2015 IEEE 54th Annual Conference on Decision and Control (CDC)
December 15-18, 2015. Osaka, Japan

Controller Synthesis with Inductive Proofs for Piecewise Linear Systems: an SMT-based Algorithm

Zhenqi Huang, Yu Wang, Sayan Mitra, Geir E. Dullerud and Swarat Chaudhuri

Abstract— **We present a controller synthesis algorithm for reach-avoid problems for piecewise linear discrete-time systems. Our algorithm relies on SMT solvers and in this paper we focus on piecewise constant control strategies. Our algorithm generates feedback control laws together with inductive proofs of unbounded time safety and progress properties with respect to the reach-avoid sets. Under a reasonable robustness assumption, the algorithm is shown to be complete. That is, it either generates a controller of the above type along with a proof of correctness, or it establishes the impossibility of the existence of such controllers. To achieve this, the algorithm iteratively attempts to solve a weakened and strengthened versions of the SMT encoding of the reach-avoid problem. We present preliminary experimental results on applying this algorithm based on a prototype implementation.**

I. Introduction

A Satisfiability Modulo Theory (SMT) problem is a classical decision problem in computer science [6]. It takes as input a logical formula in first-order logic that can involve combinations of background theories, and requires one to decide whether or not the formula has a satisfying solution. For a bounded time horizon k, a simplest SMT problem in Equation (1), for instance, is an encoding of a search for a sequence of control inputs vectors $u_1, \ldots, u_k$ that drives a discrete time linear open-loop control system from every initial state in the hypercube $[0, 0.1]^n$ to the hypercube $[0.9, 1]^n$ in k steps, while always keeping the state inside the hypercube $[0, 1]^n$.

$$\begin{aligned}
&\exists\, u_1, \ldots u_k, \forall\, x_0 \in [0, 0.1]^n, \forall\, t \in \{1, \ldots, k-1\},\\
&\text{let } x_{t+1} = Ax_t + Bu_t\\
&\text{such that } x_t \in [0,1]^n \text{ and } x_k \in [0.9, 1]^n. \qquad (1)
\end{aligned}$$

This example has several constraints that are defined in terms of the quantified variables u_i and x_i, numerical constants (including those in the matrices A and B), and the background theory of linear real arithmetic. An SMT solver is a software tool that solves SMT problems by either giving an assignment to the variables that satisfy all the constraints or by saying that none exists. Modern SMT solvers routinely handle linear problems with thousands of variables and millions of constraints, so much as they have become the engines for innovation in verification and synthesis for computer software and hardware [2], [10], [13].

Zhenqi Huang, Yu Wang, Sayan Mitra and Geir E. Dullerud are with Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA {zhuang25, yuwang8, mitras, dullerud}@illinois.edu

Swarat Chaudhuri is with the Department of Computer Science, Rice University, Houston, TX 77005, USA swarat@rice.edu

Although many control systems can only be modeled by means of nonlinear arithmetic over the real numbers involving transcendental functions that make the corresponding SMT problems undecidable, the solvers are evolving rapidly and several incorporate approximate decision procedures for nonlinear arithmetic [15]. These technological developments motivated us (and others [20], [21]) to explore SMT-based controller synthesis.

In this paper, we present an algorithm that uses SMT solvers for synthesizing controllers for discrete time systems. The dynamics of the system is given as a piecewise linear feedback control system. The control requirements are the standard *reach-avoid* specification [7], [8]: a set of states $Goal$ that has to be reached while always staying inside a $Safe$ set. A key difficulty in using SMT for synthesis, is that the resulting SMT problem has to encode the unrolled dynamics of the system with the unknown controller inputs. In the above simple example, this gave rise to k control input variables and the intermediate states. For more general nonlinear models, the intermediate states cannot be written down in closed form and one has to unroll the over-approximations of the dynamics. This can then lead to overtly conservative answers from the solver. We present a technique that avoids this problem by synthesizing the control law together with an inductive proof of its correctness. The proof has two parts: (a) an inductive invariant that implies safety and (b) a ranking function that implies progress. A positive side-effect of this is that it can not only synthesize controllers with correctness proofs, but it can also establish the nonexistence of provably correct controllers (of a certain template).

In Section III we define the system model, the reach-avoid synthesis problem and a particular notion of robustness of models. In Section IV we first present a basic SMT encoding of the synthesis problem and then a strengthened and a weakened version this encoding. Using these two encodings, in Section V we present the synthesis algorithm, its soundness and relative completeness. In Section VI we illustrate an application of the algorithm in a vehicle navigation problem and conclude in Section VII.

II. Related Work

Researchers have recently used SMT solvers for synthesizing programs and strategies in games. The approach in [21] uses SMT to find controllers for general linear temporal logic (LTL) specifications by stitching together motion primitives from a library. Unlike our encoding with inductive proofs, the approach of [21] involves bounded unrolling of the dynamics.

In [20], the authors used SMT solvers to synthesize integrated task and motion plans by constructing a placement graph. In [4], a constraint-based approach was developed to solve games on infinite graphs between the system and the adversary. Our work can be seen as introducing control theoretic constraints to the SMT formulation.

The authors of [11], [25] proposed a game theoretical approach to synthesize controller for the reach-avoid problem, first for continuous and later for switched systems. In these approaches, the reach set of the system is computed by solving a non-linear Hamilton-Jacobi-Isaacs PDE. Our methodology, instead of formulating a general optimization problem for which the solution may not be easily computable, solves a special case exactly and efficiently. With this building block, we are able to solve more general problems through abstraction and refinement.

Model predictive control (MPC) can as well be used to solve the reach-avoid problem [3], [23]. In each cycle of an MPC, the optimal input for reaching the goal while avoiding the obstacle, is computed for a fixed prediction horizon. Then, the first part of the optimal control input is applied, and a new input is computed from the new state, and so on. As the prediction horizon increases, the applied input converges to the optimal reach-avoid input. In the contrast, our approach can be used to synthesis controls for possibly unbounded horizon with safety and progress guarantees and can establish nonexistence of controller of certain type.

There is a large body of results on automata theoretic approaches for controller synthesis [1], [17], [18], [22], [24]. The approach here is to construct a finite abstraction of the dynamical system and then invoke the LTL synthesis algorithms such as the one in [5]. This approach has been applied to several systems and several software tools for synthesis have been implemented [9], [19].

III. PRELIMINARIES AND BACKGROUND

Sets and Functions: For a natural number N, $[N]$ denotes the set $\{0, 1, \ldots, N-1\}$. Given two functions $f, g : A \to \mathbb{R}^n$, we use $d(f, g) = |f(a) - g(a)|_\infty$ to denote the ℓ_∞ distance between f and g. We will use finite collections of sets to approximate arbitrary compact subsets in $\mathbb{R}^n$. For a finite collection $\mathcal{P}$ of subsets of $\mathbb{R}^n$ and a subset $S \subseteq \mathbb{R}^n$, we say that $\mathcal{P}$ *preserves* S if there exists a subset $\mathcal{P}' \subseteq \mathcal{P}$ such that (i) $\bigcup_{P \in \mathcal{P}'} P = S$, and (ii) $\forall\ P \in \mathcal{P} \backslash \mathcal{P}', P \cap S = \varnothing$. A *finite partition* $\mathcal{P}$ of a compact subset $S \subseteq \mathbb{R}^n$ is a finite disjoint collection of sets that exactly cover S. The *resolution* of a partition $\mathcal{P}$ is the maximum diameter of the sets in $\mathcal{P}$. For two partitions $\mathcal{P}, \mathcal{P}'$ of a compact set S, we say that $\mathcal{P}$ *subsumes* $\mathcal{P}'$, if for any $I \in \mathcal{P}$, there exists $I' \in \mathcal{P}'$ such that $I \subseteq I'$.

Piecewise Linear Systems and Robustness: A piecewise linear system M is a tuple $(\mathcal{X}, \mathcal{U}, loc, \mathcal{I}, \mathcal{F})$ where (a) $\mathcal{X} \subseteq \mathbb{R}^n$ is a compact set called the *state space*, (b) $\mathcal{U} \subseteq \mathbb{R}^m$ is a compact set called the *input space*, (c) *loc* is a finite set called the *set of locations*, (d) $\mathcal{I} = \{P_l\}_{l \in loc}$ is a partition of $\mathcal{X}$ and each element of $\mathcal{I}$ is called a *location invariant*, and (e) $\mathcal{F} = \{f_l\}_{l \in loc}$ is a collection of linear *dynamic functions* $f_l : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$. The evolution of the continuous state of the system is governed by the dynamic function of the location invariant it is currently in. For any time $t \in \mathbb{N}$, a state $x_t \in P_l$ and an input $u_t \in \mathcal{U}$ the next state of the system is given by the discrete-time dynamics:

$$x_{t+1} = f_l(x_t, u_t). \qquad (2)$$

A general static state-feedback control law can be thought of as a function $\mathbf{u} : \mathcal{X} \to \mathcal{U}$ In many systems, sensors and controller hardware have a finite resolution, and therefore, such a general law cannot be implemented. In this paper, we assume that M is associated with a *controller table* $\mathcal{C}$ which is a partition of the state space $\mathcal{X}$ and the $\mathbf{u} : \mathcal{C} \to \mathcal{U}$ maps each partition in $\mathcal{C}$ to an input. Essentially $\mathbf{u}$ is a look-up table, which assign an input for every equivalence class defined by $\mathcal{C}$. For a feedback control policy $\mathbf{u}$ and a system (2), the next state is just a function of the current state. We denote $post_M(x, \mathbf{u}) = f_l(x, \mathbf{u}(x))$ if $x \in P_l$. The subscript M is dropped if it is clear in the context. We denote by $post^t(x, \mathbf{u})$ the state reached from x after the t^{th} step. For a compact set of states $S \subseteq \mathcal{X}$, we define $post(S, \mathbf{u}) = \{x' : \exists x \in S \text{ such that } x' = post(x, \mathbf{u})\}$. The t step post operation $post^t(S, \mathbf{u})$ is defined similarly.

Our synthesis algorithm will be complete for system models upto some imprecision in the model. For a system $M = (\mathcal{X}, \mathcal{U}, loc, \mathcal{I}, \mathcal{F})$ and $\epsilon > 0$, another system M' is an *ϵ-perturbation* of M if it is identical to M except that the set of dynamic functions for M' is $\mathcal{F}' = \{f'_l\}_{l \in loc}$, such that for each $l \in loc$, $d(f_l, f'_l) \leqslant \epsilon$. We denote by $\mathcal{B}_\epsilon(M)$ the set of all models that are ϵ-perturbations of M.

Reach-Avoid Control Problem (RAC): A *reach-avoid control*(RAC) problem is parameterized by the system model M, the controller table $\mathcal{C}$, and three sets of states *Init*, *Safe*, *Goal* $\subseteq \mathcal{X}$ called the *initial, safe* and *goal* states. We will assume that these sets have some finite representation (for example, hyperrectangles, polytopes). We define what it means to solve a RAC problem with a feedback control policy $\mathbf{u}$.

Definition 1. *A* solution *to a RAC is a feedback control policy* $\mathbf{u} : \mathcal{C} \to \mathcal{U}$ *such that for any initial state* $x \in$ *Init, the states visited by the system satisfies the condition:*

- (Safety) *for all* $t \in \mathbb{N}$, $x_t \in$ *Safe and*
- (Progress) *there exists* $T \in \mathbb{N}$ *such that* $x_T \in$ *Goal.*

Throughout the paper a RAC is uniquely specified by a model M as the rest parameters are fixed.

IV. CONSTRAINT-BASED SYNTHESIS

A major barrier in encoding RAC as an SMT problem is that the safety and progress requirements are over unbounded time. Moreover, these requirements are stated in terms of the future reachable states of the system and computing that in and on itself is a hard problem. Instead of working with unbounded time reach sets, we address this problem by encoding a set of rules that inductively prove safety and progress of the control system.

A. Inductive Synthesis Rules

In addition to searching for the feedback control law $\mathbf{u} : \mathcal{C} \to \mathcal{U}$, the SMT problem will encode the search for (a) an inductive invariant $Inv \subseteq \mathcal{X}$ that proves safety with $\mathbf{u}$, and (b) a ranking function $\mathcal{V}$ that proves progress with $\mathbf{u}$.

In order to constrain the search, we will fix a *template* for the ranking function. For this paper, we will use the template $\mathcal{C} \to \mathbb{N}$, that is, any function that is piecewise constant on the partition of the state space $\mathcal{C}$. This choice has an easy interpretation: each entry in the controller table gives the rank of the controller along with the feedback law. Let $\mathcal{V}$ denote the countable set of all such functions. Each ranking function $V \in \mathcal{V}$ maps every state in $\mathcal{X}$ to a natural number. For any $C \in \mathcal{C}$, $V(C)$ is the natural number that all the states $x \in C$ map to. Now we are ready to present the basic rules encoding inductive synthesis of RAC:

Find $\mathbf{u} : \mathcal{C} \to \mathcal{U}$, $V \in \mathcal{V}$, $Inv \subseteq \mathcal{X}$ such that:

R1: $Init \subseteq Inv$

R2: $post(Inv, \mathbf{u}) \subseteq Inv$

R3: $Inv \subseteq Safe$

R4: $C \subseteq Goal \iff V(C) = 0$

R5: $C \subseteq Inv \wedge post(C, \mathbf{u}) \cap C' \neq \varnothing$
$\Rightarrow V(C) \geqslant V(C')$

R6: $C \subseteq Inv \backslash Goal \wedge post^k(C, \mathbf{u}) \cap C' \neq \varnothing$
$\implies V(C) > V(C')$.

Fig. 1: Basic rules $\Pi(M, \mathcal{C}, \mathcal{V})$ for synthesis for RAC.

Rules R1-R3 imply that Inv is a fixed-point of $post$ with control $\mathbf{u}$ that contains $Init$ and is contained in $Safe$, and therefore, is adequate for proving safety. Rule R4 states the the rank of any region C vanishes iff it is in $Goal$. Rule R5 encodes the (Lyapunov-like) property that the rank of any region C is nonincreasing along trajectories. Finally, rule R6 states that for any non-$Goal$ region C, the rank decreases with $\mathbf{u}$ within k steps, where k is an induction parameter of this encoding. For RAC specified by model M with *controller table* $\mathcal{C}$ and a template $\mathcal{V}$, we denote the SMT problem (Figure 1) as $\Pi(M, \mathcal{C}, \mathcal{V})$ For some control $\mathbf{u}$, ranking function $V \in \mathcal{V}$ and $Inv \subseteq \mathcal{X}$, we write $\mathbf{u}, V, Inv \models \Pi(M, \mathcal{C}, \mathcal{V})$[1] if the Rules R1-R6 are satisfied. The proof of the following theorem can be found in the online full version of the paper [16].

Theorem 2 (Soundness)**.** *If* $\mathbf{u}, V, Inv \models \Pi(M, \mathcal{C}, \mathcal{V})$*, then* $\mathbf{u}$ *solves* RAC *specified by* M.

a) Robustness Modulo Templates: In Section III we defined perturbations of system models, here we lift the definition to the corresponding synthesis rules: $\Pi(M', \mathcal{C}', \mathcal{V}')$ is an *ϵ-perturbation* of $\Pi(M, \mathcal{C}, \mathcal{V})$ if (i) the *controller table* and the ranking templates are identical $\mathcal{C} = \mathcal{C}', \mathcal{V} = \mathcal{V}'$, and (ii) the model $M' \in \mathcal{B}_\epsilon(M)$ is an ϵ-perturbation of M.

Definition 3. *For a controller table $\mathcal{C}$ and a template of ranking functions $\mathcal{V}$, a RAC specified by M is* robust *modulo $(\mathcal{C}, \mathcal{V})$ if there exists $\epsilon > 0$ such that either of the following holds:*

(i) there exists a control $\mathbf{u}$ and a ranking function $V \in \mathcal{V}$ such that for any $M' \in \mathcal{B}_\epsilon(M)$, $\mathbf{u}, V, Inv \models \Pi(M', \mathcal{C}, \mathcal{V})$ with some $Inv \subseteq \mathcal{X}$, or

(ii) for none of $M' \in \mathcal{B}_\epsilon(M)$, the synthesis problem $\Pi(M', \mathcal{C}, \mathcal{V})$ is satisfiable.

In Theorem 7 we will show that our synthesis algorithm is also relatively complete with respect to this notion of robustness.

B. Weakened and Strengthened Rules

The main challenge in solving $\Pi(M, \mathcal{C}, \mathcal{V})$ is the $post$ operator in Rules R2, R5, and R6. We need a reasonable representation of $post$ for this computation to be effective. In this work, we use a finite partition $\mathcal{P}$ of the state space (which preserves $\mathcal{C}, Init, Safe, Goal$) for computing the $post$. This choices is somewhat independent of the rest of the methodology and any other template (for example, linear functions, support functions, zonotopes) could be used instead of the fixed partitions.

The key idea to solve it is to create *over* and *under* approximations of the $post$ operator with respect to the representation of choice—in this case representation using the fixed partition $\mathcal{P}$. These operators are then used to create weakened and strengthened versions of the basic inductive rules that can be effectively solved as SMT problems.

We define an over-approximation ($\mathcal{P}$-$\overline{post}$) and an under-approximation ($\mathcal{P}$-$\underline{post}$) of the $post$ operator with respect to a partition $\mathcal{P}$ as follows: for any compact $S \subseteq \mathcal{X}$,

$$\mathcal{P}\text{-}\overline{post}(S, \mathbf{u}) = \bigcup_{P \in \mathcal{P} \wedge P \cap post(S, \mathbf{u}) \neq \varnothing} P, \tag{3}$$

$$\mathcal{P}\text{-}\underline{post}(S, \mathbf{u}) = \bigcup_{P \in \mathcal{P} \wedge P \subseteq post(S, \mathbf{u})} P. \tag{4}$$

Roughly, the over-approximation $\mathcal{P}\text{-}\overline{post}(S, \mathbf{u})$ computes the minimum superset of S which is preserved by $\mathcal{P}$ and the under-approximation $\mathcal{P}\text{-}\underline{post}(S, \mathbf{u})$ computes the maximum subset of S which is preserved by $\mathcal{P}$. We define $\mathcal{P}\text{-}\overline{post}^t(S, \mathbf{u})$ and $\mathcal{P}\text{-}\underline{post}^t(S, \mathbf{u})$ as the t step over and under-approximations in the usual way.

Proposition 4. *For any measurable $S \subseteq \mathcal{X}$, a post operator and any partition $\mathcal{P}$, the following properties hold:*

(i) $\mathcal{P}\text{-}\underline{post}(S, \mathbf{u}) \subseteq post(S, \mathbf{u}) \subseteq \mathcal{P}\text{-}\overline{post}(S, \mathbf{u})$,

(ii) If $\mathcal{P}'$ subsumes $\mathcal{P}$, then $\mathcal{P}'\text{-}\underline{post}(S, \mathbf{u}) \supseteq \mathcal{P}\text{-}\underline{post}(S, \mathbf{u})$ *and* $\mathcal{P}'\text{-}\overline{post}(S, \mathbf{u}) \subseteq \mathcal{P}\text{-}\overline{post}(S, \mathbf{u})$*, and*

(iii) For any $\epsilon > 0$, $\exists\ \delta > 0$ such that for any $\mathcal{P}$ with resolution less than δ, $d(\mathcal{P}\text{-}\underline{post}(S, \mathbf{u}), \mathcal{P}\text{-}\overline{post}(S, \mathbf{u})) < \epsilon$.

Instead of searching for an exact inductive invariant Inv, the weakened and strengthened versions of the synthesis rules presented below try to find under ($Must$)

[1]For the sake of clarity, we supress the dependence on k.

and over-approximations (May) of the invariant using the $\mathcal{P}\text{-}\overline{post}(S,\mathbf{u})$ and $\mathcal{P}\text{-}\underline{post}(S,\mathbf{u})$ operators. Lemma 5 and 6 establish the soundness and completeness of the weakening and strengthening rules. The proofs of the two lemmas can be found in the full version [16].

Find $\mathbf{u} : \mathcal{C} \to \mathcal{U}$, $V \in \mathcal{V}$, $Must \subseteq \mathcal{X}$ such that

W1: $Init \subseteq Must$

W2: $\mathcal{P}\text{-}\underline{post}(Must,\mathbf{u}) \subseteq Must$

W3: $Must \subseteq Safe$

W4: $C \subseteq Goal \iff V(C) = 0$

W5: $C \subseteq Must \wedge C' \subseteq \mathcal{P}\text{-}\underline{post}(C,\mathbf{u}) \implies V(C) \geqslant V(C')$

W6: $C \subseteq Must \backslash Goal \wedge C' \subseteq \mathcal{P}\text{-}\underline{post}^k(C,\mathbf{u}) \implies V(C) > V(C')$

Fig. 2: Weakened rules $\Pi^w_{\mathcal{P}}(M,\mathcal{C},\mathcal{V})$ for synthesis.

Find $\mathbf{u} : \mathcal{C} \to \mathcal{U}$, $V \in \mathcal{V}$, $May \subseteq \mathcal{X}$ s.t.:

S1: $Init \subseteq May$

S2: $\mathcal{P}\text{-}\overline{post}(May,\mathbf{u}) \subseteq May$

S3: $May \subseteq Safe$

S4: $C \subseteq Goal \iff V(C) = 0$

S5: $C \subseteq May \wedge C' \subseteq \mathcal{P}\text{-}\overline{post}(C,\mathbf{u}) \implies V(C) \geqslant V(C')$

S6: $C \subseteq May \backslash Goal \wedge C' \subseteq \mathcal{P}\text{-}\overline{post}^k(C,\mathbf{u}) \implies V(C) > V(C')$

Fig. 3: Strengthened rules $\Pi^s_{\mathcal{P}}(M,\mathcal{C},\mathcal{V})$ for synthesis.

Lemma 5. *For any $\mathcal{P}$, the following hold:*

(i) if $\mathbf{u}, V, Inv \models \Pi(M,\mathcal{C},\mathcal{V})$, then there exist $Must \subseteq \mathcal{X}$ such that $\mathbf{u}, V, Must \models \Pi^w_{\mathcal{P}}(M,\mathcal{C},\mathcal{V})$; and

(ii) if $\mathbf{u}, V, May \models \Pi^s_{\mathcal{P}}(M,\mathcal{C},\mathcal{V})$, then exists $Inv \subseteq \mathcal{X}$ such that $\mathbf{u}, V, Inv \models \Pi(M,\mathcal{C},\mathcal{V})$.

The above lemma states that the weakening and strengthening of the synthesis rules are sound. With the additional robustness condition, we can show that either the former is unsatisfiable or the latter is satisfiable.

Lemma 6. *If a RAC specified by M is robust modulo $\mathcal{C}, \mathcal{V}$, then there exists a sufficiently fine partition $\mathcal{P}$ such that either (i) $\Pi^w_{\mathcal{P}}(M,\mathcal{C},\mathcal{V})$ is unsatisfiable or (ii) $\Pi^s_{\mathcal{P}}(M,\mathcal{C},\mathcal{V})$ is satisfiable.*

V. SMT-based Synthesis Algorithm

We introduce an algorithm for controller synthesis for RAC using the strengthened and weakened inductive SMT encodings of the previous section. The algorithm takes as input the model M, the controller table/partition $\mathcal{C}$, the template for the ranking function $\mathcal{V}$ and the three sets $Init$, $Safe$ and $Goal$ that define RAC problem. It iteratively refines the partition $\mathcal{P}$ for representing invariants and makes subroutine calls to the SMT solver with the strengthened and weakened rules until it either finds a controller law $\mathbf{u}$ or outputs $\perp$. Specifically, in each iteration, (a) if the strengthened problem $\Pi^s_{\mathcal{P}}$ is satisfiable then it returns the satisfying $\mathbf{u}$. (b) if the weakened problem $\Pi^w_{\mathcal{P}}$ is unsatisfiable then it returns $\perp$. Otherwise, (c) it refines the partition $\mathcal{P}$ (using the $Must$ set computed from $\Pi^w_{\mathcal{P}}$). The $Refine(\mathcal{P}, Must)$ function creates a finer partition of $\mathcal{P}$. For the completeness result, we require that for any $\mathcal{P}$, by iteratively applying $Refine$, the resolution of the resulting partition can be made arbitrarily fine. In Section V-B, we discuss several heuristics for refinement that potentially improve the performance of the algorithm.

Algorithm 1: SMT-based Synthesis Algorithm

```
input: $M, \mathcal{C}, \mathcal{V}, Init, Safe, Goal$;
$\mathcal{P} \leftarrow initPartition$;
while True do
    $(val_s, \mathbf{u}) \leftarrow Solve(\Pi^s_{\mathcal{P}}(M,\mathcal{C},\mathcal{V}))$;
    $(val_w, Must) \leftarrow Solve(\Pi^w_{\mathcal{P}}(M,\mathcal{C},\mathcal{V}))$;
    if $val_s = SAT$ then
        return $\mathbf{u}$
    else if $val_w = UNSAT$ then
        return $\perp$
    else
        $\mathcal{P} \leftarrow Refine(\mathcal{P}, Must)$;
    end
end
```

A. Soundness and Relative Completeness

We will next sketch the arguments for the correctness of the algorithm. Soundness of the algorithm implies that whenever it outputs $\mathbf{u}$, (i) that $\mathbf{u}$ is a control law that solves the RAC problem, (ii) the May set obtained from solving $\Pi^s_{\mathcal{P}}$ in the final iteration is an inductive proof certificates for safety with $\mathbf{u}$, and (iii) the V is a k-step inductive proof certificate for progress with $\mathbf{u}$. And, whenever the algorithm outputs $\perp$ then there does not exists a controller $\mathbf{u}$, a ranking function $V \in \mathcal{V}$ and an invariant $Inv \subseteq$ such that the above (i)-(iii) holds.

In addition, we show that the algorithm is relative complete. That is, if RAC is robust modulo $\mathcal{C}, \mathcal{V}$, then the algorithm terminates with one of the above answers.

Theorem 7. *The algorithm is sound and relatively complete.*

Proof. Soundness. If the algorithm terminates and return $\mathbf{u}$, then for some partition $\mathcal{P}$, the SMT solver returns a satisfying solution $\mathbf{u}$ with $\Pi^s_{\mathcal{P}}(M)$. From Lemma 5, $\mathbf{u}$ solves the RAC. Otherwise if the algorithm terminates and returns $\perp$, then for some partition $\mathcal{P}$, the SMT solver on $\Pi^w_{\mathcal{P}}(M)$ returns UNSAT. From Lemma 5, there is no control that solves the RAC problem modulo $\mathcal{V}$.

Relative Completeness. Since $\Pi(M)$ is a robust RAC modulo $\mathcal{V}$, from Lemma 6, we know that for a sufficiently fine partition, either $\Pi^w_{\mathcal{P}}(M)$ is unsatisfiable or $\Pi^s_{\mathcal{P}}(M)$ is are satisfiable. Thus the while-loop will terminate as the algorithm creates fine enough partitions.

□

B. Guided Refinement

There are different ways in which the refinement of the partition $\mathcal{P}$ can be implemented without compromising the soundness and the relative completeness guarantees. The naive strategy of subdividing every equivalence class in $\mathcal{P}$, increases the size of the SMT problems quickly. As our algorithm solves both the weakened and strengthened versions of the problem simultaneously, we can marshall extra information in performing refinement. For example, when the weakened rules return a possible control $\mathbf{u}$ along with its proof V, Must, even though this controller $\mathbf{u}$ cannot be proven (to be safe and progress making) with the strengthened rules, it can provide useful information for guiding the refinement.

Definition 8. *For a partition $\mathcal{P}$ and a set S that is preserved by $\mathcal{P}$, $\mathcal{P}'$ is a S-*guided refinement *of $\mathcal{P}$ if $\mathcal{P}'$ is derived by refining the cells of $\mathcal{P}$ that are in S.*

One key observation is that, a $\mathcal{X}\backslash \mathit{Must}$-guided refinement helps in generating safety proofs (S3 and W3), while a Must-guided refinement can improve the precesion of progress proofs (S5-S6 and W5-W6). The following proposition formalizes part of this intuition and states that for given a controller $\mathbf{u}$, refining the cells in Must does not improve the precision of the fixed-point $\mathit{Must}, \mathit{May}$ computed by rules S1-S2 and W1-W2.

Proposition 9. *For any control $\mathbf{u}$, any set Init and any partitions $\mathcal{P}$, let $\mathit{Must}, \mathit{May}$ be the fixed point of operator $\mathcal{P}$-$\underline{\mathit{post}}(\cdot, \mathbf{u})$ and $\mathcal{P}$-$\overline{\mathit{post}}(\cdot, \mathbf{u})$ containing Init. Let $\mathcal{P}'$ be a Must-guided refinement of $\mathcal{P}'$ and $\mathit{Must}', \mathit{May}'$ be the fixed point of $\mathcal{P}'$-$\underline{\mathit{post}}(\cdot, \mathbf{u})$ and $\mathcal{P}'$-$\overline{\mathit{post}}(\cdot, \mathbf{u})$ containing Init. Then, $\mathit{Must} = \mathit{Must}'$ and $\mathit{May} = \mathit{May}'$.*

By above proposition, a Must-guided refinement provides no help in generating better safety proofs. However, from Proposition 4, a finer partition $\mathcal{P}$ increase the precision of $\mathcal{P}$-$\underline{\mathit{post}}(C, \mathbf{u})$ and $\mathcal{P}$-$\overline{\mathit{post}}(C, \mathbf{u})$. Since the rules S5-S6 and W5-W6 involve computing $\mathcal{P}$-$\underline{\mathit{post}}(C, \mathbf{u})$ and $\mathcal{P}$-$\overline{\mathit{post}}(C, \mathbf{u})$ for cells in May and Must respectively, a Must-guided refinement possibly increases the precision of these rules. Based on the above observations, we can adopt to the following heuristics for refinement: If the Must set is close to the unsafe set, perform $\mathcal{X}\backslash \mathit{Must}$-guided refinement, otherwise perform Must-guided refinement.

VI. Prototype Implementation and Experiments

We implemented the synthesis algorithm in Python using the the CVC4 SMT solver [2]. In this section, we briefly report preliminary results on applying it to a simple class of navigation problems. With this implementation, we were able to automatically synthesize correct controls (and their inductive proofs) for some configurations and proved impossibility for others.

Vehicle Navigation Problem: We consider a reach-avoid problem for a vehicle that follows piecewise linear approximation of Dubin's dynamics. The system model has 4 state variables $[x, y, v, \theta]^T$: position, velocity and heading angle of the vehicle. It has input variables $[\alpha, \beta]^T$: the acceleration and the turning rate. From the continuous Dubin's vehicle model: $\dot{x} = v\cos\theta$, $\dot{y} = v\sin\theta$, $\dot{v} = \alpha$, $\dot{\theta} = v\beta$. we construct a switched linear model by partitioning the domain of θ and v into 24 locations, and for each location we compute an approximate linear dynamics. The result is a switched-linear model:

$$x^+ = x + av + b,\ y^+ = y + cv + d,\ v^+ = v + \alpha,\ \theta^+ = \theta + e\beta, \tag{5}$$

where a, b, c, d, e have different values in different locations. The piecewise linearized model preserves some properties of the original system. For example, the linearized model cannot turn in place: if the velocity is close to 0, the heading θ cannot change. Moreover, the velocity is non-negative, which further restricts its maneuverability. These properties give rise to interesting RAC problem instances where the system has no satisfying control law.

We allow finitely many discrete input values and compute $\mathcal{P}$-$\underline{\mathit{post}}(C, \mathbf{u})$ and $\mathcal{P}$-$\overline{\mathit{post}}(C, \mathbf{u})$ offline as follows: For a given partition $\mathcal{P}$, and a cell $C \in \mathcal{P}$, we first identify a set of cells $\mathcal{N}(C)$ such that $C' \in \mathcal{N}(C)$ can visit some state in C in one step. Then, for each possible input $\mathbf{u}$, we compute the one step reach set of $\mathit{post}(\mathcal{N}(C), \mathbf{u})$ with the help from reachability tools such as [12], [14]. Thus we just need to identify the input combinations such that C is covered by or intersected with $\mathit{post}(\mathcal{N}(C), \mathbf{u})$.

Experimental Results: We performed several experiments for the above class of problems using our prototype implementation. We search for a control policy as a look-up table, specified by a *controller table*. We utilize a *controller table* $\mathcal{C}$ with 768 cells in total. In Figure 5 and 4, the grids illustrate the projection of controller table to x, y coordinates.

We create a partition $\mathcal{P}$ by further partitioning each cell in $\mathcal{C}$ into 4 pieces, with which we construct both the weakened and the strengthened rules. For some cases, we proved the impossibility of synthesis. We visualize such a case in Figure 4. While for other cases, we successfully synthesized a control policy. An example is illustrated in Figure 5. The satisfying control policy is synthesized with an inductive proof, namely the May set and the ranking function $\mathcal{V}$.

VII. Conclusion

In this work, we studied the controller synthesis problem of discrete-time systems with possibly unbounded time safety and progress specifications. Leveraging the growing strength of modern SMT tools, we propose an algorithm that finds controllers as well as inductive proofs of their correctness. Specifically, the algorithm creates a weaker and a stronger

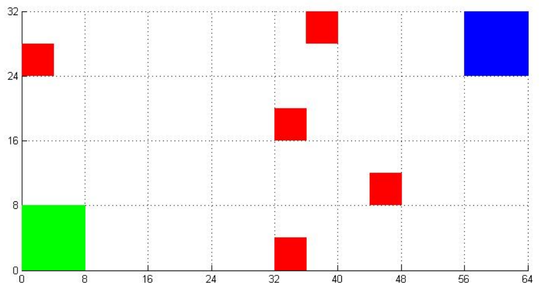

Fig. 4: A RAC instance that is impossible to solve. The grid illustrates the controller table, the green block at the bottom left corner is $Init$, the blue rectangle at the top right is $Goal$, the smaller red blocks are unsafe.

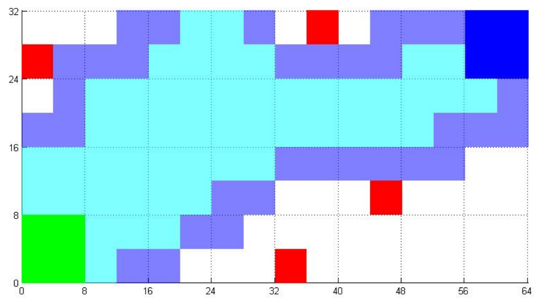

Fig. 5: A RAC instance that has a satisfying control law. The lighter connected region is the $Must$ set and the darker region together with the lighter region is the May set.

version of the synthesis problem and encodes them as SMT problems. By solving the controller synthesis problems for these two bounding systems automatically with SMT solvers, we can solve the synthesis problem for the original system. We prove that this algorithm is sound and relatively complete and show that the solution given by the strengthened system provide a guidance for refining the bounding system. Our experimental results based on a prototype implementation suggest that this can be a promising direction of investigation for controller synthesis research. Since the core problem of computing over-approximations of $post$ are decoupled from synthesis in this formulation, one future direction of research that this work opens up is to extend this framework to nonlinear system models.

References

[1] E. Aydin Gol, X. Ding, M. Lazar, and C. Belta. Finite Bisimulations for Switched Linear Systems. *IEEE Transactions on Automatic Control*, 59(12):3122–3134, Dec. 2014.

[2] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli. Cvc4. In *Computer aided verification*, pages 171–177. Springer, 2011.

[3] A. Bemporad, F. Borrelli, M. Morari, et al. Model predictive control based on linear programming˜ the explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1974–1985, 2002.

[4] T. Beyene, S. Chaudhuri, C. Popeea, and A. Rybalchenko. A constraint-based approach to solving games on infinite graphs. In *Proceedings of the 41st annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 221–234. ACM, 2014.

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

[6] E. Börger, E. Grädel, and Y. Gurevich. *The classical decision problem*. Perspectives in mathematical logic. Springer, Berlin, Heidelberg, New York, 1997.

[7] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and S. Sastry. Challenges for securing cyber physical systems. In *Workshop on future directions in cyber-physical systems security*, 2009.

[8] A. A. Cárdenas, S. Amin, and S. Sastry. Research challenges for the security of control systems. In *HotSec*, 2008.

[9] F. Chen and G. Rosu. Mop: Reliable software development using abstract aspects. *Technical Report UIUCDCS-R-2006- 2776, Department of Computer Science, University of Illinois at Urbana-Champaign*, Oct. 2006.

[10] L. De Moura and N. Bjørner. Z3: An efficient smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[11] J. Ding, E. Li, H. Huang, and C. J. Tomlin. Reachability-based synthesis of feedback policies for motion planning under bounded disturbances. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2160–2165. IEEE, 2011.

[12] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok. C2e2: A verification tool for stateflow models.

[13] B. Dutertre and L. De Moura. The yices smt solver. *Tool paper at http://yices. csl. sri. com/tool-paper. pdf*, 2(2), 2006.

[14] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Computer Aided Verification*, pages 379–395. Springer, 2011.

[15] S. Gao, S. Kong, and E. M. Clarke. dReal: An SMT Solver for Nonlinear Theories over the Reals. In M. P. Bonacina, editor, *Automated Deduction CADE-24*, number 7898 in Lecture Notes in Computer Science, pages 208–214. Springer Berlin Heidelberg, 2013.

[16] Z. Huang, Y. Wang, S. Mitra, G. E. Dullerud, and S. Chaudhuri. Controller Synthesis with Inductive Proofs for Piecewise Linear Systems: an SMT-based Algorithm. *http://adsabs.harvard.edu/abs/2015arXiv150904623H*, Sept. 2015.

[17] H. Kress-Gazit, G. Fainekos, and G. Pappas. Temporal-Logic-Based Reactive Mission and Motion Planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, Dec. 2009.

[18] J. Liu and N. Ozay. Abstraction, Discretization, and Robustness in Temporal Logic Control of Dynamical Systems. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, HSCC '14, pages 293–302, New York, NY, USA, 2014. ACM.

[19] M. Mazo Jr, A. Davitian, and P. Tabuada. Pessoa: A tool for embedded controller synthesis. In *Computer Aided Verification*, pages 566–569. Springer, 2010.

[20] S. Nedunuri, S. Prabhu, M. Moll, S. Chaudhuri, and L. Kavraki. SMT-based synthesis of integrated task and motion plans from plan outlines. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 655–662, May 2014.

[21] I. Saha, R. Ramaithitima, V. Kumar, G. Pappas, and S. Seshia. Automated composition of motion primitives for multi-robot systems from safe LTL specifications. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 1525–1532, Sept. 2014.

[22] M. Svorenova, J. Kretinsky, M. Chmelik, K. Chatterjee, I. Cerna, and C. Belta. Temporal Logic Control for Stochastic Linear Systems using Abstraction Refinement of Probabilistic Games. *arXiv:1410.5387 [cs]*, Oct. 2014. arXiv: 1410.5387.

[23] V. Turri, A. Carvalho, H. E. Tseng, K. H. Johansson, and F. Borrelli. Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pages 378–383. IEEE, 2013.

[24] T. Wongpiromsarn, U. Topcu, and R. Murray. Receding Horizon Temporal Logic Planning. *IEEE Transactions on Automatic Control*, 57(11):2817–2830, Nov. 2012.

[25] Z. Zhou, R. Takei, H. Huang, and C. J. Tomlin. A general, open-loop formulation for reach-avoid games. In *CDC*, pages 6501–6506, 2012.