Identifying and exploiting tolerance to unexpected jumps in synthesized strategies for GR(1) specifications

Sumanth Dathathri      Scott C. Livingston      Richard M. Murray

Abstract—When used as part of a hybrid controller, finite-memory strategies synthesized from LTL specifications rely on an accurate dynamics model in order to ensure correctness of trajectories. In the presence of uncertainty about this underlying model, there may exist unexpected trajectories that manifest as unexpected transitions under control of the strategy. While some disturbances can be captured by augmenting the dynamics model, such approaches may be conservative in that bisimulations may fail to exist for which strategies can be synthesized. In this paper, we characterize the tolerance of such hybrid controllers - synthesized for generalized reactivity(1) specifications - to disturbances that appear as unexpected jumps (transitions) to states in the discrete strategy part of the controller. As a first step, we show robustness to certain unexpected transitions that occur in a finite-manner, i.e., despite a certain number of unexpected jumps, the sequence of states obtained will still meet a stricter specification and hence the original specification. Additionally, we propose algorithms to improve robustness by increasing tolerance to additional disturbances. A robot gridworld example is presented to demonstrate the application of the developed ideas and also to obtain empirical computational and memory cost estimates.

I. INTRODUCTION

The ability of strategies synthesized from formal specifications to be tolerant to unexpected perturbations (disturbances or uncertainty or unexpected failures) is important - more so for safety-critical applications. This is an area of concern with reactive strategies because they are not error-resilient. Even with disturbances that are not critical to the system, but were not accurately modeled during synthesis, no guarantees can be provided about satisfaction of the temporal-formula used for synthesis. Though sometimes these uncertainties can be modelled through the dynamics, it may be the case that it is not possible to synthesize a winning strategy with the uncertainty.

After a disturbance, if resynthesis is done from the perturbed point, there are no current results that provide guarantees about the execution with segments from two separate strategies. In this paper, we make progress towards enhancing the tolerance of strategies synthesized to satisfy specifications in the generalized reactivity(1) (GR(1)) fragment of linear-temporal logic (LTL) [12], [13]. GR(1) formulae are considered because they are quite expressive in terms of temporal properties captured, yet symbolic synthesis algorithms are possible if relatively low computational complexity [3], [7], [11]. The first result we show is that by trivially refining a strategy synthesized to satisfy a GR(1)

 formula, a strategy that is robust to certain unexpected perturbations and guarantees i.e winning against a stricter formula can be generated. Then, exploiting this tolerance, we propose multiple algorithms that combine separately synthesized strategies to form a single robust winning strategy. It is often desired that the system can recover from these glitches (uncertainties/noise) and function normally and this be done without resynthesizing the entire strategy again. It is also desirable that the strategies allow for recovery from faults whenever possible. In this regard, we propose one such approach which lets us recover from glitches without a complete resynthesis.

Understanding the behavior of systems to disturbances and uncertainties has been extensively studied in control theory and more recently, for reactive controllers and their synthesis. In [10], [15], [14], the robustness considered is in terms of bounded input-output deviation. This relates directly to the prevalent notion in control for robustness[17], where controllers are designed to ensure bounded disturbances lead to bounded deviations from nominal-behavior for the system. In this work, the tolerance to disturbances is in the form of satisfaction of a formula representing the desired system behavior. However, we do not yet propose a measure on this interpretation of robustness. In [2], the effect of disturbances on system behavior is quantified. The focus here is to synthesize robust systems that degrade gracefully - smallest number of system failures possible but not primarily directed on GR(1) specifications. Some existing work on notions of robustness in terms of satisfaction or violation of a formula can be found in [16], [4], [1]. The main objective in our paper is to understand and augment the robustness of pre-existing strategies for recovery from disturbances in constrast to those in [8] where robustness margins are introduced during abstraction for model inaccuracies. [5] uses a similarly motivated underlying idea to completely re-synthesize new robust strategies against a new GR(1) formula. Often uncertainties are not foreseen at the time of synthesis occur. In cases such as these, where unforeseen perturbations occur when the controller is implemented on the cyber-physical system, the results presented in this work allow for continued execution with guarantees in terms of formula satisfaction.

In summary, the main contributions of this work are the following: 1) to characterize the inherent tolerance of GR(1) strategies to unexpected perturbations; 2) to propose and prove approaches to refine GR(1) strategies to augment their tolerance to unexpected perturbations; 3) to quantify empirically the cost of augmenting the tolerance (robustness)
using the proposed approaches.

II. PRELIMINARIES

For a finite set \( \Sigma \), the set of all finite strings formed from concatenating elements of \( \Sigma \) is denoted by \( \Sigma^* \), which is known as the Kleene closure [6]. The set of all countably infinite strings of \( \Sigma \) is \( \Sigma^\omega \). In this paper, a subscript notation is used, e.g., \( \sigma_0 \sigma_1 \sigma_2 \cdots \sigma_n \in \Sigma^* \), but observe that infinite strings can also be regarded as functions of the natural numbers \( \mathbb{N} \) into \( \Sigma \).

Let \( \text{AP}_{\text{in}} \) be a set of input atomic propositions, and \( \text{AP}_{\text{out}} \) be a set of output atomic propositions such that \( \text{AP}_{\text{in}} \cap \text{AP}_{\text{out}} = \emptyset \). A state \( s \) is an assignment of True and False to the atomic propositions in \( \text{AP}_{\text{in}} \cup \text{AP}_{\text{out}} \). We use subset notation to indicate states and thus, for brevity, introduce \( \Sigma = 2^{\text{AP}_{\text{in}} \cup \text{AP}_{\text{out}}} \).

A finite-memory strategy is a pair \((f, m_0)\) where \( f : M \times 2^{\text{AP}_{\text{in}}} \rightarrow M \times 2^{\text{AP}_{\text{out}}} \) is a partial function and \( m_0 \in M \), where \(|M| < \infty \). Intuitively the set \( M \) represents the memory of the strategy. At each move, a new output is given depending on the input and the current memory value. As part of the move, a memory value is selected. Since we are only concerned with finite-memory strategies in this paper, we simply refer to them as strategies. The set of input-output sequences that may occur under \( f \) is defined as

\[
\text{Plays}(f) = \left\{ \sigma \in \Sigma^\omega \mid \exists m \in \Sigma. \forall k \geq 0. \right. \\
\left. f(m_k, \sigma_k \cap \text{AP}_{\text{in}}) = (m_{k+1}, \sigma_k \cap \text{AP}_{\text{out}}) \right\},
\]

where every \( m \in \Sigma^\omega \) has the same first element, \( m_0 \). Elements of \( \text{Plays}(f) \) are referred to as plays. The set of prefixes that may be extended into a play is

\[
\text{Pref}(f) = \left\{ \sigma \in \Sigma^* \mid \exists \alpha \in \Sigma^\omega. \sigma \alpha \in \text{Plays}(f) \right\}.
\]

**Remark 1:** For each \( \sigma \in \text{Plays}(f) \), there exists a unique \( m \in \Sigma^\omega \) satisfying \( f(m_k, \sigma_k \cap \text{AP}_{\text{in}}) = (m_{k+1}, \sigma_k \cap \text{AP}_{\text{out}}) \) for \( k \geq 0 \).

It follows from the remark that a sequence of inputs determines precisely one output sequence.

We describe specifications for these strategies in linear temporal logic (LTL) [ref] in this paper. LTL formulae over propositions (\( \text{AP}_{\text{in}} \cup \text{AP}_{\text{out}} \)) are evaluated over positions i in \( \sigma = \sigma_0 \sigma_1 \ldots \in \Sigma^\omega \). In addition to the Boolean operators, the standard LTL operators \( \boxdot \) (always), \( \text{eventually} \) and \( \circ \) (next) are used here for the specification.

A finite-memory strategy \((f, m_0)\) is said to be

- **input-enabled** iff for every \( \sigma^m \in (2^{\text{AP}_{\text{in}}})^\omega \), there exists \( \sigma \in \text{Plays}(f) \) such that \( \sigma^m_k = \sigma_k \cap \text{AP}_{\text{in}} \) for \( k \geq 0 \).
- **a realization of an LTL formula \( \varphi \)** iff \( \text{Plays}(f) \subseteq L(\varphi) \) (also written as \((f, m_0) \text{ realizes } \varphi\)), i.e., for every \( \sigma \in \text{Plays}(f) \), \( \sigma \models \varphi \).

A state \( s \in \Sigma \) is said to be **reachable under** \((f, m_0)\) iff there exists \( \sigma \in \text{Plays}(f) \) such that \( \sigma_k = s \) for some \( k \geq 0 \).

A GR(1) formula is an LTL formula of the form

\[
\Theta_{\text{env}}^\land \land \rho_{\text{env}}^\land \left( \bigwedge_{j=1}^{J} \boxdot \land \psi_j^\land \right)
\]

\[
\implies \Theta_{\text{sys}}^\land \land \rho_{\text{sys}}^\land \left( \bigwedge_{k=1}^{K} \land \boxdot \land \psi_k^\land \right),
\]

where \( \Theta_{\text{env}}^\land \) is a state formula (i.e., without temporal operators) that is a function of \( \text{AP}_{\text{in}} \), \( \Theta_{\text{sys}}^\land \) is a state formula that is a function of \( \text{AP}_{\text{out}} \), and all \( \psi_j^\land \), \( \psi_k^\land \) subformulae are functions of \( \text{AP}_{\text{in}} \cup \text{AP}_{\text{out}} \) and also without temporal operators. The subformula \( \rho_{\text{env}}^\land \) is a function of \( \text{AP}_{\text{in}} \cup \text{AP}_{\text{out}} \cup \text{AP}_{\text{in}} \), where

\[
\bigcirc \text{AP}_{\text{in}} = \{ \bigcirc x \mid x \in \text{AP}_{\text{in}} \}.
\]

Except for \( \bigcirc \) operators appearing as subformulae from \( \text{AP}_{\text{in}} \), there are no other temporal operators in \( \rho_{\text{env}}^\land \). Finally, \( \rho_{\text{sys}}^\land \) is defined similarly to \( \rho_{\text{env}}^\land \) but as a function of \( \text{AP}_{\text{in}} \cup \text{AP}_{\text{out}} \cup \text{AP}_{\text{in}} \).

To facilitate working with (3), and in particular the subformulae \( \rho_{\text{env}}^\land \) and \( \rho_{\text{sys}}^\land \), we extend the semantics of the operator \( \models \) for finite strings. Let \( \sigma \in \Sigma^\omega \). Define

\[
\sigma \models \rho \iff \sigma \alpha \models \rho \text{ for any } \alpha \in \Sigma^\omega,
\]

where \( \rho \) is any Boolean formula that is a function of \( \text{AP}_{\text{in}} \cup \text{AP}_{\text{out}} \cup \text{AP}_{\text{in}} \). Because at most one \( \bigcirc \) operator binds to each atomic proposition, it follows that only \( \sigma_0, \sigma_1 \) determine whether the formula is satisfied.

Given a GR(1) formula \( \varphi \) as in (3), a state \( s \in \Sigma \) is said to be \( \varphi \)-reachable under \((f, m_0)\) iff there exists \( \sigma \in \text{Plays}(f) \) such that for some \( k \geq 0 \),

\[
\sigma_k = s,
\]

\[
\sigma \models \Theta_{\text{env}}^\land,
\]

\[
\sigma_{j:(j+1)} \models \rho_{\text{sys}}^\land \text{ for } j < k - 1.
\]

A finite-memory strategy \((f, m_0)\) is said to be a **strict realization of** (or to strictly realize a) GR(1) formula (3) if the following conditions are met

\[
\sigma \models \Theta_{\text{env}}^\land \implies \sigma \models \Theta_{\text{sys}}^\land
\]

\[
\sigma \models (\neg \bigcirc \rho_{\text{env}}^\land) \implies \neg \bigcirc \neg \rho_{\text{sys}}^\land
\]

for any \( \sigma \in \text{Plays}(f) \). Intuitively, strict realizability ensures that blocking of an environment liveness condition when the other assumptions are met only occurs when the system is following transition rules. Here, \( \neg \bigcirc \) is the Past LTL operator which semantics are as defined in [13].

III. INHERENT ROBUSTNESS OF GR(1) STRATEGIES

**Definition 2:** A perturbation for a given finite-memory strategy is a deviant transition to a state \( \mathcal{S}' \) in \( \Sigma \) from a state \( s \), such that \( f(m_j, s' \cap \text{AP}_{\text{in}}) \neq f(m_{j+1}, s' \cap \text{AP}_{\text{out}}) \lor ss' \not\models \rho_{\text{env}}^\land \).

A perturbation occurs when the system control action fails to drive the system to the state indicated by the strategy or the environment violates a safety assumption. In this section,
we show that the GR(1) strategy with trivial refinement can satisfy a stricter formula - one that allows for finite perturbations when the transitions meet certain constraints. First we prove a lemma and which is then used to propose the refinement to allow for unexpected perturbations.

Let $f,m_0$ be a finite-memory strategy that strictly realizes a GR(1) formula $\varphi$, let $\sigma \in \text{Pref}(f)$ with $|\sigma| \geq 1$, and let $p^i \in \Sigma$. Let $\tilde{f}$ be the set of $\varphi$-reachable states and $p^i \in \tilde{f}$, $\gamma_i \in \Sigma^\omega$. Let $\tau^i \xi^i \in \text{Plays}(f)$ where $\tau^i \xi^i \in \Sigma^*$ for $i \in \{1,2,...,n\}$. Define $1_{\text{jump}}$ to be a formula such that for $\gamma_{j+1} \equiv 1_{\text{jump}}$ iff $f(m_{k+1},\gamma_{j+1} \cap \text{AP}_{\text{in}}) \neq f(m_{k+2},\gamma_{j+1} \cap \text{AP}_{\text{out}})$ where $k = j$ if $I_0$ is True and $k = j - |\tau^i \xi^i| - \sum_{i=1}^{n} |\xi^i| - |\sigma|$ for $I_1$ being True. Define $\varphi_{\text{jump}}$ as below:

$$\varphi_{\text{jump}} := \Theta^{\text{env}} \land \Box (\rho^{\text{env}} \lor 1_{\text{jump}}) \land \left( \bigwedge_{j=1}^{J} \Box \psi_{j}^{\text{env}} \right) \implies \Theta^{\text{sys}} \land \Box \rho^{\text{sys}} \land \left( \bigwedge_{k=1}^{K} \Box \psi_{k}^{\text{sys}} \right).$$

**Lemma 3:** If $\sigma_{-1} p^i \equiv \rho^{\text{sys}}$ and $\xi^i_{-1} p^i \equiv \rho^{\text{sys}}$ with $\tau^i \xi^i \in \text{Plays}(f)$ where $\tau^i \xi^i \in \Sigma^*$, then $\rho^{\text{sys}}$ is a path through which $p^i$ is $\varphi$-reachable, the following holds $\sigma = \sigma^i \xi^i_{1} \cdots \xi^i_{k+1} \cdots \xi^i_{n} \equiv \varphi_{\text{jump}}$.

**Proof:** The pre-ordered set of $n$-strategies chosen in Lemma 6 is chosen with replacement from the set of $n$ strategies. This lemma arises as a direct consequence of Lemma 6 for the case where $m = 1$, that is the number of strategies synthesized is just 1. The same strategy is chosen $n$ times and traces generated by the strategies are concatenated to generate a word satisfying $\varphi_{\text{jump}}$.

Intuitively, the practical significance of Lemma 3 is that, if there is a disturbance that causes an unexpected transition to some state that is $\varphi$-reachable in some other play and if there are only finitely many such disturbances, then execution of the finite-state machine can continue after an appropriate change of its internal state and still result in a correct input-output sequence. This result also allows for actions even when $\rho^{\text{env}}$ is violated. If $\rho^{\text{env}}$ is violated during a particular transition between a state $s$ and its successor $s'$ i.e $s s' \neg \rho^{\text{env}}$ and we end up at a $\varphi$-reachable state, this allows for a sequence of input-outputs that satisfies $\varphi_{\text{jump}}$ if these disturbances occur in a finite manner. This suggests the refinement proposed subsequently. This result is useful because in practice once the symbolic computation during the synthesis of GR(1) strategies is done, only the $\varphi$-reachable paths are enumerated from the symbolic computation. In this instance, all the states stored are $\varphi$-reachable and only the $\rho^{\text{sys}}$ condition must be checked before concatenating two paths and continuing further execution along the new path.

Consider a controller based on a GR(1) strategy. Consider the formula

$$\bar{\varphi} := \Theta^{\text{env}} \land \Box (\rho^{\text{env}} \lor 1_{\text{jump}}) \land \left( \bigwedge_{j=1}^{J} \Box \psi_{j}^{\text{env}} \right) \land (\Box \Box \neg 1_{\text{jump}}) \implies \Theta^{\text{sys}} \land \Box \rho^{\text{sys}} \land \left( \bigwedge_{k=1}^{K} \Box \psi_{k}^{\text{sys}} \right).$$

Algorithm 1 with $n = 1$, generates an output sequence for a controller given an input sequence. The lemma above guarantees that this input-output sequence satisfies the formula $\bar{\varphi}$. It also considers for disturbances during application of the output action to a cyber-physical system.

The added $\Box \Box \neg 1_{\text{jump}}$ segment on the environment-assumption side in $\bar{\varphi}$ ensures that the perturbations do not occur infinitely often. And, for all instances of environment violation if a feasible $\varphi$-reachable state can be found, the system part of the formula is satisfied.

Also, we arrive at the following corollaries which help us augment the robustness of a given strategy strictly-realizing a GR(1) formula.

**Corollary 4:** Let $(f,m_0)$ be a finite-memory strategy that realizes a GR(1) formula $\varphi$ and $\varphi_{\text{jump}}$ be the corresponding LTL formula as defined above. Let $\sigma \in \Sigma^\omega$, then $\sigma \models \varphi_{\text{jump}} \implies \sigma \models \varphi$.

**Proof:** If $\sigma \models (\neg \Theta^{\text{env}} \lor \Box \rho^{\text{env}} \lor (\bigwedge_{k=1}^{K} \Box \neg \psi_{k}^{\text{sys}}))$ then $\sigma \models \varphi$.

**Corollary 5:** Let $(f,m_0)$ be a finite-memory strategy that realizes a GR(1) formula $\varphi$, and let $p$ be reachable under $(f,m_0)$. If $\tau \rho a \in \text{Plays}(f)$ (at least one must exist), then $\rho a$ satisfies

$$\Box \rho^{\text{env}} \land \left( \bigwedge_{j=1}^{J} \Box \psi_{j}^{\text{env}} \right) \implies \Box \rho^{\text{sys}} \land \left( \bigwedge_{k=1}^{K} \Box \psi_{k}^{\text{sys}} \right).$$

**IV. AUGMENTING ROBUSTNESS**

**A. Approach 1: Concatenating multiple strategies with same safety/progess specifications**

In this section, the intuition from Section III is used and a more general Lemma is presented and proved. The results in this section allow for the concatenation of multiple strategies synthesized with formulae differing in the initial condition and the approach for concatenation is described.

Given the specification $\varphi_0$ and a synthesized finite-memory strategy $(f_0,m_0)$ that realizes $\varphi_0$, let $I(f_0,m_0)$ be the set of all states in the strategy.

Let $\eta_0, \eta_1, \eta_2,..., \eta_n$ be the additional states in $\Sigma$ that the strategy must visit to provide additional robustness. Define $\eta_{I}^i = \eta_i \cap \text{AP}_{\text{in}}$ and $\eta_{I}^o = \eta_i \cap \text{AP}_{\text{out}}$. Define $\Theta_{\text{env}}^{\text{env}}$ as a Boolean formula which is True for a state $s$ in $\Sigma$ iff $s \cap \text{AP}_{\text{in}} = \eta_{I}^i$. Similarly, define $\Theta_{\text{sys}}^{\text{sys}}$ as a Boolean formula which is True for a state $s$ in $\Sigma$ iff $s \cap \text{AP}_{\text{out}} = \eta_{I}^o$.
Then, construct a set of finite-memory strategies \( \{f_i, m_i^0\}\) such that \( \forall i \in \{0, 1, 2, \ldots, n\}\), \( (f_i, m_i^0) \) realizes \( \varphi_i \), where \( \varphi_i \) is as defined below:

\[
\varphi_i = T_i^{out} \land \Box \rho_{env} \land \left( \bigwedge_{j=1}^{j=k} \Box \Diamond \psi_j^{env} \right) \implies T_i^{out} \land \Box \rho_{sys} \land \left( \bigwedge_{k=1}^{K} \Box \Diamond \psi_k^{sys} \right). \tag{13}
\]

Let \( (f_0, m_0^0) \) be a finite-memory strategy that strictly realizes a GR(1) formula \( \varphi_0 \), let \( \sigma \in \text{Pref}(f_0) \) with \( |\sigma| \geq 1 \), and let \( p^i \in \Sigma \).

Let there be a set of finite-memory strategies \( \{f_i, m_i^0\} \) such that \( \forall i \in I = \{1, 2, \ldots, m\} \), \( (f_i, m_i^0) \) strictly-realizes \( \varphi_i \), where \( \varphi_i \) is as defined above.

Consider a fixed ordering of the strategies, \( \{f_1, f_2, f_3, \ldots, f_m\} \) where \( i_l \in \{1, 2, \ldots, m\} \) and \( l \in \{1, 2, \ldots, n\} \). Let \( \tau^l_\xi^\alpha \in \text{Plays}(f_i) \) where \( \tau^l_\xi^\alpha \in \Sigma^* \).

Define \( I_1 \) and \( 1_{jump} \) as below: \( 1_{jump} \land I_1 \rightarrow \bigcirc (I_{i+1} \land \neg I_1) \). \( I_0 \) is initialized to True.

Define \( 1_{jump} \) to be a boolean formula such that for \( \gamma_{j:j+1} = 1_{jump} \) iff \( f_i(mk+1, \gamma_{j:j+1} \land \text{AP}_{in}) \neq f_i(mk+2, \gamma_{j+1} \land \text{AP}_{out}) \).

Define \( I_1 \) and \( 1_{jump} \) as below: \( 1_{jump} \land I_1 \rightarrow \bigcirc (I_{i+1} \land \neg I_1) \).

Lemma 6: If \( \forall l \in \{1, \ldots, m\} \), \( p^j \) is \( \varphi_i \)-reachable under \( \{f_i, m_i^0\} \) through the path \( \tau^l_\xi^\alpha \) with \( \xi^0 = p^j \).

\[
\sigma \models \xi^0 \models \rho_{sys}, \xi_{i+1}^{l+1} \models \rho_{sys} \forall l \in \{1, \ldots, m-1\} \text{ then } \sigma \models \xi^l \models \rho_{sys}, \xi_n^{l+1} \models \varphi_{jump}.
\]

Proof: By definition, there exists \( \beta^l \in \text{Plays}(f_i) \) and \( k < j-1 \) such that \( p^j = \beta^l_k, \beta^l \models \Theta_{env} \), and

\[
\beta^l_{j+1} \models \rho_{sys} \text{ for } j < k - 1. \tag{14}
\]

Thus, we write \( \tau^l_\xi^\alpha = \beta^l \) by taking \( \tau^l_\xi^\alpha = \beta^l \) for \( 0 < j < k \). And, \( \xi^l_{j+1} = \beta^l_{j+1} \) where \( 0 \leq r < ml \) for some \( m_i \) and \( \alpha^_j - k = m_1 \) for \( j \leq k + m \). We want to show that \( \sigma^l \models \Box \psi_{env} \). Since \( \varphi_{jump} \) has the form (3), it is equivalent to at least one of the following subformulas being satisfied:

\[\Theta_{env} \land \Diamond (\neg \rho_{sys} \land \neg \Diamond \psi_{env})\]

Since \( \sigma \in \text{Pref}(f_0) \) by hypothesis, there exists \( \gamma \in \Sigma^* \) such that \( \sigma \gamma \models \text{Plays}(f_0) \). Also, by hypothesis, \( (f_0, m_0) \) realizes \( \varphi_0 \), i.e., \( \text{Plays}(f_0) \subseteq L(\varphi_0) \), hence \( \sigma \gamma \models \varphi_0 \). Since \( |\sigma| \geq 1 \) by hypothesis, \( \sigma \gamma \models \neg \Theta_{env} \) if and only if \( \sigma \gamma \models \neg \Theta_{env} \).

Thus, if \( \sigma \gamma \models \neg \Theta_{env} \), then \( \sigma \gamma \models \varphi_{jump} \). Otherwise, i.e., if \( \sigma \gamma \models \Theta_{env} \), consider the subformula \( \Diamond (\neg \rho_{sys} \land \neg \Diamond \psi_{env}) \).

For all \( k < |\sigma| - 1, \sigma_{k:k+1} \models \neg \Diamond \psi_{env} \) by definition of a Play(f_0) and that \( \sigma \in \text{Pref}(f_0) \). Also, for \( k \geq 1, \sigma^l \models \Box \psi_{sys} \text{ if } \sigma^l \models \Diamond \psi_{sys} \), and \( \alpha_0^\gamma \models \neg \Diamond \psi_{env} \).

Recall the suffix \( \gamma \) such that \( \sigma \gamma \models \text{Plays}(f_0) \). \( (f_0, m_0) \) strictly realizes \( \varphi_0 \), therefore if \( \gamma \models \Theta_{env} \), it must be that \( \sigma \gamma \models \Theta_{env} \). Since \( \sigma \models \Theta_{env} \) (8), furthermore, because in this case we are assuming there is no \( 0 \leq k < |\sigma| - 1 \) such that \( \sigma_{k:k+1} \models \neg \rho_{sys} \) (otherwise we would have \( \sigma \gamma \models \varphi_{sys} \), it follows from strict realizability (cf. (9)) that \( 0 \leq k < |\sigma| - 1, \sigma_{k:k+1} \models \rho_{sys} \), and therefore \( \sigma \gamma \models \varphi_{jump} \).
Consider GR(1) strategy based controllers, as discussed earlier, where only the \( \varphi \)-reachable states are retained and the environment moves are restricted to ones that do not violate \( \rho^{env} \). Combine the finite-state controllers by adding transitions from all states to all other \( \varphi \)-reachable states that satisfy \( s s' \models \rho^{sys} \). This combined set of controllers (strategies) will satisfy the formula \( \varphi \) with \( 1_{jump} \) being as defined in Lemma 6.

Algorithm 1 gives a formal description of the approach to combine strategies using the lemma proposed above. The notation used in the description is as defined in this section. This controller formed by the combined set of controllers (strategies) will satisfy the formula \( \varphi \) with \( 1_{jump} \) being as defined in Lemma 6.

**Procedure 1** Implements controller based on Section IV  

**Input:** finite-memory strategy \( (f_i, m_i) \) \( \forall i \in \{1, 2, ..., n\} \), sequence of inputs \( \sigma^{env} \in AP^{in}_{\alpha} \), a system the control sequence \( \sigma^{sys} \) can be applied to and its state measured \( s \in \Sigma \), set \( I - \) union of \( \varphi \)-reachable states for strategy \( f_i \) and \( M \) memory states corresponding to \( I \) and a mapping for every \( m \) in \( M \) to the strategy \( f_{m} \) it was taken from.  

**Output:** Sequence of output actions \( \sigma^{sys} \in AP^{out}_{\omega} \) satisfying \( \varphi \) when conditions in Lemma 6 are satisfied memory=\( m_{0} \)

\( i=1 \)

(memoryNew, \( \sigma^{sys}_i \)) = Strategy \( f : \) (memory, \( \sigma^{env}_0 \)) if safety=1

\( l=0 \)

**while** (\( l \geq 0 \))

**if** \( (\sigma^{env}_{i-1}, \sigma^{sys}_{i-1}) (\sigma^{env}_i) \models \rho^{env} \) and safety=1 **then**

(memoryNew, \( \sigma^{sys}_i \)) = Strategy \( f_i : \) (memory, \( \sigma^{env}_i \)) Run: SafetyCheck  

**else** \( (\sigma^{env}_{i-1}, \sigma^{sys}_{i-1}) (\sigma^{env}_i) \not\models \rho^{env} \) OR safety=0 **then**

**if** \( \exists p \) and a corresponding \( m \) such that \( p \in I \) and \( m \) in \( M \) and \( i \in \{1, 2, ..., m\} \) and \( (\sigma^{env}_{i-1}, \sigma^{sys}_{i-1}) p \models \rho^{sys} \) and \( p \cap AP_{in} = \sigma^{sys}_{i-1} \) **then**

\( (\sigma^{env}_{i}, \sigma^{sys}_{i}) = p, \) memoryNew=\( m, l = i \_m \)  

Run: SafetyCheck  

**else**

EXIT  

**end if**

**end if**

\( i=i+1 \)

memory=memoryNew

**end while**

**B. Approach 2: Augment Initial States**

Let \( (f^0, m_0) \) be a finite-memory strategy that strictly realizes a GR(1) formula \( \varphi \), with \( |\sigma| \geq 1 \), and let \( p^1 \in \Sigma \). Let \( \sigma \) belong to Pref\( (f^0) \) and \( \sigma \models \Theta^{env} \land \Theta^{sys} \).

For a set of states \( \eta \in \Sigma \) and \( \eta \not\in I(f, m_0) \), let \( \chi^\eta \) be a Boolean formula indicating these states. Let \( (f^*, m_0) \) be a finite memory strategy that strictly realizes the formula \( \varphi^* \)

**Procedure 2** SafetyCheck

\begin{align*}
\text{apply } \sigma^{sys} \text{, measure } s \\
\text{if } (\sigma^{env}_{i-1}, \sigma^{sys}_{i-1}) s \not\models \rho^{sys} \text{ then} \\
\text{EXIT} \\
\text{end if} \\
\text{if } (\sigma^{env}_{sys}_i) = s \text{ then} \\
\text{safety=1} \\
\text{else} \\
\text{safety=0, } \sigma^{env}_i \cap AP_{in}, \sigma^{sys}_i = s \cap AP_{out} \\
\text{end if}
\end{align*}

that is defined as below:

\[
\varphi^* := (I_m \land \chi^\eta_{m}) \land \square \rho^{env} \land (\bigwedge_{j=1}^{J} \square \psi^{env}_j)
\]

\[
\implies (I \lor \chi^\eta) \land \square \rho^{sys} \land (\bigwedge_{k=1}^{K} \square \psi^{sys}_k).
\]

**Corollary 7:** If \( p_i \) is \( \varphi^* \)-reachable under \( (f^*, m^*) \) through the path \( \tau^i \xi^1 \) with \( \xi^1 = p_i \forall \xi \in 1, ..., m, \tau^i \xi^1 \in \text{Plays}(f^*) \), \( \text{ssssss}_{\xi^1} \models \rho^{sys}, \xi^1 \cdot \xi^1 \models \rho^{sys} \forall \xi \in 1, ..., m - 1 \) then \( \sigma = \sigma^{\xi^1}, \xi^2, ..., \xi^m \models \varphi^{jump} \).

This corollary results as a special case of Lemma 6 with the number of strategies, \( n = 1 \) and picking a \( \sigma \) such that \( \Theta^{env} \land \Theta^{sys} \) is satisfied. The utility in this approach is that if there a certain set of states that are recognized as states the system is likely to be perturbed to after executing the strategy, these states can be augmented to the initial-set of states visited by the strategy and using these as the initial states, synthesis can be done. This ensures that there is no loss of coverage in terms of the states visited by the initial strategy and there a single strategy that is robust to likely disturbances. Again, this strategy can be refined similarly as proposed earlier.

**C. Approach 3: Patching**

Let \( (f, m_0) \) be a finite-memory strategy that strictly realizes a GR(1) formula \( \varphi \), let \( \sigma \in \text{Pref}(f) \) with \( |\sigma| \geq 1 \), and let \( p \in \Sigma \) and \( p \not\in I(f, m_0) \). Let \( \eta \) be an element in \( \Sigma \) and \( \eta \not\in I(f, m_0) \). Define \( T_{reach} \) as a Boolean formula that evaluates to \( \text{True} \) at a state \( s \in \Sigma \) iff \( s \in I(f) \).

Let \( (f_{reach}, M) \) be a finite-memory strategy that strictly realizes (defined similarly to the GR(1) specification) \( \varphi_{reach} \) where \( \varphi_{reach} \) is defined as

\[
(\chi^\eta_{m} \land \square \rho^{env} \land (\bigwedge_{k=1}^{K} \square \psi^{env}_k)) \implies (\chi^\eta_{out} \land \square \rho^{sys} \land \square T_{reach}).
\]

Define \( 1_{jump} \) to be a boolean formula such that for \( \gamma \in \Sigma^*, j \geq 1, \gamma_{j:j+1} \models 1_{jump} \) if \( f(m_j, \gamma_j \land AP_{in}) \neq f(m_{j+1}, \gamma_j \land AP_{out}) \) and \( \forall i < j, \gamma_{i:i+1} \models \neg 1_{jump} \). Otherwise, \( \gamma_{j:j+1} \models \neg 1_{jump} \).
By corollary 5, \( \psi_{\text{jump}} := \Theta_{\text{env}} \land \Box (\rho_{\text{env}} \lor 1_{\text{jump}}) \land \left( \bigwedge_{j=1}^{K} \Box \Diamond \psi^\text{sys}_j \right) \Rightarrow \Theta^\text{sys} \land \Box \rho^\text{sys} \land \left( \bigwedge_{k=1}^{K} \Box \Diamond \psi^\text{sys}_k \right) \)

**Lemma 8:** If \( p \) is \( \varphi_{\text{reach}} \)-reachable (defined similarly as for a GR(1) specification) under \( (f_{\text{reach}}, M) \) and \( \sigma_{-1} p \models (\rho^\text{sys}) \) then for \( \sigma_{\text{reach}} \in \{ \text{Plays}(f_{\text{reach}}) \} \) with the following properties holding:

(i) \( \sigma_{\text{reach}} \models \Diamond T_{\text{reach}} \)
(ii) \( k^* = \min\{ k : \sigma_{k_{\text{reach}}} = p \} < j^* = \min\{ j : \sigma_{j_{\text{reach}}} = \Diamond T_{\text{reach}} \} \)
(iii) \( \sigma_{j_{\text{reach}}} \) is \( \varphi_{\text{reachable}} \)
(iv) \( \sigma_{-1} p \models \rho^\text{sys} \)
then firstly, \( \sigma_{\text{reach}} \models \left( \bigwedge_{k=1}^{K} \Box \Diamond \psi^\text{sys}_k \right) \Rightarrow \sigma_{\text{reach}} \models \varphi_{\text{jump}} \) where \( \gamma_{j_{\text{reach}}} \) is in \( \text{Plays}(f_{\text{reach}}) \) and \( \gamma_{j_{\text{reach}}} \in \Sigma^+ \). And secondly, \( \sigma_{\text{reach}} \models \left( \bigwedge_{k=1}^{K} \Box \Diamond \not\psi^\text{sys}_k \right) \Rightarrow \sigma_{\text{reach}} \models \varphi_{\text{jump}} \).

**Proof.**

Let \( \sigma_{\text{reach}} \in \text{Plays}(f_{\text{reach}}) \) such that it meets the four assumptions in the lemma and be the path through which \( p \) is \( \varphi_{\text{reach}} \)-reachable. For the case when \( \sigma_{\text{reach}} \models \left( \bigwedge_{k=1}^{K} \Box \Diamond \not\psi^\text{sys}_k \right) \land \left( \bigwedge_{k=1}^{K} \Box \Diamond \not\psi^\text{sys}_k \right) \) then \( \sigma_{\text{reach}} \models \varphi_{\text{relaxed}} \).

Consider the other scenario, \( \varphi_{\text{relaxed}} \) is defined similarly as above.

**Case 2:** \( \sigma_{\text{reach}} \models \left( \bigwedge_{k=1}^{K} \Box \Diamond \psi^\text{sys}_k \right) \Rightarrow \)

\( \varphi_{\text{relaxed}} \) has less expressive power than \( \varphi_{\text{reach}} \) and allows for faster computation. Since, \( \varphi_{\text{relaxed}} \Rightarrow \varphi_{\text{reach}} \) by Lemma 7 holds with \( \varphi_{\text{reach}} \) replaced by \( \varphi_{\text{relaxed}} \).

**V. EXAMPLE IMPLEMENTATION AND ANALYSIS**

Examples are implemented for the analysis of the techniques described in sections III, IV for the task of planar robot motion planning in the environments shown (See Figure 1). The robot is required to visit a set of locations infinitely often (progress-states). A moving obstacle whose behavior dynamics mimic those of the robot with different progress-states and initial positions is added to the setup. The planned trajectories for the robot must be such that they do not collide with any of the walls (regions shaded black in Figure 1) or the non-deterministic moving obstacle.

**A. Complexity for refinement**

An empirical analysis of the computational costs involved in each of the approaches to augment robustness is presented here. The computations were performed on a 2.40GHz Quadcore machine with 16 GB of RAM. The experiment described below is repeated 50 times and the average synthesis...
Procedure 3 Algorithm for executing single patch from perturbed state

Input: GR(1) formula $\varphi$, finite-memory strategy $(f, m_0)$, sequence of inputs $\sigma^{\text{env}} \in \text{AP}_{\text{in}}^w$, a system control sequence $\sigma^{\text{sys}}$ can be applied and its state measured $s \in \Sigma$, set $I$ of $\varphi$-reachable states for strategy $f$ and $M$ memory states, such that each $m \in M$ is the memory corresponding to some $p \in I$ along a $\varphi$ reachable path.

Output: Sequence of output actions $\text{sys}^2 \text{AP}^!$ satisfying $\overline{\varphi}$ when conditions in Lemma 8 are satisfied

```
memory = m_0
i=1
l=0
(memoryNew, $\sigma_i^{\text{sys}}$) = Strategy $f_i : (\text{memory}, \sigma_0^{\text{env}})

while True do
  if ($\sigma_i^{\text{env}}(\text{env}_i), \sigma_i^{\text{sys}}(\text{sys}_i)$) $\not\models \varphi^{\text{env}}$ and safety=1 then
    (memoryNew, $\sigma_i^{\text{sys}}$) = Strategy $f : (\text{memory}, \sigma_i^{\text{env}})$
    Run: SafetyCheck
  else
    if ($\sigma_i^{\text{env}}(\text{env}_i), \sigma_i^{\text{sys}}(\text{sys}_i)$) $\not\models \varphi^{\text{env}}$ OR safety=0 then
      if Synthesize $f^{\text{reach}}$ for $\varphi^{\text{reach}}$ with ($\sigma_i^{\text{env}}, \sigma_i^{\text{sys}}$) as initial state is successful then
        reached=0
        memoryNew=0
        while reached=0 do
          if ($\sigma_i^{\text{env}}(\text{env}_i), \sigma_i^{\text{sys}}(\text{sys}_i)$) $\models \varphi^{\text{env}}$ and safety=1 then
            (memoryNew, $\sigma_i^{\text{sys}}$) = Strategy $f^{\text{reach}} : (\text{memory}, \sigma_i^{\text{env}})$
            Run: SafetyCheck
            i+=1
            if safety=0 then
              EXIT
          end if
          if ($\sigma_i^{\text{env}}(\text{env}_i), \sigma_i^{\text{sys}}(\text{sys}_i)$) $\in I$ then
            reached=0
            memoryNew= memory corresponding to ($\sigma_i^{\text{env}}, \sigma_i^{\text{sys}}$) in $M$
          end if
        end while
        if reached=0 then
          EXIT
        end if
      else
        EXIT
      end if
    end if
  end if
end while
```

Fig. 1: Grid-World Setup

Random 5x5 gridworlds are generated with a wall density of 0.2. The moving obstacle and robot have two different progress-locations which they visit infinitely often and two different initial positions. For each of the approaches 5 perturbation points are chosen as described:

- Multiple Strategy Approach: A single perturbation point is chosen that is not visited by the initial strategy and a strategy is synthesized. The states visited by the new strategy are stored. And, a new perturbation point is chosen not in any of the earlier strategies. This repeated 5 times.
- Patching: The points are chosen as the previous approach except only those states from the new strategy are stored that occurred before the trajectories hit the old set of states.
- Patching without progress: Similarly done as patching, if synthesis from a perturbed point is not feasible, another point is chosen. This is done so till a point is found from which a feasible patch exists.
- Augmented Initial States Approach: Five arbitrary points not visited by the original strategy are chosen and a new strategy is synthesized.

The coverage i.e the number of unique states - robot, moving obstacle position combinations - visited by each strategy is also presented. It loosely characterizes the robustness for the concatenated-strategies as this count represents the $\varphi$-reachable states. Approach 3 is implemented recursively, with the visited states augmented in each patch. With recursive patching, the time for synthesis tends to decrease progressively with each patch for a given gridworld because the number of unique visited states tends to go up. Also, the numbers indicate that the synthesis for patching without the progress condition is faster than that with progress, as expected because the synthesis formula has smaller length.
TABLE I: Runtimes and unique states visited

<table>
<thead>
<tr>
<th>Approach for refinement</th>
<th>Average time for synthesis(s)</th>
<th>Coverage (unique states)</th>
</tr>
</thead>
<tbody>
<tr>
<td>New strategy from perturbed state</td>
<td>0.22</td>
<td>145.14</td>
</tr>
<tr>
<td>Augment Initial States</td>
<td>0.21</td>
<td>144.92</td>
</tr>
<tr>
<td>Patching</td>
<td>2.98</td>
<td>173.63</td>
</tr>
<tr>
<td>Patching without progress</td>
<td>1.90</td>
<td>150.36</td>
</tr>
</tbody>
</table>

VI. CONCLUSION AND FUTURE DIRECTIONS

We demonstrated the inherent tolerance of strategies synthesized to satisfy GR(1) specifications and described approaches to augment the tolerance of a strategy to perturbations by refinement or concatenation with other strategies in a provably correct manner. It was shown that these refined strategies satisfy a stricter formula than the one used for synthesis. This tolerance is useful when the model is not exact for either the system behavior or the environment behavior.

In the future, we plan to extend the framework built here to the case of infinite jumps. We also intend to develop a metric that would quantify the robustness added to a strategy through a given concatenation and prescribe approaches for refinement of strategies to make them more robust with optimal synthesis time/memory costs. Also, we plan to implement the approaches in Sections IV-A and IV-B using enumeration from a stored BDD computed during the original synthesis as that would remove the need for re-synthesis to obtain the new strategies. This is not directly facilitated by the solver ‘gr1c’ [9] used for the work presented here.

ACKNOWLEDGMENTS

This work was partially supported by United Technologies Corporation and IBM, through the industrial cyber-physical systems (iCyPhy) consortium.

REFERENCES