arXiv:1504.07702v1 [math.OC] 29 Apr 2015

CONTROLLER SYNTHESIS FOR MODE-TARGET GAMES

AYCA BALKAN, MOSHE VARDI, AND PAULO TABUADA

Abstract. Cyber-Physical Systems (CPS) are notoriously difficult to verify due to the intricate interactions between the cyber and the physical components. To address this difficulty, several researchers have argued that the synthesis paradigm is better suited to ensure the correct operation of CPS than the verification paradigm. The key insight of synthesis is that design should be constrained so that resulting systems are easily verified and, ideally, synthesis algorithms should directly provide a proof of correctness.

In this paper we present a Linear Temporal Logic fragment inspired by specifications that frequently occur in control applications where we have a set of modes and corresponding targets to be reached for each mode. The synthesis problem for this fragment is formulated as a mode-target game and we show that these games can be solved in polynomial time by providing two embeddings of mode-target games into Generalized Reactivity(1) (GR(1)) games. While solving GR(1) games requires $O(mnN^2)$ symbolic steps when we have m assumptions, n guarantees, and a game graph with N states, mode-target games can be solved in $O(nN^2)$ symbolic steps when we have n modes and a game graph with N states. These embeddings, however, do not make full use of the specificity of mode-target games. For this reason we investigate in this paper a solution to mode-target games that does not rely on GR(1) embeddings. The resulting algorithm has the same worst case time complexity and we illustrate through experimental results the extent to which it improves upon the algorithms obtained via GR(1) embeddings. In doing so, we highlight the commonalities between mode-target games and GR(1) games while providing additional insight into the solution of GR(1) games.

1. Introduction

Traditionally, implementation and verification are distinct phases of the design process. When the implementation does not satisfy the specifications, the designer attempts to identify the source of the problem, to correct it, and returns to the verification phase. A more appealing alternative to this iterative design process is to put more effort in the implementation phase so as to produce systems that have as few bugs as possible. Ideally, the implementation would be automatically synthesized from a formal specification so as to guarantee that the specification is met. Since the designed system interacts with its environment, the specifications have to be met regardless of the actions taken by the environment. This is precisely the objective of *reactive synthesis*. In this paradigm, the interaction of the environment and the system is modeled as a game where the environment is modeled as an adversary whose objective is to prevent the system from meeting its specifications [19, 2].

Despite the promise of the reactive synthesis paradigm, it is known that synthesizing reactive programs from Linear Temporal Logic (LTL) specifications is doubly exponential in the length of the specification. While in verification problems the

specification tends to be small, for synthesis problems the specification can be large since multiple designs with different objectives have to be produced until the right trade-off between competing specifications is met. To address these challenges, the research community identified fragments of LTL for which the synthesis problem has lower complexity, yet are still interesting for practical applications [5, 4, 6]. Generalized Reactivity(1) (GR(1)), which has polynomial time complexity, is the most commonly used fragment among these [6] and it has been shown to be practically useful via several controller synthesis applications, e.g., [15, 16]. The largest fragment for which the reactive synthesis problem can be solved in polynomial time was shown to be Generalized Rabin(1) by Ehlers in [9]. A discussion on reactive synthesis for a fragment of Generalized Rabin (1) can be found in [24].

This paper introduces a new fragment of LTL that is motivated by various control problems encountered in practice. We consider scenarios where the control objective can be described by modes and corresponding targets. Each target specifies a certain region in the game where the system should enter and stay unless a mode change occurs. This organization of the specification in terms of modes and targets is quite natural and occurs in many different application domains. In Section 4 we provide one example taken from an adaptive cruise control international standard and one example from engine control published by researchers at Toyota. We show that these specifications can be expressed by a type of LTL formulas we call *mode-target formulas*. We model the synthesis problem as finding a winning strategy in a game whose winning objective is given by a mode-target formula. We term such games, under some additional requirements, *mode-target games*.

The contributions of this paper can be summarized as follows:

- We propose mode-target formulas as a practically useful[1] LTL fragment from a modeling perspective. We provide two concrete control applications as an illustration of the large class of problems that can be naturally modeled as mode-target games.
- We show that synthesizing wining strategies for mode-target games has lower complexity than synthesizing winning strategies for GR(1) games and we compare two different synthesis approaches. The first approach is based on an embedding of mode-target games into GR(1) games. The second approach is based on an embedding into a class of games that is rich enough to contain GR(1) games and to bring to the fore how to maximally leverage the specificity of mode-target games.
- Through the aforementioned class of games we expose the commonalities between GR(1) games and mode-target games while providing further insight on the solution of GR(1) games. In particular, we provide a more transparent derivation of the solution of GR(1) games.

Although our motivation to study mode-target games comes from control applications, in this paper we assume that a finite-state game is given. Such finite-state games can be extracted from the differential equations modeling control systems using different abstraction techniques available in the literature, e.g., [23], [14], [20]. Moreover, several of these abstraction techniques also guarantee that a controller or

[1]To further substantiate the usefulness of mode-target games, and although mode-target games had not yet been singled out, we refer the reader to [18] where adaptive cruise control controllers have been automatically synthesized for specifications given via mode-target games.

strategy for the finite-state game abstraction can be refined to a controller enforcing the desired specification on the original control system as detailed in [23].

The rest of the paper is organized as follows. In Section 2, we review the syntax and semantics of LTL and introduce LTL games. We formally define mode-target games in Section 3 and illustrate their usefulness via examples from control. In Section 4 we present the first solution to mode-target games, obtained via an embedding into GR(1) games. We then propose an alternative direct solution in Section 5, which yields a different algorithm. We experimentally compare all the algorithms for the solution of mode-target games in Section 6 and conclude with Section 7.

2. Preliminaries

We start by reviewing the syntax and semantics of Linear Temporal Logic (LTL) and corresponding games.

2.1. **Linear Temporal Logic.** Consider a set of atomic propositions P. LTL formulas are constructed according to the following grammar:

$$\varphi ::= p \in P \,|\, \neg\varphi \,|\, \varphi \vee \varphi \,|\, \bigcirc\varphi \,|\, \varphi \mathcal{U} \varphi.$$

We denote the set 2^P by Σ, where 2^P is the set of all subsets of P. An *infinite word* is an element of Σ^ω where Σ^ω denotes the set of all infinite strings or words obtained by concatenating elements or letters in Σ. We also regard elements of $w \in \Sigma^\omega$ as maps $w : \mathbb{N} \to \Sigma$. Using this interpretation we denote $w(i)$ by w_i. In the context of LTL, the index i models time and w_i is interpreted as the set of atomic propositions that hold at time i.

The semantics of an LTL formula φ is described by a satisfaction relation $\models$ that defines when the string $w \in \Sigma^\omega$ satisfies the formula φ at time $i \in \mathbb{N}$, denoted by $w, i \models \varphi$:

- For $p \in P$, we have $w, i \models p$ iff $p \in w_i$,
- $w, i \models \neg\varphi$ iff $w, i \not\models \varphi$,
- $w, i \models \varphi \vee \psi$ iff $w, i \models \varphi$ or $w, i \models \psi$,
- $w, i \models \bigcirc\varphi$ iff $w, i+1 \models \varphi$,
- $w, i \models \varphi \mathcal{U} \psi$ iff there exists $k \geq i$ such that $w, k \models \psi$ and for all $i \leq j < k$, we have $w, j \models \varphi$.

We use the short hand notation $\varphi \wedge \psi$, for $\neg(\neg\varphi \vee \neg\psi)$, and **True** for $\neg\varphi \vee \varphi$. We further abbreviate $\mathbf{True}\mathcal{U}\varphi$ as $\Diamond\varphi$ which means that φ *eventually* holds and $\neg\Diamond\neg\varphi$ by $\Box\varphi$, which says that φ *always* holds.

We write $W(\varphi)$ to denote the set of all infinite words which satisfy φ, i.e., $W(\varphi) := \{\sigma \in \Sigma^\omega | \sigma \models \varphi\}$. We say that ψ_1 and ψ_2 are *semantically equivalent*, and write $\psi_1 \equiv \psi_2$, if $W(\psi_1) = W(\psi_2)$.

2.2. **Games.** *A game graph* is a tuple $G = (V, E, P, L)$ consisting of:

- A finite set V of states partitioned into V_0 and V_1, i.e., $V = V_0 \cup V_1$ and $V_0 \cap V_1 = \varnothing$;
- A *transition relation* $E \subseteq V \times V$;
- A finite set of atomic propositions P;
- A *labeling function* $L : V \to 2^P$ mapping every state in V to the set of atomic propositions that hold true on that state.

In this definition, V_0 and V_1 are the states from which only player 0 and player 1 can move, respectively. Thus, the state determines which player can move. We assume that for every state $v \in V$, there exists some $v' \in V$ such that $(v, v') \in E$. The function L can be naturally extended to infinite strings $r \in V^\omega$ by $L(r) = L(r_0)L(r_1)L(r_2)\ldots \in \Sigma^\omega$.

A *play* r in a game graph G is an infinite sequence of states $r = v_0 v_1 \ldots \in V^\omega$, such that for all $i \geq 0$, we have $(v_i, v_{i+1}) \in E$. We denote the set of all possible plays for a given game graph G by $\Omega(G)$. For a given LTL formula φ and a game graph $G = (V, E, P, L)$, we use $W_G(\varphi)$ as the short-hand notation for $W(\varphi) \cap L(\Omega(G))$. A strategy for player 0 is a partial function $f : V^* \times V_0 \to V$ such that whenever $f(r, v)$ is defined $(v, f(r, v)) \in E$.

For the purposes of this paper, an *LTL game* is a pair (G, φ) consisting of a game graph G, and a *winning condition* φ which is an LTL formula. A play r in a game (G, φ) is winning for player 0 if $L(r) \in W(\varphi)$. A strategy f for player 0 is *winning* from state v, if all plays starting in v which follow f are winning for player 0. For a given game (G, φ), $[\![\varphi]\!]_G$ denotes the set of states from which player 0 has a winning strategy, this is the *winning region* of player 0. When it is clear from the context which game graph we are referring to, we drop the subscript and just write $[\![\varphi]\!]$.

The sets from which player 0 can force a visit to a set of states V' is denoted by $\mathrm{Pre}(V')$, i.e.,

$$\mathrm{Pre}\,(V') = \{v \in V_0 \,|\, \exists_{v' \in V'}\ (v, v') \in E\} \cup \{v \in V_1 \,|\, \forall_{v' \in V}\ (v, v') \in E \Rightarrow v' \in V'\}$$

We introduce the following fixed-point notation for a given monotone mapping $F : 2^V \to 2^V$:

$$\begin{aligned} &\nu X.F(X) = \cap_i X_i, \text{ where } X_0 = V, \text{ and, } X_{i+1} = F(X_i), \text{ and} \\ &\mu X.F(X) = \cup_i X_i, \text{ where } X_0 = \varnothing, \text{ and } X_{i+1} = F(X_i). \end{aligned}$$

In other words, $\nu X.F(X)$ and $\mu X.F(X)$ are the greatest and least fixed-point of the mapping $F(X)$, respectively.

In the rest of the paper, we abuse notation and sometimes use a set of states $V' \subseteq V$ as an LTL formula. In this case V' is to be interpreted as an atomic proposition that holds only on the states in V'. Whenever, V' defines an atomic proposition not in P, we can always extend P to contain V'. However, for the sake of simplicity we will not explicitly do so.

We call φ a *positional formula* if it does not contain any temporal operators and a *reachability formula* if $\varphi = \Diamond p$ for some positional formula p. We say that φ is a *GR(1) formula* if it has the following form:

$$\varphi = \bigwedge_{i_1 \in I_1} \Box\Diamond a_{i_1} \implies \bigwedge_{i_2 \in I_2} \Box\Diamond g_{i_2}, \tag{2.1}$$

for some positional formulas a_{i_1}, g_{i_2} and finite sets I_1 and I_2. We call games with winning conditions given as a GR(1) formula *GR(1) games*. We refer the reader to [6] for further details on GR(1) formulas.

3. Mode-Target Games

3.1. **Motivation.** Adaptive Cruise Control (ACC) is a more intelligent form of conventional cruise control (CCC). ACC has two *modes* of operation: the speed mode and the distance mode. In order to explain the modes we will call *lead car* to the car in front of the car equipped with ACC. In the speed mode, which is only active when there is no lead car, ACC behaves just like a CCC, i.e., it reaches a pre-set speed and maintains it. The distance mode is what makes ACC more involved than CCC. This mode is active whenever a lead car is present. In the distance mode, ACC attains and sustains a safe headway to the lead car. The headway is a quantity that captures the time required by ACC equipped vehicles to break and avoid a collision when the lead car suddenly reduces its velocity. We consider the specifications for ACC set by the International Organization of Standardization (ISO) in [11]. In these specifications the target region corresponding to the speed mode is defined as $v \in [v_{des} - \epsilon, v_{des} + \epsilon]$, where v, v_{des} and ϵ denote the velocity of the car, the desired velocity, and the allowable tolerance respectively. Similarly, the target region of the distance mode is formalized as $\tau > \tau_{des}$ and $v < v_{des}$, where τ is the headway and τ_{des} is the desired headway[2] Although mode-target games had not yet been singled out, we refer the reader to [18] where controllers for ACC were automatically synthesized for specifications given by mode-target games.

We now consider a different example, the control of a combustion engine, where the specification can again be naturally expressed in terms of modes and targets. The control objectives for an internal combustion engine air-fuel (A/F) ratio controller have a similar flavor to those of ACC [12]. There are four modes of operation: start-up mode, normal mode, power-enrichment mode, and fault mode. In all of these modes, the control objective is to bring the A/F ratio to a certain allowable interval and keep it there. The modes and the corresponding target A/F ratios are summarized in Table 1, where λ_{ref}, and λ_{ref}^{pwr} are the optimal A/F ratios for normal and "full throttle" driving conditions respectively. The car starts in the start-up mode and then transitions to the normal mode after the engine reaches a certain operating temperature. The transition to the power-enrichment mode and to the fault mode is determined by the throttle angle, and an external fault signal, respectively. Note that this scenario is different from the ACC in the sense that mode changes can be triggered by an external signal (fault signal, throttle angle) as well as the current state (temperature) of the system.

Table 1. The modes and the corresponding target A/F ratios as given in [12]. In this table, λ_{ref}, λ_{ref}^{pwr} correspond to the optimal A/F ratios in normal and power-enrichment mode.

Mode	**Target A/F Ratio**
Start-up	$[0.9\lambda_{ref}, 1.1\lambda_{ref}]$
Normal	$[0.98\lambda_{ref}, 1.02\lambda_{ref}]$
Power-Enrichment	$[0.8\lambda_{ref}^{pwr}, 1.2\lambda_{ref}^{pwr}]$
Fault	$[0.9\lambda_{ref}, 1.1\lambda_{ref}]$

[2]In addition to the mode-target behavior, [11] requires the headway to be kept above a certain value regardless of the mode and at all times. However, this is a simple safety specification for which a controller can be synthesized separately and composed with the mode-target controller afterwards.

3.2. **Mode-Target Formulas and Games.** The preceding examples illustrate the scenarios that we want to capture with a suitable LTL fragment. Both ACC and A/F ratio control problems share the following properties that define our setting:
(P1) There are modes and corresponding targets.
(P2) If the system enters mode i, it should reach target i and remain there.
(P3) If the mode changes, there is no obligation to reach the target of the previous mode anymore.

We also make the following observation regarding the dynamics of the modes:
(P4) There is at most one mode active at any given time.

With these properties in mind, we now formally define mode-target formulas and games. For a game to be a mode-target game, its winning condition must be given by a mode-target formula and the corresponding game graph should have certain properties capturing **(P1)**-**(P4)**.

Let I be a finite set and let T and M be finite sets of atomic propositions: $T = \{T_1, T_2, \dots T_{|I|}\}$, and $M = \{M_1, M_2, \dots M_{|I|}\}$. Here, the T_i's represent the targets and the M_i's represent the modes. We start with a game graph G labeled with modes and targets, i.e., $G = (V, E, M \cup T, L)$ where $L : V \to 2^{M \cup T}$. The winning condition for player 0 is given by a *mode-target formula* defined as

$$\varphi := \bigwedge_{i \in I} (\Diamond\Box M_i \implies \Diamond\Box T_i).$$

We can interpret φ as: *if the system eventually settles in M_i, then it should eventually settle in T_i.* This formula captures **(P2)** because it guarantees that the system will reach the target region T_i if the system stays in mode M_i from a point on. If the mode changes at some point, then $\Diamond\Box M_i$ becomes false and the system does not have to stay in the corresponding target anymore, as asserted by **(P3)**. It is true that φ can also be satisfied by switching between modes infinitely often. However, as it is the case in the ACC and A/F ratio control examples, the modes can be partially if not fully determined by an external signal that the controller cannot change. In these cases, by construction, the controller will make progress towards the target of the current mode since it cannot predict if the system will remain in the current mode or switch to a different mode.

To address **(P4)** we make the following assumption on the modes:
(A1) Modes are mutually exclusive, i.e., $M_i \in L(v) \Longrightarrow M_j \notin L(v)$, $\forall j \neq i, \forall v \in V$.

For the rest of the paper, we call games with winning condition given by a mode-target formula and a labeling function L that satisfies **(A1)**, *mode-target games*. Note that, a mode-target game is a Streett game [22] with additional structure imposed by the assumption **(A1)** on the labeling function.

4. Solving Mode-Target Games via GR(1) Games

In this section, we present two different embeddings of mode-target games into GR(1) games.

Proposition 4.1. *Every mode-target game with game graph G is equivalent to the GR(1) game (G, φ), where*

$$\varphi = \bigwedge_{i \in I} \Box\Diamond(\neg M_i \vee \neg T_i) \implies \bigwedge_{i \in I} \Box\Diamond\neg M_i. \tag{4.1}$$

Proof. See Appendix B.1. □

The proof of Proposition 4.1 relies on assumption **(A1)** capturing the mutually exclusive nature of modes in mode-target games. This embedding immediately provides an algorithm for the solution of mode-target games consisting of the algorithm for the solution of GR(1) games, presented in [6], applied to the winning condition defined by (4.1). The essence of this algorithm is the computation of the following fixed-point:

(4.2)
$$[\![\varphi]\!]{=}\nu Z\left(\bigcap_{i\in I}\mu Y\nu X\left(\bigcup_{j\in I}(\mathrm{Pre}(X)\cap[\![M_j\wedge T_j]\!])\cup\mathrm{Pre}(Y)\cup([\![\neg M_i]\!]\cap\mathrm{Pre}(Z))\right)\right).$$

We refer to the algorithm defined by the iterative computation of the preceding fixed-point as **E1** for *Embedding 1*. Algorithm **E1** has time complexity $O(n^2N^2)$ where n is the number of modes in the mode-target formula and N is the number of vertices in the game graph G. This follows from the fact that solving GR(1) games according to the algorithm in [6] has time complexity $O(mnN^2)$ where m is the number of guarantees and n us the number of assumptions. The bound $O(n^2N^2)$ then follows from the equality $n = m$ in (4.1).

An improved algorithm for the solution of mode-target games is obtained by the parsimonious embedding[3], in what regards assumptions, described in the following proposition.

Proposition 4.2. *Every mode-target game with game graph G is equivalent to the GR(1) game (G, φ), where*

$$\varphi = \Box\Diamond\bigwedge_{i\in I}(\neg M_i\vee\neg T_i)\implies\bigwedge_{i\in I}\Box\Diamond\neg M_i. \tag{4.3}$$

Proof. See Appendix B.2. □

While the GR(1) formula (4.1) has n assumptions, the GR(1) formula (4.3) only has 1 assumption. Therefore, by applying the algorithm for the solution of GR(1) games to the winning condition given by (4.3) we obtain an algorithm for the solution of mode-target games with time complexity $O(nN^2)$. We call this algorithm **E2**, for *Embedding 2*, and note that it consists in the computation of the following fixed-point:

(4.4)
$$[\![\varphi]\!]{=}\nu Z\left(\bigcap_{i\in I}\mu Y\nu X(\mathrm{Pre}(X)\cap\cup_{j\in I}[\![M_j\wedge T_j]\!])\cup\mathrm{Pre}(Y)\cup([\![\neg M_i]\!]\cap\mathrm{Pre}(Z))\right).$$

The following result summarizes the discussion in this section.

Theorem 4.3. *Mode-target games can be solved by the symbolic algorithm **E2** requiring $O(nN^2)$* Pre *computations.*

Proof. In [7] Browne et al. show that a fixed-point expression with *alternation depth* k can be computed in $O(N^{\lfloor 1+k/2\rfloor})$ iterations. Note that, given a fixed-point expression the alternation depth is simply the number of alternating largest and greatest fixed-point operators.

The alternation depth of the fixed-point expression (4.4) is three. Moreover, observe that the number of fixed-point computations performed in each iteration

[3]We thank an anonymous reviewer for bringing this embedding to our attention.

of the fixed-point over Z is n. Therefore, one can construct an algorithm that performs $O(nN^2)$ symbolic Pre computations in the worst case. □

Although (4.2) and (4.4) denote the same fixed-point, expression (4.4) results in a more efficient algorithm. In the next section, instead of converting mode-target games into GR(1) games, we investigate whether a direct solution to mode-target games can lead to further simplifications. We also note that Theorem 5.5 only addresses the computation of the winning set. For the sake of completeness we provide all the details on the synthesis of the winning strategy in Appendix F.

5. Solving Mode-Target Games Directly

5.1. **Decomposition of the Winning Set.** We start by introducing a few notions that are critical to understand the direct solution of mode-target games described in this section.

Let $S_1 \subseteq \Sigma^*$ and $S_2 \subseteq \Sigma^* \cup \Sigma^\omega$. We define the concatenation of these sets as $S_1S_2 := \{\sigma \in \Sigma^* \cup \Sigma^\omega | \sigma = \sigma_1\sigma_2, \sigma_1 \in S_1, \sigma_2 \in S_2\}$. A *property* Φ is a subset of Σ^ω. The set of suffixes of a property Φ is denoted by $\text{Post}(\Phi)$, i.e., $\text{Post}(\Phi) := \{\sigma' \in \Sigma^\omega | \sigma\sigma' \in \Phi, \text{ for some } \sigma \in \Sigma^*\}$. A property Φ is an *absolute liveness property* iff $\Sigma^*\Phi \subseteq \Phi$. We call φ an *absolute liveness formula* if $W(\varphi)$ is an absolute liveness property. A formula φ is an absolute liveness formula iff $\varphi \equiv \Diamond\varphi$ (see [21]). It follows that any formula of the form $\Diamond\phi$, for some ϕ is an absolute liveness formula.

We now introduce a class of games that includes both GR(1) games and mode-target games. The definition of this class of games distills the properties that are essential for a simple and transparent derivation of its solution and brings to the fore how we can leverage the nature of mode-target games to further simplify the computation of its solution.

Definition 5.1. An LTL game (G, φ) is said to be simple if the winning condition defined by φ can be written as:

$$\varphi = \Box \bigwedge_{i \in I} \varphi_i, \qquad \varphi_i = \Diamond p_i \vee \psi_i, \tag{5.1}$$

where p_i is a positional formula and ψ_i is an absolute liveness formula that satisfies:

$$W_G(\psi_i) \subseteq W(\varphi). \tag{5.2}$$

The next result shows that GR(1) games are simple.

Lemma 5.2. *Every GR(1) game is a simple game.*

Proof. See Appendix C. □

Although it directly follows from this result that mode-target games are also simple, by virtue of the embeddings described in Section 4, we state this fact in a separate lemma since its proof does not rely on the aforementioned embeddings.

Lemma 5.3. *Every mode-target game is simple.*

Proof. See Appendix D. □

The different translation of mode-target games into simple games has important consequences. The absolute liveness formulas ψ_i that result from the conversion of GR(1) games into simple games are of the form $\psi_i = \vee_{i_1 \in I_1} \Diamond\Box\neg a_{i_1}$ where a_{i_1} are

the assumptions in the GR(1) formula. In contrast, the translation of mode-target games into simple games avoids the disjunction in the absolute liveness formulas resulting in $\psi_i = \Diamond\Box M_i \wedge T_i$.

The winning condition for simple games can be written as a conjunction of formulas φ_i preceded by $\Box$ where each φ_i can be decomposed as a disjunction between a reachability formula and an absolute liveness formula. We now show that it is easy to modify algorithms that synthesize winning strategies for reachability games to obtain algorithms for a conjunction of reachability formulas preceded by $\Box$. The absolute liveness formulas are also easy to handle by virtue of (5.2). This inclusion ensures that a play in (G, ψ_i) that is winning for player 0, is also winning in (G, φ). Therefore, one can adopt a compositional approach to the solution of simple games. A small modification to an algorithm that computes $[\![\varphi_i]\!]$ leads to an algorithm computing $[\![\Box \bigwedge_{i\in I} \varphi_i]\!]$. The next result makes these ideas precise.

Theorem 5.4. *The winning set for player* 0 *in a simple game* (G, φ) *is given by:*

$$[\![\varphi]\!] = \nu Z. \bigcap_{i\in I} [\![\psi_i \vee \Diamond(p_i \wedge \circ Z)]\!]. \tag{5.3}$$

Proof. See Appendix E. □ □

The proof of the first part of Theorem 5.4, follows the existing methods for constructing winning strategies for Generalized Büchi games [8], in which the winning condition is given by:

$$\bigwedge_{i\in I} \Box\Diamond B_i \equiv \Box \bigwedge_{i\in I} \Diamond B_i, \tag{5.4}$$

for some subset of states $B_i \subseteq V$. The winning condition we are interested in, given in (5.1), is slightly different from the one given in (5.4) due to the additional ψ_i term. However, inclusion (5.2) ensures that any play that is winning for (G, ψ_i) is also winning for (G, φ). Hence, by simply computing $\nu Z. \bigcap_{i\in I} [\![\psi_i \vee \Diamond(p_i \wedge \circ Z)]\!]$ we can obtain a winning strategy for player 0 in a simple game. Moreover, this strategy can be seen as the composition of the strategies for games with the simpler winning condition $\psi_i \vee \Diamond(p_i \wedge \circ Z)$.

Note how Theorem 5.4 describes the solution to both GR(1) and mode-target games. For later reference we instantiate (5.3) for mode-target games:

$$[\![\varphi]\!] = \nu Z \bigcap_{i\in I} [\![\Diamond\Box(M_i \wedge T_i) \vee \Diamond(\neg M_i \wedge \circ Z)]\!] \tag{5.5}$$

and explain in the next section how to compute the winning sets

$$[\![\Diamond\Box(M_i \wedge T_i) \vee \Diamond(\neg M_i \wedge \circ Z)]\!] \tag{5.6}$$

so as to make use of (5.5). Note that if we instead instantiate (5.3) for the GR(1) formula (2.1) we obtain:

$$\nu Z \left[\!\!\left[\bigvee_{i_1\in I_1} \Diamond\Box\neg a_{i_1} \vee \Diamond(\neg g_{i_2} \wedge \circ Z) \right]\!\!\right]. \tag{5.7}$$

In comparing (5.7) with (5.5) we note again the difference that was discussed after Lemma 5.3. While GR(1) games require a disjunction of persistence properties ($\vee_{i_i\in I_1} \Diamond\Box\neg a_{i_1}$), mode-target games only require a single persistence property ($\Diamond\Box M_i \wedge T_i$). This suggests that solving mode-target games is, in general, less expensive than solving GR(1) games.

5.2. **Computation of the Winning Set.** In [13], Kesten, Piterman and Pnueli presented a μ-calculus formula which characterizes $[\![\Diamond\Box p \vee \Diamond q]\!]$, where p and q are positional formulas. This μ-calculus formula yields the following fixed-point expression:

$$[\![\Diamond\Box p \vee \Diamond q]\!] = \mu Y \left(\nu X(\mathrm{Pre}(X) \cap [\![p]\!]) \cup [\![q]\!] \cup \mathrm{Pre}(Y)\right).$$

Using this insight, it is easy to see that the winning set (5.5) is given by the following fixed-point:

(5.8)

$$[\![\varphi]\!] = \nu Z \left(\bigcap_{i \in I} \mu Y \left(\nu X(\mathrm{Pre}(X) \cap [\![M_i \wedge T_i]\!]) \cup ([\![\neg M_i]\!] \cap \mathrm{Pre}(Z)) \cup \mathrm{Pre}(Y)\right)\right).$$

Computation of this fixed-point gives us the third algorithm, that we call **MT**, for the solution of mode-target games.

Theorem 5.5. *Mode-target games can be solved by the symbolic algorithm **MT** requiring $O(nN^2)$* Pre *computations.*

Proof. Similar to the proof of Theorem 5.5, this result follows from the fact that the given fixed-point expression is of alternation depth three. Moreover, in each iteration of the algorithm, we sequentially perform n fixed-point expressions which results in $O(nN^2)$ Pre computations in the worst case. □

Even though algorithms **E2** and **MT** compute the same winning set in the same worst case number of iterations, the computations performed by these algorithms differ. While the fixed-point expression (4.4) has $\cup_{i \in I}[\![M_i \wedge T_i]\!]$ for every mode index $i \in I$, the fixed-point in (5.8) replaces this set with $[\![M_i \wedge T_i]\!]$ for each $i \in I$. Since $[\![M_i \wedge T_i]\!] \subseteq \cup_{i \in I}[\![M_i \wedge T_i]\!]$ for all $i \in I$, due to the monotonicity of the given fixed-point operator, algorithm **MT** should perform no worse than **E2** in terms of number of iterations. Moreover, for a given $i \in I$, in order to compute the fixed-point in the variable X, algorithm **MT** only requires the storage of the set $[\![M_i \wedge T_i]\!]$ instead of $\cup_{i \in I}[\![M_i \wedge T_i]\!]$. This suggests that algorithm **MT** might also have better space complexity. To investigate these differences in practice, we provide in the next section an experimental comparison of two implementations for each of the 3 algorithms presented in this paper: **E1**, **E2**, and **MT**.

6. Experimental Comparison

The fixed-points given by (4.2), (4.4), and (5.8) can be computed via iteration of the operators on the right-hand sides of (4.2), (4.4), and (5.8). Two important improvements to a direct implementation of this iteration have appeared in the literature. In [10], Emerson and Lei observed that it is unnecessary, in some cases, to reinitialize the computation of fixed-points to V and $\varnothing$. Instead, for example if one wants to compute a largest fixed-point, the computation can be reinitialized with any approximation that is known to be above (contain) the fixed-point. We refer to an implementation using the techniques in [10] as the **EL** method. In [7], Browne et. al. further exploit monotonicity to obtain an even more efficient implementation of the same fixed-point computation. We refer to such implementation as the **BCJLM** method. While the **BCJLM** method leads to improved time efficiency, it requires storing the value of intermediate computations and thus leads to worse memory efficiency. This is an important factor to be considered since, as we argued

in the introduction, the motivation for mode-target games comes from problems in the control of CPS. In this context, the game graph is given as the abstraction of the CPS to be controlled which is typically very large since it is obtained by discretizing the state space of differential equations as detailed in [23]. For this reason, the main bottleneck is memory usage rather than time.

6.1. **Unicycle Model.** We start with the simplest benchmark used in nonlinear control: the unicycle. The differential equations describing the unicycle:

$$\dot{x} = v\cos(\theta),\ \dot{y} = v\sin(\theta),\ \dot{\theta} = \omega,$$

offer a simplified model for a 3-wheel robot equipped with differential drive, for an airplane moving on a plane, and for several other concrete systems. The pair $(x, y) \in \mathbb{R}^2$ denotes the position of the robot, $\theta \in [-\pi, \pi[$ denotes its orientation, and $(v, \omega) \in \mathbb{R}^2$ are the control inputs, linear velocity v and angular velocity ω. For this example we restrict the position to the set $[1, 5] \times [1, 5]$ and create an abstraction[4] using the PESSOA [17] tool. This abstraction is stored as an Ordered Binary Decision Diagram [3] (OBDD) and constitutes the game graph of the mode-target game. It describes the 13851 vertices or states and the 15 inputs that are available at each state. The second player in this game arises due to the conservative nature of the abstraction, as explained in [23].

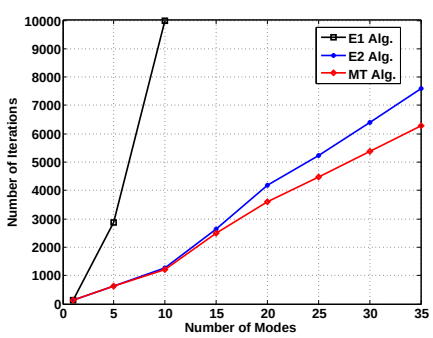
We compare the performance of algorithms **E1**, **E2**, and **MT** as we increase the number of modes from 1 to 35. The mode sets are disjoint hyberboxes of various dimensions and the target sets are smaller hyperboxes contained in the mode sets. For all the experiments in this paper, we measure the memory usage in terms of the maximum total number of nodes in the stored OBDDs during the algorithm's execution.

Figure 1 summarizes our findings. We only provide results for the **EL** method because in this particular example, **BCJLM** does not provide any improvement in the number of iterations over **EL**. Moreover, since **E1** performs significantly worse in terms of time compared to **E2** and **MT**, we only show its first three data points. Figure 1(a) and Figure 1(b) illustrate that, as the number of modes increases, the gap between the performance of algorithm **MT** and algorithm **E2** increases both in terms of number of iterations as well as memory usage. As shown in Fig. 1(c), this gap is also reflected on the time performance of the algorithms. In terms of memory usage, **E2** does slightly worse than **E1** and **MT**, as can be seen from Fig. 1(b). This is expected since **E2** has to store the set $\cup_{i\in I}[\![M_i \wedge T_i]\!]$ for each inner fixed-point computation while **E1** and **MT** only need to store $[\![M_i \wedge T_i]\!]$.

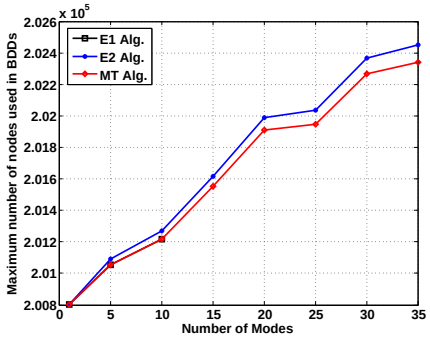
6.2. **Random Tests.** To illustrate the typical performance of the algorithms under comparison we randomly generated 75 linear time-invariant control systems of dimension 2 that were abstracted into a finite game graph. We used PESSOA to construct the abstraction[5] of the randomly generated systems described by the linear differential equation $\dot{x} = Ax + Bu$. The entries of the matrices A and B were chosen from the compact set $[-1, 1]$ and the state space was the set $[-7.5, 7.5] \times [-2.5, 2.5]$. We constructed 12 polyhedral mode-target sets whose descriptions can be found in

[4]The parameters used for the abstraction were $\eta = 0.4$, $\mu = 0.5$, and $\tau = 0.25$. An explanation of the meaning of these parameters is given in [17].

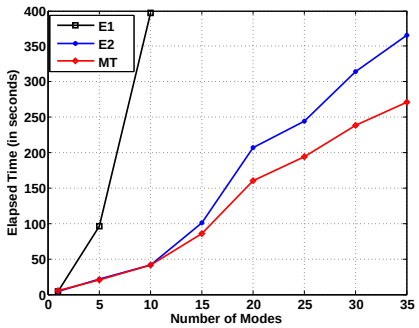
[5]The parameters used for the abstraction were $\eta = 0.15$, $\mu = 0.1$, and $\tau = 0.1$.



(a) Number of iterations of algorithms **E1**, **E2**, and **MT**.



(b) Maximum number of stored OBDD nodes during the execution of Algorithms **E1**, **E2**, and **MT**.



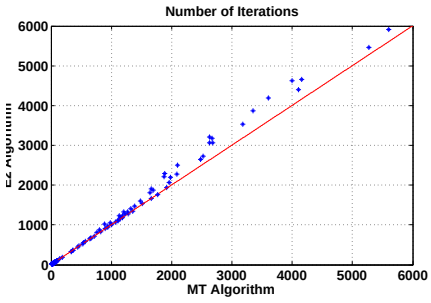
(c) Elapsed time until convergence for Algorithms **E1**, **E2**, and **MT**.

FIGURE 1. Comparison of algorithms **E1**, **E2**, and **MT** using the **EL** method on a unicycle model abstraction

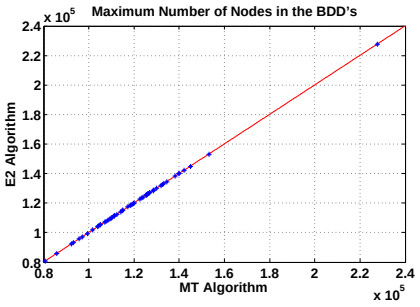
[1]. In each experiment we randomly picked modes from this set of 12 while also randomizing over the number of picked modes.

The results are illustrated in Fig. 2. Fig. 2(a) compares the number of iterations until a fixed-point is reached when the algorithms **E2** and **MT** are implemented using the **BCJLM** method. In this figure, we do not observe a significant gap between the two algorithms. However, as the number of iterations increases, the gap between the number of iterations of both algorithms becomes more distinct. In Figure 2(b), we compare the memory performance of algorithms **E2** and **MT** when using the **EL** method and observe that both algorithms have similar memory usage. Fig. 2(c) and Fig. 2(d) illustrates that even though the **BCJLM** method uses considerably more memory than the **EL** method in most instances, the improvement it provides in terms of number of iterations is not significant.

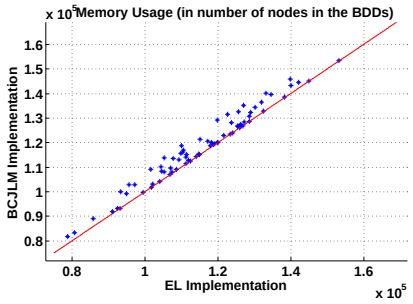
These results suggests the following: **(1)** the **BCJLM** method does not significantly improve the time performance over the **EL** method; **(2)** as expected, algorithm **E1** has the worst time performance even though its memory usage is the



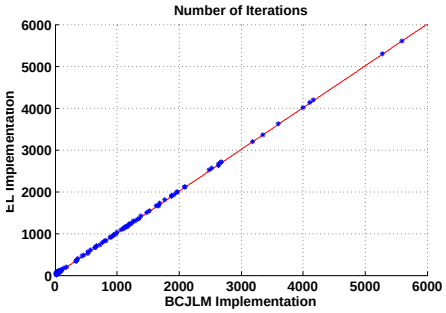
(a) Comparison of algorithms **E2** and **MT**, using the **BCJLM** method, in terms of the number of iterations.



(b) Comparison of algorithms **E2** and **MT**, using the **EL** implementation, in terms of memory usage.



(c) Comparison of **EL** and **BCJLM** implementations on **MT** Algorithm, in terms of memory usage.



(d) Comparison of **EL** and **BCJLM** implementations on **MT** Algorithm, in terms of number of iterations.

FIGURE 2. Comparison of **E2** and **MT** algorithms on random test cases.

same as algorithm **MT**; **(3)** algorithm **MT** is consistently better than algorithms **E2** and **E1** in every aspects even though the increase in performance is not always considerable.

7. CONCLUSIONS

We presented a new class of LTL games called mode-target games inspired by a class of specifications that occurs frequently in control applications. These games can be solved in polynomial time with respect to the size of the underlying game graph and the number of modes. Although mode-target games can be converted to GR(1) games, the complexity of solving mode-target games is lower than the complexity of solving GR(1) games. Experimental results showed that among the three algorithms for the solution of mode-target games algorithm **MT** is always better in terms of time and memory usage although the improvement may not be substantial. As future work, we plan to explore additional structure arising from control applications that may allow further simplifications in the solution of these games.

References

[1] `http://www.cyphylab.ee.ucla.edu/Home/members/ayca-balkan/mode_target`.

[2] M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reactive systems. In *Proceedings of the 16th International Colloquium on Automata, Languages and Programming*, ICALP '89, pages 1–17, London, UK, 1989. Springer-Verlag.

[3] S.B. Akers. Binary decision diagrams. *Computers, IEEE Transactions on*, C-27(6):509–516, June 1978.

[4] R. Alur and S. La Torre. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Logic*, 5(1):1–25, January 2004.

[5] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata, 1998.

[6] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Saar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3):911–938, May 2012.

[7] A. Browne, E.M. Clarke, S. Jha, D.E. Long, and W. Marrero. An improved algorithm for the evaluation of fixpoint expressions. *Theoretical Computer Science*, 178(1-2):237 – 255, 1997.

[8] S. Dziembowski, M. Jurdzinski, and I. Walukiewicz. How much memory is needed to win infinite games? In *Logic in Computer Science, 1997. LICS '97. Proceedings., 12th Annual IEEE Symposium on*, pages 99–110, Jun 1997.

[9] R. Ehlers. Generalized Rabin(1) synthesis with applications to robust system synthesis. In M. Bobaru, K. Havelund, G. J. Holzmann, and R. Joshi, editors, *NASA Formal Methods*, volume 6617 of *Lecture Notes in Computer Science*, pages 101–115. Springer Berlin Heidelberg, 2011.

[10] E. Allen Emerson and Chin-Laung Lei. Efficient Model Checking in Fragments of the Propositional Mu-Calculus (Extended Abstract). In *Proceedings of the First Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 267–278, June 1986.

[11] ISO 15622:2010 (E). Intelligent transport systems – adaptive cruise control systems – performance requirements and test procedures. Technical report, International Organization for Standardization, 2010.

[12] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts. Powertrain control verification benchmark. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, HSCC '14, pages 253–262, New York, NY, USA, 2014. ACM.

[13] Y. Kesten, N. Piterman, and A. Pnueli. Bridging the gap between fair simulation and trace inclusion. *Information and Computation*, 200(1):35 – 61, 2005.

[14] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *Automatic Control, IEEE Transactions on*, 53(1):287–297, Feb 2008.

[15] H. Kress-Gazit and G. E. Fainekos. Where's Waldo? sensor-based temporal logic motion planning. In *in IEEE International Conference on Robotics and Automation*, pages 3116–3121, 2007.

[16] Jun L., N. Ozay, U. Topcu, and R.M. Murray. Synthesis of reactive switching protocols from temporal logic specifications. *Automatic Control, IEEE Transactions on*, 58(7):1771–1785, July 2013.

[17] M. Mazo, A. Davitian, and P. Tabuada. Pessoa: A tool for embedded controller synthesis. In *Proceedings of the 22Nd International Conference on Computer Aided Verification*, CAV'10, pages 566–569, Berlin, Heidelberg, 2010. Springer-Verlag.

[18] P. Nilsson, O. Hussien, Y. Chen, A. Balkan, M. Rungger, A. Ames, J. Grizzle, N. Ozay, H. Peng, and P. Tabuada. Preliminary results on correct-by-construction control software synthesis for adaptive cruise control. In *53rd IEEE Conference on Decision and Control*, 2014.

[19] 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.

[20] G. Reissig. Computing abstractions of nonlinear systems. *Automatic Control, IEEE Transactions on*, 56(11):2583–2598, Nov 2011.

[21] A.P. Sistla. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6(5):495–511, 1994.

[22] R. S. Streett. Propositional dynamic logic of looping and converse. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, STOC '81, pages 375–383, New York, NY, USA, 1981. ACM.
[23] P. Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer, 2009.
[24] E.M. Wolff, U. Topcu, and R.M. Murray. Efficient reactive controller synthesis for a fragment of linear temporal logic. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5033–5040, May 2013.

Appendix A. Preliminary Lemmas

A property Φ is a *stable property* iff $\text{Post}(\Phi) \subseteq \Phi$, i.e., if Φ is closed under suffixes. We call φ a *stable* formula if $W(\varphi)$ is a stable property. It is proved in [21] that a formula φ is a stable formula iff $\Box\varphi \equiv \varphi$. Then it follows that any formula of the form $\Box\phi$, for some ϕ is a stable formula. Moreover, the conjunction of stable formulas is also a stable formula. Take two stable formulas φ_1 and φ_2; then $\varphi_1 \wedge \varphi_2 \equiv \Box\varphi_1 \wedge \Box\varphi_2 \equiv \Box(\varphi_1 \wedge \varphi_2)$, which is a stable formula.

Also recall that a property Φ is an *absolute liveness property* iff $\Sigma^*\Phi \subseteq \Phi$. We call φ an *absolute liveness formula* if $W(\varphi)$ is an absolute liveness property. A formula φ is an absolute liveness formula iff $\varphi \equiv \Diamond\varphi$ as stated in [21]. Therefore, for any formula ϕ, $\Diamond\phi$ is an absolute liveness formula. It is also true that absolute liveness is closed under disjunctions. This is easy to see since if φ_1 and φ_2 are absolute liveness formulas we have: $\varphi_1 \vee \varphi_2 \equiv \Diamond\varphi_1 \vee \Diamond\varphi_2 \equiv \Diamond(\varphi_1 \vee \varphi_2)$, which is an absolute liveness formula as well.

Lemma A.1. *Let φ be an absolute liveness formula. Then $\Box\varphi$ is an absolute liveness formula as well.*

Proof. $\Box\varphi \equiv \Box\Diamond\varphi \equiv \Diamond\Box\Diamond\varphi \equiv \Diamond(\Box\Diamond\varphi)$, which is an absolute liveness formula. □

Lemma A.2. *If φ_1 and φ_2 are absolute liveness formulas, then so is $\varphi_1 \wedge \varphi_2$.*

Proof.

$$\Sigma^* W(\varphi_1 \wedge \varphi_2) = \Sigma^*(W(\varphi_1) \cap W(\varphi_2)) \overset{(1)}{=} \Sigma^* W(\varphi_1) \cap \Sigma^* W(\varphi_2)$$
$$\overset{(1)}{\subseteq} W(\varphi_1) \cap W(\varphi_2) = W(\varphi_1 \wedge \varphi_2),$$

where $\overset{(1)}{=}$ follows from the fact that

$$S_1(S_2 \cap S_3) = S_1S_2 \cap S_1S_3, \text{ for } S_1 \in \Sigma^* \text{ and } S_2, S_3 \in \Sigma^\omega,$$

and $\overset{(1)}{\subseteq}$ holds since φ_1 and φ_2 are absolute liveness formulas. □

Lemma A.3. *Let $\{\varphi_i\}_{i\in I}$ be a collection of formulas indexed by a finite set I. If $\varphi \implies \varphi_i \; \forall\, i \in I$ and φ is a stable formula, then $[\![\varphi]\!] \subseteq \bigcap_{i\in I} [\![\varphi_i \wedge \Box[\![\varphi]\!]]\!]$.*

Proof. We note that it suffices to establish that $[\![\varphi]\!] \subseteq [\![\varphi_i \wedge \Box[\![\varphi]\!]]\!]$ holds. Let $v \in [\![\varphi]\!]$ and let f be a winning strategy for the game (G, φ) from v. It follows from $\varphi \implies \varphi_i$ that strategy f is also winning for (G, φ_i). Moreover, since φ is closed under suffixes f is winning for $(G, \Box[\![\varphi]\!])$ as well. Therefore, it is winning for $(G, \varphi_i \wedge \Box[\![\varphi]\!])$. Hence, $[\![\varphi]\!] \subseteq [\![\varphi_i \wedge \Box[\![\varphi]\!]]\!]$ and the result follows. □

Lemma A.4. *If φ is an absolute liveness formula, then $[\![\Diamond[\![\varphi]\!]]\!] = [\![\varphi]\!]$.*

Proof. The inclusion $[\![\varphi]\!] \subseteq [\![\Diamond[\![\varphi]\!]]\!]$ is clear. We just show the other direction. Note that $[\![\Diamond[\![\varphi]\!]]\!] \subseteq [\![\Diamond\varphi]\!]$ because from any state in $[\![\Diamond[\![\varphi]\!]]\!]$, player 0 can first reach a state $v \in [\![\varphi]\!]$ in finitely many steps, and then follow the strategy to win for the game with the winning condition φ. Also, we know from [21] that $\Diamond\varphi \equiv \varphi$ iff φ is an absolute liveness formula. Therefore $[\![\Diamond[\![\varphi]\!]]\!] \subseteq [\![\Diamond\varphi]\!] = [\![\varphi]\!]$. □

Lemma A.5. *Let p and q be positional formulas, then*

$$\Box(\Diamond p \vee \Diamond\Box q) \equiv \Box\Diamond p \vee \Diamond\Box q.$$

Proof.

$$\Box(\Diamond p \vee \Diamond\Box q) \overset{(1)}{\equiv} \Box(\Diamond(p \vee \Box q)) \equiv \Box\Diamond(p \vee \Box q) \overset{(2)}{\equiv} \Box\Diamond p \vee \Box\Diamond\Box q \overset{(3)}{\equiv} \Box\Diamond p \vee \Diamond\Box q,$$

where $\overset{(1)}{\equiv}$ holds since $\Diamond\varphi_1 \vee \Diamond\varphi_2 \equiv \Diamond(\varphi_1 \vee \varphi_2)$,and $\overset{(2)}{\equiv}$ is true because

$$\Box\Diamond(\varphi_1 \vee \varphi_2) \equiv \Box\Diamond\varphi_1 \vee \Box\Diamond\varphi_2.$$

Finally, $\overset{(3)}{\equiv}$ follows from $\Box\Diamond\Box q \equiv \Diamond\Box q$.

□

Lemma A.6. *Let p and q be positional formulas, then*

$$(\Diamond\Box p \implies \Diamond\Box q) \equiv (\Diamond\Box p \implies \Diamond\Box(p \wedge q)).$$

Proof.

$$\begin{aligned}
&(\Diamond\Box p \implies \Diamond\Box(p \wedge q)) \equiv (\Diamond\Box p \implies (\Diamond\Box p \wedge \Diamond\Box q)) \equiv (\neg(\Diamond\Box p) \vee (\Diamond\Box p \wedge \Diamond\Box q)) \\
&\overset{(1)}{\equiv} (\neg(\Diamond\Box p) \vee \Diamond\Box p) \wedge (\neg(\Diamond\Box p) \vee \Diamond\Box q) \\
&\equiv \mathbf{True} \wedge (\neg(\Diamond\Box p) \vee \Diamond\Box q) \equiv (\Diamond\Box p \implies \Diamond\Box q),
\end{aligned}$$

where $\overset{(1)}{\equiv}$ holds because $\vee$ distributes over $\wedge$. □

Appendix B. Proof of Propositions in Section 4

B.1. **Proof of Proposition 4.1.** Let (G, φ) be a mode-target game. Then the following holds:

$$\varphi = \bigwedge_{i \in I} (\Diamond\Box M_i \implies \Diamond\Box T_i) \overset{(1)}{\equiv} \bigwedge_{i \in I} \Diamond\Box M_i \implies \Diamond\Box(M_i \wedge T_i) \tag{B.1}$$

$$\equiv \bigwedge_{i \in I} (\Box\Diamond\neg M_i \vee \Diamond\Box(M_i \wedge T_i)), \tag{B.2}$$

where $\overset{(1)}{\equiv}$ follows from Lemma A.6. We continue with explaining how this winning condition is equivalent to a winning condition in GR(1) fragment.

Since the modes are mutually exclusive, i.e.,

$$M_i \in L(v) \implies M_j \notin L(v), \forall\, j \neq i, \forall v \in V,$$

the following holds for all $i \in I$:

$$W_G(\Diamond\Box(M_i \wedge T_i) \wedge \Diamond\Box(M_j \wedge T_j)) = \varnothing, \forall\, j \neq i. \tag{B.3}$$

We also have: $W_G(\Diamond\Box(M_i \wedge T_i)) \overset{(1)}{\subseteq} W_G(\Diamond\Box\neg M_j) \overset{(2)}{\subseteq} W_G(\Box\Diamond\neg M_j)$, $\forall\, j \neq i$, where $\overset{(1)}{\subseteq}$ follows from the fact that for all $i \in I$:$[\![M_i]\!] \subseteq [\![\neg M_j]\!]$, $\forall j \neq i$, due to the mutual exclusiveness of the modes. Also note that $\overset{(2)}{\subseteq}$ holds because $\Diamond\Box p \implies \Box\Diamond p$, for any positional formula p.

Therefore we can conclude that for all $i \in I$

$$W_G(\Diamond\Box(M_i \wedge T_i)) \subseteq W_G(\Box\Diamond\neg M_j), \quad \forall\, j \neq i. \tag{B.4}$$

Moreover, the following holds:

$$\begin{aligned} &W_G\left(\bigwedge_{i\in I}(\Box\Diamond\neg M_i \vee \Diamond\Box(M_i \wedge T_i))\right) = W_G\left(\varphi_1 \vee \varphi_2 \vee \varphi_3\right) \\ &= W_G(\varphi_1) \cup W_G(\varphi_2) \cup W_G(\varphi_3), \text{ where} \end{aligned}$$

$$\varphi_1 = \bigwedge_{i\in I} \Diamond\Box(M_i \wedge T_i), \quad \varphi_2 = \bigwedge_{i\in I} \Box\Diamond\neg M_i, \text{ and}$$

$$\varphi_3 = \bigvee_{i\in I}\left(\Diamond\Box(M_i \wedge T_i) \wedge \bigwedge_{j\in I\setminus\{i\}} (\Diamond\Box(M_j \wedge T_j) \vee \Box\Diamond\neg M_j)\right).$$

Due to Equation (B.3), $W_G(\varphi_1) = \varnothing$, and it follows from inclusion in (B.4) that $W_G(\varphi_3) = W_G\left(\bigvee_{i\in I} \Diamond\Box(M_i \wedge T_i)\right)$.

Therefore, we can conclude that

$$W_G\left(\bigwedge_{i\in I}(\Box\Diamond\neg M_i \vee \Diamond\Box(M_i \wedge T_i))\right) = W_G\left(\bigvee_{i\in I} \Diamond\Box(M_i \wedge T_i) \vee \bigwedge_{i\in I} \Box\Diamond\neg M_i\right). \tag{B.5}$$

Note that

$$\bigvee_{i\in I} \Diamond\Box(M_i \wedge T_i) \vee \bigwedge_{i\in I} \Box\Diamond\neg M_i \equiv \left(\bigwedge_{i\in I} \Box\Diamond(\neg M_i \vee \neg T_i) \implies \bigwedge_{i\in I} \Box\Diamond\neg M_i\right). \tag{B.6}$$

Let $\phi := \left(\bigwedge_{i\in I} \Box\Diamond(\neg M_i \vee \neg T_i) \implies \bigwedge_{i\in I} \Box\Diamond\neg M_i\right)$, which is a GR(1) specification with $|I|$ assumptions and $|I|$ guarantees. Then, by combining Equation (B.2), (B.5), and (B.6), we get $W_G(\varphi) = W_G(\phi)$.

This means that finding a winning strategy for player 0 in the mode-target game (G, φ) is equivalent to finding a winning strategy for player 0 in the game (G, ϕ), which is a GR(1) game.

B.2. **Proof of Proposition 4.2.** Suppose $r \in V^\omega$. We start with the following observation:

$$L(r) \models \Box \bigvee_{i\in I} p_i \implies L(r) \models \bigvee_{i\in I} \Box p_i \vee \bigvee_{\substack{J\subseteq I,\\ |J|>1}} \bigwedge_{j\in J} \Box\Diamond p_j, \tag{B.7}$$

where each p_i is a positional formula.

Note that this follows form the fact that any word that satisfies $\Box \bigvee_{i\in I} p_i$ should either always stay in one of the p_i's forever, and hence satisfy $\vee_{i\in I}\Box p_i$ or shuffle between at least two different p_i's, i.e., satisfy $\bigvee_{\substack{J\subseteq I,\\ |J|>1}} \bigwedge_{j\in J} \Box\Diamond p_j$.

Due to Proposition 4.1, we know that a mode-target game is equivalent to the GR(1) game with winning condition $\left(\bigvee_{i\in I} \Diamond\Box(M_i \wedge T_i) \vee \bigwedge_{i\in I} \Box\Diamond\neg M_i\right)$. Next, we show that $W_G(\varphi) = W_G\left(\Diamond\Box \bigvee_{i\in I}(M_i \wedge T_i) \vee \bigwedge_{i\in I} \Box\Diamond\neg M_i\right)$.

Note that the $W_G(\varphi) \subseteq W_G\left(\Diamond\Box \bigvee_{i\in I}(M_i \wedge T_i) \vee \bigwedge_{i\in I} \Box\Diamond\neg M_i\right)$ is immediate since $\bigvee_{i\in I} \Diamond\Box(M_i \wedge T_i)$ implies $\Diamond\Box \bigvee_{i\in I}(M_i \wedge T_i)$. We show the other direction as follows:

$$
\begin{aligned}
& W_G\left(\Diamond\Box \bigvee_{i\in I}(M_i \wedge T_i) \vee \bigwedge_{i\in I} \Box\Diamond\neg M_i\right) \\
& \overset{(1)}{\subseteq} W_G\left(\bigvee_{i\in I} \Diamond\Box(M_i \wedge T_i) \vee \bigvee_{\substack{J\subseteq I,\\ |J|>1}} \bigwedge_{j\in J} \Box\Diamond(M_j \wedge T_j) \vee \bigwedge_{i\in I} \Box\Diamond\neg M_i\right) \\
& \overset{(2)}{\subseteq} W_G\left(\bigvee_{i\in I} \Diamond\Box(M_i \wedge T_i) \vee \bigwedge_{i\in I} \Box\Diamond\neg M_i\right),
\end{aligned}
$$

where $\overset{(1)}{\subseteq}$ follows from the inclusion given in (B.7), distributivity of $\Diamond$ with respect to $\vee$ and the syntactic equivalance :

$$\Diamond \wedge_{j\in J} \Box\Diamond(M_j \wedge T_j) \equiv \wedge_{j\in J}\Box\Diamond(M_j \wedge T_j).$$

Due to the disjointness of the modes, we have $W_G(M_i) \subseteq W_G(\neg M_j)$, $\forall\, j \neq i$ and therefore $\overset{(2)}{\subseteq}$ follows from the fact that $\bigvee_{\substack{J\subseteq I,\\ |J|>1}} \bigwedge_{j\in J} \Box\Diamond(M_j \wedge T_j)$ implies $\bigwedge_{i\in I} \Box\Diamond\neg M_i$.

We therefore obtain $W_G(\varphi) = W_G\left(\Box\Diamond \bigwedge_{i\in I}(\neg M_i \vee \neg T_i) \implies \bigwedge_{i\in I} \Box\Diamond\neg M_i\right)$, which means that mode-target games can be formulated as a GR(1) game with one assumption.

Appendix C. Proof of Lemma 5.2

Given a GR(1) formula φ, the following holds:

$$\varphi \equiv \bigvee_{i_1\in I_1} \Diamond\Box\neg a_i \vee \bigwedge_{i_2\in I_2} \Box\Diamond g_{i_2} \overset{(1)}{\equiv} \Box \bigwedge_{i_2\in I_2} \left(\left(\bigvee_{i_1\in I_1} \Diamond\Box\neg a_{i_1}\right) \vee \Diamond g_{i_2}\right),$$

where $\overset{(1)}{\equiv}$ follows from very similar arguments to those in the proof of Lemma A.5 in Appendix A. Note that $\bigvee_{i_1\in I_1} \Diamond\Box\neg a_{i_1}$ is an absolute liveness formula and moreover

$\bigvee_{i_1 \in I_1} \Diamond\Box\neg a_{i_1}$ implies φ, i.e, $W\left(\bigvee_{i_1 \in I_1} \Diamond\Box\neg a_{i_1}\right) \subseteq W(\varphi)$. Therefore,

$$\varphi = \Box\left(\bigwedge_{i_2 \in I_2} \Diamond g_{i_2} \vee \psi_{i_2}\right), \text{ where } \psi_{i_2} := \bigvee_{i_1 \in I_1} \Diamond\Box\neg a_{i_1}, \text{for all } i_2 \in I_2,$$

which completes the proof of the lemma.

Appendix D. Proof of Lemma 5.3

$$\begin{aligned}\varphi = \bigwedge_{i\in I}(\Diamond\Box M_i \implies \Diamond\Box T_i) &\overset{(1)}{\equiv} \bigwedge_{i\in I}(\Box\Diamond\neg M_i \vee \Diamond\Box(M_i \wedge T_i)) \\ &\overset{(2)}{\equiv} \Box\bigwedge_{i\in I}(\Diamond\neg M_i \vee \Diamond\Box(M_i \wedge T_i)),\end{aligned}$$

where $\overset{(1)}{\equiv}$ has been already shown in the Proof of Proposition 4.1, while $\overset{(2)}{\equiv}$ follows from Lemma A.5 and $\Box\varphi_1 \wedge \Box\varphi_2 \equiv \Box(\varphi_1 \wedge \varphi_2)$.

The last formula has the form given in the statement of the lemma, where p_i is $\neg M_i$ and ψ_i is $\Diamond\Box(M_i \wedge T_i)$. Since $\Diamond\Box(M_i \wedge T_i)$ is an absolute liveness formula, we are only left with showing that $W_G(\psi_i) \subseteq W_G(\varphi)$.

Recall that in mode-target games, if $M_i \in L(v)$ then $M_j \notin L(v)$ for all $j \neq i$. It follows that for any $r \in V^\omega$ we have: $L(r) \models \Diamond\Box M_i \implies L(r) \models \Diamond\Box\neg M_j$, for all $j \neq i$. Moreover, note that $W(\Diamond\Box\neg M_j) \subseteq W(\Box\Diamond\neg M_j)$. Therefore, the following holds:

$$\begin{aligned}W_G(\Diamond\Box(M_i \wedge T_i)) &\subseteq W\left(\Box\bigwedge_{\substack{j\in I,\\ j\neq i}} \Diamond\neg M_j\right) \\ &\subseteq W\left(\Box\left(\bigwedge_{\substack{j\in I,\\ j\neq i}} \Diamond\neg M_j \vee \Diamond\Box(M_j \wedge T_j)\right)\right).\end{aligned} \tag{D.1}$$

Also note that

$$\begin{aligned}W_G\left(\Diamond\Box(M_i \wedge T_i)\right) &\subseteq W_G(\Box\Diamond\neg M_i \vee \Diamond\Box(M_i \wedge T_i)) \\ &= W_G\left(\Box(\Diamond\neg M_i \vee \Diamond\Box(M_i \wedge T_i))\right),\end{aligned} \tag{D.2}$$

where the last equality is due to Lemma A.5.

By combining the inclusions (D.2) and (D.1) we get

$$\begin{aligned}&W_G\left(\Diamond\Box(M_i \wedge T_i)\right) \subseteq W\left(\Box(\Diamond\neg M_i \vee \Diamond\Box(M_i \wedge T_i)\right) \\ &\cap W\left(\Box\bigwedge_{\substack{j\in I,\\ j\neq i}} (\Diamond\neg M_j \vee \Diamond\Box(M_j \wedge T_j))\right) = W\left(\Box\bigwedge_{i\in I}(\Diamond\neg M_i \vee \Diamond\Box(M_i \wedge T_i))\right),\end{aligned}$$

which completes the proof of the lemma.

Appendix E. Proof of Theorem 5.4

Let $Z^* = \nu Z. \bigcap_{i\in I} [\![\psi_i \vee \Diamond(p_i \wedge \circ Z)]\!]$. We start by proving $Z^* \subseteq \left[\!\!\left[\Box \bigwedge_{i\in I} \varphi_i\right]\!\!\right]$. We make the following observation:

$$\left((\Sigma^* p_1)(\Sigma^* p_2)\dots(\Sigma^* p_{|I|})\right)^\omega = W\left(\Box \bigwedge_{i\in I} \Diamond p_i\right) \subseteq W\left(\Box \bigwedge_{i\in I} \Diamond p_i \vee \psi_i\right).$$

This suggests that a strategy that visits all p_i's in a circular fashion is winning for player 0. We pick the visiting order $p_1 p_2 \dots p_i \dots p_{|I|}$, since it is enough to find one winning strategy. Therefore, whenever a play visits a state that satisfies p_i player 0 should be able to switch to a strategy that is winning for the game with the winning condition $\Diamond p_{i+1(mod|I|)}$. Next, we explain that this is in fact possible on Z^*.

The game starts at a state in Z^*. Player 0 follows the strategy that is winning for the game $(G, \psi_i \vee \Diamond(p_i \wedge \circ Z^*))$, from Z^*. If the game reaches a state $v \in [\![p_i]\!]$, then player 0 forces a visit to Z^*. After that player 0 starts following a strategy that is winning for the game with the winning condition:

$$\psi_{i+1(mod|I|)} \vee \Diamond(p_{i+1(mod|I|)} \wedge \circ Z^*).$$

This switching is possible since $Z^* \subseteq [\![\psi_i \vee \Diamond(p_i \wedge \circ Z^*)]\!]$, for all $i \in I$. The circular switching can be implemented using a counter, with $|I|$ states.

Due to the disjunction of the reachability part of the formula with ψ_i, it is true that a play that follows the above strategy can be winning for (G, ψ_i) for some $i \in I$, instead of $(G, \Diamond p_i)$ for some $i \in I$. However, since we assumed that for each $i \in I$, $W_G(\psi_i) \subseteq W\left(\Box \bigwedge_{i\in I} \varphi_i\right)$, even in this case the play is winning for $\Box \bigwedge_{i\in I} \varphi_i$. Therefore, $Z^* \subseteq \left[\!\!\left[\Box \bigwedge_{i\in I} \varphi_i\right]\!\!\right]$.

Now, we also assume that each ψ_i is an absolute liveness formula and show that the other direction, i.e., $\left[\!\!\left[\Box \bigwedge_{i\in I} \varphi_i\right]\!\!\right] \subseteq Z^*$.

Since Z^* is the greatest fixed-point of the operator

$$F(Z) := \bigcap_{i\in I} [\![\psi_i \vee \Diamond(p_i \wedge \circ Z)]\!],$$

to show that $\left[\!\!\left[\Box \bigwedge_{i\in I} \varphi_i\right]\!\!\right] \subseteq Z^*$, it is sufficient to show $\left[\!\!\left[\Box \bigwedge_{i\in I} \varphi_i\right]\!\!\right]$ is a fixed-point of $F(Z)$.

We start by proving that

$$\bigcap_{i\in I} \left[\!\!\left[\psi_i \vee \Diamond\left(p_i \wedge \circ \left[\!\!\left[\Box \bigwedge_{i\in I} \varphi_i\right]\!\!\right]\right)\right]\!\!\right] \subseteq \left[\!\!\left[\Box \bigwedge_{i\in I} \varphi_i\right]\!\!\right].$$

This inclusion can be shown as follows:

$$\bigcap_{i\in I}\left[\!\!\left[\psi_i \vee \Diamond\left(p_i \wedge \circ\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right)\right]\!\!\right] \overset{(1)}{\subseteq} \bigcap_{i\in I}\left[\!\!\left[\psi_i \vee \Diamond\circ\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right]\!\!\right]$$
$$\overset{(2)}{\subseteq} \bigcap_{i\in I}\left[\!\!\left[\psi_i \vee \Diamond\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right]\!\!\right] \overset{(1)}{=} \left[\!\!\left[\Diamond\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right]\!\!\right] \overset{(2)}{=} \left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right].$$

Note that $\overset{(1)}{\subseteq}$ holds since $\Diamond(\phi_1 \wedge \phi_2) \implies \Diamond\phi_1$, for arbitrary formulas ϕ_1, ϕ_2, and $\overset{(2)}{\subseteq}$ follows from the fact that $\Diamond\circ p \implies \Diamond p$ for any positional formula p. Next, we explain the equalities $\overset{(1)}{=}$ and $\overset{(2)}{=}$.

We start with $\overset{(1)}{=}$. Suppose $r \in V^\omega$. Then

$$L(r) \models \psi_i \implies L(r) \models \Box\bigwedge_{i\in I}\varphi_i \implies L(r) \models \left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]$$
$$\implies L(r) \models \Diamond\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right],$$

where the first implication follows from the fact that for all $i \in I$, we have

$$W_G(\psi_i) \subseteq W\left(\Box\bigwedge_{i\in I}\varphi_i\right).$$

Therefore, for all $i \in I$, any play r that is winning for ψ_i, is also winning for $\Diamond\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]$. Then, we can conclude that

$$\left[\!\!\left[\psi_i \vee \Diamond\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right]\!\!\right] = \left[\!\!\left[\Diamond\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right]\!\!\right] \text{ for all } i \in I.$$

We continue with the justification of the equality $\overset{(2)}{=}$. First, note that $\Diamond p_i$ is an absolute liveness formula. Therefore φ_i is an absolute liveness formula as well since it is the disjunction of absolute liveness formulas, $\Diamond p_i$ and ψ_i. By Lemma A.1 $\Box\varphi_i$ is an absolute liveness formula as well, then by Lemma A.2, so is $\Box\bigwedge_{i\in I}\varphi_i$. Therefore, it follows from Lemma A.4 that $\left[\!\!\left[\Diamond\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right]\!\!\right] = \left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]$.

Next, we show the other direction, i.e.,

$$\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right] \subseteq \bigcap_{i\in I}\left[\!\!\left[\psi_i \vee \Diamond\left(p_i \wedge \circ\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right)\right]\!\!\right].$$

Note that $\Box\bigwedge_{i\in I}\varphi_i$ is a stable formula. Therefore, we can invoke Lemma A.3, with $\varphi = \Box\bigwedge_{i\in I}\varphi_i$ which is a stable formula that satisfies $\varphi \implies \varphi_i$, for all $i \in I$.

Then using Lemma A.3, we conclude that

$$\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right] \subseteq \bigcap_{i\in I}\left[\!\!\left[\varphi_i \wedge \Box\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right]\!\!\right].$$

Moreover, the following inclusions hold:

$$\bigcap_{i\in I}\left[\!\!\left[\varphi_i \wedge \Box\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right]\!\!\right] = \bigcap_{i\in I}\left[\!\!\left[(\psi_i \vee \Diamond p_i) \wedge \Box\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right]\!\!\right]$$
$$\overset{(1)}{=} \bigcap_{i\in I}\left[\!\!\left[\left(\psi_i \wedge \Box\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right) \vee \left(\Diamond p_i \wedge \Box\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right)\right]\!\!\right]$$
$$\overset{(1)}{\subseteq} \bigcap_{i\in I}\left[\!\!\left[(\psi_i \vee \left(\Diamond p_i \wedge \Box[\![\Box\bigwedge_{i\in I}\varphi_i]\!]\right)\right]\!\!\right] \subseteq \bigcap_{i\in I}\left[\!\!\left[\psi_i \vee \Diamond\left(p_i \wedge \circ\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right)\right]\!\!\right],$$

where $\overset{(1)}{=}$ follows from the fact that $\wedge$ distributes over $\vee$ and $\overset{(1)}{\subseteq}$ holds since $\phi_1 \wedge \phi_2 \implies \phi_1$. Let's explain the last inclusion in detail. For any positional formula p and q, we have $(\Diamond p) \wedge \Box q \implies (\Diamond p) \wedge \Box\circ q$, because $\Box q \implies \Box\circ q$ for any q. It is also true that $(\Diamond p) \wedge \Box\circ q \implies \Diamond(p \wedge \circ q)$. Hence,

$$\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right] \subseteq \bigcap_{i\in I}\left[\!\!\left[\psi_i \vee \Diamond\left(p_i \wedge \circ\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]\right)\right]\!\!\right].$$

Therefore $\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right]$ is a fixed-point of $F(Z) := \bigcap_{i\in I}[\![\psi_i \vee \Diamond(p_i \wedge \circ Z)]\!]$, and this shows that $\left[\!\!\left[\Box\bigwedge_{i\in I}\varphi_i\right]\!\!\right] \subseteq Z^*$.

Appendix F. Strategy Synthesis

Recall that a strategy is a partial function $f : V^* \times V_0 \to V$ such that whenever $f(r, v)$ is defined, $(v, f(r, v)) \in E$. We next construct a strategy with finite memory that can also be described by a finite-state automaton $A_f = (S, s_0, \delta_M, \delta)$, where S is a finite set that represents the memory, $s_0 \in S$ is the initial memory content, $\delta : V_0 \times S \to V$ is the transition function which determines the next state to be visited based on the current memory content and the current state of the game, and $\delta_M : S \times V \to S$ is the transition function that describes how the memory content should be updated according to the current memory content and current state of the game. We denote the natural extension of δ_M from $S \times V$ to $S \times V^*$ by δ_M^*. Once we specify A_f we can construct f as:

$$f(v_0 \dots v_{i-1}, v_i) := \delta(v_i, \delta_M^*(s_0, v_0 \dots v_{i-1})). \tag{F.1}$$

We now consider the problem of constructing A_f. The set of memory states S is given by $S = \{1, 2, \dots, |I|\}$ and the initial memory state is $s_0 = 1$. The memory state being i indicates that the player 0 should follow a strategy that is winning for the game with the winning condition $\Diamond\Box(M_i \wedge T_i) \vee \Diamond\neg M_i$. If the play satisfies $\Diamond\neg M_i$ then the memory state is incremented. Otherwise, the play satisfies $\Diamond\Box(M_i \wedge T_i)$ and the memory state remains the same. Moreover, we decide (other choices lead to equally valid solutions) that the memory states should evolve according to: $1, 2, \dots, |I|, 1, 2, \dots$ This leads us to define the transition function δ_M by:

$$\delta_M(v, i) = \begin{cases} i + 1 (mod |I|) & \text{if } v \in [\![\neg M_i]\!] \\ i & \text{if } v \notin [\![\neg M_i]\!] \end{cases}.$$

The final step is the definition of δ. It will be based on a set of edges that can be computed from the intermediate results obtained when computing the fixed-point in (5.8).

We start with some additional notation. We use Y_j^{*i} to denote the set computed at the i^{th} iteration of the following fixed-point computation over Y:

$$\mu Y.(\nu X.(\mathrm{Pre}(X) \cap [\![M_j \wedge T_j]\!]) \cup ([\![\neg M_j]\!] \cap [\![\varphi]\!]) \cup \mathrm{Pre}(Y)).$$

Similarly, X_j^{*i*} denotes

$$\nu X.(\mathrm{Pre}(X) \cap [\![M_j \wedge T_j]\!]) \cup ([\![\neg M_j]\!] \cap [\![\varphi]\!]) \cup \mathrm{Pre}(Y_j^{*i})).$$

For each counter value $j \in |I|$, we define the set of edges $E_j := E_{j,1} \cup E_{j,2} \cup E_{j,3}$, where

$$\begin{aligned}
E_{j,1} &= \{(v, v') \in E \mid v \in [\![\neg M_j]\!], \wedge v' \in [\![\varphi]\!]\}, \\
E_{j,2} &= \bigcup_{k>1} \{(v, v') \in E \mid v \in Y_j^{*k} \wedge v \notin Y_j^{*<k} \wedge v' \in Y_j^{*<k}\}, \\
E_{j,3} &= \bigcup_{k} \{(v, v') \in E \mid v \in X_j^{*k*} \cap [\![M_i \wedge T_i]\!] \wedge v \notin X_j^{*<k*} \wedge v' \in X_j^{*k*}\},
\end{aligned}$$

where $Y_j^{*<k} = \bigcup_{0 \le i < k} Y_j^{*i}$ and $X_j^{*<k*} = \bigcup_{0 \le i < k} X_j^{*i*}$.

The set $E_{j,1}$ contains the transitions that player 0 can use to force the game to move from a state in $[\![\neg M_j]\!]$ to a state in $[\![\varphi]\!]$. $E_{j,2}$ corresponds to the transitions, that player 0 can force the game to make progress towards a state in $[\![\neg M_j]\!]$. The edges in $E_{j,3}$ are the transitions, where the game is at a state in $[\![M_i \wedge T_i]\!]$ and player 0 cannot force the game to make progress towards $[\![\neg M_i]\!]$. Note that, player 0 still wins even if there is no progress towards $[\![\neg M_i]\!]$ forever as long as the game stays in $[\![M_i \wedge T_i]\!]$ forever as well.

We make use of edges E_j to define δ when the memory state is at j as follows:

$$\delta(v_0, j) = v', \text{ where } (v_0, v') \in E_j.$$

Now, we have all the ingredients to construct the winning strategy f according to (F.1).

Department of Electrical Engineering, University of California at Los Angeles, CA 90095-1594, United States

E-mail address: abalkan@ucla.edu

Department of Computer Science, Rice University, Houston, TX 77005-1892, United States

E-mail address: vardi@cs.rice.edu

Department of Electrical Engineering, University of California at Los Angeles, CA 90095-1594, United States

E-mail address: tabuada@ee.ucla.edu