \emptyset$. The earlier segment of the proof was to show the existence of these strategies when $\bar{\varphi}$ is winnable.

Now we show these can be combined to form a winning strategy $f_G^{\bar{\varphi}}$ winning against $\bar{\varphi}$. First, consider the strategy f^{reach_n}. Replace all the memory values corresponding to reachable (w, s) for f^{reach_n} where $s \models \psi_1^{\text{sys}}$ with m_0^1. Note that $s \models \psi_1^{\text{sys}}$ corresponds to a valid initial state for the game with condition φ_1^{reach}. Let this modified strategy be $\bar{f}^{\text{reach}_n}$. Effectively, we have patched f^{reach_n} with f^{reach_1} so that after reaching a state that satisfies ψ_1^{sys} it switches from f^{reach_n} to f^{reach_1}. Call the new resulting strategy $f_{1,2}^*$ where $f_{1,2}^*$ is defined as:

$$f_{1,2}^*(w', y) = \begin{cases} \bar{f}^{\text{reach}_n}(w, s, x) & \text{if } w \in M^n, \\ f^{\text{reach}_1}(w, s, x) & \text{if } w \in M^1. \end{cases}$$

Consider σ in $\text{Plays}(f_{1,2}^*)$ such that $\sigma \models \theta \wedge \bigwedge_{i=1}^{m} \Box\Diamond\psi_i^{\text{env}}$ (the other case is trivial). Initially we start at memory m_0^n and a state $\sigma_0 : \sigma_0 \models \theta$ and continue execution along strategy f^{reach_n} till we reach ψ_1^{sys} in a finite number of steps– this is guaranteed by the definition of φ_n^{reach}. Subsequently, execution is continued along f^{reach_1} till we reach a state that satisfies ψ_2^{sys} in a finite number of steps. This is because if $\sigma \models \bigwedge_{i=1}^{m} \Box\Diamond\psi_i^{\text{env}}$, then $\sigma \models \Diamond\psi_1^{\text{sys}} \wedge \Diamond(\psi_1^{\text{sys}} \to \Diamond\psi_2^{\text{sys}})$. Otherwise from the definition of f_1^{reach} and φ_1^{reach}, we end up with a contradiction as before. Similarly, we extend $f_{1,2}^*$ to replace the memory corresponding to the reachable (w, s) for f_1^{reach} where $s \models \psi_1^{\text{sys}}$ with m_0^2. Define the resulting strategy $f_{1,2,3}^*$ as:

$$f_{1,2,3}^*(w', y) = \begin{cases} \bar{f}^{\text{reach}_n}(w, s, x) & \text{if } w \in M^n, \\ \bar{f}^{\text{reach}_1}(w, s, x) & \text{if } w \in M^1, \\ f^{\text{reach}_2}(w, s, x) & \text{if } w \in M^2. \end{cases}$$

As before, we can show that the plays of this strategy are winning against $\Diamond\psi_1^{\text{reach}} \wedge \Diamond(\psi_1^{\text{reach}} \to \Diamond\psi_2^{\text{reach}}) \wedge \Diamond(\psi_2^{\text{reach}} \to \Diamond\psi_3^{\text{reach}})$. Continue the procedure to obtain $f_{1,2,\ldots,n,1}^*$. By construction, this stategy is winning against $\bar{\varphi}$.

We argued that for a state $s \models \theta$ that is winning for $\bar{\varphi}$, either the reachability game with condition φ_0^{reach} is winnable or the reachability games with condition φ_i^{reach} with $i \in \{1, 2, \ldots, n\}$ are winnable. And for both cases, we provided a construction for a strategy winning against $\bar{\varphi}$ from the strategies winning for the reachability games. This implies that for a state, $\bar{\varphi}$ is winnable if and only if the game with φ_0^{reach} is winnable or the games with φ_i^{reach} are winnable. Therefore, from solving the reachability games we can infer if a state is winnable or not and also construct a winning strategy for $\bar{\varphi}$ if it is winnable.

In the proof of Claim 2, we demonstrated an approach to construct a strategy that is winning against φ using the strategy winning against $\bar{\varphi}$. We also showed that the winning states for the conditions φ and $\bar{\varphi}$ are the same. Hence, the GR(1) game can be solved by solving $n + 1$ reachability games separately. ■

IV. Strategy for GR(1) from reachability games

Suppose φ_0^{reach} is winnable, then from solving φ_0^{reach} we have a strategy for $\bar{\varphi}$ and this is also winning for φ, since the environment is blocked from satisfying its assumptions.

Suppose φ_0^{reach} is not winnable and the other n reachability games are winnable. By combining the n reachability games, we construct the strategy f_G^{φ} that is winning against φ. To do this, as before we need a $\mathcal{Z}_n$ that can take values in $\{1, 2, \ldots, n\}$ to track which liveness guarantees have been satisfied with $\mathcal{Z}_n$ initialized to n. The strategy

$$f_G^{\varphi} : (M \times \{1, 2, \ldots, n\}) \times \Sigma \times \mathcal{P}(\text{AP}_{\text{env}}) \to (M \times \{1, 2, \ldots, n\}) \times \mathcal{P}(\text{AP}_{\text{sys}})$$

is constructed as

$$f_G^{\varphi}((w, \mathcal{Z}_n), s, s' \cap \text{AP}_{\text{env}}) = ((w', \mathcal{Z}_n'), s' \cap \text{AP}_{\text{sys}}),$$

where if $s \models \psi_{\mathcal{Z}_n \oplus 1}^{\text{sys}}$,

$$\mathcal{Z}_n' = \mathcal{Z}_n \oplus 1,$$

$$(w', s' \cap \text{AP}_{\text{env}}) = f^{\text{reach}_{\mathcal{Z}_n'}}(m_0^{\mathcal{Z}_n'}, s, s' \cap \text{AP}_{\text{env}},),$$

and if $s \not\models \psi_{\mathcal{Z}_n \oplus 1}^{\text{sys}}$,

$$(w', s' \cap \text{AP}_{\text{env}}) = f^{\text{reach}_{\mathcal{Z}_n}}(w, s, s' \cap \text{AP}_{\text{env}}),$$

$$\mathcal{Z}_n' = \mathcal{Z}_n.$$

Note that this construction combines the two constructions from proofs of Claim 2 and Lemma 3 to get a strategy winning against φ. For an initial condition θ, if φ_0^{reach} is not winnable and for some i such that $n \geq i > 0$, φ_i^{reach} is not winnable, then φ is not winnable from θ.

V. Discussion

In general GR(1) games do not allow for such a decomposition, but the singleton nature of the sets corresponding to the liveness goals allows us to decompose the specifications here. To see why, we take a closer look at the μ-calculus formula correspoding to a GR(1) game with two liveness guarantees for the system as stated in equation (7).

To simplify things, suppose that the environment cannot be blocked from satisfying its liveness assumptions (i.e. φ_0^{reach} is not reazliable). The fixed points in Z_1 and Z_2 are such that after starting at a state in Z_1, the system forces the play into Z_2 while visiting a state satisfying ψ_1^{sys}. Similarly, from Z_2 the system forces the play into a state in Z_1 while visiting ψ_2^{sys}. This could take a few iterations for the fixed points in

Z_1 and Z_2 to converge because a change in Z_1 may cause a change in Z_2, and this goes on till the greatest fixed point is met. Figure 2 illustrates this process, where initially Z_2 is the set of all states and Z_1 is the set of states from which the system can force the play into Z_2 from visiting `Goal 1`. Subsequently, in the next computation, Z_2 is the set of states from where the play can be forced into Z_1 after visitng `Goal 2`. The iterations are continued till convergence.

But, if the sets corresponding to the liveness guarantees are singletons, we can chain a sequence of strategies at the liveness guarantees to enforce the cyclic satisfaction of the liveness guarantees as opposed to going through the iterations in Z_i.

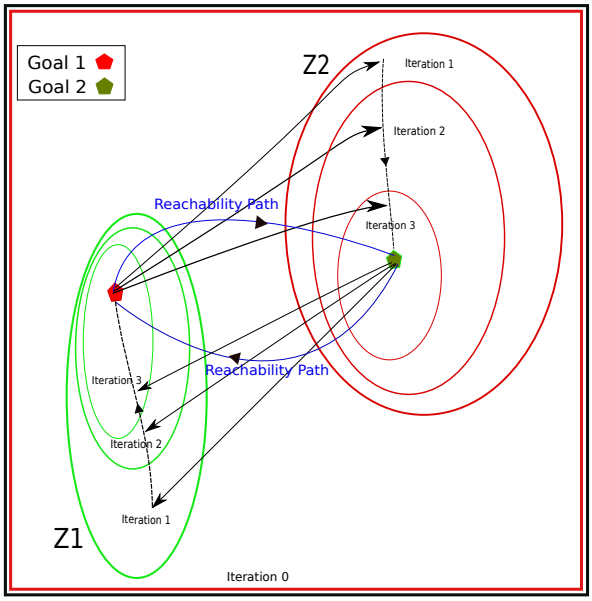

Fig. 2: Visualization of the different approaches [1]

VI. Experiments

For comparing the performance of the synthesis algorithm proposed here with standard GR(1) synthesis, we consider the problem of coordinated planar reactive robot motion planning on a gridworld. For a given set of cells $\{A_1, A_2, \ldots, A_n\}$ and $\{B_1, B_2, \ldots, B_n\}$, the controlled robot has to coordinate with a moving agent such that the robot is in cell B_i when the agent is in cell A_i. The robot has to complete this coordination task infinitely often. To make sure the problem is feasible, the cells $\{B_1, B_2, \ldots, B_n\}$ are added as liveness conditions for the agent. The robot's motion constraints allow movement to any of its non-diagonally adjacent cells. The agent's motion is constrained in a similar way. Additionally, as a part of its safety the robot has to avoid collisions with the agent and the walls (shaded). Figure 3 shows an example gridworld instance. The runtimes for the approach presented here using the decomposed reachability games is compared with those for the solvers `gr1c`[13] and `slugs`[6]. The solvers are accessed using the interfaces

[1] Z_2 after iteration i is a subset of Z_1 from iteration i, Z_1 from iteration $i+1$ is a subset of Z_1 from iteration i. The figure is not an accurate representation and is drawn so for better visualization.

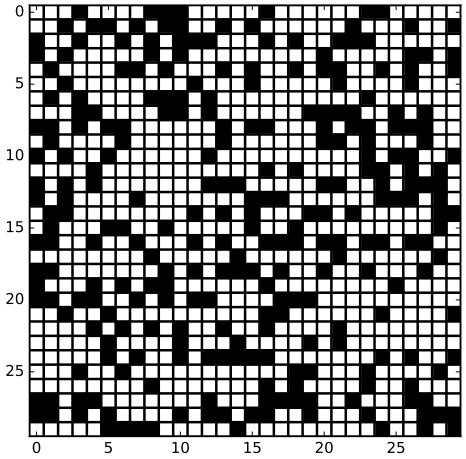

Fig. 3: Gridworld of size 30×30 with wall density of 0.3

in the Temporal Logic Planning Toolbox (`TuLiP`)][9]. The reachability games are solved using the `rg` module from `gr1c`. The computations were performed on a 2.40GHz Quadcore machine with 16 GB of RAM [2].

Gridworld instances with varying number of liveness guarantees and gridsizes are used for benchmarking. Figure 4 depicts the mean runtimes from the benchmarking experiments on $t \times t$– sized gridworld instances, with varying t. For each grid size, 50 random problem instances (with a wall density of 10 percent and 6 liveness guarantees) are created. We see that the decomposition based approach outperforms GR(1) synthesis (using `slugs` and `gr1c`).

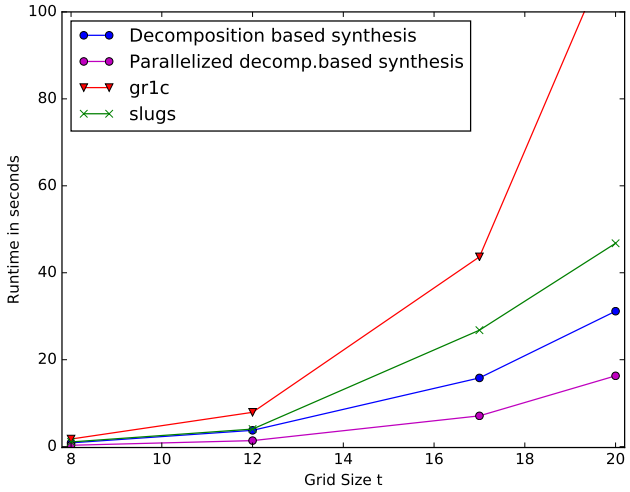

Fig. 4: Performance on gridworld problems with varying grid size $(t \times t)$

Figure 5 depicts the performance for gridworld instances of size 14×14 (wall density 0.3) with the number of liveness constraints changing. Here again we observe similar trends with the decomposition approach outperforming GR(1) synthesis using `slugs` and `gr1c`. When the reachability games

[2] Code for the implemented examples is available at `https://www.dropbox.com/sh/9pfqysj2z572f5b/AAATa7VlGmpT91jpCRgyIwhRa?dl=0`

are solved in parallel, we observe improved scaling for the decomposition based approach. The slope for the parallelized decompositioned-based synthesis is lesser than that of decomposition-based synthesis without parallelization. This is because the complexity of each of the reachability games is independent of n, where n is the number of liveness-guarantees. Since the $n+1$ reachability games are solved in parallel, the runtime approximately stays constant even with the varying number of liveness guarantees.

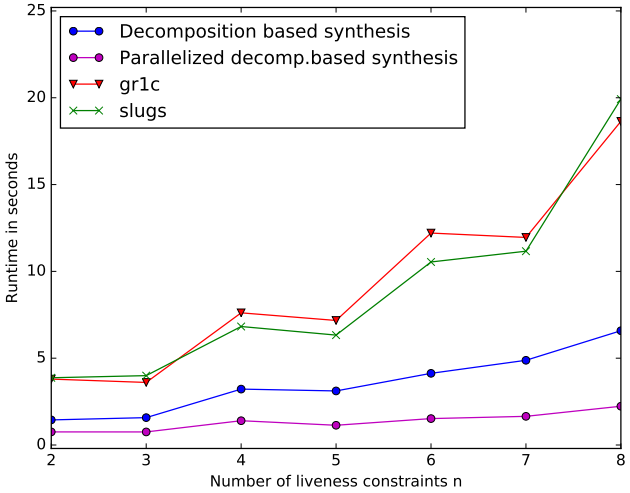

Fig. 5: Performance on gridworld problems with varying number of liveness constraints

VII. Conclusion

We identified a class of synthesis problems where GR(1) games can be solved by solving a set of reachability games that arise from decomposing the original GR(1) formula. The decomposed reachability games are independent from each other allowing for parallelization. Synthesis from the decomposed set of games also results in better performance from eliminating the outer greatest fixed point in GR(1) synthesis. For benchmarking, the decomposition based approach is used to solve LTL motion planning problems that involve coordination between an external agent and a controlled robot on a gridworld. The experiments demonstrate the improved performance of the decomposition-based approach.

Directions for future work include extending the results here to the case when one of the liveness guarantees corresponds to a set of states that is a singleton. Even though in that case the problem cannot be fully decomposed for parallelization–the alternation depth can still be reduced by eliminating the outer greatest fixed point corresponding to GR(1) synthesis. Using the intuiton from here, we wish to further explore approaches for the decomposition of the GR(1) synthesis problems in more general settings to allow for parallelization–heuristic based if complete decomposition is not possible. Another direction of research is to explore approaches for abstraction where

1) Existing discrete transition systems can be abstracted into those with liveness conditions that correspond to singleton sets;
2) For abstraction based synthesis for systems with continous dynamics, the abstraction into a discrete transition system is done so that the sets corresponding to the liveness guarantees are singletons.

Acknowledgments

The authors would like to thank Scott C. Livingston and Tung M. Phan for helpful input. This work was supported by STARnet, a Semiconductor Research Corporation program, sponsored by MARCO and DARPA.

References

[1] R. Alur, S. Moarref, and U. Topcu. *Pattern-Based Refinement of Assume-Guarantee Specifications in Reactive Synthesis*, pages 501–516. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[2] A. Bhatia, L. E. Kavraki, and M. Y. Vardi. Sampling-based motion planning with temporal goals. In *2010 IEEE International Conference on Robotics and Automation*, pages 2689–2696, May 2010.

[3] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. *Journal of Computer and System Sciences*, 78:911–938, May 2012.

[4] A. Browne, E. Clarke, S. Jha, D. Long, and W. Marrero. An improved algorithm for the evaluation of fixpoint expressions. *Theoretical Computer Science*, 178(1):237 – 255, 1997.

[5] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009. Electronically available at http://coordinationbook.info.

[6] R. Ehlers and V. Raman. *Slugs: Extensible GR(1) Synthesis*, pages 333–339. Springer International Publishing, Cham, 2016.

[7] E. A. Emerson and C. Lei. Efficient model checking in fragments of the propositional mu-calculus. In *Proc. Symp. on Logic in Computer Science*, Cambridge, MA, 1986.

[8] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 2020–2025, Barcelona, Spain, 2005.

[9] I. Filippidis, S. Dathathri, S. C. Livingston, N. Ozay, and R. M. Murray. Control design for hybrid systems with TuLiP: The temporal logic planning toolbox. In *2016 IEEE Conference on Control Applications (CCA)*, pages 1030–1041, Sept 2016.

[10] Y. Kantaros and M. M. Zavlanos. Simultaneous intermittent communication control and path optimization in networks of mobile robots. *CoRR*, abs/1609.07038, 2016.

[11] Y. Kesten, N. Piterman, and A. Pnueli. Bridging the gap between fair simulation and trace inclusion. *Information and Computation*, 200:35–61, 2005.

[12] D. Kozen. Results on the propositional -calculus. *Theoretical Computer Science*, 27(3):333 – 354, 1983.

[13] S. C. Livingston. gr1c: a collection of tools for GR(1) synthesis and related activities. `http://scottman.net/2012/gr1c`. [Online; accessed 15-March 2016].

[14] S. C. Livingston and R. M. Murray. Just-in-time synthesis for reactive motion planning with temporal logic. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5033–5038, Karlsruhe, Germany, May 2013.

[15] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag New York, Inc., New York, NY, USA, 1992.

[16] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.

[17] 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*, POPL '89, pages 179–190, New York, NY, USA, 1989. ACM.

[18] M. Rungger, M. Mazo, and P. Tabuada. Scaling up controller synthesis for linear systems and safety specifications. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 7638–7643, Dec 2012.

[19] K. Schneider. *Verification of Reactive Systems: Formal Methods and Algorithms*. SpringerVerlag, 2004.