Environmentally-friendly GR(1) Synthesis

Rupak Majumdar[1], Nir Piterman[2⋆], and Anne-Kathrin Schmuck[1⋆⋆]

[1] MPI-SWS, Kaiserslautern, Germany
[2] University of Leicester, Leicester, UK

Abstract Many problems in reactive synthesis are stated using two formulas —an *environment assumption* and a *system guarantee*— and ask for an implementation that satisfies the guarantee in environments that satisfy their assumption. Reactive synthesis tools often produce strategies that formally satisfy such specifications by actively preventing an environment assumption from holding. While formally correct, such strategies do not capture the intention of the designer. We introduce an additional requirement in reactive synthesis, *non-conflictingness*, which asks that a system strategy should always allow the environment to fulfill its liveness requirements. We give an algorithm for solving GR(1) synthesis that produces non-conflicting strategies. Our algorithm is given by a 4-nested fixed point in the μ-calculus, in contrast to the usual 3-nested fixed point for GR(1). Our algorithm ensures that, in every environment that satisfies its assumptions on its own, traces of the resulting implementation satisfy both the assumptions and the guarantees. In addition, the asymptotic complexity of our algorithm is the same as that of the usual GR(1) solution. We have implemented our algorithm and show how its performance compares to the usual GR(1) synthesis algorithm.

1 Introduction

Reactive synthesis from temporal logic specifications provides a methodology to automatically construct a system implementation from a declarative specification of correctness. Typically, reactive synthesis starts with a set of requirements on the system and a set of assumptions about the environment. The objective of the synthesis tool is to construct an implementation that ensures all guarantees are met in every environment that satisfies all the assumptions; formally, the synthesis objective is an implication $A \Rightarrow G$. In many synthesis problems, the system can actively influence whether an environment satisfies its assumptions. In such cases, an implementation that prevents the environment from satisfying its assumptions is considered correct for the specification: since the antecedent of the implication $A \Rightarrow G$ does not hold, the property is satisfied.

⋆ Supported by project "d-SynMA" that is funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 772459).

⋆⋆ corresponding author: akschmuck@mpi-sws.org

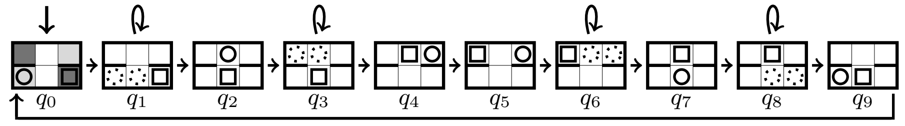

Figure 1. Pictorial representation of a *desired* strategy for a robot (square) moving in a maze in presence of a moving obstacle (circle). Obstacle and robot start in the lower left and right corner, can move at most one step at a time (to non-occupied cells) and cells that they should visit infinitely often are indicated in light and dark gray (see q_0), respectively. Nodes with self-loops ($q_{\{1,3,6,8\}}$) can be repeated finitely often with the obstacle located at one of the dotted positions.

Such implementations satisfy the letter of the specification but not its intent. Moreover, assumption-violating implementations are not a theoretical curiosity but are regularly produced by synthesis tools such as `slugs` [14]. In recent years, a lot of research has thus focused on how to model environment assumptions [19,11,2,5,4], so that assumption-violating implementations are ruled out. Existing research either removes the "zero sum" assumption on the game by introducing different levels of co-operation [5], by introducing equilibrium notions inspired by non-zero sum games [7,21,16], or by introducing richer quantitative objectives on top of the temporal specifications [3,1].

Contribution In this paper, we take an alternative approach. We consider the setting of GR(1) specifications, where assumptions and guarantees are both conjunctions of safety and Büchi properties [6]. GR(1) has emerged as an expressive specification formalism [25,29,18] and, unlike full linear temporal logic, synthesis for GR(1) can be implemented in time quadratic in the state/transition space. In our approach, the environment is assumed to satisfy its assumptions provided the system does not prevent this. Conversely, the system is required to pick a strategy that ensures the guarantees whenever the assumptions are satisfied, but additionally ensures *non-conflictingness*: along each finite prefix of a play according to the strategy, there exists the persistent possibility for the environment to play such that its liveness assumptions will be met.

Our main contribution is to show a μ-calculus characterization of winning states (and winning strategies) that rules out system strategies that are winning by preventing the environment from fulfilling its assumptions. Specifically, we provide a 4-nested fixed point that characterizes winning states and strategies that are *non-conflicting* and ensure all guarantees are met if all the assumptions are satisfied. Thus, if the environment promises to satisfy its assumption if allowed, the resulting strategy ensures both the assumption and the guarantee.

Our algorithm does not introduce new notions of winning, or new logics or winning conditions. Moreover, since μ-calculus formulas with d alternations can be computed in $O(n^{\lceil d/2 \rceil})$ time [27,8], the $O(n^2)$ asymptotic complexity for the new symbolic algorithm is the same as the standard GR(1) algorithm.

Motivating Example Consider a small two-dimensional maze with 3x2 cells as depicted in Figure 1, state q_0. A robot (square) and an obstacle (circle) are located in this maze and can move at most one step at a time to non-occupied cells. There is a wall between the lower and upper left cell and the lower and

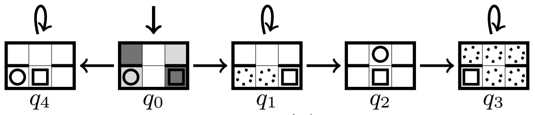

Figure 2. Pictorial representation of the *GR(1) winning strategy* synthesized by `slugs` for the robot (square) in the game described in Figure 1.

upper right cell. The interaction between the robot and the object is as follows: first the environment chooses where to move the obstacle to, and, after observing the new location of the obstacle, the robot chooses where to move.

Our objective is to synthesize a strategy for the robot s.t. it visits both the upper left and the lower right corner of the maze (indicated in dark gray in Figure 1, state q_0) infinitely often. Due to the walls in the maze the robot needs to cross the two white middle cells infinitely often to fulfill this task. If we assume an arbitrary, adversarial behavior of the environment (e.g., placing the obstacle in one white cell and never moving it again) this desired robot behavior cannot be enforced. We therefore assume that the obstacle is actually another robot that is required to visit the lower left and the upper right corner of the maze (indicated in light gray in Figure 1, state q_0) infinitely often. While we do not know the precise strategy of the other robot (i.e., the obstacle), its liveness assumption is enough to infer that the obstacle will always eventually free the white cells. Under this assumption the considered synthesis problem has a solution.

Let us first discuss one intuitive strategy for the robot in this scenario, as depicted in Figure 1. We start in q_0 with the obstacle (circle) located in the lower left corner and the robot (square) located in the lower right corner. Recall that the obstacle will eventually move towards the upper right corner. The robot can therefore wait until it does so, indicated by q_1. Here, the dotted circles denote possible locations of the obstacle during the (finitely many) repetitions of q_1 by following its self loop. Whenever the obstacle moves to the upper part of the maze, the robot moves into the middle part (q_2). Now it waits until the obstacle reaches its goal in the upper right, which is ensured to happen after a finite number of visits to q_3. When the obstacle reaches the upper right, the robot moves up as well (q_4). Now the robot can freely move to its goal in the upper left (q_5). This process symmetrically repeats for moving back to the respective goals in the lower part of the maze (q_6 to q_9 and then back to q_0). With this strategy, the interaction between environment and system goes on for infinitely many cycles and the robot fulfills its specification.

The outlined synthesis problem can be formalized as a two player game with GR(1) winning condition. When solving this synthesis problem using the tool `slugs` [14], we obtain the strategy depicted in Figure 2 (not the desired one in Figure 1). The initial state, denoted by q_0 is the same as in Figure 1 and if the environment moves the obstacle into the middle passage (q_1) the robot reacts as before; it waits until the object eventually proceeds to the upper part of the maze (q_2). However, after this happens the robot takes the chance to simply move to the lower left cell of the maze and stays there forever (q_3). By this, the robot prevents the environment from fulfilling its objective. Similarly, if the obstacle does not immediately start moving in q_0, the robot takes the chance to place itself in the middle passage and stays there forever (q_4). This obviously prevents the environment from fulfilling its liveness properties.

In contrast, when using our new algorithm to solve the given synthesis problem, we obtain the strategy given in Figure 1, which satisfies the guarantees while allowing the environment assumptions to be satisfied.

Related Work Our algorithm is inspired by supervisory controller synthesis for non-terminating processes [24,28], resulting in a fixed-point algorithm over a Rabin-Büchi automaton. This algorithm has been simplified for two interacting Büchi automata in [23] without proof. We adapt this algorithm to GR(1) games and provide a new, self-contained proof in the framework of two-player games, which is distinct from the supervisory controller synthesis setting (see [13,26] for a recent comparison of both frameworks).

The problem of correctly handling assumptions in synthesis has recently gained attention in the reactive synthesis community [4]. As our work does not assume precise knowledge about the environment strategy (or the ability to impose the latter), it is distinct from cooperative approaches such as assume-guarantee [9] or rational synthesis [17]. It is closest related to obliging games [10], cooperative reactive synthesis [5], and assume-admissible synthesis [7]. Obliging games [10] incorporate a similar notion of non-conflictingness as our work, but do not condition winning of the system on the environment fulfilling the assumptions. This makes obliging games harder to win. Cooperative reactive synthesis [5] tries to find a winning strategy enforcing $A \cap G$. If this specification is not realizable, it is relaxed and the obtained system strategy enforces the guarantees if the environment cooperates “in the right way”. Instead, our work always assumes the same form of cooperation; coinciding with just one cooperation lever in [5]. Assume-admissible synthesis [7] for two players results in two individual synthesis problems. Given that both have a solution, only implementing the system strategy ensures that the game will be won if the environment plays *admissible*. This is comparable to the view taken in this paper, however, assuming that the environment plays *admissible* is stronger then our assumption on an environment attaining its liveness properties if not prevented from doing so. Moreover, we only need so solve one synthesis problem, instead of two. However, it should be noted that [10,5,7] handle ω-regular assumptions and guarantees. We focus on the practically important GR(1) fragment and our method better leverages the computational benefits for this fragment.

All proofs of our results and additional examples can be found in the extended version [22]. We further acknowledge that the same problem was independently solved in the context of reactive robot mission plans [12] which was brought to our attention only shortly before the final submission of this paper.

2 Two Player Games and the Synthesis Problem

2.1 Two Player Games

Formal Languages Let Σ be a finite alphabet. We write Σ^*, Σ^+, and Σ^ω for the sets of finite words, non-empty finite words, and infinite words over Σ. We write $w \leq v$ (resp., $w < v$) if w is a prefix of v (resp., a strict prefix of v). The set of all prefixes of a word $w \in \Sigma^\omega$ is denoted $\mathrm{pfx}(w) \subseteq \Sigma^*$. For $L \subseteq \Sigma^*$, we have $L \subseteq \mathrm{pfx}(L)$. For $\mathcal{L} \subseteq \Sigma^\omega$ we denote by $\overline{\mathcal{L}}$ its complement $\Sigma^\omega \setminus \mathcal{L}$.

Game Graphs and Strategies A *two player game graph* $H = (Q^0, Q^1, \delta^0, \delta^1, q_0)$ consists of two finite disjoint state sets Q^0 and Q^1, two transition functions $\delta^0 : Q^0 \to 2^{Q^1}$ and $\delta^1 : Q^1 \to 2^{Q^0}$, and an initial state $q_0 \in Q^0$. We write $Q = Q^0 \cup Q^1$. Given a game graph H, a *strategy* for player 0 is a function $f^0 : (Q^0Q^1)^*Q^0 \to Q^1$; it is *memoryless* if $f^0(\nu q^0) = f^1(q^0)$ for all $\nu \in (Q^0Q^1)^*$ and all $q^0 \in Q^0$. A *strategy* $f^1 : (Q^0Q^1)^+ \to Q^0$ for player 1 is defined analogously. The infinite sequence $\pi \in (Q^0Q^1)^\omega$ is called a play over H if $\pi(0) = q_0$ and for all $k \in \mathbb{N}$ holds that $\pi(2k{+}1) \in \delta^0(\pi(2k))$ and $\pi(2k{+}2) \in \delta^1(\pi(2k{+}1))$; π is compliant with f^0 and/or f^1 if additionally holds that $f^0(\pi|_{[0,2k]}) = \pi(2k+1)$ and/or $f^1(\pi|_{[0,2k+1]}) = \pi(2k+2)$. We denote by $\mathcal{L}(H, f^0)$, $\mathcal{L}(H, f^1)$ and $\mathcal{L}(H, f^0, f^1)$ the set of plays over H compliant with f^0, f^1, and both f^0 and f^1, respectively.

Winning Conditions We consider winning conditions defined over sets of states of a given game graph H. Given $F \subseteq Q$, we say a play π satisfies the *Büchi condition* F if $\mathrm{Inf}(\pi) \cap F \neq \emptyset$, where $\mathrm{Inf}(\pi) = \{q \in Q \mid \pi(k) = q$ for infinitely many $k \in \mathbb{N}\}$. Given a set $\mathcal{F} = \{F_1, \ldots, F_m\}$, where each $F_i \subseteq Q$, we say a play π satisfies the *generalized Büchi condition* $\mathcal{F}$ if $\mathrm{Inf}(\pi) \cap F_i \neq \emptyset$ for each $i \in [1; m]$. We additionally consider generalized reactivity winning conditions with rank 1 (GR(1) winning conditions in short). Given two generalized Büchi conditions $\mathcal{F}^0 = \{F^0_1, \ldots, F^0_m\}$ and $\mathcal{F}^1 = \{F^1_1, \ldots, F^1_n\}$, a play π satisfies the GR(1) condition if either $\mathrm{Inf}(\pi) \cap F^0_i = \emptyset$ for some $i \in [1; m]$ or $\mathrm{Inf}(\pi) \cap F^1_j \neq \emptyset$ for each $j \in [1; m]$. That is, whenever the play satisfies $\mathcal{F}^0$, it also satisfies $\mathcal{F}^1$. We use the tuples (H, F), $(H, \mathcal{F})$ and $(H, \mathcal{F}^0, \mathcal{F}^1)$ to denote a Büchi, generalized Büchi and GR(1) game over H, respectively, and collect all winning plays in these games in the sets $\mathcal{L}(H, F)$, $\mathcal{L}(H, \mathcal{F})$ and $\mathcal{L}(H, \mathcal{F}^0, \mathcal{F}^1)$. A strategy f^l is *winning* for player l in a Büchi, generalized Büchi, or GR(1) game, if $\mathcal{L}(H, f^l)$ is contained in the respective set of winning plays.

Set Transformers on Games Given a game graph H, we define the existential, universal, and player 0-, and player 1-controllable pre-operators. Let $P \subseteq Q$.

$$\mathsf{Pre}^\exists(P) = \left\{q^0 \in Q^0 \middle| \delta^0(q^0) \cap P \neq \emptyset\right\} \cup \left\{q^1 \in Q^1 \middle| \delta^1(q^1) \cap P \neq \emptyset\right\}, \text{ and} \tag{1}$$

$$\mathsf{Pre}^\forall(P) = \left\{q^0 \in Q^0 \middle| \delta^0(q^0) \subseteq P\right\} \cup \left\{q^1 \in Q^1 \middle| \delta^1(q^1) \subseteq P\right\}, \tag{2}$$

$$\mathsf{Pre}^0(P) = \left\{q^0 \in Q^0 \middle| \delta^0(q^0) \cap P \neq \emptyset\right\} \cup \left\{q^1 \in Q^1 \middle| \delta^1(q^1) \subseteq P\right\}, \text{ and} \tag{3}$$

$$\mathsf{Pre}^1(P) = \left\{q^0 \in Q^0 \middle| \delta^0(q^0) \subseteq P\right\} \cup \left\{q^1 \in Q^1 \middle| \delta^1(q^1) \cap P \neq \emptyset\right\}. \tag{4}$$

Observe that $Q \setminus \mathsf{Pre}^\exists(P) = \mathsf{Pre}^\forall(Q \setminus P)$ and $Q \setminus \mathsf{Pre}^1(P) = \mathsf{Pre}^0(Q \setminus P)$.

We combine the operators in (1)-(4) to define a *conditional predecessor* $\mathsf{CondPre}$ and its dual $\overline{\mathsf{CondPre}}$ for sets $P, P' \subseteq Q$ by

$$\mathsf{CondPre}(P,\, P') := \mathsf{Pre}^\exists(P) \cap \mathsf{Pre}^1(P \cup P'), \text{ and} \tag{5}$$

$$\overline{\mathsf{CondPre}}(P,\, P') := \mathsf{Pre}^\forall(P) \cup \mathsf{Pre}^0(P \cap P'). \tag{6}$$

We see that $Q \setminus \mathsf{CondPre}(P,\, P') = \overline{\mathsf{CondPre}}(Q \setminus P,\, Q \setminus P')$.

μ-Calculus We use the μ-calculus as a convenient logical notation used to define a symbolic algorithm (i.e., an algorithm that manipulates sets of states rather then individual states) for computing a set of states with a particular property over a given game graph H. The formulas of the μ-calculus, interpreted over a two-player game graph H, are given by the grammar

$$\varphi \ ::= \ p \mid X \mid \varphi \cup \varphi \mid \varphi_1 \cap \varphi_2 \mid \mathit{pre}(\varphi) \mid \mu X.\varphi \mid \nu X.\varphi$$

where p ranges over subsets of Q, X ranges over a set of formal variables, $\mathit{pre} \in \{\mathsf{Pre}^{\exists}, \mathsf{Pre}^{\forall}, \mathsf{Pre}^{0}, \mathsf{Pre}^{1}, \mathsf{CondPre}, \overline{\mathsf{CondPre}}\}$ ranges over set transformers, and μ and ν denote, respectively, the least and greatest fixpoint of the functional defined as $X \mapsto \varphi(X)$. Since the operations $\cup$, $\cap$, and the set transformers pre are all monotonic, the fixpoints are guaranteed to exist. A μ-calculus formula evaluates to a set of states over H, and the set can be computed by induction over the structure of the formula, where the fixpoints are evaluated by iteration. We omit the (standard) semantics of formulas [20].

2.2 The Considered Synthesis Problem

The GR(1) synthesis problem asks to synthesize a winning strategy for the system player (player 1) for a given GR(1) game $(H, \mathcal{F}_{\mathcal{A}}, \mathcal{F}_{\mathcal{G}})$ or determine that no such strategy exists. This can be equivalently represented in terms of ω-languages, by asking for a system strategy f^1 over H s.t.

$$\emptyset \neq \mathcal{L}(H, f^1) \subseteq \overline{\mathcal{L}(H, \mathcal{F}_{\mathcal{A}})} \cup \mathcal{L}(H, \mathcal{F}_{\mathcal{G}}).$$

That is, the system wins on plays $\pi \in \mathcal{L}(H, f^1)$ if either $\pi \notin \mathcal{L}(H, \mathcal{F}_{\mathcal{A}})$ or $\pi \in \mathcal{L}(H, \mathcal{F}_{\mathcal{A}}) \cap \mathcal{L}(H, \mathcal{F}_{\mathcal{G}})$. The only mechanism to ensure that *sufficiently* many computations will result from f^1 is the usage of the environment input, which enforces a minimal branching structure. However, the system could still win this game by *falsifying the assumptions*; i.e., by generating plays $\pi \notin \mathcal{L}(H, \mathcal{F}_{\mathcal{A}})$ that prevent the environment from fulfilling its liveness properties.

We suggest an alternative view to the usage of the assumptions on the environment $\mathcal{F}_{\mathcal{A}}$ in a GR(1) game. The condition $\mathcal{F}_{\mathcal{A}}$ can be interpreted abstractly as modeling an underlying mechanism that ensures that the environment player (player 0) generates only inputs (possibly in response to observed outputs) that conform with the given assumption. In this context, we would like to ensure that the system (player 1) allows the environment, as much as possible, to fulfill its liveness and only *restricts* the environment behavior if needed to enforce the guarantees. We achieve this by forcing the system player to ensure that the environment is always able to play such that it fulfills its liveness, i.e.

$$\mathrm{pfx}(\mathcal{L}(H, f^1)) = \mathrm{pfx}(\mathcal{L}(H, f^1) \cap \mathcal{L}(H, \mathcal{F}_{\mathcal{A}}))\,.$$

As the $\supseteq$-inclusion trivially holds, the constraint is given by the $\subseteq$-inclusion. Intuitively, the latter holds if every finite play α compliant with f^1 over H can be extended (by a suitable environment strategy) to an infinite play π compliant with f^1 that fulfills the environment liveness assumptions. It is easy to see that not every solution to the GR(1) game $(H, \mathcal{F}_\mathcal{A}, \mathcal{F}_\mathcal{G})$ (in the classical sense) supplies this additional requirement. We therefore propose to synthesize a system strategy f^1 with the above properties, as summarized in the following problem statement.

Problem 1. Given a GR(1) game $(H, \mathcal{F}_\mathcal{A}, \mathcal{F}_\mathcal{G})$ synthesize a system strategy f^1

$$\text{s.t.} \quad \emptyset \neq \mathcal{L}(H, f^1) \subseteq \overline{\mathcal{L}(H, \mathcal{F}_\mathcal{A})} \cup \mathcal{L}(H, \mathcal{F}_\mathcal{G}), \tag{7a}$$

$$\text{and} \quad \mathrm{pfx}(\mathcal{L}(H, f^1)) = \mathrm{pfx}(\mathcal{L}(H, f^1) \cap \mathcal{L}(H, \mathcal{F}_\mathcal{A})) \tag{7b}$$

both hold, or verify that no such system strategy exists. □

Problem 1 asks for a strategy f^1 s.t. every play π compliant with f^1 over H fulfills the system guarantees, i.e., $\pi \in \mathcal{L}(H, \mathcal{F}_\mathcal{G})$, if the environment fulfills its liveness properties, i.e., if $\pi \in \mathcal{L}(H, \mathcal{F}_\mathcal{A})$ (from (7a)), while the latter always remains possible (by a suitably playing environment) due to (7b). Inspired by algorithms solving the supervisory controller synthesis problem for non-terminating processes [24,28], we propose a solution to Problem 1 in terms of a vectorized 4-nested fixed-point in the remaining part of this paper. We show that Problem 1 can be solved by a finite-memory strategy, if a solution exists.

We note that (7b) is not a linear time but a branching time property and can therefore not be "compiled away" into a different GR(1) or even ω-regular objective. Satisfaction of (7b) requires checking whether the set $F_\mathcal{A}$ remains reachable from any reachable state in the game graph realizing $\mathcal{L}(H, f^1)$[3].

3 Algorithmic Solution for Singleton Winning Conditions

We first consider the GR(1) game $(H, \mathcal{F}_\mathcal{A}, \mathcal{F}_\mathcal{G})$ with singleton winning conditions $\mathcal{F}_\mathcal{A} = \{F_\mathcal{A}\}$ and $\mathcal{F}_\mathcal{G} = \{F_\mathcal{G}\}$, i.e., $n = m = 1$. It is well known that a system winning strategy f^1 for this game can be synthesized by solving a three color parity game over H. This can be expressed by the μ-calculus formula (see [15])

$$\varphi_3 := \nu Z \;.\; \mu Y \;.\; \nu X \;.\; (F_\mathcal{G} \cap \mathsf{Pre}^1(Z)) \cup \mathsf{Pre}^1(Y) \cup (Q \setminus F_\mathcal{A} \cap \mathsf{Pre}^1(X)). \tag{8}$$

It follows that $q_0 \in [\![\varphi_3]\!]$ if and only if the synthesis problem has a solution and the winning strategy f^1 is obtained from a ranking argument over the sets computed during the evaluation of (8).

To obtain a system strategy f^1 solving Problem 1 instead, we propose to extend (8) to a 4-nested fixed-point expressed by the μ-calculus formula

$$\begin{aligned} \varphi_4 = {} & \nu Z \;.\; \mu Y \;.\; \nu X \;.\; \mu W \;. \\ & (F_\mathcal{G} \cap \mathsf{Pre}^1(Z)) \;\cup\; \mathsf{Pre}^1(Y) \;\cup\; ((Q \setminus F_\mathcal{A}) \cap \mathsf{CondPre}(W, X \setminus F_\mathcal{A}))\,. \end{aligned} \tag{9}$$

Compared to (8) this adds an inner-most largest fixed-point and substitutes the last controllable pre-operator by the conditional one. Intuitively, this distinguishes between states from which player 1 can force visiting $F_\mathcal{G}$ and states from

[3] It can indeed be expressed by the CTL* formula $\mathsf{AGEF}F_\mathcal{A}$ (see [13], Sec. 3.3.2).

which player 1 can force avoiding $F_\mathcal{A}$. This is in contrast to (8) and allows to exclude strategies that allow player 1 to win by falsifying the assumptions.

The remainder of this section shows that $q_0 \in [\![\varphi_4]\!]$ if and only if Problem 1 has a solution and the winning strategy f^1 fulfilling (7) can be obtained from a ranking argument over the sets computed during the evaluation of (9).

Soundness

We prove soundness of (9) by showing that every state $q \in [\![\varphi_4]\!]$ is winning for the system player. In view of Problem 1 this requires to show that there exists a system strategy f^1 s.t. all plays starting in a state $q \in [\![\varphi_4]\!]$ and evolving in accordance to f^1 result in an infinite play that fulfills (7a) and (7b).

We start by defining f^1 from a ranking argument over the iterations of (9). Consider the last iteration of the fixed-point in (9) over Z. As (9) terminates after this iteration we have $Z = Z^\infty = [\![\varphi_4]\!]$. Assume that the fixed point over Y is reached after k iterations. If Y^i is the set obtained after the i-th iteration, we have that $Z^\infty = \bigcup_{i=0}^{k} Y^i$ with $Y^i \subseteq Y^{i+1}$, $Y^0 = \emptyset$ and $Y^k = Z^\infty$. Furthermore, let $X^i = Y^i$ denote the fixed-point of the iteration over X resulting in Y^i and denote by W^i_j the set obtained in the jth iteration over W performed while using the value X^i for X and Y^{i-1} for Y. Then it holds that $Y^i = X^i = \bigcup_{j=0}^{l_i} W^i_j$ with $W^i_j \subseteq W^i_{j+1}$, $W^i_0 = \emptyset$ and $W^i_{l_i} = Y^i$ for all $i \in [0;k]$.

Using these sets, we define a ranking for every state $q \in Z^\infty$ s.t.

$$\mathsf{rank}(q) = (i,j) \text{ iff } q \in \left(Y^i \setminus Y^{i-1}\right) \cap \left(W^i_j \setminus W^i_{j-1}\right) \text{ for } i,j > 0. \qquad (10)$$

We order ranks lexicographically. It further holds that (see [22])

$$q \in D \;\Leftrightarrow\; \mathsf{rank}(q) = (1,1) \qquad \Leftrightarrow\; q \in F_\mathcal{G} \cap Z^\infty \qquad (11a)$$

$$q \in E^i \;\Leftrightarrow\; \mathsf{rank}(q) = (i,1) \wedge i > 1 \qquad \Leftrightarrow\; q \in (F_\mathcal{A} \setminus F_\mathcal{G}) \cap Z^\infty \qquad (11b)$$

$$q \in R^i_j \;\Leftrightarrow\; \mathsf{rank}(q) = (i,j) \wedge j > 1 \qquad \Leftrightarrow\; q \in (Z^\infty \setminus (F_\mathcal{A} \cup F_\mathcal{G})), \qquad (11c)$$

where D, E^i and R^i_j denote the sets *added* to the winning state set by the first, second and third term of (9), respectively, in the corresponding iteration.

Figure 3 (left) shows a schematic representation of this construction for an example with $k = 3$, $l_1 = 4$, $l_2 = 2$ and $l_3 = 3$. The set $D = F_\mathcal{G} \cap Z^\infty$ is represented by the diamond at the top where the label $(1,1)$ denotes the associated rank (see (11a)). The ellipses represent the sets $E^i \subseteq (F_\mathcal{A} \setminus F_\mathcal{G}) \cap Z^\infty$, where the corresponding $i > 1$ is indicated by the associated rank $(i,1)$. Due to the use of the controllable pre-operator in the first and second term of (9), it is ensured that progress out of D and E^i can be enforced by the system, indicated by the solid arrows. This is in contrast to all states in $R^i_j \subseteq Z^\infty \setminus F_\mathcal{A} \setminus F_\mathcal{G}$, which are represented by the rectangular shapes in Figure 3 (left). These states allow the environment to increase the ranking (dashed lines) as long as $Z^\infty \setminus F_\mathcal{A} \setminus F_\mathcal{G}$ is not left and there exists a possible move to decrease the j-rank (dotted lines). While this does not strictly enforce progress, we see that whenever the environment plays such that states in $F_\mathcal{A}$ (i.e., the ellipses) are visited infinitely often (i.e., the environment fulfills its assumptions), the system can enforce progress w.r.t. the

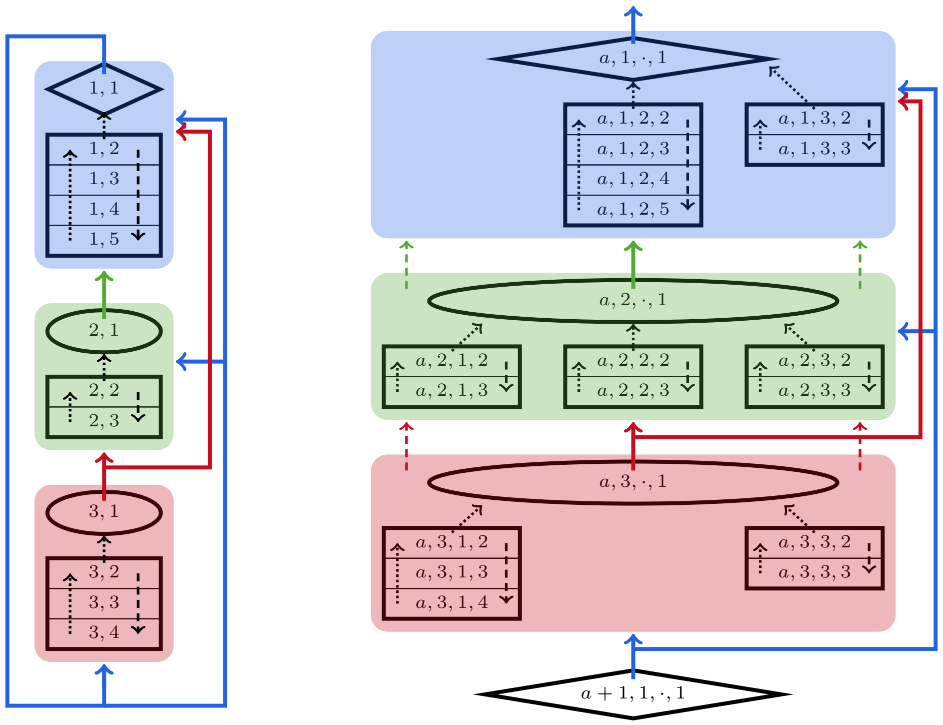

Figure 3. Schematic representation of the ranking defined in (10) (left) and in (16) (right). Diamond, ellipses and rectangles represent the sets D, E^i and R^i_j, while blue, green and red indicate the sets Y^1, $Y^2 \setminus Y^1$ and $Y^3 \setminus Y^2$ (annotated by ${}^a/{}^{ab}$ for the right figure). Labels (i, j) and (a, i, b, j) indicate that all states q associated with this set fulfill $\mathsf{rank}(q) = (i, j)$ and ${}^{ab}\mathsf{rank}(q) = (i, j)$, respectively. Solid, colored arcs indicate system-enforceable moves, dotted arcs indicate existence of environment or system transitions and dashed arcs indicate possible existence of environment transitions.

defined ranking and states in $F_\mathcal{G}$ (i.e., the diamond shape) is eventually visited. The system is restricted to take the existing solid or dotted transitions in Figure 3 (left). With this, it is easy to see that the constructed strategy is winning if the environment fulfills its assumptions, i.e., (7a) holds. However, to ensure that (7b) also holds, we need an additional requirement. This is necessary as the used construction also allows plays to cycle through the blue region of Figure 3 (left) only, and by this not surely visiting states in $F_\mathcal{A}$ infinitely often. However, if $\mathcal{L}(H, F_\mathcal{G}) \subseteq \mathcal{L}(H, F_\mathcal{A})$ we see that (7b) holds as well. It should be noted that the latter is a sufficient condition which can be easily checked symbolically on the problem instance but not a necessary one.

Based on the ranking in (10) we define a memory-less system strategy $f^1 : Q^1 \cap Z^\infty \to Q^0 \subseteq \delta^1$ s.t. the rank is always decreased, i.e.,

$$q' = f^1(q) \Rightarrow \begin{cases} \mathsf{rank}(q') < \mathsf{rank}(q), & \mathsf{rank}(q) > (1,1) \\ q' \in Z^\infty, & \text{otherwise} \end{cases}. \tag{12}$$

The next theorem shows that this strategy indeed solves Problem 1.

Theorem 1. *Let $(H, \mathcal{F}_{\mathcal{A}}, \mathcal{F}_{\mathcal{G}})$ be a GR(1) game with singleton winning conditions $\mathcal{F}_{\mathcal{A}} = \{F_{\mathcal{A}}\}$ and $\mathcal{F}_{\mathcal{G}} = \{F_{\mathcal{G}}\}$. Suppose f^1 is the system strategy in* (12) *based on the ranking in* (10). *Then it holds for all $q \in [\![\varphi_4]\!]$ that*[4]

$$\mathcal{L}_q(H, f^1) \subseteq \overline{\mathcal{L}_q(H, \mathcal{F}_{\mathcal{A}})} \cup \mathcal{L}_q(H, \mathcal{F}_{\mathcal{G}}), \tag{13a}$$

$$\mathcal{L}_q(H, f^1) \cap \mathcal{L}_q(H, \mathcal{F}_{\mathcal{G}}) \neq \emptyset, \text{ and} \tag{13b}$$

$$\mathcal{L}_q(H, \mathcal{F}_{\mathcal{G}}) \subseteq \mathcal{L}_q(H, \mathcal{F}_{\mathcal{A}}) \Rightarrow \mathrm{pfx}(\mathcal{L}_q(H, f^1)) = \mathrm{pfx}(\mathcal{L}_q(H, f^1) \cap \mathcal{L}_q(H, \mathcal{F}_{\mathcal{A}})). \tag{13c}$$

Completeness
We show completeness of (9) by establishing that every state $q \in Q \setminus [\![\varphi_4]\!] = [\![\overline{\varphi}_4]\!]$ is losing for the system player. In view of Problem 1 this requires to show that for all $q \in [\![\overline{\varphi}_4]\!]$ and all system strategies f^1 either (7a) or (7b) does not hold. This is formalized in [22] by first negating the fixed-point in (9) and deriving the induced ranking of this negated fixed-point. Using this ranking, we first show that the environment can (i) render the negated winning set $\overline{Z}^{\infty}$ invariant and (ii) can always enforce the play to visit $F_{\mathcal{G}}$ only finitely often, resulting in a violation of the guarantees. Using these observations we finally show that whenever (7a) holds for an arbitrary system strategy f^1 starting in $[\![\overline{\varphi}_4]\!]$, then (7b) cannot hold. With this, completeness, as formalized in the following theorem, directly follows.

Theorem 2. *Let $(H, \mathcal{F}_{\mathcal{A}}, \mathcal{F}_{\mathcal{G}})$ be a GR(1) game with singleton winning conditions $\mathcal{F}_{\mathcal{A}} = \{F_{\mathcal{A}}\}$ and $\mathcal{F}_{\mathcal{G}} = \{F_{\mathcal{G}}\}$. Then it holds for all $q \in [\![\overline{\varphi}_4]\!]$ and all system strategies f^1 over H that either*

$$\emptyset \neq \mathcal{L}_q(H, f^1) \subseteq \overline{\mathcal{L}_q(H, \mathcal{F}_{\mathcal{A}})} \cup \mathcal{L}_q(H, \mathcal{F}_{\mathcal{G}}), \text{ or} \tag{14a}$$

$$\mathrm{pfx}(\mathcal{L}_q(H, f^1)) = \mathrm{pfx}(\mathcal{L}_q(H, f^1) \cap \mathcal{L}_q(H, \mathcal{F}_{\mathcal{A}})) \text{ does not hold.} \tag{14b}$$

A Solution for Problem 1
We note that the additional assumption in Theorem 1 is required only to ensure that the resulting strategy fulfills (7b). Suppose that this assumption holds for the initial state q_0 of H. That is, consider a GR(1) game $(H, \mathcal{F}_{\mathcal{A}}, \mathcal{F}_{\mathcal{G}})$ with singleton winning conditions $\mathcal{F}_{\mathcal{A}} = \{F_{\mathcal{A}}\}$ and $\mathcal{F}_{\mathcal{G}} = \{F_{\mathcal{G}}\}$ s.t. $\mathcal{L}(H, F_{\mathcal{G}}) \subseteq \mathcal{L}(H, F_{\mathcal{A}})$. Then it follows from Theorem 2 that Problem 1 has a solution iff $q_0 \in [\![\varphi_4]\!]$. Furthermore, if $q_0 \in [\![\varphi_4]\!]$, based on the intermediate values maintained for the computation of φ_4 in (10) and the ranking defined in (12), we can construct f^1 that wins the GR(1) condition in (7a) and is non-conflicting, as in (7b).

We can check symbolically whether $\mathcal{L}(H, F_{\mathcal{G}}) \subseteq \mathcal{L}(H, F_{\mathcal{A}})$. For this we construct a game graph H' from H by removing all states in $F_{\mathcal{A}}$, and then check whether $\mathcal{L}(H', F_{\mathcal{G}})$ is empty. The latter is decidable in logarithmic space and polynomial time. If this check fails, then $\mathcal{L}(H, F_{\mathcal{G}}) \not\subseteq \mathcal{L}(H, F_{\mathcal{A}})$. Furthermore, we can replace $\mathcal{L}(H, \mathcal{F}_{\mathcal{G}})$ in (7a) by $\mathcal{L}(H, \mathcal{F}_{\mathcal{G}}) \cap \mathcal{L}(H, \mathcal{F}_{\mathcal{A}})$ without affecting the restriction (7a) imposes on the choice of f^1. Given singleton winning conditions $F_{\mathcal{G}}$ and $F_{\mathcal{A}}$, we see that $\mathcal{L}(H, F_{\mathcal{G}}) \cap \mathcal{L}(H, F_{\mathcal{A}}) = \mathcal{L}(H, \{F_{\mathcal{G}}, F_{\mathcal{A}}\})$ and it trivially

[4] Given a state $q \in Q = Q^0 \cup Q^1$ we use the subscript q to denote that the respective set of plays is defined by using q as the initial state of H.

holds that $\mathcal{L}(H, \{F_{\mathcal{G}}, F_{\mathcal{A}}\}) \subseteq \mathcal{L}(H, F_{\mathcal{A}})$. That is, we fulfill the conditional by replacing the system guarantee $\mathcal{L}(H, \mathcal{F}_{\mathcal{G}})$ by $\mathcal{L}(H, \{F_{\mathcal{G}}, F_{\mathcal{A}}\})$. However, this results in a GR(1) synthesis problem with $m = 1$ and $n = 2$, which we discuss next.

4 Algorithmic Solution for GR(1) Winning Conditions

We now consider a general GR(1) game $(H, \mathcal{F}_{\mathcal{A}}, \mathcal{F}_{\mathcal{G}})$ with $\mathcal{F}_{\mathcal{A}} = \{{}^1F_{\mathcal{A}}, \ldots, {}^mF_{\mathcal{A}}\}$ and $\mathcal{F}_{\mathcal{G}} = \{{}^1F_{\mathcal{G}}, \ldots, {}^nF_{\mathcal{G}}\}$ s.t. $n, m > 1$. The known fixed-point for solving GR(1) games in [6] rewrites the three nested fixed-point in (8) in a vectorized version, which induces an order on the guarantee sets in $\mathcal{F}_{\mathcal{G}}$ and adds a disjunction over all assumption sets in $\mathcal{F}_{\mathcal{A}}$ to every line of this vectorized fixed-point. Adapting the same idea to the 4-nested fixed-point algorithm (9) results in

$$\varphi_4 = \nu \begin{bmatrix} {}^1Z \\ {}^2Z \\ \vdots \\ {}^nZ \end{bmatrix} . \begin{bmatrix} \mu\ {}^1Y\ .\ \left(\bigvee_{b=1}^m\ \nu\ {}^{1b}X\ .\ \mu\ {}^{1b}W\ {}^{1b}\Omega\right) \\ \mu\ {}^2Y\ .\ \left(\bigvee_{b=1}^m\ \nu\ {}^{2b}X\ .\ \mu\ {}^{2b}W\ {}^{2b}\Omega\right) \\ \vdots \\ \mu\ {}^nY\ .\ \left(\bigvee_{b=1}^m\ \nu\ {}^{nb}X\ .\ \mu\ {}^{nb}W\ {}^{nb}\Omega\right) \end{bmatrix}, \tag{15}$$

where, ${}^{ab}\Omega = ({}^aF_{\mathcal{G}} \cap \mathsf{Pre}^1({}^{a^+}Z)) \cup \mathsf{Pre}^1({}^aY) \cup (Q \setminus {}^bF_{\mathcal{A}} \cap \mathsf{CondPre}(W, X \setminus {}^bF_{\mathcal{A}}))$ and a^+ denotes $(a \mod n) + 1$.

The remainder of this section shows how soundness and completeness carries over from the 4-nested fixed-point algorithm (9) to its vectorized version in (15).

Soundness and Completeness
We refer to intermediate sets obtained during the computation of the fixpoints by similar notations as in Section 3. For example, the set ${}^aY^i$ is the i-th approximation of the fixpoint computing ${}^aY$ and ${}^{ab}W_j^i$ is the j-th approximation of ${}^{ab}W$ while computing the i-th approximation of ${}^aY$, i.e., computing ${}^aY^i$ and using ${}^aY^{i-1}$. Similar to the above, we define a mode-based rank for every state $q \in {}^aZ^\infty$; we track the currently chased guarantee $a \in [1; n]$ (similar to [6]) and the currently avoided assumption set $b \in [1, m]$ as an additional internal mode. In analogy to (10) we define

$${}^{ab}\mathsf{rank}(q) = (i, j) \text{ iff } q \in \left({}^aY^i \setminus {}^aY^{i-1}\right) \cap \left({}^{ab}W_j^i \setminus {}^{ab}W_{j-1}^i\right) \text{ for } i, j > 0. \tag{16}$$

Again, we order ranks lexicographically, and, in analogy to (11), we have

$$q \in {}^aD \Leftrightarrow {}^{a\cdot}\mathsf{rank}(q) = (1, 1) \quad \Rightarrow q \in {}^aF_{\mathcal{G}}, \tag{17a}$$

$$q \in {}^aE^i \Leftrightarrow {}^{a\cdot}\mathsf{rank}(q) = (i, 1) \wedge i > 1, \tag{17b}$$

$$q \in {}^{ab}R_j^i \Leftrightarrow {}^{ab}\mathsf{rank}(q) = (i, j) \wedge j > 1 \quad \Rightarrow q \notin {}^bF_{\mathcal{A}}. \tag{17c}$$

The sets ${}^aY^i$, ${}^{ab}W_j^i$, ${}^aD$, ${}^aE^i$ and ${}^{ab}R_j^i$ are interpreted in direct analogy to Section 3, where a and b annotate the used line and conjunct in (15).

Figure 3 (right) shows a schematic representation of the ranking for an example with ${}^ak = 3$, ${}^{a1}l_1 = 0$, ${}^{a2}l_1 = 4$, ${}^{a3}l_1 = 2$, ${}^{a\cdot}l_2 = 2$, ${}^{a1}l_3 = 3$, ${}^{a2}l_3 = 0$, and ${}^{a3}l_3 = 2$. Again, the set ${}^{a}D \subseteq {}^{a}F_{\mathcal{G}}$ is represented by the diamond at the top of the figure. Similarly, all ellipses represent sets ${}^{a}E^{i}$ added in the i-th iteration over line a of (15). Again, progress out of ellipses can be enforced by the system, indicated by the solid arrows leaving those shapes. However, this might not preserve the current b mode. It might be the environment choosing which assumption to avoid next. Further, the environment might choose to change the b mode along with decreasing the i-rank, as indicated by the colored dashed lines[5]. Finally, the interpretation of the sets represented by rectangular shapes in Figure 3 (right), corresponding to (17c), is in direct analogy to the case with singleton winning conditions. It should be noticed that this is the only place where we preserve the current b-mode when constructing a strategy.

Using this intuition we define a system strategy that uses enforceable and existing transitions to decrease the rank if possible and preserves the current a mode until the diamond shape is reached. The b mode is only preserved within rectangular sets. This is formalized by a strategy

$$f^1 : \bigcup_{a\in[1;n]} \big((Q^1 \cap {}^{a}Z^{\infty}) \times a \times [1;m]\big) \to Q^0 \times [1;n] \times [1;m] \tag{18a}$$

s.t. $(q', \cdot, \cdot) = f^1(q, \cdot, \cdot)$ implies $q' \in \delta^1(q)$ and $(q', a', b') = f^1(q, a, b)$ implies

$$\begin{cases} q' \in {}^{a^+}Z^{\infty} \wedge a' = a^+, & {}^{ab}\mathsf{rank}(q) = (1,1) \\ {}^{a'b'}\mathsf{rank}(q') \leq (i-1,\cdot) \wedge a' = a, & {}^{ab}\mathsf{rank}(q) = (i,1), i > 1 \\ {}^{a'b'}\mathsf{rank}(q') \leq (i, j-1) \wedge a' = a \wedge b' = b, & {}^{ab}\mathsf{rank}(q) = (i,j), j > 1 \end{cases} . \tag{18b}$$

We say that a play π over H is compliant with f^1 if there exist mode traces $\alpha \in [1;n]^{\omega}$ and $\beta \in [1;m]^{\omega}$ s.t. for all $k \in \mathbb{N}$ holds $(\pi(2k+2), \alpha(2k+2), \beta(2k+2)) = f^1(\pi(2k+1), \alpha(2k+1), \beta(2k+1))$, and (i) $\alpha(2k+1) = \alpha(2k)^+$ if ${}^{ab}\mathsf{rank}(\pi(2k+1)) = (1,1)$, (ii) $\alpha(2k+1) = \alpha(2k)$ if ${}^{ab}\mathsf{rank}(\pi(2k+1)) = (i,1), i > 1$, and (iii) $\alpha(2k+1) = \alpha(2k)$ and $\beta(2k+1) = \beta(2k)$ if ${}^{ab}\mathsf{rank}(\pi(2k+1)) = (i,j), j > 1$.

With this it is easy to see that the intuition behind Theorem 1 directly carries over to every line of (15). Additionally, using $\mathsf{Pre}^1({}^{a^+}Z)$ in ${}^{a}D$ allows to cycle through all the lines of (15), which ensures that every set ${}^{a}F_{\mathcal{G}} \in \mathcal{F}_{\mathcal{G}}$ is tried to be attained by the constructed system strategy in a pre-defined order. See [22] for a formalization of this intuition and a detailed proof.

To prove completeness, it is also shown in [22] that the negation of (15) can be over-approximated by negating every line separately. Therefore, the reasoning for every line of the negated fixed-point carries over from Section 3, resulting in the analogous completeness result. With this we obtain soundness and completeness in direct analogy to Theorem 1-2, formalized in Theorem 3.

Theorem 3. *Let $(H, \mathcal{F}_{\mathcal{A}}, \mathcal{F}_{\mathcal{G}})$ be a GR(1) game with $\mathcal{F}_{\mathcal{A}} = \{{}^{1}F_{\mathcal{A}}, \ldots, {}^{m}F_{\mathcal{A}}\}$ and $\mathcal{F}_{\mathcal{G}} = \{{}^{1}F_{\mathcal{G}}, \ldots, {}^{n}F_{\mathcal{G}}\}$. Suppose f^1 is the system strategy in* (18) *based on the ranking in* (16). *Then it holds for all $q \in [\![\varphi_4^v]\!]$ that* (13) *holds. Furthermore, it holds for all $q \notin [\![\varphi_4^v]\!]$ and all system strategies f^1 over H that either* (14a) *or* (14b) *does not hold.*

[5] The strategy extraction in (18) prevents the system from choosing a different b mode. The strategy choice could be optimized w.r.t. fast progress towards ${}^{a}F_{\mathcal{G}}$ in such cases.

A Solution for Problem 1
Given that $\mathcal{L}(H,\mathcal{F}_\mathcal{G}) \subseteq \mathcal{L}(H,\mathcal{F}_\mathcal{A})$ it follows from Theorem 3 that Problem 1 has a solution iff $q_0 \in [\![\varphi_4^v]\!]$. Furthermore, if $q_0 \in [\![\varphi_4^v]\!]$ we can construct f^1 that wins the GR(1) condition in (7a) and is non-conflicting, as in (7b).

Using a similar construction as in Section 3, we can symbolically check whether $\mathcal{L}(H,\mathcal{F}_\mathcal{G}) \subseteq \mathcal{L}(H,\mathcal{F}_\mathcal{A})$. For this, we construct a new game graph H_b for every ${}^bF_\mathcal{A}$, $b \in [1;m]$ by removing the latter set from the state set of H and checking whether $\mathcal{L}(H_b,\mathcal{F}_\mathcal{G})$ is empty. If some of these m checks fail, we have $\mathcal{L}(H,\mathcal{F}_\mathcal{G}) \not\subseteq \mathcal{L}(H,\mathcal{F}_\mathcal{A})$. Now observe that by checking every ${}^bF_\mathcal{A}$ separately, we know which goals are not necessarily passed by infinite runs which visit all ${}^aF_\mathcal{G}$ infinitely often and can collect them in the set $\mathcal{F}_\mathcal{A}^{\text{failed}}$. Using the same reasoning as in Section 3, we can simply add the set $\mathcal{F}_\mathcal{A}^{\text{failed}}$ to the system guarantee set to obtain an equivalent synthesis problem which is solvable by the given algorithm, if it is realizable. More precisely, consider the new system guarantee set $\mathcal{F}'_\mathcal{G} = \mathcal{F}_\mathcal{G} \cup \mathcal{F}_\mathcal{A}^{\text{failed}}$ and observe that $\mathcal{L}(H,\mathcal{F}'_\mathcal{G}) \subseteq \mathcal{L}(H,\mathcal{F}_\mathcal{A})$ by definition, and therefore substituting $\mathcal{L}(H,\mathcal{F}_\mathcal{G})$ by $\mathcal{L}(H,\mathcal{F}'_\mathcal{G})$ in (7a) does not change the satisfaction of the given inclusion.

5 Complexity Analysis

We show that the search for a more elaborate strategy does not affect the worst case complexity. In Section 6 we show that this is also the case in practice. We state this complexity formally below.

Theorem 4. *Let $(H,\mathcal{F}_\mathcal{A},\mathcal{F}_\mathcal{G})$ be a GR(1) game. We can check whether there is a winning non-conflicting strategy f^1 by a symbolic algorithm that performs $O(|Q|^2|\mathcal{F}_\mathcal{G}||\mathcal{F}_\mathcal{A}|)$ next step computations and by an enumerative algorithm that works in time $O(m|Q|^2|\mathcal{F}_\mathcal{G}||\mathcal{F}_\mathcal{A}|)$, where m is the number of transitions of the game.*

Proof. Each line of the fixed-point is iterated $O(|Q|^2)$ times [8]. As there are $|\mathcal{F}_\mathcal{G}||\mathcal{F}_\mathcal{A}|$ lines the upper bound follows. As we have to compute $|\mathcal{F}_\mathcal{G}||\mathcal{F}_\mathcal{A}|$ different ranks for each state, it follows that the complexity is $O(m|Q|^2|\mathcal{F}_\mathcal{G}||\mathcal{F}_\mathcal{A}|)$. □

We note that *enumeratively* our approach is theoretically worse than the classical approach to GR(1). This follows from the straight forward reduction to the rank computation in the rank lifting algorithm and the relative complexity of the new rank when compared to the general GR(1) rank. We conjecture that more complex approaches, e.g., through a reduction to a parity game and the usage of other enumerative algorithms, could eliminate this gap.

6 Experiments

We have implemented the 4-nested fixed-point algorithm in (15) and the corresponding strategy extraction in (18). It is available as an extension to the GR(1) synthesis tool `slugs` [14]. In this section we show how this algorithm (called 4FP) performs in comparison to the usual 3-nested fixed-point algorithm for GR(1) synthesis (called 3FP) available in `slugs`. All experiments were run on a computer with an Intel i5 processor running an x86 Linux at 2 GHz with 8 GB of memory.

We first run both algorithms on a benchmark set obtained from the maze example in the introduction by changing the number of rows and columns of the maze. We first increased the number of lines in the maze and added a goal state for both the obstacle and the robot per line. This results in a maze where in the first and last column, system and environment goals alternate and all adjacent cells are separated by a horizontal wall. Hence, both players need to cross the one-cell wide white space in the middle infinitely often to visit all their goal states infinitely often. The computation times and the number of states in the resulting strategy are shown in Table 1, upper part, column 3-6. Interestingly, we see that the 3FP always returns a strategy that blocks the environment. In contrast, the non-conflicting strategies computed by the 4FP are relatively larger (in state size) and computed about 10 times slower compared to the 3FP (compare column 3-4 and 5-6). When increasing the number of columns instead (lower part of Table 1), the number of goals is unaffected. We made the maze wider and left only a one-cell wide passage in the middle of the maze to allow crossings between its upper and lower row. Still, the 3FP only returns strategies that falsify the assumption, which have fewer states and are computed much faster than the environment respecting strategy returned by the 4FP. Unfortunately, the speed of computing a strategy or its size is immaterial if the winning strategy so computed wins only by falsifying assumptions.

To rule out the discrepancy between the two algorithms w.r.t. the size of strategies, we slightly modified the above maze benchmark s.t. the environment assumptions are not falsifiable anymore. We increased the capabilities of the obstacle by allowing it to move at most 2 steps in each round and to "jump over" the robot. Under these assumptions we repeated the above experiments. The computation times and the number of states in the resulting strategy are shown in Table 1, column 9-12. We see, that in this case the size of the strategies computed by the two algorithms are more similar. The larger number for the 4FP is due to the fact that we have to track both the a and the b mode, possibly

Table 1. Experimental results for the maze benchmark. The size of the maze is given in columns/lines, the number of goals is given per player. The states are counted for the returned winning strategies. Strategies preventing the environment from fulfilling its goals are indicated by a *. Recorded computation times are rounded wall-clock times.

		falsifiable assumptions						non-falsifiable assumptions					
		3FP		4FP		Heuristic		3FP		4FP		Heuristic	
size	goals	states	time	states	time	states	time	states	time	states	time	states	time
3/2	2	10^*	< 1s	46	< 1s	12	< 1s	35	< 1s	50	< 1s	40	< 1s
3/10	10	34^*	< 1s	1401	8s	1307	3s	1119	1s	1513	13s	1533	5s
3/20	20	64^*	21s	5799	201s	5732	337s	3926	37s	6000	163s	6378	105s
25/2	2	94^*	< 1s	2144	4s	n.r.	6s	744	< 1s	2318	4s	n.r.	5s
63/2	2	397^*	< 1s	14259	32s	n.r.	101s	4938	2s	15465	54s	n.r.	66s

resulting in multiple copies of the same a-mode state. We see that the state difference decreases with the number of goals (upper part of Table 1, column 9-12) and increases with the number of (non-goal) states (lower part of Table 1, column 9-12). In both cases, the 3FP still computes faster, but the difference decreases with the number of goals.

In addition to the 3FP and the 4FP we have also tested a sound but incomplete heuristic, which avoids the disjunction over all b's in every line of (15) by only investigating $a = b$. The state count and computation times for this heuristic are shown in Table 1, column 7-8 for the original maze benchmark, and in column 13-14 for the modified one. We see that in both cases the heuristic only returns a winning strategy if the maze is not wider then 3 cells. This is due to the fact that in all other cases the robot cannot prevent the obstacle from attaining a particular assumption state until the robot has moved from one goal to the next. The 4FP handles this problem by changing between avoided assumptions in between visits to different goals. Intuitively, the computation times and state counts for the heuristic should be smaller then for the 4FP, as the exploration of the disjunction over b's is avoided, which is true for many scenarios of the considered benchmark. It should however be noted that this is not always the case (compare e.g. line 3, column 6 and 8). This stems from the fact that restricting the synthesis to avoiding one particular assumption might require more iterations over W and Y within the fixed-point computation.

7 Discussion

We believe the requirement that a winning strategy be *non-conflicting* is a simple way to disallow strategies that win by actively preventing the environment from satisfying its assumptions, without significantly changing the theoretical formulation of reactive synthesis (e.g., by adding different winning conditions or new notions of equilibria). It is not a trace property, but our main results show that adding this requirement retains the algorithmic niceties of GR(1) synthesis: in particular, symbolic algorithms have the same asymptotic complexity.

However, non-conflictingness makes the implicit assumption of a "maximally flexible" environment: it is possible that because of unmodeled aspects of the environment strategy, it is not possible for the environment to satisfy its specifications in the precise way allowed by a non-conflicting strategy. In the maze example discussed in Section 1, the environment needs to move the obstacle to precisely the goal cell which is currently rendered reachable by the system. If the underlying dynamics of the obstacle require it to go back to the lower left from state q_3 before proceeding to the upper right (e.g., due to a required battery recharge), the synthesized robot strategy prevents the obstacle from doing so.

Finally, if there is no non-conflicting winning strategy, one could look for a "minimally violating" strategy. We leave this for future work. Additionally, we leave for future work the consideration of non-conflictingness for general LTL specifications or (efficient) fragments thereof.

References

1. S. Almagor, O. Kupferman, J. Ringert, and Y. Velner. Quantitative assume guarantee synthesis. In *Computer Aided Verification (CAV)*, volume 10427 of *Lecture Notes in Computer Science*, pages 353–374. Springer, 2017.
2. R. Bloem, K. Chatterjee, K. Greimel, T. Henzinger, G. Hofferek, B. Jobstmann, B. Könighofer, and R. Könighofer. Synthesizing robust systems. *Acta Informatika*, 51(3-4):193–220, 2014.
3. R. Bloem, K. Chatterjee, T. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In *Computer Aided Verification (CAV)*, volume 5643 of *Lecture Notes in Computer Science*, pages 140–156. Springer, 2009.
4. R. Bloem, R. Ehlers, S. Jacobs, and R. Könighofer. How to handle assumptions in synthesis. In *SYNT'14, Vienna, Austria*, pages 34–50, 2014.
5. R. Bloem, R. Ehlers, and R. Könighofer. Cooperative reactive synthesis. In *ATVA 2015, Shanghai, China*, pages 394–410, 2015.
6. R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sahar. Synthesis of reactive(1) designs. *Journal of Computer and System Sciences*, 78(3):911 – 938, 2012.
7. R. Brenguier, J.-F. Raskin, and O. Sankur. Assume-admissible synthesis. *Acta Informatica*, 54(1):41–83, Feb 2017.
8. A. Browne, E. Clarke, S. Jha, D. Long, and W. Marrero. An improved algorithm for the evaluation of fixpoint expressions. *Theor. Comput. Sci.*, 178(1-2):237–255, 1997.
9. K. Chatterjee and T. A. Henzinger. Assume-guarantee synthesis. In *TACAS*, Lecture Notes in Computer Science, 2007.
10. K. Chatterjee, F. Horn, and C. Löding. Obliging games. In *Concur*, Lecture Notes in Computer Science, pages 284–296. Springer, 2010.
11. N. D'Ippolito, V. Braberman, N. Piterman, and S. Uchitel. Synthesis of live behavior models. In *18th International Symposium on Foundations of Software Engineering*, pages 77–86. ACM, 2010.
12. R. Ehlers, R. Könighofer, and R. Bloem. Synthesizing cooperative reactive mission plans. In *IROS*, pages 3478–3485, 2015.
13. R. Ehlers, S. Lafortune, S. Tripakis, and M. Y. Vardi. Supervisory control and reactive synthesis: a comparative introduction. *Discrete Event Dynamic Systems*, 27(2):209–260, 2017.
14. R. Ehlers and V. Raman. Slugs: Extensible GR(1) synthesis. In *CAV'16*, pages 333–339, 2016.
15. E. Emerson and C. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS'91*, pages 368–377, Oct 1991.
16. D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *TACAS 2010: Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 6015, pages 190–204. Springer, 2010.
17. D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *TACAS'10*, pages 190–204, 2010.
18. B. Johnson, F. Havlak, H. Kress-Gazit, and M. Campbell. Experimental evaluation and formal analysis of high-level tasks with dynamic obstacle anticipation on a full-sized autonomous vehicle. *Journal of Field Robotics*, 2017.
19. U. Klein and A. Pnueli. Revisiting synthesis of GR(1) specifications. In *6th International Haifa Verification Conference*, volume 6504 of *Lecture Notes in Computer Science*, pages 161–181. Springer, 2010.

20. D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.
21. O. Kupferman, G. Perelli, and M. Vardi. Synthesis with rational environments. *Ann. Math. Artif. Intell.*, 78(1):3–20, 2016.
22. R. Majumdar, N. Piterman, and A.-K. Schmuck. Environmentally-friendly GR(1) synthesis (extended version). *arXiv preprint*, 2019.
23. T. Moor. Supervisory control on non-terminating processes: An interpretation of liveness properties. Technical report, Lehrstuhl für Regelungstechnik, Friedrich-Alexander Universität Erlangen-Nürnberg, 2017.
24. P. J. Ramadge. Some tractable supervisory control problems for discrete-event systems modeled by Büchi automata. *IEEE Transactions on Automatic Control*, 34:10–19, 1989.
25. R. Rogersten, H. Xu, N. Ozay, U. Topcu, and R. M. Murray. Control software synthesis and validation for a vehicular electric power distribution testbed. *J. Aerospace Inf. Sys.*, 11(10):665–678, 2014.
26. A.-K. Schmuck, T. Moor, and R. Majumdar. On the relation between reactive synthesis and supervisory control of non-terminating processes. 2018. WODES'18.
27. H. Seidl. Fast and simple nested fixpoints. *Inf. Process. Lett.*, 59(6):303–308, 1996.
28. J. G. Thistle and W. M. Wonham. Supervision of infinite behavior of discrete event systems. *SIAM J. Control and Optimization*, 32:1098–1113, 1994.
29. H. Xu, U. Topcu, and R. M. Murray. Specification and synthesis of reactive protocols for aircraft electric power distribution. *IEEE Transactions on Control of Network Systems*, 2(2):193–203, 2015.