On Synthesizing Robust Discrete Controllers under Modeling Uncertainty*

Ufuk Topcu, Necmiye Ozay, Jun Liu, and Richard M. Murray
California Institute of Technology, Pasadena, CA
{utopcu, necmiye, liu, murray}@cds.caltech.edu

ABSTRACT

We investigate the robustness of reactive control protocols synthesized to guarantee system's correctness with respect to given temporal logic specifications. We consider uncertainties in open finite transition systems due to unmodeled transitions. The resulting robust synthesis problem is formulated as a temporal logic game. In particular, if the specification is in the so-called generalized reactivity [1] fragment of linear temporal logic, so is the augmented specification in the resulting robust synthesis problem. Hence, the robust synthesis problem belongs to the same complexity class with the nominal synthesis problem, and is amenable to polynomial time solvers. Additionally, we discuss reasoning about the effects of different levels of uncertainties on robust synthesizability and demonstrate the results on a simple robot motion planning scenario.

Categories and Subject Descriptors

D.2.4 [**SOFTWARE ENGINEERING**]: Software/Program Verification

Keywords

Discrete controller synthesis; Robustness; Temporal logic

1. INTRODUCTION

Robustness—a system's ability to function correctly under uncertainties, for example, due to imperfections in the way the evolution of the system and its interactions with its environment are modeled—is a key attribute to predictable operation (and graceful failure). Though well-studied for physical engineering artifacts, it has been hardly explored for distributed embedded systems. Approaching this from a computer science perspective, a reason for the lack of suitable robustness notions is that computing systems are conveniently modeled as discrete mathematical objects with no underlying (non-trivial) topology where uncertainties and their impact can be quantified [9]. Furthermore, even though controls have explicitly modeled such uncertainties and developed dedicated methods and tools, they have been limited to rather restrictive representations and cannot directly address the critical interplay between physical components and computing/communication. Consequently, there is a need for characterizations and computable metrics to support the analysis, design, and construction of robust embedded control systems.

As a step toward addressing this need, we consider robustness of discrete, reactive control protocols synthesized to guarantee system's correctness with respect to given temporal logic specifications. Such control protocols—in the context of embedded control systems, robot motion planning, and hybrid systems—have attracted considerable attention recently. The special case, where there is no uncontrolled, environment, has been studied, for example, in [2, 11]. These methods generally formulate a problem that is amenable to model checking [5]. More recently, the case with a priori unknown, dynamic environment has been investigated, for example, in [12, 22]. In this case, the problem is generally formulated as a temporal logic game [1].

We consider uncertainties in open finite transition systems, i.e., transition systems with uncontrolled inputs, due to unmodeled transitions. We reformulate the resulting robust synthesis problem as a temporal logic game. In particular, we utilize specifications that belong to the so-called generalized reactivity [1] (GR[1]) fragment of linear temporal logic for which there exist polynomial complexity solvers [19]. We show that if the specification is a GR[1] formula, so is the augmented specification in the resulting robust synthesis problem. Hence, robustification of protocol synthesis with the specific uncertainty model consider in this paper does not change its complexity class. Finally, we discuss reasoning about the effects of different levels of uncertainties on robust synthesizability. Specifically, we embed partial orders on the family of sets of unmodeled transitions, which reflect on designer's intent or prior knowledge, as a means to compare different uncertainty sets. For a class of partial orders that satisfy certain monotonicity conditions, we propose a bisection-type search to compute maximum level robustness a system possesses with respect to given specifications and a partially ordered family of uncertainty sets.

The work in [20] extends notions, such as input-output gains and small-gain theorems, well-known in controls to systems over finite alphabets. This study is limited to the

*This work is supported by the Boeing Corporation, NSF grants CNS 0911041, the NSERC of Canada, the FCRP consortium through the Multiscale Systems Center (MuSyC).

verification of stability and amplification of a measure of energy from the input channels to the output channels. A recent collection of papers including [3, 13] consider the *sensitivity* of the outputs of discrete systems to the variations in the inputs. For example, [3] focuses on safety properties and uses the ratio of the "distance" between allowed and observed system behavior to that of the environment behavior as a measure for sensitivity. The recent work [?] considers fault tolerance in the context of discrete controller synthesis, where such tolerance is achieved by taking various fault conditions as adversarial inputs and modifying the state space according to each fault hypotheses against which fault-tolerance is desired. We emphasize, though, that what matters for satisfactory operation of engineering systems is *robustness* (i.e., how much the input specifications can be violated and still the output specifications hold), beyond mere *sensitivity* (i.e., how much the output specifications are violated due to the violations in the input specifications).

The organization of the paper is straightforward. We continue with some background material used in the rest of the paper. The problem formulation in section 4 is followed by the main results of the paper and a demonstration of these results on a simple robot motion planning problem. We conclude with a critique of the problems and progress reported in the paper focusing on the limitations and potential extensions.

2. BACKGROUND

We now present some of the definitions and background material used throughout the paper.

2.1 Finite transition system

We consider two types of finite transition system models. Roughly speaking, the first one does not interact with its external environment (after possibly being started by the environment). The other one explicitly accounts for the interactions with possibly adversarial environments and reacts to the changes in the environment.

Definition 1. A *finite transition system* is a tuple $TS = (Q, I, \mathcal{A}, R)$ where Q is the finite set of states, $I \subseteq Q$ is the set of initial states, $\mathcal{A}$ is the finite set of actions (i.e. controllable input variables) and $R \subseteq Q \times \mathcal{A} \times Q$ is a transition relation.

TS is called *action deterministic* if I is a singleton and for all $q \in Q$, for all $u \in \mathcal{A}$, there is at most one $q' \in Q$ such that $(q, u, q') \in R$. TS is called *non-blocking* if for all $q \in Q$, there exists a pair $(u, q') \in \mathcal{A} \times Q$ such that $(q, u, q') \in R$.

An *execution* of the transition system TS is a sequence $(q_0, u_0), (q_1, u_1), (q_2, u_2), \ldots$ such that $q_0 \in I$ and $(q_i, u_i, q_{i+1}) \in R$ for all $i \geq 0$. For simplicity, we assume TS is non-blocking and consider only infinite executions. This assumption introduces no loss of generality since one can add an auxiliary sink state with a self-transition and complete the finite executions to infinite ones.

We also consider a more general class of transition systems, so-called *open* systems, where part of the actions are uncontrollable (e.g., controlled by the environment).

Definition 2. An *open finite transition system* is defined as a tuple $TS = (Q, I, \mathcal{A}_{uc}, \mathcal{A}_c, R)$ where Q is the finite set of states, $I \subseteq Q$ is the set of initial states, $\mathcal{A}_{uc}$ is the finite set of uncontrollable actions (i.e. environment decisions), $\mathcal{A}_c$ is the finite set of controllable actions (i.e. control inputs) and $R \subseteq Q \times \mathcal{A}_{uc} \times \mathcal{A}_c \times Q$ is a transition relation.

An open finite transition system TS is called *action deterministic* if I is a singleton and for any pair of transitions $(q_1, e_1, u_1, q_1') \in R$ and $(q_2, e_2, u_2, q_2') \in R$, if $q_1 = q_2$, $e_1 = e_2$ and $u_1 = u_2$, then $q_1' = q_2'$. TS is called *non-blocking* if for all $q \in Q$, there exists a triple $(e, u, q') \in \mathcal{A}_{uc} \times \mathcal{A}_c \times Q$ such that $(q, e, u, q') \in R$. An execution for an open transition system is defined similarly.

2.2 Linear temporal logic

We use linear temporal logic (LTL) [17, 14] as a formal language to specify correct behaviors of systems and admissible behaviors of their environments. LTL is a rich specification language that can express properties often used in control, robot motion planning, and embedded systems, including safety, reachability, invariance, response, and/or a combination of these [14] (see [21] for examples). Roughly speaking, LTL specifications impose constraints on the infinite sequences of the values a certain set of variables can take (e.g., the executions in finite transition systems).

LTL is an extension of propositional logic by including temporal operators. Apart from the logical connectives negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$) and implication ($\rightarrow$), it includes temporal operators next ($\bigcirc$), always ($\Box$), eventually ($\Diamond$) and until ($\mathcal{U}$). By combining these operators, it is possible to specify a wide range of requirements on the desired behavior of a system.

An *atomic proposition* is a statement that has a unique truth value (*True* or *False*) at a given state[1]. Given a set Π of atomic propositions, an LTL formula is defined inductively as follows: (i) any atomic proposition $p \in \Pi$ is an LTL formula; and (ii) given LTL formulas φ and ψ, $\neg\varphi$, $\varphi \vee \psi$, $\bigcirc\varphi$ and $\varphi \,\mathcal{U}\, \psi$ are also LTL formulas. Formulas involving other operators can be derived from these basic ones.

Semantics of LTL: An LTL formula is interpreted over an infinite sequence of states. Given a sequence of states $\sigma = q_0q_1q_2\ldots$ and an LTL formula φ, we say that φ *holds at position* $i \geq 0$ of σ, written $q_i \models \varphi$, if and only if (iff) φ holds for the remainder of the sequence σ starting at position i. The semantics of LTL is defined inductively as follows:

- For an atomic proposition p, $q_i \models p$ iff $q_i \Vdash p$;
- $q_i \models \neg\varphi$ iff $q_i \not\models \varphi$;
- $q_i \models \varphi \vee \psi$ iff $q_i \models \varphi$ or $q_i \models \psi$;
- $q_i \models \bigcirc\varphi$ iff $q_{i+1} \models \varphi$; and
- $q_i \models \varphi \,\mathcal{U}\, \psi$ iff there exists $j \geq i$ such that $q_j \models \psi$ and $\forall k \in [i, j), q_k \models \varphi$.

Based on this definition, $\bigcirc\varphi$ holds at position i of σ iff φ holds at the next state q_{i+1}, $\Box\varphi$ holds at position i iff φ holds at every position in σ starting at position i, and $\Diamond\varphi$ holds at position i iff φ holds at some position $j \geq i$ in σ.

2.3 Control strategy

Given a transition system TS and an LTL specification of the admissible environment behaviors and requirements on

[1]Here, state refers to a valuation of the variables the atomic proposition involves or is evaluated over; e.g., some $(q, e, u) \in Q \times \mathcal{A}_{uc} \times \mathcal{A}_c$ if one wants to reason about an element in an execution.

the system behavior, we aim to synthesize control protocols that, when implemented on the system, guarantee that the executions of the system satisfy the specification.

A *control strategy* for an open transition system TS is a partial function

$$f : (q_0, e_0, u_0, \cdots, q_{i-1}, e_{i-1}, u_{i-1}, q_i, e_i) \mapsto u_i,$$

such that $(q_i, e_i, u_i, q_{i+1}) \in R$ for all $i \geq 0$. A control strategy is said to be *memoryless* if f is a partial function $f : (q_i, e_i) \mapsto u_i$, for all $i \geq 0$.

Similarly, if the transition system TS is closed in the sense that all actions are controllable (i.e., $\mathcal{A}_{uc}$ is empty), a control strategy is given by

$$f : (q_0, u_0, \cdots, q_{i-1}, u_{i-1}, q_i) \mapsto u_i,$$

such that $(q_i, u_i, q_{i+1}) \in R$ for all $i \geq 0$. A memoryless control strategy is given by $f : q_i \mapsto u_i$, for all $i \geq 0$.

A *controlled execution* of a transition system TS is an execution of TS, where for each $i \geq 0$, u_i is chosen according to the control strategy f. Note that, in each step of a controlled execution, an uncontrollable action is followed by a controlled action, i.e., the control decision is made after the uncontrolled action is observed.

Synthesis Problem: Given a transition system TS and a temporal logic formula φ, compute a strategy f of the form $u_i = f(e_0, u_0, \cdots, e_{i-1}, u_{i-1}, q_i, e_i)$ such that any controlled execution of TS satisfies φ. If such a strategy exists, we call the tuple (TS, φ) to be *synthesizable.*

We use a simple example to illustrate the meaning of a control strategy for transition systems.

Example 1. Consider a finite transition system TS with $I = \{q_0\}$, $Q = \{q_0, q_1, q_2\}$, $\mathcal{A} = \{a, b\}$. The transition relation R is shown in Figure 1. We consider two memoryless

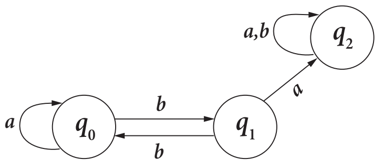

Figure 1: A finite transition system TS with three states $Q = \{q_0, q_1, q_2\}$ and two controllable actions $\mathcal{A} = \{a, b\}$.

control strategies. Let f_a be the memoryless control strategy that chooses action a for each $q \in Q$, and f_b be the memoryless control strategy that chooses action b for each $q \in Q$. We want the system state to always stay in $\{q_0\} \cup \{q_1\}$, i.e., satisfies the LTL specification $\varphi = \Box(q_0 \vee q_1)$. It is clear that, with either of the two strategies f_a or f_b, the controlled executions of TS satisfy φ.

More generally, if the transition system is an open system that has environment inputs, the corresponding control strategy provides a reactive control mechanism such that the controlled executions of the overall system satisfy certain assumption-guarantee type specification $\varphi_e \to \varphi_s$, as shown schematically in Figure 2 where φ_e and φ_s are LTL formulas. In other words, the control strategy should ensure correct behavior of the system (specified by φ_s) for all allowable behavior of the environment (specified by φ_e).

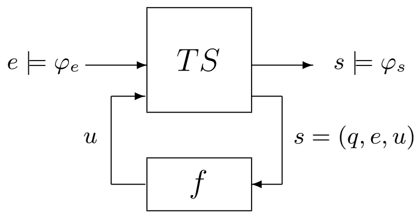

Figure 2: The interconnection of a finite transition system model TS and a control strategy f.

Before formulating the robust synthesis problem, we present a simple motivating example.

Example 2. With the transition system in Figure 1, we consider the case where such a model may not be an exact model of the plant to be controlled, e.g., there are possible unmodeled transitions. Let $\delta_1 = (q_0, a, q_1)$ be such an uncertain transition. With this uncertainty, the control strategy f_a can lead to executions that no longer satisfy the same specification $\varphi = \Box(q_0 \vee q_1)$, whereas the control strategy f_b still ensures the same specification is satisfied. As another case, adding the uncertain transition $\delta_2 = (q_1, b, q_2)$ will make the strategy f_b no longer ensure correct executions, whereas the strategy f_a still does so.

Example 2 shows that, due to different modeling uncertainties, we may prefer one control strategy over another. The goal of this paper is to synthesize robust control strategies that ensure correct behavior of the system under such modeling uncertainties. The problem will be formally stated in the next section.

3. PROBLEM FORMULATION

Most control design procedures are based on the use of a design model of the plant under control. Since no single fixed model can respond exactly like the true plant, one needs to account for the mismatch between the model and the plant in the design procedure. A common approach is to design controllers that are guaranteed to be correct when implemented not only on a design model but also on a family of models that contain the design model as the nominal representation of the plant behavior. We will call such controllers to be robust to modeling uncertainties.

The effectiveness of the design of robust controllers heavily relies on the extent as well as the representation of modeling uncertainties due to a number of factors including: (a) As the extent at which the modeling uncertainties are accounted for in the design procedure increases, the likelihood that the true plant behavior is covered increases. Hence, it is more likely that the resulting control protocol works when implemented on the true plant. On the other hand, increasing modeling uncertainty may increase the conservatism of the procedure. (b) The representation of uncertainties—coupled with that of the plant itself—will determine the technique which can be used for synthesis and its computational complexity. In general, synthesis of robust control protocols requires a computational cost higher than that for nominal control protocols and may even become intractable.

3.1 Robust synthesis problem

With the issues discussed earlier in this section, we now introduce the uncertainty model used in this paper. In the following, unless noted otherwise, we consider open transition systems and drop the adjective "open" for brevity. Consider an action-deterministic and non-blocking nominal transition system $TS = (Q, I, \mathcal{A}_{uc}, \mathcal{A}_c, R_{nom})$ and the set of transitions

$$\mathbf{\Delta} := \{(q, e, u, q') \in (Q \times \mathcal{A}_{uc} \times \mathcal{A}_c \times Q) \backslash R_{nom} \ : \ \exists q'' \in Q \ \text{s.t.} \, (q, e, u, q'') \in R_{nom}\} .$$

The set $\mathbf{\Delta}$, for each nominal transition (q, e, u, q') under the uncontrolled action e and controlled action u, includes all the possible transitions that can be taken from the state q excluding those already in R_{nom}.

Definition 3. Let an action-deterministic and non-blocking nominal transition system $TS = (Q, I, \mathcal{A}_{uc}, \mathcal{A}_c, R_{nom})$ and a subset Δ of $\mathbf{\Delta}$, associated with TS as defined above, be given. Then, an uncertain transition system (TS, Δ) is defined as

$$(TS, \Delta) := (Q, I, \mathcal{A}_{uc}, \mathcal{A}_c, R_{nom} \cup \Delta).$$

An execution of (TS, Δ) is a sequence

$$(q_0, e_0, u_0), (q_1, e_1, u_1), (q_2, e_2, u_2), \ldots$$

such that $q_0 \in I$ and $(q_i, e_i, u_i, q_{i+1}) \in R_{nom} \cup \Delta$ for all $i \geq 0$.

Then, the robust synthesis problem is as follows.

Robust Synthesis Problem: Given a nominal transition system TS, a set $\Delta \subseteq \mathbf{\Delta}$ of possible unmodeled transitions, and a temporal logic formula φ, compute a strategy f of the form $u_i = f(q_0, e_0, u_0, \cdots, q_{i-1}, e_{i-1}, u_{i-1}, q_i, e_i)$ such that any controlled execution of the uncertain transition system (TS, Δ) satisfies φ. If such a strategy exists, we call the tuple $((TS, \Delta), \varphi)$ to be *robustly synthesizable.*

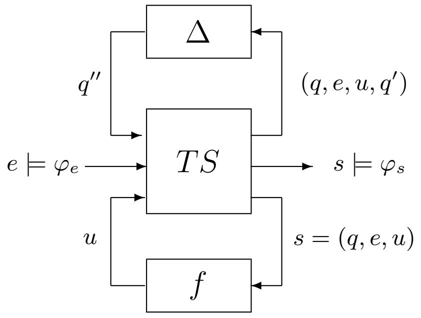

Figure 3: The interconnection of nominal plant model TS, the perturbations due to the unmodeled transitions in Δ, and the strategy f.

The resulting system structure is shown in Figure 3. The transition (q, e, u, q') to be taken by the system under the controlled action u and the uncontrolled, environment action e may be perturbed to a transition $(q, e, u, q'') \in \Delta$ (but not in R_{nom}). The robust strategy f picks the control action u so that the system interconnection between the nominal transition system TS perturbed by the possible unmodeled transitions, and the strategy satisfies an assumption-guarantee type specification $\varphi_e \rightarrow \varphi_s$. In each step of the controlled execution of the uncertain transition system, an uncontrollable action is followed by a controlled action which, in turn, is followed by an "action" by Δ.

Remark 1. In this section and throughout the paper, the "action" due to the unmodeled transitions in Δ is not explicitly notated and it is implicitly accounted for as nondeterminism in the state space. On the other hand, if this action is denoted by δ, then each transition $(q, e, u, q') \in R_{nom} \cup \Delta$ can be considered as $(e, q) \xrightarrow{u} q'' \xrightarrow{\delta} q'$ for some $q'' \in Q$. As such the system in Figure 3 can be considered as a combination of the so-called Mealy ($(e, q) \xrightarrow{u} q''$ part) and Moore ($q'' \xrightarrow{\delta} q'$ part) machines [6].

3.2 Assessing the level of robustness

In order to reason about the effects of different sets of unmodeled transitions, we consider a finite family $\mathcal{C}$ of subsets of $\mathbf{\Delta}$ and partial orders over $\mathcal{C}$ that formalize a notion for comparing the sets in $\mathcal{C}$.

Definition 4. Let P be a set. A partial order on P is a binary relation $\leq$ on P such that, for all $x, y, z \in P$, (i) $x \leq x$, (ii) $x \leq y$ and $y \leq x$ imply $x = y$, and (ii) $x \leq y$ and $y \leq z$ imply $x \leq z$. A set P equipped with a partial order relation $\leq$ is said to be a partially ordered set. We sometimes use the shorthand notation $(P, \leq)$ to denote that the set P is partially ordered with respect to the order relation $\leq$.

Now, assume that the partially ordered set $(\mathcal{C}, \leq)$ is equipped with a rank function $r : \mathcal{C} \rightarrow \mathbb{N}$ such that (a) $r(x) < r(y)$ whenever $x < y$, and (b) $r(y) = r(x) + 1$ whenever $y > x$ is an immediate neighbor of x. For $\alpha \in \mathbb{N}$, let $C_\alpha := \{\Delta \in \mathcal{C} \ : \ r(\Delta) = \alpha\}$.

A partial order on $\mathcal{C}$ may represent a qualitative preference over the sets of unmodeled transitions in $\mathcal{C}$. For example, in a robot motion planning problem, a designer may consider uncertainties in translation to be more critical than those in rotation and prefer those strategies that are robust to uncertainties in translation. Another natural choice as a partial order $\leq$ is defined through set inclusion $\subseteq$, that is, for $\Delta_1, \Delta_2 \in \mathcal{C}$, $\Delta_1 \leq \Delta_2$ if and only if $\Delta_1 \subseteq \Delta_2$. Partial orders induced by set inclusion will be of particular interest in the subsequent sections.

The representation of $\mathbf{\Delta}$ is explicitly parametrized in the state q and the actions e and u for simplicity. Under certain conditions, more compact representations of uncertainty may be available. Uncertain transitions may only depend on the controlled actions. For example, in a robot motion planning scenario, they may depend on which one of the translation and rotation motion primitives are applied rather than at what particular state they are applied. Furthermore, a partial order may be induced by a metric on the state space Q in cases where a notion of "closeness" in Q exists.

Example 3. Consider the example introduced in section 2 and the family $\mathcal{C}$ composed of the following sets of unmodeled transitions.

$$\begin{aligned}
&\Delta_0 = \{\}, \ \Delta_1 = \{(q_1, a, q_1)\}, \ \Delta_2 = \{(q_1, b, q_2)\}, \\
&\Delta_3 = \{(q_1, a, q_1), (q_1, b, q_2)\}, \\
&\Delta_4 = \{(q_1, a, q_1), (q_2, a, q_1)\}, \ \text{and} \\
&\Delta_5 = \{(q_2, a, q_1), (q_1, a, q_1), (q_1, b, q_2)\}.
\end{aligned}$$

A partial order $\leq$ on $\mathcal{C} = \{\Delta_0, \Delta_1, \ldots, \Delta_5\}$ induced by set inclusion is depicted in Figure 4. For example, $\Delta_3 \geq \Delta_2$

because $\Delta_2 \subseteq \Delta_3$. On the other hand, Δ_3 and Δ_4 cannot be distinguished by $\leq$.

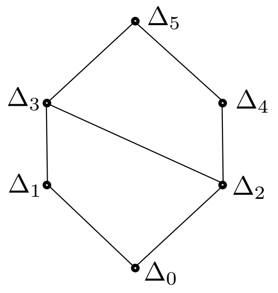

Figure 4: Graphical representation of $(\mathcal{C}, \leq)$ in Example 3.

Given a nominal plant model TS, a temporal logic specification φ, and a partially ordered set $(\mathcal{C}, \leq)$, define the maps $h : \mathcal{C} \to \{0,1\}$ and $h_\forall : \mathbb{N} \to \{0,1\}$ as

$$h(\Delta) := \begin{cases} 1, & \text{if } ((TS,\Delta),\varphi) \text{ is robustly synthesizable,} \\ 0, & \text{otherwise;} \end{cases}$$

$$h_\forall(\alpha) := \begin{cases} 1, & \text{if } h(\Delta) = 1 \text{ for all } \Delta \in C_\alpha, \\ 0, & \text{otherwise} \end{cases}$$

and consider the following optimization, which aims to assess the level (induced by the partial order $(\mathcal{C}, \leq)$) of robustness.

$$\text{maximize} \;\; \alpha \text{ subject to } h_\forall(\alpha) = 1 \qquad (1)$$

over $\alpha \in \mathbb{N}$. The problem in (1) computes the maximum α^* such that $((TS,\Delta),\varphi)$ is robustly synthesizable for all Δ with $r(\Delta) = \alpha^*$. This problem is relevant when the system operates under different modes associated with different types of uncertainties of varying levels and correct operation is desired in all these modes. Consider now a different scenario where the sets in $\mathcal{C}$ correspond to the unmodeled transitions in different modes of the system among which a selection is possible. For example, such modes may be due to the availability of a collection of different sensing or actuation equipment. In this case, the following optimization searches for a maximal uncertainty set Δ—and its rank in the partial order—for which $(TS,\Delta),\varphi)$ is robustly synthesizable.

$$\text{maximize} \;\; \alpha \text{ subject to } h_\exists(\alpha) = 1 \qquad (2)$$

over $\alpha \in \mathbb{N}$. The problem in (2) computes the maximum rank α^* in the partial order such that $((TS,\Delta),\varphi)$ is robustly synthesizable for at least one Δ with $r(\Delta) = \alpha^*$.

Both problems (1) and (2) are optimizations over discrete sets and, as such, may require to check the robust synthesizability of $((TS,\Delta),\varphi)$ for all $\Delta \in \mathcal{C}$. Under certain conditions on the partial order, systematic search (which is faster than exhaustive search over $\mathcal{C}$) may be possible. We discuss one such case, where the partial order is induced from set inclusion, in section 4.3.

4. ROBUST DISCRETE SYNTHESIS

We now present the main results of the paper: reformulation of the robust synthesis problem as a temporal logic game, a polynomial complexity solution procedure for a fragment of LTL specifications, and methods for assessing the level of robustness as characterized by the optimizations in (1) and (2).

4.1 Game reformulation

One approach to the problem of synthesizing a control strategy for an open system is recasting the problem as an infinite horizon game (also known as infinite games [8, 15]). We first introduce the elements of such games and then apply to the robust synthesis problem.

In particular, we consider two-player turn-based games where, at each turn, a move by player 1 is followed by a move by player 2. Roughly speaking, we treat player 1 as an adversarial environment that tries to falsify the specification and player 2 as the system that tries to satisfy it. Formally, a game is defined as follows.

Definition 5. [4] A *game structure* is a tuple

$$G = \langle \mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, \phi \rangle,$$

where

- $\mathcal{V}$ is a finite set of *state variables*,
- $\mathcal{X} \subseteq \mathcal{V}$ is a set of *input variables*, i.e., variables controlled by player 1,
- $\mathcal{Y} = \mathcal{V} \setminus \mathcal{X}$ is the set of *output variables*, i.e., variables controlled by player 2,
- θ_e is an atomic proposition over $\mathcal{X}$ characterizing initial states of the input variables,
- θ_s is an atomic proposition over $\mathcal{Y}$ characterizing initial states of the output variables,
- $\rho_e(\mathcal{V}, \mathcal{X}')$ is the transition relation for player 1, which is an atomic proposition that relates a state and possible next input values (primed variables represent the value of a variable in the next step),
- $\rho_s(\mathcal{V}, \mathcal{X}', \mathcal{Y}')$ is the transition relation for player 2, which is an atomic proposition that relates a state and an input value to possible output values,
- ϕ is the winning condition given by an LTL formula over $\mathcal{V}$.

Valuations of the variables are denoted by lower case letters (e.g., $v \in dom(V)$ for $V \in \mathcal{V}$); and the valuations of the state variables $\mathcal{V}$ are called *states.* A *play* (i.e., a sequence of states) is said to be winning for player 2 if (i) either at a given state v in the sequence there is no $x' \in \mathcal{X}$ such that (v, x') satisfies ρ_e or (ii) the sequence is infinite and it satisfies ϕ. A *strategy* for player 2 is a partial function $f : (v_0, v_1, \ldots, v_{k-1}, x_k) \mapsto y_k$ which chooses a move of player 2 among its allowable moves based on the state sequence so far and the last move of player 1. Strategy f is said to be *winning* for player 2, if all plays starting from the initial states characterized by θ_e and θ_s, and compliant with f are winning. For LTL specifications, if there exists a winning strategy for player 2, it is known that there always exists a finite memory winning strategy [8].

Next, we discuss how synthesis problems and finding a winning strategy in a game are related. Consider an open nominal transition system $TS = (Q, I, \mathcal{A}_{uc}, \mathcal{A}_c, R_{nom})$ and an LTL specification φ. Define the game structure $G_o(TS)$

with the following elements:

$$\begin{aligned}
&\mathcal{X} = Q \times \mathcal{A}_{uc},\\
&\mathcal{Y} = \mathcal{A}_c,\\
&\theta_e = \{(q,e) \ : \ q \in I \text{ and } \exists u, q' \text{ s.t. } (q,e,u,q') \in R_{nom}\},\\
&\theta_s = \{(q,e,u) \ : \ q \in I \text{ and } \exists q' \text{ s.t. } (q,e,u,q') \in R_{nom}\},\\
&\rho_e = \{(q,e,u,q',e') \ : \ (q,e,u,q') \in R_{nom} \text{ and }\\
&\qquad\qquad\qquad\qquad \exists u', q'' \text{ s.t. } (q',e',u',q'') \in R_{nom}\},\\
&\rho_s = \{(q,e,u,q',e',u') \ : \ (q,e,u,q') \in R_{nom} \text{ and }\\
&\qquad\qquad\qquad\qquad \exists q'' \text{ s.t. } (q',e',u',q'') \in R_{nom}\},\\
&\phi = \varphi,
\end{aligned}$$

where, by slight abuse of notation, θ_α, ρ_α for $\alpha = \{e, s\}$ are given as sets but they should be understood as the indicator functions of the corresponding sets. Given an uncertainty set Δ, the game structure $G_o((TS, \Delta))$ is defined similarly by replacing R_{nom} with $R_{nom} \cup \Delta$ in the above formulation associating the elements of the game structure with those of the transition system.

Proposition 1. $((TS, \Delta), \varphi)$ is robustly synthesizable if and only if there exists a winning strategy for player 2 in game $G_o((TS, \Delta))$.

Note that, by construction, there is a one-to-one correspondence between the plays of the game $G_o((TS, \Delta))$ and the executions of the transition system (TS, Δ). Since we assume that TS is non-blocking, so is (TS, Δ). Hence all executions of (TS, Δ) and all plays of the game will be infinite sequences which implies a play winning for player 2 should satisfy ϕ (equivalently, φ). Therefore, Proposition 1 follows from the definitions of winning strategies for the games and synthesizability of control strategies for transition systems.

The relation between the game structures and synthesis problems is quite generally applicable (see [4] for more details). For example, the nominal synthesis problem for an open transition system can be solved through $G_o(TS)$. Furthermore, for a nominal closed transition system $TS = (Q, I, \mathcal{A}, R_{nom})$, a robust control strategy can be computed through a slight modification, namely by redefining

$$\begin{aligned}
&\mathcal{X} = Q, \ \mathcal{Y} = \mathcal{A}_c, \ \phi = \varphi, \ \theta_e = I,\\
&\theta_s = \{(q,u) \ : \ q \in I \text{ and } \exists q' \text{ s.t. } (q,u,q') \in R_{nom} \cup \Delta\},\\
&\rho_e = \{(q,u,q') \ : \ (q,u,q') \in R_{nom} \cup \Delta \text{ and }\\
&\qquad\qquad\qquad \exists u', q'' \text{ s.t. } (q',u',q'') \in R_{nom} \cup \Delta\},\\
&\rho_s = \{(q,u,q',u') \ : \ (q,u,q') \in R_{nom} \cup \Delta \text{ and }\\
&\qquad\qquad\qquad \exists q'' \text{ s.t. } (q',u',q'') \in R_{nom} \cup \Delta\}.
\end{aligned}$$

In summary, the two-player temporal logic game formulation provides a flexible framework to solve the (robust and nominal) synthesis questions discussed in this paper.

Remark 2. The nominal synthesis problem for a closed transition system fits the temporal logic game framework. On the other hand, this problem can also be recast (and can be more efficiently solved) as a model checking question.

4.2 Polynomial-complexity solutions

Solving the two-player game discussed in section 4.1, i.e., checking which one of the system and environment wins and, if the system can win, extracting a control strategy, is known to have 2EXPTIME complexity for general LTL specifications φ [18]. On the other hand, recent advances mostly exploiting the observation, that "typical" specifications in practice have a structure that can be algorithmically exploited [7], have focused on relatively expressive fragments of LTL. For example, if the winning condition is one of the LTL formulas $\Box p$, $\Diamond q$, $\Box\Diamond p$, or $\Diamond\Box q$, where p and q are atomic propositions, then the computational cost of solving the corresponding game is quadratic in the number possible valuations of the pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$.

A quite more expressive winning condition for which there are polynomial complexity algorithms is the so-called generalized reactivity [1] (GR[1]) specifications, i.e., formulas of the form [4]

$$\varphi_e \to \varphi_s,$$

where for $\beta \in \{e, s\}$,

$$\varphi_\beta = \bigwedge_{i \in I_\beta} \Box\sigma_i^\beta \wedge \bigwedge_{j \in J_\beta} \Box\Diamond\pi_j^\beta$$

with the finite set of indices I_e, J_e, I_s, and J_s and propositional logic formulas σ_i^e, π_j^e, σ_i^s, and σ_j^s. Here, σ_i^e and π_j^e describe the safety and liveness assumptions on the environment. Similarly,$\sigma_i^s$and σ_j^s describe the safety and liveness guarantees on the system behavior, respectively. The computational cost of solving GR[1] games is $O(N^3|J_e||J_s|)$ where $|\cdot|$ indicates the cardinality of the respective sets and N is the number of possible valuations of the pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Moreover, there exists computational tools that leverage this polynomial complexity bound, e.g., the digital design synthesis tool in the JTLV framework [19]. This tool has been used for applications including correct-by-construction synthesis of control protocols for autonomous navigation [21] and design of bus arbiter [10].

We now establish that robustification of the synthesis problem as discussed in the previous sections preserves the GR[1] property, i.e., if the nominal problem can be solved as a GR[1] game, then the robust synthesis problem can be solved as a GR[1] game as well. To this end, for a nominal transition system $TS = (Q, I, \mathcal{A}_{uc}, \mathcal{A}_c, R_{nom})$ and an uncertainty set Δ, define

$$B^\Delta_{(q,e,u)} := \{q' \in Q \ : \ (q, e, u, q') \in R_{nom} \cup \Delta\}.$$

Proposition 2. Let φ be a GR[1] specification. Then, the robust synthesizability of $((TS, \Delta), \varphi)$ can be solved as a GR[1] game.

Proof: Observe that the winning condition for the robust synthesis problem can be written as

$$\left\{ \bigwedge_{(\tilde{q},\tilde{e},\tilde{u})} \Box \Big[(q = \tilde{q} \wedge e = \tilde{e} \wedge u = \tilde{u}) \to (q' \in B^\Delta_{(\tilde{q},\tilde{e},\tilde{u})}) \Big] \right\} \to \varphi. \quad (3)$$

If φ is a GR[1] formula, then that in (3) is also a GR[1] formula and the result follows. □

By Proposition 2 and the definition of the two-player game in section 4.1, the worst-case computational complexity of the robustified GR[1] synthesis problem is cubic in the number of possible valuations of $(x, y) \in \mathcal{X} \times \mathcal{Y}$; hence, its complexity class is the same as the nominal synthesis problem. Note that the numbers $|J_e|$ and $|J_s|$ are not affected by robustification.

4.3 Algorithms for assessing the level of robustness

We now revisit the optimization problems formulated in section 3.2. We discuss procedures for computing upper and lower bound on their optimal values and for systematic reduction of the gap between these bounds.

Note that both problems, in general, involve search over a finite, discrete $\mathcal{C}$. As such they can be solved with a worst-case computational cost that is linear in the cardinality of $\mathcal{C}$, i.e., evaluate $h(\Delta)$ for each $\Delta \in \mathcal{C}$. Yet, each evaluation of h requires solving a robust synthesis problem and is computational demanding. Hence, it is desirable to limit the number of evaluations of h. To this end, we now discuss a bisection-type procedure for partially ordered sets $(\mathcal{C}, \leq)$ which satisfies the following monotonicity condition.

Monotonicity condition: For each $\Delta_1, \Delta_2 \in \mathcal{C}$ such that $\Delta_1 \geq \Delta_2$, it holds that $h(\Delta_1) \leq h(\Delta_2)$.

That is, for given nominal transition system TS and specification φ, if $((TS, \Delta_1), \varphi)$ is robustly synthesizable for some $\Delta_1 \in \mathcal{C}$ (i.e., $h(\Delta_1) = 1$), the $((TS, \Delta_2), \varphi)$ is robustly synthesizable for every $\Delta_2 \in \mathcal{C}$ such that $\Delta_2 \leq \Delta_1$.

Proposition 3. Let $\mathcal{C}$ be a finite family of subsets of $\mathbf{\Delta}$ and let $\leq$ be a partial order induced from set inclusion. Then, $(\mathcal{C}, \leq)$ satisfies the monotonicity condition.

Let $\Delta_1, \Delta_2 \in \mathcal{C}$ be such that $\Delta_1 \geq \Delta_2$. The proof of Proposition 3 relies on the fact that the robust synthesizability of $((TS, \Delta_1), \varphi)$ does not only imply robust synthesizability of $((TS, \Delta_2), \varphi)$, but also the control strategy of the former is a control strategy for the other.

For partially ordered set $(\mathcal{C}, \leq)$ that satisfies the monotonicity condition, the bisection-type algorithm in Figure 5 can be used to compute upper and lower bounds on the optimal value of the optimization in (1). In Figure 5 and hereafter, for a real number a, $\lfloor a \rfloor$ denotes the largest integer smaller than or equal to a. A number of remarks about this algorithm is in order. First, it exploits the monotonicity condition: If $h_\forall(\tilde{\alpha})$ is evaluated to be 1 for some $\tilde{\alpha}$, then it must be equal to 1 for all $\alpha \leq \tilde{\alpha}$ and, therefore, the sets in $\mathcal{C}$ with $r(\Delta) < \tilde{\alpha}$ can be disregarded in the search for a solution for the optimization in (1) and $\tilde{\alpha}$ is a lower bound. Similarly, if $h_\forall(\tilde{\alpha}) = 0$ for some $\tilde{\alpha}$, then $h_\forall(\Delta) = 0$ for all $\alpha \leq \tilde{\alpha}$ and, therefore, the sets in $\mathcal{C}$ with $r(\Delta) > \tilde{\alpha}$ can be disregarded and $\tilde{\alpha}$ is an upper bound on the optimal value. Second, assuming that the empty set is in $\mathcal{C}$ and (TS, φ) is nominally synthesizable, the algorithm maintains a lower bound b_l and an upper bound b_u at each iteration. Hence, at any iteration the algorithm is terminated, it returns a value of $\alpha = b_l$ such that $((TS, \Delta), \varphi)$ is robustly synthesizable for all $\Delta \in \mathcal{C}_\alpha$ and a measure of suboptimality, i.e., $b_u - b_l$. This measure of suboptimality is reduced, roughly, by half, in each iteration.

The algorithm in Figure 5 can be adapted for the optimization in (2) by replacing $h_\forall$ in line 5 to $h_\exists$. The main difference between the two algorithms is the potential number of evaluations of h in line 5. Each evaluation of $h_\forall(\alpha)$ to 1 requires solving as many robust synthesis problems as the cardinality of $\mathcal{C}_\alpha$ whereas it is enough to find a single $\Delta \in \mathcal{C}_\alpha$ to conclude that $h_\forall(\alpha) = 0$. On the other hand, finding a single $\Delta \in \mathcal{C}_\alpha$ to conclude that $h_\exists(\alpha) = 0$.

Remark 3. Note that while running the algorithm in Figure 5, the monotonicity condition provides a means to prune the search space which, in turn, potentially reduces the number of synthesizability checks required during the search. However, the worst-case complexity in general remains the same.

Given: nominal transition system TS, partially ordered set $(\mathcal{C}, \leq)$, upper and lower bounds b_l and b_u with $b_l < b_u$, tolerance TOL, and maximum number $N_{max} \geq 1$ of iterations.
Output: Upper and lower bounds b_l and b_u such that $b_u - b_l < TOL$ or their values at iteration N_{max}.

```
1. $N \leftarrow 1$
2. while $N \leq N_{max}$
3.   while $b_u - b_l \geq TOL$
4.     $\alpha \leftarrow \lfloor (b_l + b_u)/2 \rfloor$
5.     if $h_\forall(\alpha) = 1$ then $b_l \leftarrow \alpha$
6.     else $b_u \leftarrow \alpha$
7.     $N \leftarrow N + 1$
```

Figure 5: Bisection-type algorithm for the optimization in (1).

The problems in (1) and (2) provide means for quantifying the level of robust synthesizability of (TS, φ) with respect to the unmodeled transitions captured by $(\mathcal{C}, \leq)$. Another way of such quantification would be determining a "maximal uncertainty set" $\Delta \in \mathcal{C}$ such that $((TS, \Delta), \varphi)$ is robustly synthesizable. We now formalize this notion and discuss, through an example, why it may not be a suitable way of quantification in general.

Definition 6. Given a nominal transition system TS, an LTL specification φ, and a partially ordered set $(\mathcal{C}, \leq)$ of unmodeled transitions where $\leq$ is induced from set inclusion, a *maximal uncertainty set* Δ_{max} is one such that $((TS, \Delta_{max}), \varphi)$ is robustly synthesizable and $(TS, \Delta), \varphi)$ is not robustly synthesizable for any $\Delta \geq \Delta_{max}$.

Fact: Maximal uncertainty set Δ_{max} (as defined in Definition 6) may not be unique in general.

Indeed, consider the setup in Example 2. Note that both $((TS, \{\delta_1\}), \varphi)$ and $((TS, \{\delta_2\}), \varphi)$ robustly synthesizable. On the other hand, it can be shown that $((TS, \{\delta_1, \delta_2\}), \varphi)$ is not robustly synthesizable. Let now $\tilde{\Delta} \in \mathcal{C}$ be such that $\tilde{\Delta} \geq \{\delta_1\}$, $\tilde{\Delta} \geq \{\delta_2\}$, and $((TS, \tilde{\Delta}, \varphi)$ is robustly synthesizable. Then, the fact that $\tilde{\Delta} \geq \{\delta_1, \delta_2\}$ leads to a contradiction by Proposition 3. Therefore, such a set $\tilde{\Delta}$ cannot exist and $\{\delta_1\}$ and $\{\delta_2\}$ are either maximal uncertainty sets or contained in two different maximal uncertainty sets. Note that the counterexample, which demonstrated the non-uniqueness of maximal uncertainty sets, discussed here is a safety formula and that it is possible to construct counterexamples that include liveness formulas.

Finally, note that the maximal uncertainty set Δ—among all the maximal uncertainty sets—with the highest rank solves the optimization in (2).

5. EXAMPLE

We demonstrate the robust discrete synthesis framework presented in the previous sections on a simple robot motion planning scenario. Consider a mobile robot in a 2D plane.

Nominal transition system: We start with a nominal transition system TS with three different controlled actions, $\mathcal{A}_c = \{L, R, S\}$, which correspond to `turn_left`, `turn_right`, and `go_straight`, respectively. Let $q = (x, y, \theta)$ be a tuple, where x, y are the coordinates of the robot in an $N \times N$ grid of cells and θ is the heading angle. The nominal transitions of TS are given by the following equations.

- Action $= L$: $x' = x$, $y' = y$, $\theta' = \theta + \frac{\pi}{2}$,
- Action $= R$: $x' = x$, $y' = y$, $\theta' = \theta - \frac{\pi}{2}$,
- Action $= S$: $x' = x + \cos(\theta)$, $y' = y + \sin(\theta)$, $\theta' = \theta$,

where $q' = (x', y', \theta')$ is the valuation of (x, y, θ) at the next time step under a given transition. The transition system TS can be obtained by an abstraction of the underlying dynamics of the mobile robot. In the nominal case, we require that, following a turning action L or R, the robot completes a corresponding 90-degree turn and remains in the same cell, whereas the robot moves forward by 1 cell following an S action. Note that θ only takes value in integer multiples of $\pi/2$ modulo 2π, and (x, y) only nonnegative integers in $[0, N-1]$. We introduce an additional boolean state `Out` to indicate when the location of the robot (x, y) goes out the $N \times N$ grid.

Specifications: Depicted in Figure 6, the desired properties for the robot to satisfy are specified as follows:

(S1) Always remain inside of the $N \times N$ region.

(S2) Visit each of the red cells, labeled as P_1, P_2, and P_3, infinitely often.

(S3) Eventually go to the blue cell P_0 after a PARK signal is received.

Here, the PARK signal is an environment variable that constrains the behavior of the robot. The following assumption is made on the PARK signal.

(A1) Infinitely often, a PARK signal is not received.

We use the positive integer n to indicate the size of the target sets P_i, $i = 1, 2, 3, 4$.

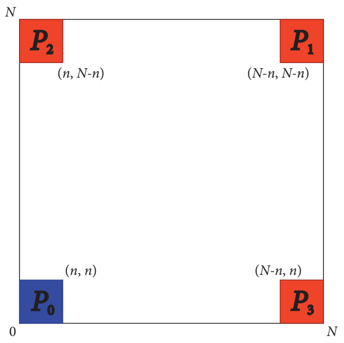

Figure 6: The robot should visit P_1, P_2, and P_3 infinitely often, and eventually go to P_0 after receiving a PARK signal.

Uncertain transition system: Furthermore, to demonstrate our robust discrete synthesis framework, we introduce unmodeled transitions following both the turning actions $\{L, R\}$ and the action $\{S\}$. These transitions are given by the following equations.

- Action $= L$:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + \delta_x\sqrt{2}\cos(\theta + \pi/4) \\ y + \delta_y\sqrt{2}\sin(\theta + \pi/4) \\ \theta + \frac{\pi}{2} \end{bmatrix}, \ \delta_x, \delta_y \in [0, \delta_1].$$

- Action $= R$:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + \delta_x\sqrt{2}\cos(\theta - \pi/4) \\ y + \delta_y\sqrt{2}\sin(\theta - \pi/4) \\ \theta - \frac{\pi}{2} \end{bmatrix}, \ \delta_x, \delta_y \in [0, \delta_1].$$

- Action $= S$:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + (\delta_x + 1)\cos(\theta) \\ y + (\delta_y + 1)\sin(\theta) \\ \theta \end{bmatrix}, \ \delta_x, \delta_y \in [0, \delta_2].$$

Here, δ_1 and δ_2 are nonnegative integers used to indicate different levels of uncertainties, and hence demonstrate different levels of robustness of a control strategy. The nominal transitions are given by letting $\delta_1 = \delta_2 = 0$. Note that, again, θ only takes value in integer multiples of $\pi/2$, and (x, y) only nonnegative integers. We keep the same additional state `Out` to indicate when the location of the robot (x, y) goes out the $N \times N$ grid.

Assessing the level of robustness: Different values of δ_1 and δ_2 introduce different number of uncertain transitions, which can be used to demonstrate different levels of robustness of a control strategy. Let R_{nom} denote the set of nominal transitions. Define $\Delta_{i,j}$ to be the set of uncertain transitions introduced by choosing $\delta_1 = i$ and $\delta_2 = j$, where $0 \leq i, j \leq 10$, excluding the nominal transitions R_{nom}. Let $\mathcal{C} = \{\Delta_{i,j} : 0 \leq i, j \leq 10\} \subset \mathbf{\Delta}$. We choose the partial order to be the one induced by set inclusion and define a natural rank function $r : \mathcal{C} \to \mathbb{N}$ to be $r(\Delta_{i,j}) = i + j$. Figure 7 gives a graphical representation of $(\mathcal{C}, \leq)$ and the rank function r, which shows a few sets of uncertainties at different levels.

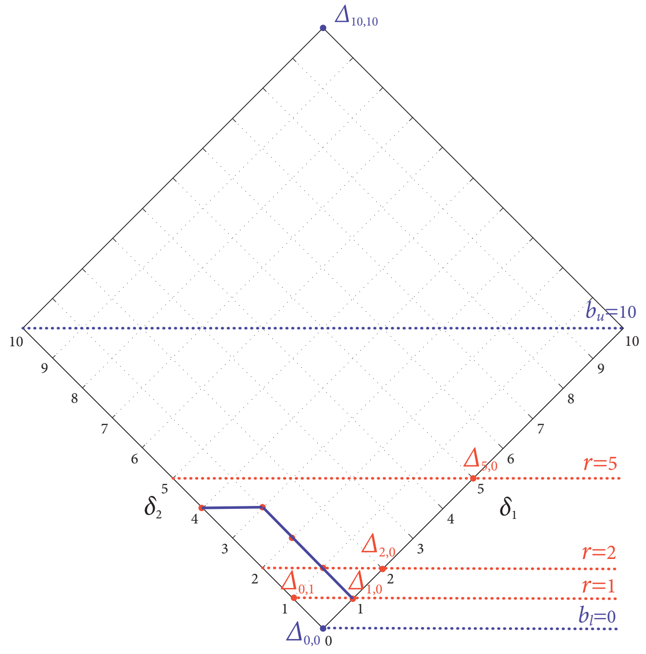

Figure 7: Graphical representation of $(\mathcal{C}, \leq)$. The Δ_{ij}'s show the the uncertainty sets checked in a bisection-type algorithm to solve the optimization problem (1) with $n = 5$. The dashed lines show relevant levels of robustness assessed. The maximal uncertainty sets are connected by thick blue lines, which bound the region of robust synthesizability for $n = 5$.

Results: Following the results in Section 4, the robust synthesis problem can be reformulated as a temporal logic game with GR[1] specifications. We then apply the Temporal Logic Planning (TuLiP) Toolbox, a Python-based code suite for automatic synthesis of correct-by-construction embedded control software [23] to solve such games. TuLiP provides an interface to the JTLV framework.

For illustration, we choose $n = 5$ and consider the optimization problem given by (1). We use a bisection-type algorithm as introduced in Section 4.3 to solve the problem. Starting with $b_l = 0$ and $b_u = 10$, we check whether $h_\forall(\alpha) = 1$ for $\alpha = \lfloor(b_l + b_u)/2\rfloor = 5$. It is found that $(TS, \Delta_{5,0})$ is not synthesizable. Therefore, $h_\forall(5) = 0$. We update $b_u = 5$. Next, we check if $h_\forall(\alpha) = 1$ holds for $\alpha = 2$. It is found that $(TS, \Delta_{2,0})$ is not synthesizable. We update $b_u = 2$. We then check if $h_\forall(1) = 1$ holds for $\alpha = 1$. It is found that both $(TS, \Delta_{1,0})$ and $(TS, \Delta_{0,1})$ are synthesizable. Therefore, $h_\forall(1) = 1$. The optimal solution for (1) is $\alpha^* = 1$. A similar bisection-type algorithm can be applied to solve the problem (2). In this case, $\alpha^* = 4$, with both $(TS, \Delta_{1,3})$ both $(TS, \Delta_{0,4})$ synthesizable. The procedure is illustrated in Figure 7. Moreover, by using a depth-first search on the graph representing the partial order on $\mathcal{C}$, we can plot all the maximal uncertainty sets for $n = 5$ (not unique as discussed earlier in Section 4.3) as shown in Figure 7.

Note that the solution for the optimization problem (1) is given by the maximum level α such that the level set $\mathcal{C}_\alpha$ is totally contained in the region of robust synthesizability, whereas the solution for the optimization problem (2) is given by the maximum level α such that the level set $\mathcal{C}_\alpha$ has nonempty intersection with this region.

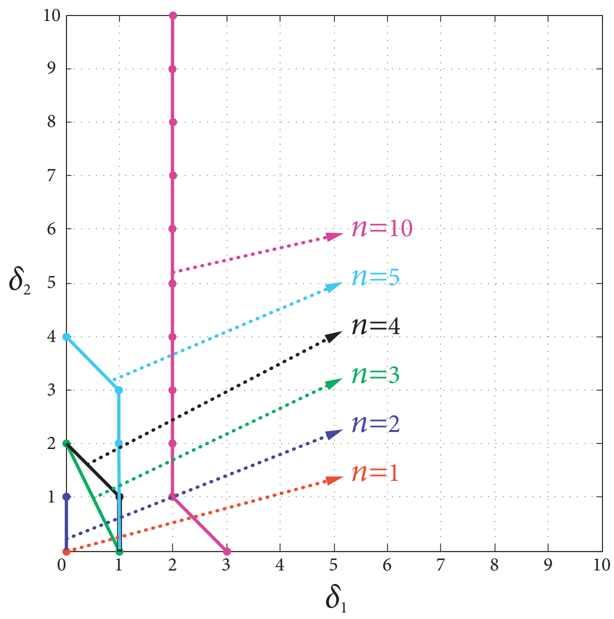

Figure 8: Robustness-performance tradeoff: the colored thick lines mark the exact regions of robust synthesizability for different n. As n increases (i.e., performance degrades), the region of robust synthesizability grows.

Robustness-performance tradeoff: By plotting the maximal uncertainty sets for different n, we can show certain tradeoffs between robustness and performance. As mentioned earlier, the size of the target sets can be seen as a performance indicator of the control strategy. With $n = 1$, only the nominal system is synthesizable. For r changing from 2 to 5, control strategies with increasing levels of robustness with respect to uncertainties can be synthesized. When $n = 10$, the specification reduces to a pure safety constraint $(S1)$. It is shown that for $\delta_1 \leq 2$, there exists a control strategy which only uses the turning actions to achieve safety. For $\delta_1 = 3$, such strategies no longer exist. A control strategy relying the action S can be found if the action S is deterministic. The results are depicted in Figure 8. As n increases (i.e., performance degrades), the exact region of robust synthesizability as bounded by the maximal uncertainty sets grows, which clearly shows a tradeoff between robustness and performance.

6. CRITIQUE

We now discuss some of the limitations and potential extensions of the preliminary results on robustness of discrete control protocols presented in this paper.

Uncertainty representation: The uncertainty representation as captured by the set $\boldsymbol{\Delta}$ explicitly enumerates unmodeled transitions from each state for every controlled and uncontrolled action possible at that state. As mentioned before and used in the presentation of the example in section 5, unmodeled transitions can be represented more compactly. For example, they may only depend on the controlled actions in the case where the controlled actions are chosen from a finite family control/motion primitives. Moreover, if there exists a metric-type function defined over the state space Q, it may be possible to compactly characterize uncertainty sets as sublevel sets of these functions. More specifically, consider the set $B_{(q,e,u)}$ defined in the proof of Proposition 2 and let $d : Q \times Q \to \mathbb{R}$ be a metric. Then, a compact representation of $B^{\Delta}_{(q,e,u)}$ as defined in section 4.2 may be of the form $\{q' \ : \ d(d, q') \leq \gamma\}$ for some $\gamma \geq 0$. Reference [13] uses a metric-based representation of uncertainty. However, the question in [13] is one of sensitivity rather than robustness and no constructive procedure for the synthesis of reactive control protocols was presented.

We emphasize that it is not straightforward to exploit this compactness in representation toward computational complexity reduction in synthesis. For example, the augmented GR[1] specification in (3) is based on explicit listing of all unmodeled transitions. Therefore, uncertainty models that can be exploited in the symbolic manipulations and fixed point iterations [16] of the underlying game solver are needed.

Other uncertainty models: The uncertainty model used here captures the transitions not modeled in the nominal system. These uncertainties, for example, may be due to inaccuracies in actuation. An enabling factor that has been exploited in the paper is that even though these uncertainties introduce uncontrolled actions, the impact of these actions can be sensed after the action is taken and before the next environment observation is made. Further work is needed in order to relax this assumption and also to account for the effects of sensing inaccuracies on the synthesis of discrete control protocols.

Game formulation of robust synthesis: We modeled effects of unmodeled transitions as a new uncontrolled input (in addition to environment in the case of open transition systems) that acts after the environment is observed and controlled action is taken. This modeling choice enabled us to straightforwardly extend the two-player game approach in a way that the two types of uncontrolled actions, the environment and the unmodeled transitions, are treated identically. A deeper understanding of the differences between the two types of uncontrolled actions and exploitation toward reductions in computation complexity are needed.

Other robustness questions: We investigated the robustness of the reactive control protocols to the uncertainties to unmodeled transitions. Investigation of the robustness properties to other types of inaccuracies is subject to current research. For example, the resulting control protocol is guaranteed to be correct with respect to the specification $\varphi_e \rightarrow \varphi_s$ as long as the environment does not violate φ_e. Otherwise, no guarantees can be established (because $\varphi_e \rightarrow \varphi_s$ holds trivially). Design of controllers that not only react to the modeled behaviors of the environment but also tolerate those that violate the assumptions is desirable. More specifically, If $\varphi_e \rightarrow \varphi_s$ is realizable, what is the "least-restrictive" assumption φ_e' such that $\varphi_e' \rightarrow \varphi_s$ is realizable and $\varphi_e \rightarrow \varphi_e'$ holds?

Assessing the level of robustness: We introduced graded partial orders over the family of uncertainty sets as a means to capture the level of robustness a system possesses and also to encode the preference/intent of the designer. In particular, partial orders for which a monotonicity condition holds, we proposed a systematic search for the solutions of the problems in (1) and (2). Effects of different choices of partial orders on the search process and optimal solutions of (1) and (2) are of particular interest. Moreover, often the discrete control protocols are utilized in higher level of hierarchical control structures where the lower levels correspond to continuous evolution controlled by continuous controllers. Therefore, it may be possible to induce metrics from the underlying continuous state space.

In robust control theory, there are quantifiable notions to establish robustness and performance tradeoffs. The example in section 5 demonstrates a similar tradeoff in the context of discrete control protocols where the performance is interpreted as the level of "relaxation" in the liveness part of the specification (until practically reducing to a safety property). Further work is needed for systematically establishing robustness and performance tradeoff in this context.

7. CONCLUSIONS

We studied the robustness of reactive control protocols synthesized to guarantee system's correctness with respect to given temporal logic specifications. Specifically, we investigated the effects of unmodeled transitions on (open) finite transition systems.We formulated the robust synthesis problem as a temporal logic game and showed that robustification preserves attracting worst-case computational complexity bounds for a fragment of linear temporal logic specifications. Finally, we discussed preliminary results on the assessment of the effects of different levels of uncertainties on robust synthesizability.

8. REFERENCES

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

[2] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas. Symbolic planning and control of robot motion: State of the art and grand challenges. *Robotics and Automation Magazine*, 14(1):61–70, 2007.

[3] R. Bloem, K. Greimel, T. A. Henzinger, and B. Jobstmann. Synthesizing robust systems. In *Formal Methods in Computer Aided Design*, 2009.

[4] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Saar. Synthesis of reactive (1) designs. *Journal of Computer and System Sciences, to appear*, 2011.

[5] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.

[6] D. I. A. Cohen. *Introduction to Computer Theory*. Wiley, 1997.

[7] R. Ehlers. Experimental aspects of synthesis. In *Proc. of Workshop on Interactions, Games and Protocols*, 2011. Available at `http://arxiv.org/pdf/1102.4117`.

[8] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[9] T. A. Henzinger. Two challenges in embedded systems design: predictability and robustness. *Philosophical Transactions of the Royal Society - Series A: Mathematical, Physical and Engineering Sciences*, 366(1881):3727–3736, 2008.

[10] B. Jobstmann, S. Galler, M. Weiglhofer, and R. Bloem. Anzu: A Tool for Property Synthesis. In *Computer Aided Verification*, pages 258–262, 2007.

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

[12] H. Kress-Gazit, T. Wongpironsarn, and U. Topcu. Correct, reactive robot control from abstraction and temporal logic specifications. *Robotics and Automation Magazine*, 18(3):65–74, 2011.

[13] R. Majumdar, E. Render, and P. Tabuada. Robust discrete synthesis against unspecified disturbances. In *Proc. Hybrid Systems: Computation and Control*, pages 211–220, 2011.

[14] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems*. Springer-Verlag, 1992.

[15] D. Perrin and J.-E. Pin. *Infinite Words: Automata, Semigroups, Logic and Games*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004.

[16] N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. In *Verification, Model Checking and Abstract Interpretation*, volume 3855 of *Lecture Notes in Computer Science*, pages 364 – 380. Springer-Verlag, 2006.

[17] A. Pnueli. The temporal logic of programs. In *Proc. of the 18th Annual Symposium on the Foundations of Computer Science*, pages 46–57. IEEE, 1977.

[18] A. Pnueli and R. Rosner. On the Synthesis of an Asynchronous Reactive Module. In *Proc. of Colloquium on Automata, Languages and Programming*, pages 652–671. Springer-Verlag, 1989.

[19] A. Pnueli, Y. Sa'ar, and L. Zuck. JTLV: A framework for developing verification algorithms. In *Proc. of Conference on Computer Aided Verification*, volume 6174, pages 171–174, 2010.

[20] D. C. Tarraf, A. Megretski, and M. A. Dahleh. A framework for robust stability of systems over finite alphabets. *IEEE Transactions on Automatic Control*, 53:1133–1146, 2008.

[21] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon control for temporal logic specifications. In *Proc. of Conf. on Hybrid Systems: Computation and Control*, 2010.

[22] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Formal synthesis of embedded control software for vehicle management systems. In *Proc. AIAA Infotech@Aerospace*, 2011.

[23] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. Murray. TuLiP: a software toolbox for receding horizon temporal logic planning. In *Proc. of Conference on Hybrid Systems: Computation and Control*, pages 313–314, 2011.