Formal Synthesis of Stabilizing Controllers for Switched Systems

Pavithra Prabhakar[1][0000−0002−5368−3234] and
Miriam García Soto[2][0000−0003−2936−5719]

[1] Kansas State University, Manhattan, KS
pprabhakar@ksu.edu ***
[2] IMDEA Software Institute &
Universidad Politécnica de Madrid, Spain
miriamgs@ucm.es †

Abstract. In this paper, we describe an abstraction-based method for synthesizing a state-based switching control for stabilizing a family of dynamical systems. Given a set of dynamical systems and a set of polyhedral switching surfaces, the algorithm synthesizes a strategy that assigns to every surface the linear dynamics to switch to at the surface. Our algorithm constructs a finite game graph that consists of the switching surfaces as the existential nodes and the choices of the dynamics as the universal nodes. In addition, the edges capture quantitative information about the evolution of the distance of the state from the equilibrium point along the executions. A switching strategy for the family of dynamical systems is extracted by finding a strategy on the game graph which results in plays having a bounded weight. Such a strategy is obtained by reducing the problem to the strategy synthesis for an energy game, which is a well-studied problem in the literature. We have implemented our algorithm for polyhedral inclusion dynamics and linear dynamics. We illustrate our algorithm on examples from these two classes of systems.

Keywords: Hybrid systems; stability; control synthesis; game theory; switched systems

1 Introduction

One of the fundamental requirements in control design for cyber-physical systems is stability. Stability is a robustness property that captures the notion that small perturbations in the initial state of the system result in only small deviations in the behaviors of the system. This robustness property accounts for certain deviations in the implementation as compared to the design. In this paper, we consider the problem of computing a supervisory controller that switches between a set

*** Pavithra Prabhakar is partially supported by NSF CAREER award no. 1552668.
† Miriam García Soto is partially supported by BES-2013-065076 grant from the Spanish Ministry of Economy and Competitiveness.

of given dynamical systems to obtain a switched system that is stable. Switching strategies are broadly of two kinds, one that specifies the switching times and the other that specifies the switching surfaces [15]. Here, we focus on strategies of the latter kind.

Stability analysis is a classical problem studied in control theory. For a linear dynamical system $\dot{x} = Ax$, stability can be characterized in terms of the eigenvalues of the matrix A, which can be efficiently determined when the elements of A are rational. However, complete characterization of stability is not known even for switched linear systems. For instance, eigenvalue based analysis does not carry over to switched linear systems; one can construct two systems — one stable and the other unstable — by switching between the same two linear dynamical systems [4]. Hence, restrictions on the dynamics and switching law that enforce stability have been investigated. For instance, a system that switches arbitrarily among a set of linear dynamics is stable, if the corresponding matrices are pairwise commutative or symmetric [20, 33].

More generally, Lyapunov functions are used to establish stability. A Lyapunov function is a function from the state space to non-negative reals such that the value of function along any execution decreases. Again, for linear systems, it is well known that there exist quadratic Lyapunov functions that can be computed by solving certain Linear Matrix Inequalities (LMIs). For switched systems, this framework has been extended to common and multiple Lyapunov functions. A common Lyapunov function consists of a single function which serves as a Lyapunov function for all the dynamical systems in the switched system. A multiple Lyapunov function consists of multiple functions, each of which serves as a Lyapunov function for certain dynamical system in the switched system, and in addition, consists of certain constraints that need to be satisfied by these functions at the switching surfaces. In [26, 27], necessary and sufficient conditions for the existence of common quadratic Lyapunov functions for arbitrary switched linear systems have been explored. Necessary and sufficient condition for robust asymptotic stability has been investigated in [3], which provides a semi decidable algorithm for stability analysis. Constraints on the switching, such as, slow switching characterized by average dwell time, that ensure stability have been investigated in [19]. For constrained switching, multiple Lyapunov functions have been studied in [9] including piecewise quadratic Lyapunov functions [13].

This paper focuses on the stabilization problem, namely, the synthesis of a switching logic that renders the system stable. Control Lyapunov functions are the analogue of Lyapunov functions for stabilizability. Quadratic and piecewise quadratic switching stabilization of linear systems has been explored in [32, 28, 22] and necessary and sufficient conditions for stabilization that lead to semi decidable algorithms have been explored in [16]. Both stability analysis and stabilization problem are undecidable for very simple dynamics, and hence, there are no complete algorithmic methods for stabilizable control synthesis. See [17] for a recent survey on stability and stabilization of switched linear systems.

In this paper, we explore an alternate method based on abstractions. Abstractions [12] construct simpler systems for the purpose of design and analysis. More precisely, to analyse the correctness of a system $\mathcal{S}$, a simpler system $\hat{\mathcal{S}}$ is constructed, such that, the correctness of $\hat{\mathcal{S}}$ implies the correctness of $\mathcal{S}$. Hence, the simpler system $\hat{\mathcal{S}}$ is analysed to infer the correctness of $\mathcal{S}$. In general, the abstraction is conservative and hence, the violation of the property by $\hat{\mathcal{S}}$ does not imply the violation of the property by $\mathcal{S}$. Abstractions for safety analysis have been explored in [1, 6, 2, 30]; more recently, abstractions have been explored for stability analysis [24]. Abstractions for controller synthesis follow the same general paradigm. Here, a simpler finite state abstract game is constructed from the continuous/switched system and a strategy for this abstract game is used to extract a controller for the original system [18, 11, 21, 31, 29]. These works consider safety and linear temporal logic objectives.

In this paper, we explore abstraction based controller synthesis for the stabilization problem. Given a set of dynamical systems and polyhedral switching surfaces, we construct a finite weighted (game) graph, that abstracts the switching synthesis problem into a strategy synthesis problem on the game graph. More precisely, if there exists a strategy on the game graph such that all the paths in the graph that conform with the strategy have bounded weights, then a switching control for the family of dynamical systems can be extracted from the strategy. We solve the strategy synthesis problem by reducing it into an energy game problem for which several algorithms have been proposed in the literature [5]. We apply our algorithm to the stabilization problem of a family of polyhedral inclusion dynamics and a family of linear dynamical systems. Here, we do not explore the choice of the polyhedral switching surfaces, and discuss briefly the computational aspects in the graph construction. These will be explored in detail in a future work.

2 Preliminaries

In this section, we present some basic definitions and concepts required in the rest of the paper.

Numbers, functions and sequences. Let $\mathbb{R}$, $\mathbb{Q}$ and $\mathbb{N}$ denote the set of real, rational and natural numbers, respectively. $\mathbb{R}_{\geqslant 0}$ denotes the set of non-negative real numbers and $\mathbb{R}^{\infty}_{\geqslant 0}$ denotes the set $\mathbb{R}_{\geqslant 0} \cup \{+\infty\}$.

Given a function $f : A \to B$, the domain of f, namely, A, is denoted $dom(f)$. The restriction of f to a subset $C \subseteq A$ is denoted as $f \restriction_C$.

Given a set A, the set of finite sequences over A with zero or more elements is denoted as A^* and the set of finite sequences over A with one or more elements is denoted as A^+. The set A^*B denotes the set of finite sequences conformed by a concatenation of a sequence in A^* and an element in B. The last term in a sequence η is denoted as $last(\eta)$. We refer to the cardinality of a sequence η as $|\eta|$.

The space $\mathbb{R}^n$. The interior of a set $A \subseteq \mathbb{R}^n$ is denoted as $\mathring{A}$, and the closure as $\overline{A}$.

A *polyhedral partition* of an n-dimensional real space $X \subseteq \mathbb{R}^n$ is a finite set of closed convex polyhedra $R = \{\Omega_1, \ldots, \Omega_k\}$ such that $X = \cup_{\Omega \in R} \Omega$, $\mathring{\Omega}_i \neq \emptyset$ for every i and $\mathring{\Omega}_i \cap \mathring{\Omega}_j = \emptyset$ for every $i \neq j$. A *valid* set of *facets* $\mathcal{F}$ of X is a set of maximal closed convex subsets of the boundary of the polyhedral sets contained in a polyhedral partition R of X, that is, $\mathcal{F} = \{f \mid f = \Omega \cap \Omega', \Omega, \Omega' \in R, \Omega \neq \Omega', dim(f) = n - 1\}$ for some polyhedral partition R.

3 Switching stabilization problem

In this section, we formalize the switching stabilization problem for a family of differential inclusions. Given a set of differential inclusions, our goal is to synthesize a state based switching logic that results in a stable switched system.

Consider a family of n-dimensional dynamical systems determined by differential inclusions of the form:

$$\dot{x}(t) \in g_p(x(t)),\ p \in \mathcal{P}, \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state at time t, $\mathcal{P}$ is a finite set of operational modes, and for every $p \in \mathcal{P}$, the multivalued map $g_p : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ specifies the dynamics in the mode. We will refer to the above as the system $\mathcal{S} = (\mathcal{P}, \{g_p\}_{p \in \mathcal{P}})$.

A switched system is then obtained by switching between the dynamical systems at certain switching facets. Our definition is slightly different from a standard state dependent strategy, where every state is assigned a dynamics from the family. Our system will follow its current dynamics until it reaches a switching facet upon which it changes its dynamics as dictated by a switching policy.

Definition 1. *Given a system* $\mathcal{S} = (\mathcal{P}, \{g_p\}_{p \in \mathcal{P}})$ *and a finite set of valid facets* $\mathcal{F}$ *in* $\mathbb{R}^n$, *a* switching strategy *is a function* $\alpha : \mathcal{F}^+ \to \mathcal{P}$.

We seek a strategy that could potentially have memory, that is, the mode being chosen could depend not only on the current facet but also the sequence of facets seen until then. However, our algorithm will find memoryless strategies where the choice depends only on the current facet.

Definition 2. *Given a system* $\mathcal{S} = (\mathcal{P}, \{g_p\}_{p \in \mathcal{P}})$ *and a switching strategy* α, *the tuple* $(\mathcal{P}, \{g_p\}_{p \in \mathcal{P}}, \alpha)$ *is a* switched system *and it is denoted by* $\mathcal{S}_\alpha$.

Next, we define partial executions which capture the solution of the switched system starting from a switching facet before it reaches the next switching facet.

Definition 3. *Given a system* $\mathcal{S} = (\mathcal{P}, \{g_p\}_{p \in \mathcal{P}})$, *a valid set of facets* $\mathcal{F}$ *and a* $p \in \mathcal{P}$, *a* p-partial execution *is a function* $x : [T_1, T_2) \to \mathbb{R}^n$, *where* $T_1 \in \mathbb{R}_{\geqslant 0}$ *and* $T_2 \in \mathbb{R}^{\infty}_{\geqslant 0}$, *such that there exists a non-negative real* $T \in [T_1, T_2)$ *and a facet* $f \in \mathcal{F}$, *satisfying*

- *for all* $t \in [T_1, T]$, $x(t) \in f$,
- *for all* $t \in (T, T_2)$, $x(t) \notin f'$, *for any* $f' \in \mathcal{F}$, *and*
- *for all* $t \in [T_1, T_2)$, $\dot{x}(t) \in g_p(x(t))$.

A partial execution is a p-partial execution for some p. Note that it starts in some facet and then possibly goes out of the facet, but does not re-enter any other facet. We then define an execution of the system as a sequence of partial executions which can be joined together.

Definition 4. *An* execution *of a switched system* $\mathcal{S}_\alpha$ *is a continuous function* $x : [0, T) \to \mathbb{R}^n$ *with* $T \in \mathbb{R}^\infty_{\geqslant 0}$, *such that there exist finite or infinite sequences of valid facets* $f_1, f_2, \ldots$ *in* $\mathcal{F}$, *of modes* $p_1, p_2, \ldots$ *in* $\mathcal{P}$ *and of partial executions* $x_1, x_2, \ldots$ *of* $\mathcal{S}$ *and* $\mathcal{F}$, *where* $dom(x_i) = [T_{i-1}, T_i)$ *and* x_i *is a* p_i*-partial execution for every* i*; satisfying:*

- $[0, T) = \bigcup_i [T_{i-1}, T_i)$,
- $x \restriction_{[T_{i-1}, T_i)} = x_i$ *for every* i,
- $x_i(T_{i-1}) \in f_i$ *for every* i *and*
- $\alpha(f_1, \ldots, f_i) = p_i$ *for every* i.

Let $Exec(\mathcal{S}_\alpha)$ denote the set of all executions of $\mathcal{S}_\alpha$.

Next, we introduce the notion of stability for a switched system. Stability is a property of a system which ensures that small perturbations in the initial state of the system lead to small variations in its behavior. We focus on a classical notion of stability, namely, Lyapunov stability. We will fix origin $\bar{0}$ to be the equilibrium point in this paper.

Definition 5. *A switched system* $\mathcal{S}_\alpha$ *is* Lyapunov stable *with respect to the origin* $\bar{0}$, *if for every* $\varepsilon > 0$ *there exists a* $\delta > 0$ *such that every execution* $x \in Exec(\mathcal{S}_\alpha)$ *with* $||x(0)|| < \delta$ *satisfies* $||x(t)|| < \varepsilon$ *for all* $t > 0$.

Note that in the sequel every time we refer to stability it is Lyapunov stability with respect to the origin. Given a family of dynamical systems, our goal is to construct a stable switched system. The switching strategy is determined by considering a set of switching facets and assigning a dynamical system to each of them. The dynamical system associated to a facet describes the evolution of the switched system after reaching the facet.

Problem 1 (Stabilization problem). Given a system $\mathcal{S} = (\mathcal{P}, \{g_p\}_{p \in \mathcal{P}})$ and a set of valid facets $\mathcal{F}$, find a switching strategy $\alpha : \mathcal{F}^+ \to \mathcal{P}$ such that the switched system $\mathcal{S}_\alpha$ is stable.

Note that we restrict ourselves to certain switching sets. Nevertheless, we can choose any polyhedral partition in order to determine the facets for switching.

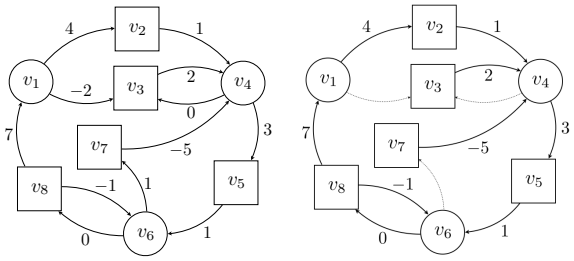
4 Games

The synthesis problem is going to be solved by reducing to certain problems on games. More precisely, we transform the stabilization problem into finding a strategy in a finite weighted game graph. Here we present the concept of game and associated notions.

A game is played between a player and an adversary, where the goal of the player is to achieve a certain objective, whereas, that of the adversary is to thwart the efforts of the player. We want to find a strategy for the player which will allow her to achieve the objective for any strategy of the environment. The game is specified using a game graph that consists of a set of nodes partitioned between the player and the adversary. At each point, depending on the node, the player or the adversary get to choose an edge and make a transition to a node, and then the game continues from the new node. The objective corresponds to ensuring that the sequence of nodes in a play satisfy certain conditions. A winning strategy corresponds to a strategy of the player which ensures that the condition holds for every play resulting from the strategy.

Definition 6. *A* game graph *is a weighted graph* $G = (V, E, W)$ *where* $V = V_0 \cup V_1$, $V_0 \cap V_1 = \emptyset$, E *is a subset of* $V_0 \times V_1 \cup V_1 \times V_0$ *and* W *is a weight function mapping edges to rational numbers,* $W : E \to \mathbb{Q}$.

The partition of the nodes indicates that the nodes in V_0 are those where the player makes a choice, while nodes in V_1 are those where the adversary makes a choice.



(a) Game graph G (b) Strategy over G

Fig. 1. Game graph and strategy

Figure 1(a) shows a game graph. The nodes represented by circles are those where the player makes a decision and the nodes represented by squares are those where the adversary makes a decision.

Definition 7. *A* play *in* G *is an infinite sequence of nodes* $\tau = v_1 v_2 \dots v_k \dots$ *such that* $(v_i, v_{i+1}) \in E$ *for every* $i \geqslant 1$. *The set of all plays in* G *is denoted as* $Plays(G)$. *The set of edges in a play* τ *is denoted as* $E(\tau)$.

We say that the weight of a play $\tau = v_1 v_2 \dots$ is bounded by b, denoted $W(\tau) \leqslant b$, if $\sum_{i=1}^{j} W(v_i, v_{i+1}) \leqslant b$ for every j.

Definition 8. *A* strategy *in G is a function $\sigma : V^* V_0 \to V_1$ such that for every $\tau \in V^* V_0$, $(last(\tau), \sigma(\tau)) \in E$. A strategy σ is said to be* memoryless *if for every $\tau_1, \tau_2 \in V^* V_0$ with $last(\tau_1) = last(\tau_2)$, $\sigma(\tau_1) = \sigma(\tau_2)$.*

Notation.- For simplicity, we denote a memoryless strategy as a function $\sigma : V_0 \to V_1$.

Definition 9. *A finite or infinite sequence of nodes $\tau = v_0 v_1 \dots$ is* consistent with a strategy σ *if for all $\tau' = v_0 v_1 \dots v_k$ subsequence of τ with $\tau' \in V^* V_0$, $\sigma(\tau') = v_{k+1}$.*

In Figure 1(b), the solid outgoing arrows from the circles show a memoryless strategy for the player in the game graph, which specifies the choices of the player when in those nodes. The strategy induces a subgraph, defined by all the solid arrows.

We consider two types of objectives for the game; one requires to ensure that the weight of the plays remains bounded, and the other requires that the energy associated with the prefixes of the play remains positive when started with some given positive energy. Next, we define the winning strategies with respect to these objectives. A winning bounded strategy states that the weight of the plays consistent with the strategy is bounded from above.

Definition 10. *A strategy σ is a* winning bounded strategy *for a game graph G if there exists a value $M \in \mathbb{Z}$ such that for every $\tau \in Plays(G)$ consistent with σ, $W(\tau) \leqslant M$.*

A winning energy strategy states that the player sustains a positive energy value whatever the choices of the adversary are.

Definition 11. *A strategy σ is a* winning energy strategy *for a game graph G if there exists a value $w_0 \in \mathbb{N}$ such that for every play in G consistent with σ of the form $\tau = v_1 v_2 \dots$, $w_0 + \sum_{i=1}^{j} W(v_i, v_{i+1}) \geqslant 0$ for all $j > 0$.*

5 Switching Strategy Synthesis for Stabilization

In this section, we present a comprehensive abstraction-based approach for synthesizing a switching strategy that stabilizes the system. We will address some of the computational issues in the next section.

Let us fix a system $\mathcal{S} = (\mathcal{P}, \{g_p\}_{p \in \mathcal{P}})$ and a valid set of switching facets $\mathcal{F}$. The synthesis algorithm is shown in Algorithm 1. It proceeds in three steps:

Step 1 It constructs a game graph, $G(\mathcal{S}, \mathcal{F})$ which *abstracts* the system $\mathcal{S}$ by using the facets $\mathcal{F}$.

Step 2 It searches for a *winning bounded strategy* σ for the game graph $G(\mathcal{S}, \mathcal{F})$.

Algorithm 1 Stabilizing switching strategy synthesis

Require: $\mathcal{S}, \mathcal{F}$
Ensure: α
1: $G :=$ *Game-graph*$(\mathcal{S}, \mathcal{F})$
2: $\sigma :=$ *Winning-bounded-strategy*(G)
3: $\alpha :=$ *Extract*(σ)
4: **return** α

Algorithm 2 Winning bounded strategy

Require: G
Ensure: σ
1: $G^e :=$ *Energy-game-graph*(G)
2: $\sigma :=$ *Winning-energy-strategy*(G^e)
3: **return** σ

Step 3 A *stabilizing switching strategy* α for $\mathcal{S}$ and $\mathcal{F}$ is extracted from the winning bounded strategy σ.

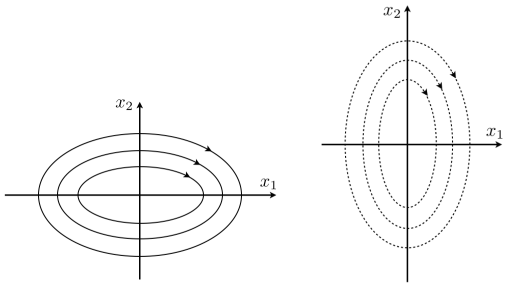
Consider, for instance, the linear dynamical systems shown in Figure 2. They are specified by $\dot{x} = A_1 x$ and $\dot{x} = A_2 x$ and the family of dynamical systems is denoted as $\mathcal{S}_1 = (\{1, 2\}, \{A_1, A_2\})$, where

$$A_1 = \begin{pmatrix} 0 & 1 \\ -0.1 & 0 \end{pmatrix} \text{ and } A_2 = \begin{pmatrix} 0 & 1 \\ -4 & 0 \end{pmatrix}$$

We consider the state space $\mathbb{R}^2$ partitioned into four operating regions, $\{\Omega_1, \Omega_2, \Omega_3, \Omega_4\}$, which correspond to the four quadrants and are separated by the following facets, $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$, where $f_1 = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 = 0, x_2 \geqslant 0\}$, $f_2 = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 \geqslant 0, x_2 = 0\}$, $f_3 = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 = 0, x_2 \leqslant 0\}$ and $f_4 = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 \leqslant 0, x_2 = 0\}$. We want to find a switching strategy $\alpha : \mathcal{F}^+ \to \mathcal{P}$ which stabilizes the system $\mathcal{S}_1$. The switching strategy will associate with each sequence of facets a dynamical system in the family, which describes the evolution of the system when starting from the last facet. Next, we explain in detail, the enumerated steps for this switching strategy construction.

5.1 Game graph construction

First, we explain the construction of the game graph which abstracts the system $\mathcal{S}$. This extends the quantitative predicate abstraction method in [24] developed for stability verification. Predicate abstraction [12] is an abstraction technique that constructs a finite graph which over approximates the behaviors of a given system by using a finite set of predicates that partition the state space. Hence, properties such as safety can be inferred by analysing the finite abstract system. Predicate abstraction of discrete and hybrid systems has been applied for safety verification [1, 2, 30]. However, qualitative abstract graphs do not suffice



(a) $\dot{x} = A_1 x$ (b) $\dot{x} = A_2 x$

Fig. 2. Sample trajectories

for stability analysis, and hence, a quantitative predicate abstraction was proposed in [24]. Here we borrow the ideas to construct a game graph for solving the stabilization problem.

Let us consider the system $\mathcal{S} = (\mathcal{P}, \{g_p\}_{p \in \mathcal{P}})$ and a valid set of facets $\mathcal{F}$ along with the polyhedral partition R that defines the facets. The player nodes of the game graph correspond to the facets, and the adversary nodes correspond to the pairs of modes and facets, $(p, f) \in \mathcal{P} \times \mathcal{F}$. From every node f, there are edges to the nodes (p, f) for every $p \in \mathcal{P}$. The edge corresponds to the fact that in the facet f, the system chooses to execute the dynamics p. From a node (p, f), there is an edge to every node f' such that the facet f' can be reached by executing the dynamics p from f (without touching any other facets). More precisely, the edge is added if some point in f can reach some point in f' by a p-partial execution. Further, the weight on an edge $((p, f), f')$ will provide an upper bound on the logarithm of the scalings associated with p-executions from f to f'; scaling of an execution captures the ratio of the distance of the execution from the origin when in f' as compared to its distance when it starts at f. It captures the factor by which the state is farther from the origin at the end as compared to where it started. A winning bounded strategy for this game graph indicates a strategy for the system $\mathcal{S}$ for which the scalings associated with the resulting executions are bounded, which in turn implies Lyapunov stability. Next, we present the concepts required to formalize the game graph construction.

Definition 12. *Given a region $\Omega \in R$, states x_1, x_2 and $p \in \mathcal{P}$, we denote by $x_1 \xrightarrow{p,\Omega} x_2$ the fact that there exists a p-partial solution $x : [T_1, T_2) \to \mathbb{R}^n$ such that $x(T_1) = x_1$, $x(t) \in \Omega$ for all $T_1 < t < T_2$, and $\lim_{t \to T_2} x(t) = x_2$.*

$x_1 \xrightarrow{p,\Omega} x_2$ denotes the fact that there exists a p-partial execution x starting at x_1 which converges to x_2 and remains within the region Ω at the intermediate times. We use converges instead of reaches only because the domain of our partial executions is right open.

Definition 13. *Given $f \in \mathcal{F}$, a region $\Omega \in R$, a set $P \subseteq \mathbb{R}^n$ and $p \in \mathcal{P}$, the scaling is given by*

$$SC(\Omega, (p, f), P) = \sup\{\frac{||x_2||}{||x_1||} \mid x_1 \in f, x_2 \in P, x_1 \xrightarrow{p,\Omega} x_2\}.$$

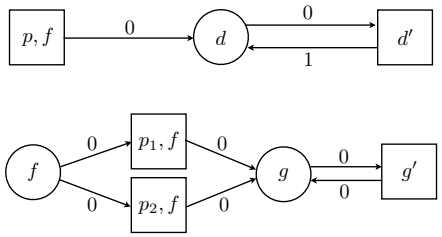


Fig. 3. (a) Unbounded executions (b) Non-existence of p-partial executions which reach another facet

Our objective is to capture executions of the switched system by paths in the game graph such that its p-partial sub-executions correspond to the edges in the game graph. The game graph construction, informally described earlier, captures all p-partial executions that traverse between two facets. However, it does not account for those which start at a facet and remain within an adjacent region, never reaching another facet. Hence, we add additional elements to the game graph to capture these executions. We check if the scaling of the executions in the region are unbounded, by checking if $SC(\Omega, (p, f), \Omega) = \infty$, that is, the scaling associated with the executions starting from p while in a region Ω do not have a bound. If the scalings evaluate to infinity for some Ω, then we add the substructure shown in Figure 3. This is also illustrated in the experiments in Section 7.1 (see Figure 9 and Figure 6). We add two nodes, d and d', and three edges, $((p, f), d)$, (d, d') and (d', d), with weights $0, 0$ and 1 respectively. If the scalings of the executions associated with a region are bounded, then they do not have any effect on the stability (they are trivially stable) and hence, we can ignore these executions. However, in some cases, all the executions out of (p, f) may not reach any other facets; hence, to ensure that all plays of the game graph are infinite we add some dummy nodes and edges as shown in Figure 3. It consists of a cycle with two new nodes, g and g' and edges (g, g') and (g', g). The node (p, f) has an edge $((p, f), g)$. All these new edges are annotated with weight equal to 0.

Remark 1. Note that the weights on the edges only capture the scalings when the executions touch the facets. It may appear that we ignore the scalings of the executions between the facets, however, the check $SC(\Omega, (p, f), \Omega) \neq \infty$, also

ensure that there is a bound on the scaling of the executions between facets. However, the exact values of these bounds are not necessary for stability analysis as will be obvious from the proof.

Next, we introduce the formal definition for the abstract game graph.

Definition 14. *A* game graph induced *by a system $\mathcal{S}$ and a set of facets $\mathcal{F}$, denoted $G(\mathcal{S},\mathcal{F}) = (V, E, W)$, satisfies:*

- $V = V_0 \cup V_1$ *where*
  - $V_0 = \mathcal{F} \cup \{d, g\}$,
  - $V_1 = (\mathcal{P} \times \mathcal{F}) \cup \{d', g'\}$;
- $E = E_0 \cup E_1 \cup E_1^d \cup E_1^g \cup E_0^* \cup E_1^*$ *where*
  - $E_0 = \{(f, (p, f)) \,|\, f \in \mathcal{F}, p \in \mathcal{P}\}$,
  - $E_1 \supseteq \{((p, f_1), f_2) \,|\, \exists \Omega \in R, x_1 \in f_1$ *and* $x_2 \in f_2$ *such that* $x_1 \xrightarrow{p,\Omega} x_2\} = E_1^{exact}$,
  - $E_1^d = \{((p, f), d) \,|\, \exists \Omega \in R : SC(\Omega, (p, f), \Omega) = \infty\}$,
  - $E_1^g = \{((p, f), g) \,|\, \nexists f' : ((p, f), f') \in E_1 \cup E_1^d\}$,
  - $E_0^* = \{(d, d'), (g, g')\}$,
  - $E_1^* = \{(d', d), (g', g)\}$;
- $W(e) : E \to \mathbb{Q}$ *where*
  - $W(e) = 0$ *if* $e \in E \setminus (E_1 \cup \{(d', d)\})$,
  - $W(((p, f_1), f_2)) \geqslant ln(SC(\Omega, (p, f_1), f_2))$ *for all* $\Omega \in R$,
  - $W((d', d)) = 1$.

Here *ln* denotes the natural logarithm.

Remark 2. The game graph is an over approximation of the executions of the system $\mathcal{S}$. Firstly, the edges are added if there is at least one execution between two facets. Note that if there is an execution from facet f_1 to f_2 and one from f_2 to f_3, it does not necessarily imply that there is an execution from f_1 to f_3 which goes through f_2. Hence, the game graph may have a path which does not correspond to any executions. However, every execution is captured by a path in the graph.

Secondly, we require E_1 to contain E_1^{exact} and not be equal to it. This is because in general we will not be able to compute E_1^{exact}. More discussion on computation of the graph will appear in Section 6. However, our results hold even for this relaxed set, that is, if we find a strategy for the game graph, we can extract one for the original system. This is because a relaxed E_1 just implies that we are considering a more general adversary than what is required. However, from the point of view of succeeding in finding a strategy for the game graph, it is desirable to have E_1 to be close to E_1^{exact}.

The game graph induced by the system $\mathcal{S}_1 = (\{1, 2\}, \{A_1, A_2\})$ described in the previous section and depicted in Figure 2, is shown in Figure 4. The circles correspond to the facets f_1, f_2, f_3 and f_4 corresponding to the four axes as explained in Section 5. In this state, the player makes the choice regarding the dynamics (potentially depending on her previous choices). From each of the

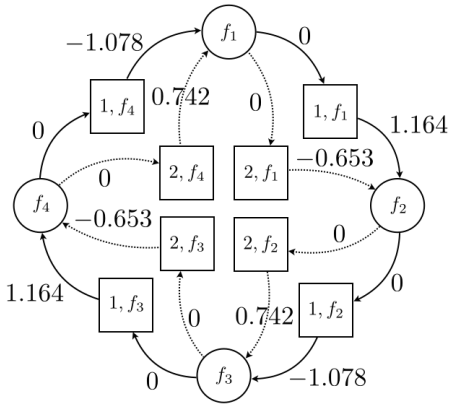


Fig. 4. Game graph

nodes f_i, there is an edge to (p, f_i) with weight 0 to capture the possible choices. The nodes (p, f_i) correspond to the adversary and has an edge to facets f_j to which the system can evolve from f_i using the dynamics p. Here, the strategy (choices of the player) need to be such that the resulting play is winning for any of the edges taken from (p, f_i). These facets have a weight that is an upper bound on the logarithm of the scaling of the executions between the facets. For instance, the edge $((2, f_1), f_2)$ states that there is potentially a 2-partial execution starting from f_1 that will evolve through some quadrant (the first) following the linear dynamics related to A_2 and reach f_2. Its weight 0.742 signifies that any 2-execution starting from f_1 and reaching f_2 will be at most $\mathrm{e}^{0.742}$ times the distance to the origin as compared to the distance when it started. A positive value indicates that the execution is moving farther, and a negative value indicates the execution is getting closer to the origin. Note that the scaling of an execution consisting of a p_1-execution x_1 followed by a p_2-execution x_2 is the product of the two scalings. The logarithm of the scaling would be the sum of the logarithms of the scaling.

Our next objective is to establish the main result of the paper. We will show that a strategy for the game graph will yield a stabilizing switching strategy for the system. First, we need the definition of a strategy for the system that can be extracted from a strategy for the game graph.

Definition 15. *Given a system $\mathcal{S}$, a valid set of facets $\mathcal{F}$, the induced game graph $G(\mathcal{S}, \mathcal{F})$ and a strategy $\sigma : V^* V_0 \to V_1$ for $G(\mathcal{S}, \mathcal{F})$, we say that $\alpha_\sigma : \mathcal{F}^+ \to \mathcal{P}$ is a* switching strategy induced *by σ if for a sequence $\tau = f_1, (p_1, f_1), f_2, (p_2, f_2), \ldots, f_k$ consistent with σ, $\alpha_\sigma(f_1, f_2, \ldots, f_k) = \sigma(\tau)_\mathcal{P}$.*

Theorem 1. *Given a system $\mathcal{S}$ and a valid set of facets $\mathcal{F}$, if there exists a winning bounded strategy σ for the induced game graph $G(\mathcal{S}, \mathcal{F})$, then any induced strategy α_σ solves the stabilization problem for $\mathcal{S}$ and $\mathcal{F}$.*

Proof. We need to show that the switched system $(\mathcal{P}, \{g_p\}_{p\in\mathcal{P}}, \alpha_\sigma)$ is stable. Let M be the bound on the weight of all consistent plays of σ. Consider an execution $x : [0, T) \to \mathbb{R}^n$ of the system $(\mathcal{P}, \{g_p\}_{p\in\mathcal{P}}, \alpha_\sigma)$, that is represented as a sequence of partial executions, $x = x_1x_2\ldots$, as in Definition 3, that is, each x_i is a p_i-partial execution and $x_i(T_{i-1}) \in f_i \in \mathcal{F}$. Since, $\alpha_\sigma(f_1, \ldots, f_k) = p_k$, the play $\tau = f_1(p_1, f_1)f_2(p_2, f_2)f_3\ldots$ in $G(\mathcal{S}, \mathcal{F})$ is consistent with σ. The weight of the path $f_i(p_i, f_i)f_{i+1}$ is a bound on the logarithm of the scaling $||x_{i+1}(T_i)||/||x_i(T_{i-1})||$, which is given by $W((f_i, (p_i, f_i))) + W(((p_i, f_i), f_{i+1}))$. Hence, the logarithm of any scaling associated with any prefix of the execution x of the form $x_1 \ldots x_k$ is bounded by the value $\sum_{i=1}^{k-1}[W((f_i, (p_i, f_i))) + W(((p_i, f_i), f_{i+1}))]$, which in turn is bounded by M. We know that the scaling of $x_1 \ldots x_k$ is bounded for every k. We want to show that the scaling $||x(t)||/||x(0)||$ is bounded for times t which correspond to interior of the domain of some x_i. More precisely, we want to show that the scalings associated with $x_i : [T_i, T_{i+1}) \to \mathbb{R}^n$ is bounded at time $t \in [T_i, T_{i+1})$. Note that since σ is a bounded winning strategy, we know that any finite sequence $\tau' = f'_1(p'_1, f_1)f'_2 \ldots f'_n$ consistent with it will not choose a p such that (p, f'_n) has an edge to d, otherwise, this play can be extended to a play with unbounded weight that is consistent with σ (since from d, σ is forced to choose d' and d' has an edge back to d). Therefore, we can conclude that every mode p that is chosen from a node f in a consistent play is such that p-executions from f are bounded. For every facet f and mode p that appears in any consistent play of the game graph, let N_p be a bound on the logarithms of the scalings associated with p executions from f. Let N be the maximum of all the N_ps. Now, we can conclude that the logarithm of the scaling associated with any x is bounded from above by $M + N$. Given any $\epsilon > 0$, we choose $\delta < \epsilon/(\mathrm{e}^{M+N})$. We know that any execution of the switched system starting within the δ ball will not leave the $\delta\mathrm{e}^{M+N}$ ball, that is, the ϵ ball. Hence, the system is Lyapunov stable.

5.2 Winning bounded strategy computation

Next we solve the problem of synthesizing a winning bounded strategy for a game graph G. Our broad approach is to reduce the problem of winning bounded strategy synthesis to that of winning energy strategy synthesis. The latter can be solved using the algorithm in [5] which outputs a memoryless winning energy strategy. First, we state the results that have been established in [5] on winning energy strategies.

Theorem 2. *[5] Given a game graph (V, E, W) where $W : E \to \mathbb{Z}$, if there exists a winning energy strategy, then there exists a memoryless winning energy strategy. Further, there is an algorithm which returns a memoryless winning energy strategy if it exists.*

We reduce the problem of finding a winning bounded strategy for the game graph G to finding a winning energy strategy for a game graph G^e. The transformation from G to G^e is achieved in two steps. Note that in Theorem 2 the

weights are required to be integers. To ensure this, we multiply all the weights in G by a value which is the least common multiple of all the denominators of the rational weights. More precisely, let the weight of an edge e in E be the rational expressed as $\frac{a_e}{b_e}$. We denote by $\text{LCM}_G \in \mathbb{Z}$ the least common multiple of $\{b_e : e \in E\}$. It is clear that by multiplying each of the weights by LCM_G, we obtain integer values. Next, note that the condition for a winning energy strategy requires that the weights along a play are lower bounded by a value, whereas that for bounded strategy requires that the weights are upper bounded by a value. To accommodate this, we negate the values of the weights.

The formal definition of the energy game is given below, followed by the theorem which states that the transformation preserves existence of corresponding kind of strategies.

Definition 16. *Let $G = (V, E, W)$ be a game graph. The* induced energy game graph *is $G^e = (V, E, W^e)$ where $W^e \equiv -\text{LCM}_G \cdot W$.*

Theorem 3. *Let G be a game graph and G^e the induced energy game graph. Then, there exists a winning bounded strategy for G if and only if there exists a winning energy strategy for G^e. Moreover, σ is a winning bounded strategy for G if and only if σ is a winning energy strategy for G^e.*

Proof. Note that if σ is a winning bounded strategy for G, then for every play $\tau = v_1, v_2, \ldots$ consistent with σ satisfies $W(\tau) \leqslant M$, that is, $\sum_{i=1}^{j} W(v_i, v_{i+1}) \leqslant M$. Then

$$\text{LCM}_G \cdot M + \sum_{i=1}^{j} (-\text{LCM}_G \cdot W(v_i, v_{i+1})) \geqslant 0,$$

that is, $\max(0, \text{LCM}_G \cdot M) + \sum_{i=1}^{j} W^e(v_i, v_{i+1}) \geqslant 0$. Hence, σ is a winning energy strategy for G^e with the initial energy $\max(0, \text{LCM}_G \cdot M) \in \mathbb{N}$. The argument in the other direction is similar.

We solve the problem of finding winning bounded strategy for G by finding the winning energy strategy for G^e and solving the latter. Since memoryless winning energy strategies are sufficient for G^e, from Theorem 3, memoryless winning bounded strategies suffice for G. We find memoryless winning energy strategy for G^e using the algorithm in [5]. The algorithm computes a "small energy progress measure" that provides an initial energy on each of the nodes such that on every edge the weight on the target is a lower bound on the sum of the weight of the source together with the weight of the edge. This condition holds for some edge from a player node and every edge from an adversary node. A memoryless strategy can then be extracted from it by choosing the edge from the player node that satisfies this property. The initial energy progress measure for every node is set to zero. After i iterations, the progress measure ensures that the condition on winning energy strategy is satisfied by all plays up to length i. The iteration is similar to a standard value iteration, and converges in a finite number of steps. A winning strategy can be constructed either during

the computation of the small energy measure or can be extracted from the small energy measure.

For the game graph shown in Figure 4, after multiplying their weights by the corresponding $-\text{LCM}_G$, a memoryless winning bounded strategy σ is obtained. It is defined as $\sigma(f_1) = 1$, $\sigma(f_2) = 2$, $\sigma(f_3) = 1$ and $\sigma(f_4) = 2$. From this strategy, the stabilizing switching strategy for the system $\mathcal{S} = (\{1, 2\}, \{A_1, A_2\})$ is as follows:

$$\alpha_\sigma(\tau) = \begin{cases} 1 \text{ if } \mathit{last}(\tau) \in \{f_1, f_3\} \\ 2 \text{ if } \mathit{last}(\tau) \in \{f_2, f_4\} \end{cases}$$

6 Game graph computation

In this section, we discuss the computational issues related to the game graph construction. The crux of the edge and the weight computation lie in computing a reachability relation $R(P_1, P_2, P_3, dyn)$, namely, given three polyhedral sets $P_1, P_2, P_3 \subseteq \mathbb{R}^n$ and a dynamical system dyn, it consists of all the pair of points (x_1, x_2) such that from the point $x_1 \in P_1$, by following the dynamics dyn, $x_2 \in P_2$ is reached, while always remaining within P_3 at the interior points. Then the weight correspond to

$$w = \sup\{\frac{||x_2||}{||x_1||} \mid (x_1, x_2) \in R(P_1, P_2, P_3, dyn)\} \tag{2}$$

Our objective is to compute the set $R(P_1, P_2, P_3, dyn)$ and the weight given in Equation 2. In this section, we discuss the computation of these entities when dyn is a polyhedral inclusion dynamics or a linear dynamics. First, it was observed in [23, 25] that Lyapunov stability is a local property whose satisfaction depends only on a small neighborhood of the origin, and hence, the only switching surface that play a role in determining stability are those that contain the origin. Further, both polyhedral inclusion dynamics $\dot{x} \in P$, where P is a polyhedron, and linear dynamics $\dot{x} = Ax$, where A is a matrix, are scaling closed, namely, if there is an execution from x to y, then there is an execution from αx to αy. This is evident from the solutions of these systems, namely, $x(t) = x(0) + ct$ for some $c \in P$, and $x(t) = e^{At}x(0)$, respectively. Hence, as argued in [23, 25], it suffices to restrict our attention to facets that are upward scaling closed, that is, facets f such that if $x \in f$, then $\beta x \in f$ for all $\beta > 0$.

First let us consider the polyhedral inclusion dynamics $\dot{x} \in P$. Note that $R(P_1, P_2, P_3, dyn)$ is the pairs (x_1, x_2) which satisfy $x_1 \in P_1$, $x_2 \in P_2$, $x_2 = x_1 + ct$, $c \in P$, $x_1 + ct' \in P_3$ for all $0 < t' < t$. These constraints can be transformed into a set of linear constraints. For instance, checking $x_1 + ct' \in P_3$ for all $0 < t' < t$ is equivalent to just checking that $x_1 \in P_3$ and $x_1 + ct \in P_3$. The non-linear term ct can be eliminated as well, see [24] for details. Hence, the edge computation is reduced to a feasibility checking of linear constraints. To compute the scaling, we need to optimize the value of $||x_2||/||x_1||$. It was shown in [24] that we can restrict the optimization to x_1 such that $||x_1||$ is 1 and the optimization can be reduced to polynomially many linear programming problems

in the dimension of the system. We use these ideas for scaling computation in our tool and experiments.

The reachability relation $R(P_1, P_2, P_3, dyn)$ cannot be computed exactly when dyn is a linear dynamical system $\dot{x} = Ax$, since the solutions for these systems involve an exponential function, and hence, needs to be approximated. There are several techniques for over approximating the reachability relation that use sample values to construct over approximations represented by polyhedra (zonotopes, boxes) or polynomials (ellipsoids) [7, 14, 10]. However, these approximations can be computed only for a bounded time, and hence work well if an upper bound on the time to reach the target polyhedron P_2 from the polyhedron P_1 is known a priori. To overcome the drawback of the time bound computation, we use an alternate technique to compute the over approximations, namely, hybridization [8, 25]. Here, we divide the state space (here P_3) into a finite number of regions and approximate the linear dynamics in each of the regions by a polyhedral dynamics (see [25] for more details). We then compute the reachable set of these polyhedral dynamics (a switched system) which provides an upper bound on the reachable set $R(P_1, P_2, P_3, dyn)$. Further the computed set is a finite set of polyhedra, hence, the weight can be computed again by solving some linear programming problems.

Synthesis Algorithm Summary Here we recap the steps for synthesizing a switching strategy that stabilizes a given system $\mathcal{S} = (\mathcal{P}, \{g_p\}_{p \in \mathcal{P}})$ by considering a valid set of switching facets $\mathcal{F}$. They are summarized at the beginning of Section 5. The construction of the game graph $G(\mathcal{S}, \mathcal{F})$, which corresponds to Step 1, is explained in Definition 14; and details on its construction are discussed at Section 6. The search for a winning bounded strategy σ for the constructed game graph $G(\mathcal{S}, \mathcal{F})$, corresponds to Algorithm 1 in [5], which is justified at the beginning of Subsection 5.2. Finally, Step 3, which corresponds to the extraction of a stabilizing switching strategy α from σ, is explained at the end of Subsection 5.2.

7 Implementation

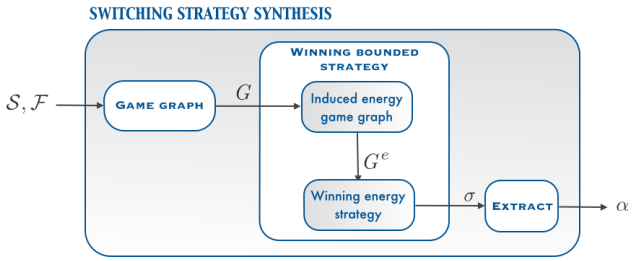


Fig. 5. Tool Architecture

In this section, we explain some details of the implementation of the algorithm for solving the stabilization problem. We have implemented the full procedure for a family of systems with polyhedral inclusion dynamics and linear dynamics. A general flowchart of the prototype tool is shown in Figure 5. The input to the procedure is a family of polyhedral or linear systems $\mathcal{S}$ and a set of valid facets $\mathcal{F}$ that are obtained from some polyhedral partition of the state space; and the output is a stabilizing switching strategy. The first module "game graph" constructs the abstract game graph $G(\mathcal{S}, \mathcal{F})$ as defined in Definition 14. This game graph is the input to the "winning bounded strategy" module, that transforms the game graph and reduces the winning bounded strategy synthesis to winning energy strategy synthesis as in Definition 16 by modifying the weights. The winning energy strategy synthesized is also a winning bounded strategy, and it is output by the module. The module "Extract" outputs a switching strategy corresponding to a winning bounded strategy as described in Definition 15. If a strategy does not exist at any stage of the procedure, an empty strategy is returned.

The prototype has been implemented in Python. The edge and weight computation involve the computation of a reachability relation and optimization. We use the Parma Polyhedra Library (PPL) to perform all the polyhedral operations involved in the reachability relation computation and the optimizations required during scaling computation are performed using the GLPK package for solving linear programming problems. Our procedure requires several preprocessing steps on the game graph as well as the implementation of the strategy synthesis algorithm for which we use the NetworkX package. The graph construction algorithm takes time polynomial in the number of facets, the dynamics and the dimension; and the size of the graph is also polynomial in these parameters. The synthesis algorithm on the abstract game graph is polynomial in the size of the graph and the maximum constant appearing on the edges.

This implementation allows us to synthesize automatically a stabilizing switching strategy for a family of polyhedral inclusion dynamics or linear dynamics. Next, we illustrate the process for each of these cases.

7.1 Experiments

Here, we enumerate the computational details for two different experiments. One considers a family of polyhedral dynamical systems and the other considers a family of linear dynamical systems.

Polyhedral dynamics We consider a family of four polyhedral dynamical systems in $\mathbb{R}^2$. They are specified by $\dot{x} \in P_i$, where P_i corresponds to a single vector for each system: $P_1 = (1, 1)$, $P_2 = (-1, -1)$, $P_3 = (-1, 1)$ and $P_4 = (1, -1)$. The family of systems is denoted as $\mathcal{S} = (\{1, 2, 3, 4\}, \{P_1, P_2, P_3, P_4\})$. The set of valid facets are the semi axes of the plane, that is $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$ where $f_1 = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 = 0, x_2 \geqslant 0\}$, $f_2 = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 \geqslant 0, x_2 = 0\}$, $f_3 = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 = 0, x_2 \leqslant 0\}$ and $f_4 = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 \leqslant 0, x_2 = 0\}$. These facets divide the state space into four operating regions, $\{\Omega_1, \Omega_2, \Omega_3, \Omega_4\}$,

which correspond to the first, second, third and fourth planar quadrants, respectively. We illustrate partially the computation of the game graph $G(\mathcal{S}, \mathcal{F}) = (V_0 \cup V_1, E, W)$. We restrict this computation to the first quadrant, which contains the pair of facets f_1 and f_2, and their common region Ω_1. These two facets will be nodes in V_0. For V_1, we include the nodes $(1, f_1)$, $(2, f_1)$, $(3, f_1)$, $(4, f_1)$, $(1, f_2)$, $(2, f_2)$, $(3, f_2)$ and $(4, f_2)$. We construct edges in E_0 from f_1 to all the nodes of the form (p, f_1), and analogously, we construct edges from f_2 to (p, f_2), where $p \in \{1, 2, 3, 4\}$. The weight for all these edges is 0. In E_1, we obtain the edges $((4, f_1), f_2)$ and $((3, f_2), f_1)$, with $SC(\Omega_1, (4, f_1), f_2) = 1$ and $SC(\Omega_1, (3, f_2), f_1) = 1$. The set E_1^d contains $((1, f_1), d)$ and $((1, f_2), d)$. with weight value 0. The edges in E_1^g, are $((2, f_1), g)$, $((2, f_2), g)$, $((3, f_1), g)$ and $((4, f_2), g)$, all of them with weight equal to 0. The graph obtained for the first quadrant is shown in Figure 6. The computation for the rest of the quadrants is similar to that presented above.

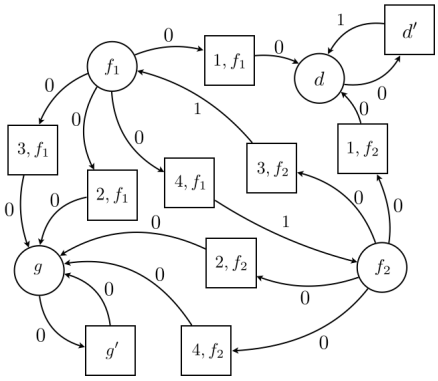


Fig. 6. Partial game graph

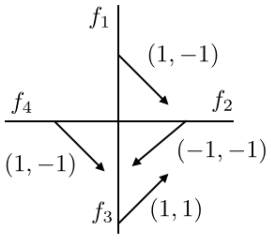


Fig. 7. Stabilizing switching strategy

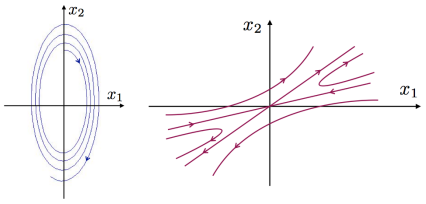
The scalings associated with the edges E_1 of the game graph $G(\mathcal{S}, \mathcal{F})$ are 1s. Hence, in the induced energy game graph, all weights are equal to 0 except for the edge (d', d), which has weight equal to -1. (Recall that all the weights in the induced graph are negated). The tool returns the following winning energy strategy; $\sigma : V_0 \to V_1$ where $\sigma(f_1) = (4, f_1)$, $\sigma(f_2) = (2, f_2)$, $\sigma(f_3) = (1, f_3)$ and $\sigma(f_4) = (4, f_4)$. From σ, the following switching strategy is computed; $\alpha_\sigma : \mathcal{F}^+ \to \mathcal{P}$ where

$$\alpha_\sigma(\tau) = \begin{cases} 4 \text{ if } \mathit{last}(\tau) \in \{f_1, f_4\} \\ 2 \text{ if } \mathit{last}(\tau) = f_2 \\ 1 \text{ if } \mathit{last}(\tau) = f_3 \end{cases}$$

This switching strategy is depicted in Figure 7, where the semi axes correspond to the facets. For every point in a facet, an execution of the synthesized switched system starting from it, evolves by following the vector outgoing from the facet.

Linear dynamics We consider a family of linear systems in the two dimensional real state space. denoted as $\mathcal{S} = (\{1, 2\}, \{A_1, A_2\})$, where $A_1 = \begin{pmatrix} 0 & 1 \\ -5 & 0.1 \end{pmatrix}$ and $A_2 = \begin{pmatrix} -1 & 4 \\ -0.2 & 1 \end{pmatrix}$.

Sample trajectories of these systems are shown in Figures 8(a) and 8(b). The



(a) $\dot{x} = A_1 x$ (b) $\dot{x} = A_2 x$

Fig. 8. Phase portraits

polyhedral partition is obtained by dividing the state space, as in the previous experiment, with the two coordinate axes $\{x_1 = 0, x_2 = 0\}$. They partition the state space into four operating regions, $\{\Omega_1, \Omega_2, \Omega_3, \Omega_4\}$, which coincide with the four quadrants and are separated by the same facets as in the previous experiment, $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$. Then, we compute the game graph $G(\mathcal{S}, \mathcal{F}) = (V, E, W)$, where $V_0 = \mathcal{F}$, $V_1 = \{(1, f_1), (2, f_1), (1, f_2), (2, f_2), (1, f_3), (2, f_3), (1, f_4), (2, f_4)\}$, $E_0 = \{(f_1, (1, f_1)), (f_1, (2, f_1)), (f_2, (1, f_2)), (f_2, (2, f_2)), (f_3, (1, f_3)), (f_3, (2, f_3)), (f_4, (1, f_4)), (f_4, (2, f_4))\}$, $E_1 = \{((1, f_1), f_2), ((1, f_2), f_3), ((2, f_2), f_3), ((1, f_3), f_4), ((1, f_4), f_1), ((2, f_4), f_1)\}$, $E_1^* = \{((2, f_1), d), ((2, f_3), d)\}$ and $E_0^* = \{(d', d)\}$. The weights for every $e \in E_0 \cup E_1^*$ are equal to 0, for (d', d) equal to 1 and for $e \in E_1$ we obtain the following scaling values: $SC(\Omega_1, ((1, f_1), f_2) =$

$1/2$, $SC(\Omega_2, ((1, f_2), f_3) = 5/2$,
$SC(\Omega_2, ((2, f_2), f_3) = 3/10$, $SC(\Omega_3, ((1, f_3), f_4) = 1/2$,
$SC(\Omega_4, ((1, f_4), f_1) = 5/2$ and $SC(\Omega_4, ((2, f_4), f_1) = 3/10$.

We compute the natural logarithm of the different scaling values: $ln(1/2) = -\frac{6243314768165359}{9007199254740992}$, $ln(5/2) = \frac{1031651649657871}{1125899906842624}$ and $ln(3/10) = -\frac{5422211472926497}{4503599627370496}$.

The game graph $G(\mathcal{S}, \mathcal{F})$ is shown in Figure 9. The winning energy strategy $\sigma : V_0 \rightarrow V_1$ computed is defined as follows: $\sigma(f_1) = (1, f_1)$, $\sigma(f_2) = (2, f_2)$, $\sigma(f_3) = (1, f_3)$ and $\sigma(f_4) = (2, f_4)$. From the winning energy strategy, we extract a stabilizing switching strategy $\alpha_\sigma : V \rightarrow \mathcal{P}$, that is

$$\alpha_\sigma(\tau) = \begin{cases} 1 \text{ if } last(\tau) \in \{f_1, f_3\} \\ 2 \text{ if } last(\tau) \in \{f_2, f_4\} \end{cases}$$

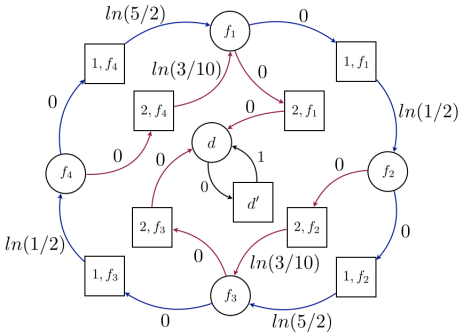


Fig. 9. Game graph

8 Conclusion

In this paper, we proposed an abstraction technique and a game based approach for synthesizing a switching logic for stabilization. While abstraction based approaches have been proposed for correct by construction of systems that satisfy certain temporal logic formulas, to the best of our knowledge, the same has not been explored for stabilization. Stability is a robustness property that is required of all control systems. Our approach can be combined with those for temporal logic properties to obtain stable controllers that satisfy temporal logic formulas. We intend to explore this in the future.

References

1. R. Alur, T. Dang, and F. Ivancic. Counter-Example Guided Predicate Abstraction of Hybrid Systems. In *TACAS*, pages 208–223, 2003.

2. R. Alur, T. Dang, and F. Ivancic. Predicate abstraction for reachability analysis of hybrid systems. *ACM Transactions on Embedded Computing Systems*, 5(1):152–199, 2006.
3. P. Bauer, K. Premaratne, and J. Duran. A necessary and sufficient condition for robust asymptotic stability of time-variant discrete systems. *Automatic Control, IEEE Transactions on*, 38(9):1427–1430, Sep 1993.
4. M. Branicky. Stability of hybrid systems. In H. Unbehauen, editor, *Encylopedia of Life Support Systems*, volume Theme 6.43:Control Sytems, Robotics and Automation, chapter Article 6.43.28.3. UNESCO Publishing, 2004.
5. L. Brim, J. Chaloupka, L. Doyen, R. Gentilini, and J. Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38(2):97–118, 2011.
6. E. Clarke, A. Fehnker, Z. Han, B. Krogh, J. Ouaknine, O. Stursberg, and M. Theobald. Abstraction and Counterexample-Guided Refinement in Model Checking of Hybrid Systems. *International Journal on Foundations of Computer Science*, 14(4):583–604, 2003.
7. T. Dang and O. Maler. Reachability analysis via face lifting. In *HSCC*, pages 96–109, 1998.
8. T. Dang, O. Maler, and R. Testylier. Accurate hybridization of nonlinear systems. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 11–20, 2010.
9. R. DeCarlo, M. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE*, 88(7):1069–1082, July 2000.
10. 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 *CAV*, pages 379–395, 2011.
11. A. Girard. Controller synthesis for safety and reachability via approximate bisimulation. *Automatica*, 48(5):947–953, 2012.
12. S. Graf and H. Saidi. Construction of abstact state graphs with PVS. In *CAV*, pages 72–83, 1997.
13. M. Johansson. *Piecewise Linear Control Systems*. Springer, 2003.
14. A. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In *HSCC*, pages 202–214, 2000.
15. D. Liberzon, J. P. Hespanha, and A. S. Morse. Stability of switched systems: a lie-algebraic condition. *Systems Control Lett*, 37:117–122, 1999.
16. H. Lin and P. J. Antsaklis. Switching stabilizability for continuous-time uncertain switched linear systems, 2007.
17. H. Lin and P. J. Antsaklis. Stability and stabilizability of switched linear systems: A survey of recent results. *IEEE Trans. Automat. Contr.*, 54(2):308–322, 2009.
18. T. Moor, J. Raisch, and S. O'Young. Discrete supervisory control of hybrid systems based on l-complete approximations. *Discrete Event Dynamic Systems*, 12(1):83–107, 2002.
19. A. S. Morse. Supervisory control of families of linear set-point controllers - part 1: Exact matching. *IEEE Trans. Automat. Contr*, 41:1413–1431, 1998.
20. K. Narendra and J. Balakrishnan. A common lyapunov function for stable lti systems with commuting a-matrices. *Automatic Control, IEEE Transactions on*, 39(12):2469–2471, Dec 1994.
21. N. Ozay, J. Liu, P. Prabhakar, and R. M. Murray. Computing augmented finite transition systems to synthesize switching protocols for polynomial switched systems. In *ACC*, pages 6237–6244, 2013.

22. S. Pettersson. Synthesis of switched linear systems. In *CDC*, volume 5, pages 5283–5288 Vol.5, 2003.
23. P. Prabhakar and M. G. Soto. Abstraction based model-checking of stability of hybrid systems. In *CAV*, pages 280–295, 2013.
24. P. Prabhakar and M. G. Soto. An algorithmic approach to stability verification of polyhedral switched systems. In *ACC*, pages 2318–2323, 2014.
25. P. Prabhakar and M. G. Soto. Hybridization for stability analysis of switched linear systems. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, 2016.
26. R. Shorten and K. Narendra. Necessary and sufficient conditions for the existence of a common quadratic lyapunov function for two stable second order linear time-invariant systems. In *ACC*, volume 2, pages 1410–1414, Jun 1999.
27. R. Shorten, K. Narendra, and O. Mason. A result on common quadratic lyapunov functions. *Automatic Control, IEEE Transactions on*, 48(1):110–113, Jan 2003.
28. E. Skafidas, R. J. Evans, A. V. Savkin, and I. R. Petersen. Stability results for switched controller systems. In *Automatica*, volume 35, pages 553–564, 1999.
29. P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer Publishing Company, Incorporated, 1st edition, 2009.
30. A. Tiwari. Abstractions for hybrid systems. *Formal Methods in System Design*, 32(1):57–83, 2008.
31. A. Ulusoy and C. Belta. Receding horizon temporal logic control in dynamic environments. *I. J. Robotic Res.*, 33(12):1593–1607, 2014.
32. G. Zhai. Quadratic stabilizability of discrete-time switched systems via state and output feedback. In *CDC*, volume 3, pages 2165–2166, 2001.
33. G. Zhai, H. Lin, and P. Antsaklis. Controller failure time analysis for symmetric h infin; control systems. In *CDC*, volume 3, pages 2459–2464, Dec 2003.