

Using Computer Games for Hybrid Systems Controller Synthesis

A. Agung Julius* and Sina Afshari
Department of Electrical, Computer and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY 12180, USA.
Email:{juliu2,afshas}@rpi.edu
* Corresponding author

Abstract—We propose a formal method for feedback controller synthesis using interactive computer programs with graphical interface (in short, computer games). The main theoretical tool used in this method is the concept of trajectory robustness, which is established using the theory of approximate bisimulation. Approximate bisimulation has been used to establish robustness (in ℓ_∞ sense) of execution trajectories of dynamical systems and hybrid systems, resulting in trajectory-based safety verification procedures.

We define control autobisimulation function (CAF), which is the analog of control Lyapunov function for approximate bisimulation. CAF is used to characterize the family of all feedback control laws, called admissible control laws, that result in a close loop system with an autobisimulation function. A computer game can then be used to construct safe and correct execution trajectory for a nominal initial state, and use the trajectory-robustness property to guarantee that the control law is also safe for other initial states in a neighborhood of the nominal initial state. As a result, a safe and correct feedback control law for a compact noncountable set of initial states can be obtained by playing finitely many games.

Keywords: hybrid system, trajectory based, controller synthesis.

I. INTRODUCTION

The issue of safety/reachability is very important in the theory of hybrid systems. The *analysis* part of this issue, i.e. the investigation whether a given hybrid system model with given initial conditions can reach a certain state, or set of states has received a lot of attention from the hybrid systems community. It has also resulted in a lot of practical applications, for example in the safety analysis of air traffic systems [1], design verification for electronic circuits [2], design verification for synthetic biology (e.g. [3]), and model analysis for biochemical processes [4]. The *synthesis* part of the safety/reachability issue deals with the construction of control laws/algorithms for systems with input that result in safe executions. Some of the methods for safety/reachability analysis can be extended for controller synthesis. For example, the optimal control method in [5], [6] and the simulation based method in [7] directly characterize the influence of the control input in the reachability formulation. The predicate abstraction technique for systems with piecewise affine dynamics in polytope sets leads to a control procedure based on the transversality of the vector field on the facets of the polytopes [8], [9]. The technique for discrete-time system presented in [10] utilizes partitioning of the state space by polygonal approximation of the reachable set. For continuous

dynamical systems, the theoretical results presented in [11] discuss some sufficient conditions for the existence of a controlled system trajectory that enters a prescribed Goal set.

The class of safety/reachability analysis methods that is closely related to this paper is the *trajectory-based analysis*. These are methods that aim to assess the safety/reachability based on the execution trajectories of the system, or the simulations thereof. The main conceptual tool that we use in this paper, the *approximate bisimulation*, was developed by Girard and Pappas [12], and has been used for trajectory based analysis of hybrid systems in [13], [14], [15]. The notion of approximate bisimulation has previously been used for controller synthesis for nonlinear dynamical systems [16], [17]. In this case, the notion is used to establish a quantization of the continuous state space, which can result in a countable transition system approximation of the original dynamics.

We introduced the notion of *control autobisimulation function (CAF)* [18], to characterize a class of feedback laws, called the *admissible feedback laws*, that result in closed loop systems that admit an (auto)bisimulation function. Therefore, the control autobisimulation function can be thought of as an analog of control Lyapunov function [19], [20] for autobisimulation. We use the term 'autobisimulation' to emphasize the fact that we are considering approximate bisimulation between a system and itself, which is a special case for the theoretical tool developed in [12] and subsequent publications.

For any given initial condition, we use a computer game to construct an admissible feedback law that results in a "valid" execution trajectory¹. The use of CAF enables us to use trajectory robustness (a la approximate bisimulation) to guarantee formally the validity of the control law for a neighborhood around that initial condition. By repeating this procedure for a finite set of initial conditions, we can cover a compact set of initial conditions. The controller design method presented in this paper therefore consists of two steps. The first step is to characterize the class of admissible feedback laws. The second step is to use a computer game to construct an admissible feedback law for each initial condition, that is guaranteed to result in a valid

¹What "valid" means will be discussed later.

trajectory. We present an example demonstrating that this is achievable by playing only a few games. This approach can thus be regarded as a highly parallelizable and lightweight (no quantization of state space is required) complement to the more formal approaches, such as [16], [17].

Futhermore, the structure of our approach enables the integration of human-based computation, which can further enable 'crowdsourcing' of controller synthesis for hybrid systems. That is, the approach presented in this paper can be further used to enable controller synthesis through (potentially large scale) collaboration of multiple human players. A recent work by Langbort et al that investigated the use of a network of human players in a collaborative computer game [21]. In this case, an online ouija board game is introduced. This is a server-based game with the goal of driving a token across an alphabetical board and spelling as many words as possible in a given time by a team of agents. The highly parallelizable nature of the trajectory-based approach makes it ideal for this setup. Pioneering efforts in the area of exploiting online computer games to construct reliable human centered computation can be found in [22], [23], [24] and the references therein.

II. THEORETICAL FOUNDATION

Consider a dynamical system with input

$$\Sigma_{\text{inp}} : \frac{dx}{dt} = f(x, u), x \in \mathbb{R}^n, u \in \mathcal{U} \subset \mathbb{R}^m. \quad (1)$$

where the function $f(x, u)$ is locally Lipschitz in x and continuous in u . Suppose that there is a given compact set of initial states $\text{Init} \subset \mathbb{R}^n$, where the state is initiated at $t = 0$, i.e. $x(0) \in \text{Init}$. Also, we assume that there is a set of goal states, $\text{Goal} \subset \mathbb{R}^n$, and a set of unsafe states $\text{Unsafe} \subset \mathbb{R}^n$. As usual, a trajectory is deemed unsafe if it enters the unsafe set. Suppose that we are given the following control problem:

Problem 1: Design a feedback control law $u = k(x)$ such that for any initial state $x_0 \in \text{Init}$, the trajectory of the closed loop system enters Goal before time $t = T_{\text{max}}$, and remains safe until it enters Goal .

Hereafter, any trajectory that satisfies the conditions above is called a **valid trajectory**. We will discuss how the notion of trajectory-robustness that can be established using bisimulation function (see e.g. [14]) can also be used in trajectory-based controller synthesis. The key concept in this approach is the *control autobisimulation function (CAF)*.

Definition 1: A continuously differentiable function $\psi : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}_+$ is a **control autobisimulation function** of (1) if for any $x, x' \in \mathbb{R}^n$,

$$\psi(x, x') \geq \|x - x'\|, \quad (2)$$

and there exists a function $k : \mathbb{R}^n \rightarrow \mathcal{U}$ such that

$$\nabla_x \psi(x, x') f(x, k(x)) + \nabla_{x'} \psi(x, x') f(x', k(x')) \leq 0. \quad (3)$$

The control autobisimulation function is an analog of the control Lyapunov function (CLF) [19], for approximate bisimulation [12], [14]. While control Lyapunov function has been used to construct control laws that guarantee stability

(e.g. [20]), we shall use the control autobisimulation function to construct control laws that guarantee trajectory robustness.

The concept of approximate bisimulation was first introduced in the seminal work of Girard and Pappas [12]. It has been used for bounding the divergence of output trajectories of continuous and hybrid systems. For autonomous systems (i.e. systems without inputs/nondeterminism), approximate bisimulation is similar to the notion of contraction metric coined by Lohmiller and Slotine (see e.g. [25]). A minor difference between the two notions lies in the fact that approximate bisimulation allows for the use of pseudometric in the state space because it emphasizes on the distance between the outputs of the systems.

Remark 1: One can compare the control autobisimulation function with control Lyapunov function of the product of the system (1) with itself

$$\frac{d}{dt} \begin{bmatrix} x \\ x' \end{bmatrix} = \begin{bmatrix} f(x, u) \\ f(x', u') \end{bmatrix}.$$

In this case, notice that unlike for CLF, for CAF we cannot set u and u' to be any functions of x and x' . Rather, the inputs u and u' must be the same function of their respective states (x and x'). Therefore, in this aspect, the requirement for CAF is more stringent than that for CLF.

Remark 2: In connection with Lyapunov stability theory, CAF can be thought of as a control Lyapunov function for incremental stability. However, the type of incremental stability associated with CAF is weaker than that of δGAS (incremental global asymptotic stability) in [26], in that CAF does not imply asymptotic stability.

Definition 2: For a given dynamical system with input Σ_{inp} and a control autobisimulation function ψ , the class of all feedback control laws $k(\cdot)$ that satisfy (3) is called the **class of admissible feedback laws**, $\eta(\Sigma_{\text{inp}}, \psi)$.

A consequence of the existence of a CAF as in Definition 1 is the existence of a feedback control law

$$u = k(x), \quad (4)$$

such that the closed loop system obtained from (1) and (4),

$$\frac{dx}{dt} = f(x, k(x)), x \in \mathbb{R}^n, \quad (5)$$

has a trajectory-robustness property in the sense of auto-bisimulation, which is defined as follows.

Notation 1: For a given dynamical system with input Σ_{inp} and a feedback control law $u = k(x)$, the closed loop trajectory with initial condition $x(0) = x_0$ is denoted by $\xi_k(t, x_0)$.

Proposition 1: [14] Given a dynamical system with input (1) and a feedback law $k(\cdot)$ such that (2) and (3) hold. For any two initial states of the closed loop system (5), $x_0 \in \mathbb{R}^n$ and $x'_0 \in \mathbb{R}^n$, we have that

$$\forall t \geq 0, \|\xi_k(t, x_0) - \xi_k(t, x'_0)\| \leq \psi(x_0, x'_0).$$

The controller synthesis paradigm in this paper can be stated as follows.

Notation 2: For any $x \in \mathbb{R}^n$ and $\delta \geq 0$, we denote the set $\{x' \in \mathbb{R}^n \mid \psi(x, x') \leq \delta\}$ as $B_\psi(x, \delta)$.

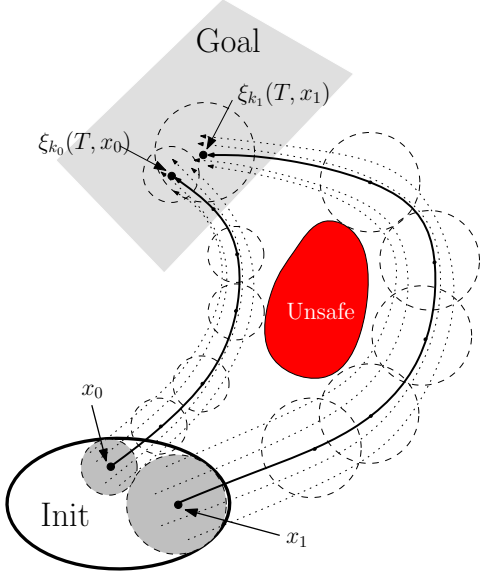


Fig. 1. An illustration for trajectory-based controller synthesis.

We construct feedback controllers from the class of feasible feedback laws. Please refer to Figure 1. Suppose that for a given initial state $x_0 \in \text{Init}$, we can design a feedback law $u = k_0(x)$ that results in a closed loop execution trajectory $\xi_{k_0}(t, x_0)$ satisfying

$$\inf_{0 \leq t \leq T} d_\psi(\xi_{k_0}(t, x_0), \text{Unsafe}) = \delta_0 > 0, \quad (6)$$

$$\xi_{k_0}(T, x_0) \in \text{Goal}, \quad (7)$$

$$d_\psi(\xi_{k_0}(T, x_0), \text{Goal}^C) > \delta_0, \quad (8)$$

where $T < T_{\max}$ and

$$d_\psi(\xi_{k_0}(t, x_0), \text{Unsafe}) := \inf_{x' \in \text{Unsafe}} \psi(\xi_{k_0}(t, x_0), x'),$$

$$d_\psi(\xi_{k_0}(T, x_0), \text{Goal}^C) := \inf_{x' \notin \text{Goal}} \psi(\xi_{k_0}(T, x_0), \xi_{k_0}).$$

Then, we can obtain a neighborhood around x_0 , $B_\psi(x_0, \delta_0)$ consisting of other initial states for which the feedback law $u = k_0(x)$ is guaranteed to result in execution trajectories that are safe and meet the goal state (see e.g. [14]).

We can repeat the procedure for a different initial state, say $x_1 \in \text{Init}$. Suppose that we can then design a feedback law $u = k_1(x)$ that results in a closed loop execution trajectory $\xi_{k_1}(t, x_1)$ that is safe and meets the goal set as shown in Figure 1. As before, we also obtain a neighborhood around x_1 for which the feedback law $u = k_1(x)$ is guaranteed to yield execution trajectories that are safe and meet the goal state. As the result of this process, we now obtain two feedback laws which are valid for two different subsets of Init (not necessarily disjoint). The goal of the controller synthesis procedure is then to cover the entire initial set Init with different control laws as such. Note that this implies that for some initial conditions, there can be more than one control laws that will result in valid trajectories.

III. CONTROLLER SYNTHESIS FOR SYSTEMS WITH AFFINE DYNAMICS

A. Two Stage Controller

Based on the exposition in the previous section, it is clear that to implement the idea of trajectory-based controller synthesis, we need to have a control autobisimulation function (CAF) ψ , and the feedback control laws for each initial condition that we evaluate. Moreover, each the feedback control laws must belong to the class of admissible controller $\eta(\Sigma_{\text{inp}}, \psi)$.

In this paper, for simplicity, we restrict our attention to systems with linear affine dynamics. In the future, we shall explore the construction for systems involving nonlinear dynamics, using ideas from previous work on verification [27]. Systems with linear affine dynamics are systems of the form

$$\Sigma_{\text{lin}} : \frac{dx}{dt} = Ax + f + Bu, \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m, \quad (9)$$

where $A \in \mathbb{R}^{n \times n}$, $f \in \mathbb{R}^n$, and $B \in \mathbb{R}^{n \times m}$. For such systems, we propose to construct CAF using quadratic functions [12], [14], [27]. That is, we assume that

$$\psi(x, x') = [(x - x')^T P (x - x')]^{\frac{1}{2}}, \quad (10)$$

where $P \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix. From Definition 1, it follows that inequality (3) is equivalent to

$$\forall x, x' \in \mathbb{R}^n, (x - x')^T P (A(x - x') + B(k(x) - k(x'))) \leq 0. \quad (11)$$

We propose to construct a feedback law of the form

$$u(t) = k(x) = Kx + v(t), \quad (12)$$

where $K \in \mathbb{R}^{m \times n}$ and $v(t) \in \mathbb{R}^m$ is a time-varying function, both to be determined later. By substituting (12) into (11), we obtain

$$\forall x, x' \in \mathbb{R}^n, (x - x')^T P (A + BK) (x - x') \leq 0. \quad (13)$$

Finding K that satisfies inequality (13) is equivalent to finding K such that $(A + BK)$ is Hurwitz. A well known result in control theory (cf. [28], [29]) states that there exist P and K such that (13) holds if and only if (A, B) is stabilizable. In this case, there are well known methods to synthesize the suitable P and K . For example, by solving the following linear matrix inequality² (LMI) [30]

$$A\tilde{P} + DB^T + \tilde{P}A^T + BD^T \leq 0, \quad \tilde{P} > 0, \quad (14)$$

for $\tilde{P} \in \mathbb{R}^{n \times n}$ and $D \in \mathbb{R}^{m \times n}$. The feedback gain K can be computed from

$$K^T = D\tilde{P}^{-1}. \quad (15)$$

From here, P can be obtained by solving the Lyapunov equation

$$(A + BK)^T P + P(A + BK) \leq 0, \quad P > 0. \quad (16)$$

²We use the fact that A is Hurwitz if and only if A^T is Hurwitz.

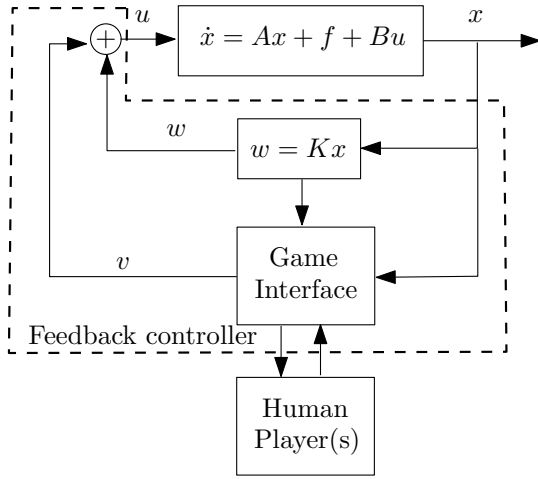


Fig. 2. The block diagram of the proposed controller synthesis method. Human player(s) will use the computer game interface to design $v(t)$ for a finite set of initial conditions.

By applying the feedback control law (12) to Σ_{lin} , we obtain a closed loop system

$$\Sigma_{\text{cl}} : \frac{dx}{dt} = (A + BK)x + f + Bv, \quad x \in \mathbb{R}^n, v \in \mathbb{R}^m. \quad (17)$$

Given that $(A + BK)$ is Hurwitz, we are still free to design $v(t)$. In other words, whatever $v(t)$ is, the control law is admissible (see Definition 2). The remaining task in the controller design is therefore to use $v(t)$ to steer the trajectories of the closed loop system. The goal is to obtain valid trajectories, i.e. steer any given initial state in Init to the goal set in T time units, without entering the unsafe set. We propose to generate $v(t)$, for any given initial condition, using human played computer game, as shown in Figure 2.

B. Bounded Input Set

Consider the case where the input set is bounded, i.e. for all $t \in \mathbb{R}_+$

$$\|u(t)\| \leq M, \quad (18)$$

for some positive bound M . We need to ensure that the feedback law given in (12) satisfies this condition. The fact that

$$\|u(t)\| \leq \bar{\sigma}(K) \|x\| + \|v(t)\|, \quad (19)$$

indicates that minimizing $\bar{\sigma}(K)$ can alleviate the difficulty of designing $v(t)$ that satisfies the control input bound. We can approach this problem by modifying the LMI (14) into the following semidefinite programming problem [31]

$$\begin{aligned} \min \bar{\sigma}(D) \text{ subject to} \\ A\tilde{P} + B D^T + \tilde{P} A^T + D B^T \leq 0, \\ \tilde{P} - I > 0. \end{aligned} \quad (20)$$

It is clear that any (\tilde{P}, D) that is feasible for (14) can be scaled so that it is feasible for (20). However, we also have

$$\bar{\sigma}(K) \leq \bar{\sigma}(D) \bar{\sigma}(\tilde{P}^{-1}) \leq \bar{\sigma}(D), \quad (21)$$

which shows that solving (20) effectively leads to the minimization of an upper bound for $\bar{\sigma}(K)$.

C. Numerical Example

The following example illustrates the controller synthesis procedure. Consider the following problem. Given an affine linear system

$$\begin{aligned} \Sigma : \frac{dx}{dt} &= Ax + f + Bu, \\ A &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0.1 \end{bmatrix}; B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}; f = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}. \end{aligned} \quad (22)$$

We want to design feedback control laws that will bring any state in the initial set

$$\text{Init} = \{x \in \mathbb{R}^4 \mid |x_1| \leq 0.2, |x_3| \leq 0.2, x_2 = 0, x_4 = 1\}, \quad (23)$$

to the goal set

$$\text{Goal} = \left\{x \in \mathbb{R}^4 \mid \sqrt{(x_1 - 2)^2 + x_2^2} \leq 0.1\right\}, \quad (24)$$

without entering the unsafe set

$$\text{Unsafe} = \left\{x \in \mathbb{R}^4 \mid \sqrt{(x_1 - 1)^2 + (x_2 - 1)^2} \leq 0.3\right\}, \quad (25)$$

within $t \in [0, 5]$ seconds and with the input constraint $\|u(t)\| \leq 2$.

We implement a feedback control

$$u(t) = Kx + v(t). \quad (26)$$

We compute the optimal K by solving (20). In this case, using the convex optimization solver `cvx` [32], we calculate the optimal gain as (up to 3 decimals)

$$K_{\text{sdp}} = D_{\text{opt}} \tilde{P}_{\text{opt}}^{-1} = \begin{bmatrix} 0.000 & -0.100 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & -0.100 \end{bmatrix}. \quad (27)$$

The control autobisimulation function

$$\psi(x, x') = [(x - x')^T P (x - x')]^{\frac{1}{2}} \quad (28)$$

needs to satisfy (13). By implementing the optimal feedback gain K_{opt} , we obtain

$$A + BK_{\text{opt}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}, \quad (29)$$

which implies that P can be taken as the identity matrix.

For any given initial condition $x_0 = [x_{1,0}, x_{2,0}, x_{3,0}, x_{4,0}]^T \in \text{Init}$, the next step in this controller design is to calculate the steering input $v(t)$ of the closed loop system

$$\Sigma_{\text{cl}} : \frac{dx}{dt} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0.1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} v, \quad (30)$$

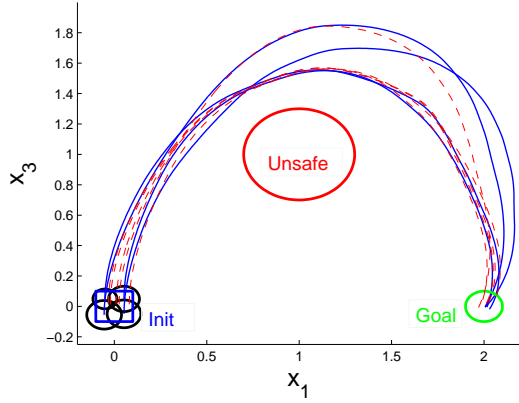


Fig. 3. The result of the controller synthesis procedure. The whole Init set is covered after only 4 games. Solid lines represent the trajectories generated by a human player during the game. Dashed lines represent the trajectories originating at initial conditions that were not picked by the human player. These trajectories are thus obtained by applying the feedback law (26).

that will bring the initial state to the goal state without entering the unsafe set³. Suppose that for the steering input $v(t)$ that we design, we can obtain a robust neighborhood around the initial state x_0 , $B_\psi(x_0, \delta_0)$, for which applying $v(t)$ to the closed loop system will result in valid trajectories. We need to make sure that

$$\forall x'_0 \in B_\psi(x_0, \delta_0), \|K_{\text{opt}}\xi_k(x'_0, t) + v(t)\| \leq 2, \quad (31)$$

where $\xi_k(x'_0, t)$ denote the trajectory of the closed loop system with the feedback control law in (26). From (6) and the fact that the radius of the goal set is 0.1, we can infer that $\delta_0 < 0$. Therefore,

$$\begin{aligned} & \|K_{\text{opt}}\xi_k(x'_0, t) + v(t)\| \\ & \leq \|K_{\text{opt}}\xi_k(x_0, t) + K_{\text{opt}}(\xi_k(x'_0, t) - \xi_k(x_0, t)) + v(t)\|, \\ & \leq \|K_{\text{opt}}\xi_k(x_0, t) + v(t)\| + \bar{\sigma}(K_{\text{opt}}) \|\xi_k(x'_0, t) - \xi_k(x_0, t)\|, \\ & \leq \|K_{\text{opt}}\xi_k(x_0, t) + v(t)\| + 0.1 \cdot 0.1. \end{aligned}$$

Notice that $\xi_k(x_0, t)$ is the trajectory of the chosen initial condition, which is also the trajectory simulated during the gameplay. Therefore, if during the gameplay we restrict $v(t)$ such that

$$\|K_{\text{opt}}\xi_k(x_0, t) + v(t)\| \leq 1.99, \quad (32)$$

the inequality (31) will be automatically satisfied.

The result of the controller synthesis procedure is shown in Figure 3. We manage to cover the entire Init set with only 4 games. At the beginning of each game, the player picks an initial state in Init that has not been previously covered. In Figure 4 we can see the control input signal $u(t)$. We can observe that the constraint on the input magnitude and the timing constraint are both satisfied.

³i.e. $v(t)$ can vary depending on the initial condition.

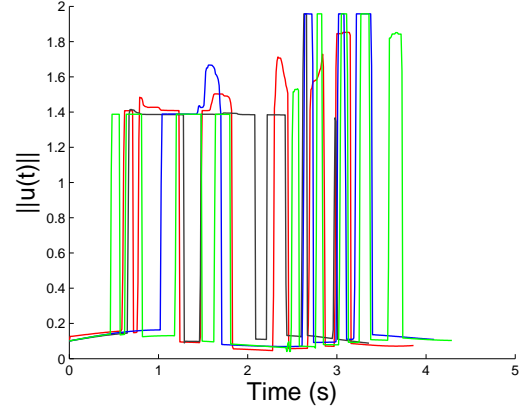


Fig. 4. Control input signal $u(t)$ for the four game trajectories shown in Figure 3. Notice the input magnitude constraint at 2 is satisfied. Also notice that all trajectories terminate before 5 seconds, as required by the problem.

D. Game Interface

We develop a prototype game interface using Matlab. A human player plays the game using a joystick controller. For the interface between Matlab and the joystick controller, we use the code published in [33], and the author's website.

We create a discrete time version of the system dynamics with constant time step δ_s . We use the assumption that within each time interval the user input $v(t)$ is kept constant (zero order hold). For the affine dynamics that we consider in this example, the discretization can be made exact by integrating the closed loop dynamics (30) analytically. We obtain

$$\begin{aligned} x(t + \delta_s) &= e^{A_{\text{cl}}\delta_s} x(t) + A_{\text{cl}}^{-1}(e^{A_{\text{cl}}\delta_s} - I)(Bv + f), \\ A_{\text{cl}} &:= A + BK_{\text{opt}}. \end{aligned}$$

The game polls for user input and refreshes the graphical display every δ_g seconds. For good video quality, we use $\delta_g = \frac{1}{30}$ second. Notice that δ_s and δ_g need not be the same. The ratio $\frac{\delta_s}{\delta_g}$ defines a speed-up factor for the game play, which can be adjusted to change the difficulty of the game. For example, if $\delta_s = 2\delta_g$ then for every second the player spend on the game, the system's dynamics advances by two seconds.

Figure 5 shows a screen capture of a game in play. The circle marked with "Covered Neighborhood" represents a robust neighborhood obtained in previous game (with different initial condition). At the beginning of each game, these neighborhood are displayed to aid the player in choosing an initial state that has not been covered. We also include two other features in the game interface that are meant to aid the player. One feature is the time display. This is helpful if the design problem requires completion of the task within a specific time horizon. The other feature is the projected future trajectory. This trajectory is obtained by assuming that $v(t) = 0$. Subjective experiences by the authors reveal that this cue helps the player in achieving the control task.

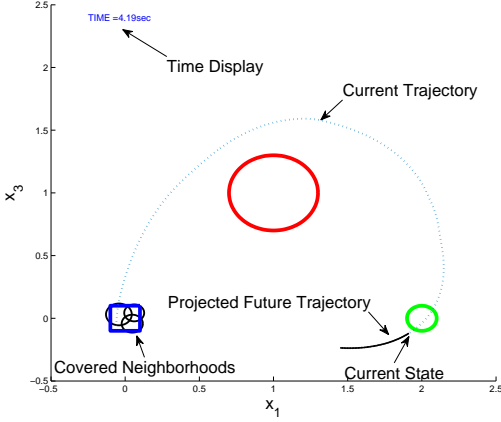


Fig. 5. The graphical interface of the game with explanation.

IV. CONTROLLER SYNTHESIS FOR HYBRID SYSTEMS

The design paradigm presented in the previous section can be also applied to hybrid systems. Consider a standard model of hybrid systems, $\mathcal{H} = (\mathcal{X}, \mathcal{L}, E, Inv, \Sigma)$, where \mathcal{X} is the continuous state space of the system, \mathcal{L} is the finite set of discrete states (locations), E is the set of transitions, $Inv : \mathcal{L} \rightarrow 2^{\mathcal{X}}$ is the invariant set of a location, and Σ is a family of dynamical systems with input that defines the continuous dynamics in each location. That is, for each location $l \in \mathcal{L}$, we define the continuous dynamics as

$$\Sigma(l) : \frac{dx}{dt} = f_l(x, u), x \in \mathcal{X}, u \in \mathcal{U}. \quad (33)$$

A transition $e \in E$ is a 4-tuple (l, l', g, r) , where $l \in \mathcal{L}$ is the origin of the transition, $l' \in \mathcal{L}$ is the target of the transition and that each location, $g \subset \partial Inv(l)$ is the guard of the transition, which is a subset of the boundary of the invariant set of location l , and $r : g \rightarrow Inv(l')$ is the reset map that resets the continuous state at the new location. We assume that the reset map r is continuous, the continuous state space is \mathbb{R}^n , the invariant sets are closed, $f_l(x, u)$ is locally Lipschitz in x and continuous in u for all $l \in \mathcal{L}$, the transitions are deterministic in the sense that the guards of all outgoing transitions from a location are disjoint, and that the system does not deadlock or possess Zeno behavior. In analyzing the safety of the system, we assume that there is a subset $Unsafe \subset \mathcal{X} \times \mathcal{L}$ of unsafe states. A trajectory of the hybrid system corresponds to an unsafe execution if it intersects with the unsafe set.

A. Control Problem Formulation and Hierarchical Control Synthesis

To define the control problem, we define a set of initial state $Init \subset \mathcal{X} \times \mathcal{L}$, in which we assume the hybrid state begins at $t = 0$. We also define a goal set, $Goal \subset \mathcal{X} \times \mathcal{L}$ in which all executions must terminate. As before, the control problem is defined as finding the feedback control strategy that is guaranteed to bring any initial state in $Init$ to the goal set without entering the unsafe set. Without any loss

of generality, we can assume that the set $Init$ is contained in (the invariant set of) one location, called $l_{init} \in \mathcal{L}$. If this is not the case, we can divide the problem into several subproblems, each with an $Init$ set contained in a specific location. Similarly, we can assume the $Goal$ is also entirely contained in one location, called $l_{goal} \in \mathcal{L}$.

We approach this problem with a *hierarchical control* design, which can be described in the following steps:

Step 1: Discrete Synthesis. We compute a discrete trajectory that starts in l_{init} and ends in l_{goal} . By discrete trajectory, we mean an alternating sequence of locations and transitions

$$l_{init} = l_0 \xrightarrow{e_1} l_1 \xrightarrow{e_2} l_2 \xrightarrow{e_3} \dots \xrightarrow{e_N} l_N = l_{goal}. \quad (34)$$

Each transition $e_{i,i \in \{1, \dots, N\}}$ is an element of E , originating in l_{i-1} , and targeting l_i . Such a discrete trajectory is not necessarily unique, but at this step we only need one. The computation of a discrete trajectory like this, albeit formally undecidable, is a standard procedure in formal verification of discrete event systems [34].

Step 2: Continuous Synthesis. In this step, we synthesize the continuous controller for each of the visited locations $(l_{0,1, \dots, N})$ in order to implement the computed discrete trajectory. Basically, in each location l_i , we define an initial set based on how l_i is reached from l_{i-1} . We then formulate the control problem of bringing the continuous state from this initial set to the goal set, which is defined as a set beyond the guard of transition e_i that will bring the state to location l_{i+1} without entering the forbidden set. The forbidden set is defined as the union of $Unsafe$, and the guards of other outgoing transitions from l_i . If we are able to construct a continuous controller that implements the discrete trajectory, then the hybrid control problem is solved. Otherwise, we go back to Step 1, and compute another discrete trajectory.

Remark 3: Similar two-step approach to solve the control problem with application in motion control synthesis for fully actuated robots has been discussed in the literature (see [35] and the references therein). The discrete part of the control goal in [35] is expressed as a temporal logic formula, which is richer than the one presented in this paper. However, we would like to point out that the continuous synthesis presented in this paper can also be applied to implement the continuous part of the controller in [35].

B. Continuous Synthesis of the Hybrid Controller

Given the discrete trajectory (34), we can synthesize the continuous controller in each location as follows.

Notation 3: We denote the set of all outgoing transitions from a location $l \in \mathcal{L}$ as $Out(l)$.

First Location ($l_{init} = l_0$). Define the $Init$ as the initial set (recall that we assume the $Init$ is entirely contained in the invariant set of l_{init}), i.e. $Init_0 := Init$. Define the initial unsafe set as

$$Unsafe_0 := Unsafe|_{l_{init}} \bigcup_{e \in Out(l_{init}) \setminus e_1} Guard(e), \quad (35)$$

where for $l \in \mathcal{L}$,

$$Unsafe|_l := \{x \in Inv(l) \mid (x, l) \in Unsafe\}, \quad (36)$$

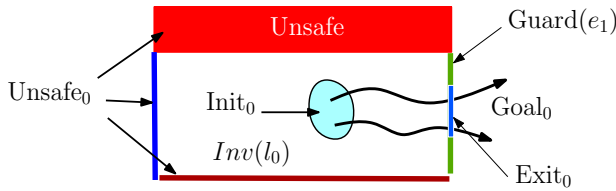


Fig. 6. An illustration for the definitions related to controller synthesis for l_0 . The feedback controller is expected to guide any initial condition in Init_0 to the goal set, beyond the guard of e_1 , while avoiding the unsafe set and other guards.

and $\text{Guard}(e)$ refers to the guard set of transition $e \in E$. Define as the goal set

$$\text{Goal}_0 = \text{Guard}(e_1) \cup \text{Inv}(l_{\text{init}})^C. \quad (37)$$

Essentially, this means that we define the guard set of e_i and beyond the invariant set of l_{init} as our goal. These definitions are illustrated in Figure 6.

We implement the controller synthesis discussed in Section II to develop a feedback control for the dynamical system $\Sigma(l_{\text{init}})$ such that any initial condition in Init_0 is guaranteed to reach Goal_0 without entering Unsafe_0 . If $\Sigma(l_{\text{init}})$ is an affine linear system, we can use the design procedure discusses in Section III-A.

After we can successfully design and implement such a controller, we define the subset of $\text{Guard}(e_1)$ that can be reached by initial state in Init_0 as Exit_0 . In practice, we can use an overapproximation to compute Exit_0 . For example, if $x_i, i \in \{0, 1, \dots, M-1\}$ are M initial conditions that we tested in order to cover Init_0 (see Figure 1), and if we define the continuous trajectories of the closed loop system starting from those initial conditions as $\xi_i, i \in \{0, 1, \dots, M-1\}(t)$, then Exit_0 can be overapproximated by the intersection of $\text{Guard}(e_1)$ with the union of all trajectory tubes around $\xi_i, i \in \{0, 1, \dots, M-1\}(t)$.

Intermediate Locations ($l_i, i \in \{1, 2, \dots, N-1\}$). Define $\text{Init}_i = r_i(\text{Exit}_{i-1})$, where r_i is the reset map of e_i . Define the unsafe set as

$$\text{Unsafe}_i := \text{Unsafe}|_{l_i} \bigcup_{e \in \text{Out}(l_i) \setminus e_{i+1}} \text{Guard}(e), \quad (38)$$

and the goal set as

$$\text{Goal}_i = \text{Guard}(e_{i+1}) \cup \text{Inv}(l_i)^C. \quad (39)$$

As before, the objective of the controller synthesis is to guide any initial condition in Init_i to reach Goal_i without entering Unsafe_i . Once we obtain and implement such a controller, we compute (or overapproximate) Exit_i in the same way as Exit_0 .

Final Location ($l_{\text{goal}} = l_N$). Define $\text{Init}_N = r_N(\text{Exit}_{N-1})$, where r_N is the reset map of e_N . Define the unsafe set as

$$\text{Unsafe}_N := \text{Unsafe}|_{l_{\text{goal}}} \bigcup_{e \in \text{Out}(l_{\text{goal}})} \text{Guard}(e), \quad (40)$$

and the goal set as

$$\text{Goal}_N = \text{Goal}. \quad (41)$$

Design a controller as before. For this location, we do not need to compute the exit set.

C. Nonforcing Guard Conditions

We can generalize the result presented here by including nonforcing guard conditions. That is, instead of assuming that a transition happens when and only when the continuous trajectory hits a guard, we can assume that there is a flexibility in the timing of the transition. This means that the guard set is 'fat', and anytime the continuous state is within this guard set, a controller-triggered transition is possible.

Continuous controller synthesis in this case is analogous to what was discussed in the previous subsection, except that we allow the human player to trigger the transition whenever it is enabled. Such input can be delivered through a joystick button or a mouse click during the game, for example. The notion of exit set is then generalized as the intersection between the robust neighborhood of the continuous state where the transition happens, and the guard set.

V. DISCUSSION

We presented a method for feedback controller synthesis for hybrid and dynamical systems using computer games. The control objective is to achieve valid trajectories from a set of possible initial states. The main tool used in this approach is the notion of trajectory robustness, which is established using the theory of approximate bisimulation. With that, we can generalize trajectories that are generated through a finite game play to cover a compact set of initial states.

In the future, we plan to explore the applicability of this approach in synthesizing controllers through multiplayer cooperation. The highly parallelizable nature of the approach makes it ideal for the setup. This can be done, for example by assigning different initial states to different players. As such, the only interaction between players happens during the initial state selection. We are inspired by the pioneering effort of von Ahn and others [22], [23], [24] in using online computer games to realize reliable human based computation schemes.

An apparent limitation of our approach is the dimension of the state space and control degree of freedom. After all, computer graphic interface is practically limited to two dimensional display⁴. However, we foresee that this limitation can be overcome in two ways. First, we can apply model abstraction in the sense of approximate bisimulation (c.f. [36], [37], [38]). This approach will allow us to abstract a higher dimensional model with a lower dimensional one. Second, we observe that many computer game products that are currently on the market can actually involve human players in controlling complex systems, such as car simulator, airplane simulator, sports simulator, etc. This means the human cognitive capacity is able to grasp high dimensional data and interaction through two dimensional display. We therefore plan to explore the human cognitive aspect of this

⁴One can argue that three dimensional display is also possible, but we consider this to be (currently) not practical.

work. Along the same line, we are interested in using this approach in designing (sub)optimal controllers. In particular, we are interested in finding how to best exploit the game interface in order to maximize the players' performance.

Finally, we would like to remark that the result presented in this paper can be generalized by replacing the game interface in Figure 2 with other means of obtaining valid trajectories for given initial conditions. These include other heuristics based methods, such as fuzzy control [39], or expert system based methods (cf. [40]) that allow for integration of human operators' experience into the control strategy. The advantage offered by the theory of trajectory-based analysis is that we can formally guarantee the safety and correctness of the resulting controllers.

Acknowledgement: This work is partially supported by an NSF CAREER grant (#0953976). The author would like to thank Jeff Ban for the stimulating discussions that lead to this paper, and the anonymous reviewers for their constructive remarks.

REFERENCES

- [1] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, "A probabilistic approach to aircraft conflict detection," *IEEE Trans. on Intelligent Transportation Systems*, vol. 1(4), pp. 199–220, 2000.
- [2] T. Dang, A. Donze, and O. Maler, "Verification of analog and mixed-signal circuits using hybrid system techniques," in *Formal Methods in Computer-Aided Design*, vol. 3312 of *LNCS*, pp. 21–36, Springer, 2004.
- [3] G. Batt, B. Yordanov, R. Weiss, and C. Belta, "Robustness analysis and tuning of synthetic gene networks," *Bioinformatics*, vol. 23, no. 18, pp. 2415–2422, 2007.
- [4] D. Riley, X. Koutsoukos, and K. Riley, "Modeling and analysis of the sugar cataract development process using stochastic hybrid systems," *IET Systems Biology*, vol. 3(3), pp. 137–154, 2009.
- [5] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Trans. Automatic Control*, vol. 50, pp. 947–957, 2005.
- [6] A. B. Kurzhanski, I. M. Mitchell, and P. Varaiya, "Optimization techniques for state-constrained control and obstacle problems," *Journal of Optimization Theory and Applications*, vol. 128, no. 3, pp. 499–521, 2006.
- [7] J. Kapinski, B. H. Krogh, O. Maler, and O. Stursberg, "On systematic simulation of open continuous systems," in *Hybrid Systems: Computation and Control*, vol. 2623 of *LNCS*, pp. 283–297, Springer, 2003.
- [8] C. Belta and L. Habets, "Controlling a class of nonlinear systems on rectangles," *IEEE Trans. Automatic Control*, vol. 51, no. 11, pp. 1749–1759, 2006.
- [9] L. Habets, P. J. Collins, and J. H. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," *IEEE Trans. Automatic Control*, vol. 51, no. 6, pp. 938–948, 2006.
- [10] G. Reissig, "Computation of discrete abstractions of arbitrary memory span for nonlinear sampled systems," in *Hybrid Systems: Computation and Control*, vol. 5469 of *LNCS*, pp. 306–320, Springer, 2009.
- [11] F. Clarke and P. R. Wolenski, "Control of systems to sets and their interiors," *Journal of Optimization Theory and Applications*, vol. 88, pp. 3–23, 1994.
- [12] A. Girard and G. J. Pappas, "Approximation metrics for discrete and continuous systems," *IEEE Trans. Automatic Control*, vol. 52, no. 5, pp. 782–798, 2007.
- [13] A. Girard and G. J. Pappas, "Verification using simulation," in *Hybrid Systems: Computation and Control*, vol. 3927 of *LNCS*, pp. 272–286, Springer Verlag, 2006.
- [14] A. A. Julius, G. Fainekos, M. Anand, I. Lee, and G. J. Pappas, "Robust test generation and coverage for hybrid systems," in *Hybrid Systems: Computation and Control*, vol. 4416 of *LNCS*, pp. 329–342, Springer Verlag, 2007.
- [15] F. Lerda, J. Kapinski, E. M. Clarke, and B. H. Krogh, "Verification of supervisory control software using state proximity and merging," in *Hybrid Systems: Computation and Control*, vol. 4981 of *LNCS*, pp. 344–357, 2008.
- [16] P. Tabuada, "An approximate simulation approach to symbolic control," *IEEE Trans. Automatic Control*, vol. 53, no. 6, pp. 1406–1418, 2008.
- [17] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, vol. 44, no. 10, pp. 2508–2516, 2008.
- [18] A. A. Julius, "Trajectory-based controller design for hybrid systems with affine continuous dynamics," submitted to IEEE CASE 2010, extended manuscript. Available online at <http://www.ecse.rpi.edu/~agung>, 2010.
- [19] Z. Artstein, "Stabilization with relaxed controls," *Nonlinear Analysis*, vol. 15, no. 11, pp. 1163–1170, 1983.
- [20] E. D. Sontag, "A 'universal' construction of Artstein's theorem on nonlinear stabilization," *Systems and Control Letters*, vol. 13, no. 2, pp. 117–123, 1989.
- [21] J. Barral, R. Wilson, and C. Langbort, "The online ouija board : a testbed for multi-party control of dynamical systems," in *Proc. American Control Conference*, (St. Louis, MO.), pp. 872–877, 2009.
- [22] L. von Ahn, M. Blum, N. Hopper, and J. Langford, "Captcha: Using hard AI problems for security," in *Advances in Cryptology*, vol. 2656 of *LNCS*, pp. 294–311, Springer, 2003.
- [23] L. von Ahn and L. Dabbish, "Labeling images with a computer game," in *Proc. SIGCHI Conference on Human Factors in Computing Systems*, (New York), pp. 319–326, ACM, 2004.
- [24] L. von Ahn and L. Dabbish, "General techniques for designing games with a purpose," *Communications of the ACM*, pp. 58–67, August 2008.
- [25] W. Lohmiller and J. J. E. Slotine, "On contraction analysis for nonlinear systems," *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.
- [26] D. Angeli, "A Lyapunov approach to incremental stability properties," *IEEE Trans. Automatic Control*, vol. 47, no. 3, pp. 410–421, 2002.
- [27] A. A. Julius and G. J. Pappas, "Trajectory based verification using local finite-time invariance," in *Hybrid Systems: Computation and Control*, vol. 5469 of *LNCS*, pp. 223–236, Springer, 2009.
- [28] W. L. Brogan, *Modern control theory*. New Jersey: Prentice Hall International, 1991.
- [29] B. Friedland, *Control System Design: An Introduction to State-Space Methods*. Dover, 2005.
- [30] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in Systems and Control Theory*. Philadelphia: SIAM, 1994.
- [31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004. Available online at www.stanford.edu/~boyd/cvxbook/.
- [32] S. Boyd and M. C. Grant, "cvx – MATLAB software for disciplined convex programming," 2005. <http://www.stanford.edu/~boyd/cvx/>.
- [33] M. Bodson, "Fun control experiments with Matlab and a joystick," in *Proc. IEEE Conf. Decision and Control*, (Maui, Hawaii, USA.), 2003.
- [34] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. MIT Press, 1999.
- [35] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic mobile robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.
- [36] A. Girard and G. J. Pappas, "Approximate bisimulation for constrained linear systems," in *Proc. of the IEEE Conf. Decision and Control*, (Seville, Spain), 2005.
- [37] A. Girard and G. J. Pappas, "Approximate bisimulations for nonlinear dynamical systems," in *Proc. of the IEEE Conf. Decision and Control*, (Seville, Spain), 2005.
- [38] A. Girard and G. Pappas, "Approximation metrics for discrete and continuous systems," *IEEE Trans. on Automatic Control*, vol. 52, no. 5, pp. 782–798, 2007.
- [39] K. M. Passino and S. Yurkovich, *Fuzzy Control*. Addison-Wesley, 1998.
- [40] B. K. Bose, "Expert system, fuzzy logic, and neural network applications in power electronics and motion control," *Proceedings of the IEEE*, vol. 82, no. 8, pp. 1303 – 1323, 1994.