Game-based Synthesis of Distributed Controllers for Sampled Switched Systems

Laurent Fribourg[1], Ulrich Kühne[2], and Nicolas Markey[1]

1 **LSV, ENS Cachan & CNRS, France**
{fribourg,markey}@lsv.ens-cachan.fr
2 **Group of Computer Architecture, University of Bremen, Germany**
ulrichk@cs.uni-bremen.de

Abstract

Switched systems are a convenient formalism for modeling physical processes interacting with a digital controller. Unfortunately, the formalism does not capture the distributed nature encountered e.g. in cyber-physical systems, which are organized as networks of elements interacting with local controllers. Most current methods for control synthesis can only produce a centralized controller, which is assumed to have complete knowledge of all the component states and can interact with all of them. In this paper, we consider a centralized-controller synthesis technique based on state-space decomposition, and use a game-based approach to extend it to a distributed framework.

1998 ACM Subject Classification B.1.2 Automatic Synthesis

Keywords and phrases Cyber-physical systems; controller synthesis; games; robustness; partial observation.

Digital Object Identifier 10.4230/OASIcs.SynCoP.2015.XXX

1 Introduction

Hybrid systems are a powerful formalism for modeling and reasoning about cyber-physical systems. They mix the continuous and discrete natures of the evolution of computerized systems. Switched systems are a special kind of hybrid systems, with restricted discrete behaviours: those systems only have finitely many different modes of (continuous) evolution, with isolated switches between modes. Such systems provide a good balance between expressiveness and controllability, and are thus in widespread use in large branches of industry such as power electronics and automotive control.

The control law for a switched system defines the way of selecting the modes during the run of the system. Controllability is the problem of (automatically) synthezing a control law in order to satisfy a desired property, such as safety (maintaining the variables within a given zone) or stabilisation (confinement of the variables in a close neighborhood around an objective point) [6].

In [12], a solution is proposed in order to achieve practical stabilization of discrete-time switched systems. It is based on the repeated bisection of the region of interest surrounding the objective point. Each resulting tile of the bisection is to be associated with a sequence of modes (or *pattern*) that should map the tile inside the zone of interest. Upon success, this naturally induces a control law that becomes cyclic and stabilizes the system along a corresponding limit cycle.

However, the decomposition method is proposed in a centralized framework, and assumes that the state of the global system is entirely known. In practice, many industrial systems,

2nd International Workshop on Synthesis of Complex Parameters (SynCoP'15).
Editors: Étienne André and Goran Frehse; pp. 1–15

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

such as cyber-physical systems, are implemented in a *distributed* manner, using actuators that are *locally* controlled. Furthermore, these controllers cannot observe the full state of the system, but have only access to the *partial* information conveyed by local sensors.

In the distributed context, the local controllers can be seen as *players*: each player has to achieve a local objective of stabilization. A sufficient condition of global stabilization is obtained when the strategy of any player meets its local objective whatever the strategy of the others. We will adopt this game-oriented view and modify the basic decomposition procedure in order to account for local controllability and partial observability.

Related work. The model of hybrid automata [16] has proven very powerful for combining discrete models issued from software and continuous models issued from the physical world. The topic of synthesizing controllers that, by construction, enforce properties such as safety or reachability, has soon attracted a lot of attention. In [3], a generic methodology is given for constructing safety controllers using iterative computation of reachable states in a backward manner. An alternative approach is proposed in [27], using a game-theoretic formulation and theory of optimal control.

Among hybrid automata, *timed automata* are a very useful class of models where all the state variables correspond to symbolic clocks evolving uniformly at the same rate [1]. In [7], an efficient control synthesis method has been proposed, extending an algorithm for solving games for finite-state systems [19]. It has been implemented in tool UPPAAL-TIGA and applied to industrial case studies [8].

As mentioned above, switched systems constitute another important subclass of hybrid systems well-suited to the modeling of many engineered systems. For this class of systems, a paradigm based on *approximate bisimulation* [25, 14, 15] allows to construct an approximately equivalent discrete model. The original control-synthesis problem can thus be solved at a discrete level, which amounts to computing winning strategies in parity games [23].

Most of the work on controller synthesis in the framework of hybrid systems has focused to the centralized framework. An exception is [20], which gives a methodology based on optimal theory in a multi-agent setting where the agents try to make optimum use of a common resource.

2 Background

2.1 Sampled Switched Systems and Decomposition Method

A *switched system* is a digital quantized control system that consists of a finite family of continuous subsystems, together with a rule that controls the switching between subsystems. Formally, a *switched system* over a set Var of variables can be described by

- a differential equation of the form $\dot{x} = f_\sigma(x)$, where $\{f_u \mid u \in U\}$ is a family of sufficiently regular functions from $\mathbb{R}^{\mathsf{Var}}$ to $\mathbb{R}^{\mathsf{Var}}$ that is parametrized by some finite index set U, called the set of *modes*,
- and a piecewise-constant function $\sigma\colon [0,\infty) \to U$, called *switching rule* [18].

We assume that the system variables have no discrete jump, i.e. the solution $x(\cdot)$ is everywhere continuous. We assume furthermore here that all the individual subsystems are affine, so we obtain an affine switched system of the form $\dot{x} = A_\sigma x + B_\sigma$. Finally, we suppose that σ changes its values *periodically*, at times $k \cdot \tau$, where $\tau \in \mathbb{R}_{>0}$ and k ranges over $\mathbb{N}$. We say that such switched systems are *sampled*. We regard these systems as *discrete-time systems*, observing the state of the system only at the switching instants $k \cdot \tau$. The integration of

the continuous equation for mode u during τ still yields an affine equation of the form $x(t+\tau) = \hat{A}_u x(t) + \hat{B}_u$.

In [12], a procedure was designed in order to synthesize a state-dependent control rule that makes all the (discrete-time) trajectories starting from a given set $R \subseteq \mathbb{R}^{\mathsf{Var}}$ repeatedly return to R (the set R is refered to as the *target set* hereafter). In our setting, we require that each variable $v \in \mathsf{Var}$ has its own target zone R_v, so that $R = \prod_{v\in\mathsf{Var}} R_v$ is a rectangular set. The method consists in decomposing R by iterative bisection until (possibly) finding, for each resulting tile, a corresponding pattern (i.e., a sequence of modes) that maps it into R. This guarantees that, starting from any tile W of R, the application of the corresponding pattern π yields a trajectory that ends within R. The crux of the method relies on a simple procedure that, given a tile W, enumerates by increasing length all the patterns, using all possible combinations of modes, until one of them, say π, maps W into R (or until we have exhausted the whole set of patterns up to a given length, in which case the system is declared uncontrollable for the given length). Formally, we write $\mathsf{Post}_\pi(W) \subseteq R$ where Post_π corresponds to the successive application of each mode composing the pattern π (note that Post_π is an affine transformation, because each mode is affine). When such a π exists, we say that the tile W is *successful*.

Each global decomposition Δ of R can be written under the form $(W_i, \pi_i)_{i\in I}$ where I is a finite set of indices, W_i is a tile, and π_i a pattern, such that $\bigcup_{i\in I} W_i = R$, and $\mathsf{Post}_{\pi_i}(W_i) \subseteq R$. For any such decomposition, one can then define an operator Post_Δ as $\mathsf{Post}_\Delta(X) = \bigcup_{i\in I} \mathsf{Post}_{\pi_i}(X \cap W_i)$, for any $X \subseteq R$. Our method aims to find a decomposition Δ of R such that $\mathsf{Post}_\Delta(R) \subseteq R$.

All along the computation of Δ, it may reveal useful to ensure not only that the trajectories return to R after application of each pattern π of the decomposition, but also that the intermediate points of trajectories, obtained after application of each single mode composing π, be confined into a given rectangular set $S \subseteq \mathbb{R}^{\mathsf{Var}}$ containing R. Such a set S is called *safety set*. In order to achieve this additional objective, we strengthen the condition for being successful by requiring the existence of a pattern satisfying:

$$\mathsf{Post}_\pi(W) \subseteq R \quad \text{and} \quad \mathsf{Interm}_\pi(W) \subseteq S \tag{1}$$

where Interm_π refers to the union of all the intermediate sets obtained by the application of the successive prefixes of π. Again, upon success, the resulting decomposition Δ enforces

$$\mathsf{Post}_\Delta(R) \subseteq R \quad \text{and} \quad \mathsf{Interm}_\Delta(R) \subseteq S, \tag{2}$$

where Interm_Δ is defined by $\mathsf{Interm}_\Delta(X) = \bigcup_{i\in I} \mathsf{Interm}_{\pi_i}(X \cap W_i)$ for all $X \subseteq R$. In other terms, it corresponds to a controller enforcing that the global system never leaves the safety zone S, and repeatedly visits the target zone R.

The decomposition method has been implemented in the tool MINIMATOR [21]. This tool makes use of zonotopes [17], and has been written in Octave [22]. At the top-level, the procedure `Decomposition` recursively bisects the target set R until, for each tile, a pattern has been found. It calls the procedure `Find_Pattern`, which implements the search for a correct pattern for a given tile W of the current decomposition. Upon success, the tool constructs a successful decomposition Δ. The tool has been used on various case studies [11, 13].

▶ **Example 1.** In this paper, we consider as a running example a two-room house equipped with a heating system (a more refined example is described in Section 4). There are heat exchanges between the rooms (characterized by parameters $\alpha_{i,j}$), and heat losses to the outside (with parameters $\alpha_{e,i}$), as schematically depicted on Fig. 1.

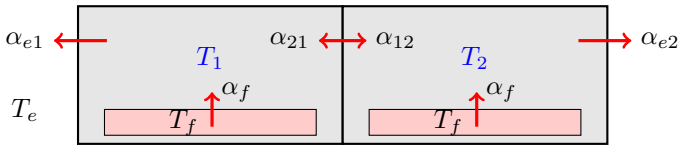


Figure 1 Two-room heating system

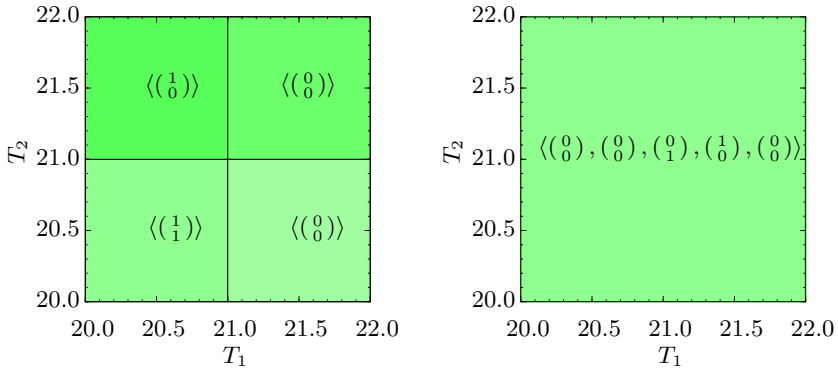


Figure 2 Two valid global controllers for Example 1

Each heater has two modes (`on` and `off`). When turned on, a heater immediately gets hot (to temperature T_f) and heats the room with heat transfer coefficient α_f. The system has two variables T_1 and T_2, which correspond to the temperatures in rooms 1 and 2. Writing $X = (T_1; T_2)^T$ for the global state of the system, and writing u_1 and u_2 for the modes of the corresponding heaters (assuming u_1 and u_2 belong to $\{0, 1\}$), we get the following equation[1] representing the evolution of the temperatures:

$$\dot{X} = f_{\binom{u_1}{u_2}}(X) = \begin{pmatrix} -\alpha_{e1} - \alpha_{21} & \alpha_{21} \\ \alpha_{12} & -\alpha_{e2} - \alpha_{12} \end{pmatrix} \cdot X + \begin{pmatrix} u_1\alpha_f T_f + \alpha_{e1} T_e \\ u_2\alpha_f T_f + \alpha_{e2} T_e \end{pmatrix}$$

The objective of our controller is to try to maintain the temperatures in both rooms within a comfort zone $R = [20; 22] \times [20; 22]$, and to ensure that both values never leave the safety zone $S = [19; 23] \times [19; 23]$. Using MINIMATOR, we are able to compute a correct controller for this problem, as depicted in Fig. 2. In fact, the two controllers in the figure represent different trade-offs: The left controller has four different tiles with patterns of length 1, while the controller on the right hand side uses a single uniform pattern of length 5.

2.2 Game-based controller synthesis

Games provide another approach to controller synthesis: in that setting, the controller is seen as one protagonist, playing against other components of the system. A strategy for a player in such a game dictates how the corresponding component must behave, and her winning condition represents the conditions under which the component is said to behave properly.

There is a huge literature on game-based techniques for synthesis [26, 2]. A very large part of these work considers *two-player zero-sum games*: zero-sum games are purely antagonist games, where the objectives of the two players are opposite. This setting corresponds to

[1] For this example, we use $T_e = 10$, $T_f = 50$, $\alpha_{e1} = 0.005$, $\alpha_{e2} = 0.0033$, $\alpha_f = 0.0083$, $\alpha_{12} = \alpha_{21} = 0.05$ and $\tau = 5$.

worst-case controller synthesis: the controller must behave correctly whatever the other players do. Winning strategies then correspond to correct controllers, ensuring correct behavior against any behavior of the environment. In various settings, notably for finite-state systems and ω-regular winning conditions, winning strategies can be computed.

Non-zero-sum games have been considered less intensively: in non-zero-sum games, the objectives are not opposite, and the players may then be interested in cooperating. Such games can be used to reason about the synthesis of *distributed systems*, where several components have their own objectives. In this setting however, winning strategies need not exist (and they would not take into account the possible cooperation between components). Instead, various notions of equilibria can be defined and studied, including the most famous *Nash equilibrium*. Several results exist in this setting; for instance, the existence of pure-strategy Nash equilibria is in general decidable in finite-state games with ω-regular objectives [5], but their existence is undecidable when considering randomized strategies.

An interesting feature of game-based controller synthesis is the ability to take *partial observability* into account. Indeed, in most applications (and especially for distributed control), the components are not able to observe the exact state of the whole system. In the game-based model, this can be taken into account in the definition of *strategies*, requiring them to return the same action for any two situations that are observationally equivalent.

Partial observation can be dealt with when considering zero-sum games with ω-regular objectives [24, 9], but it makes the problem undecidable in more complex settings [10, 4].

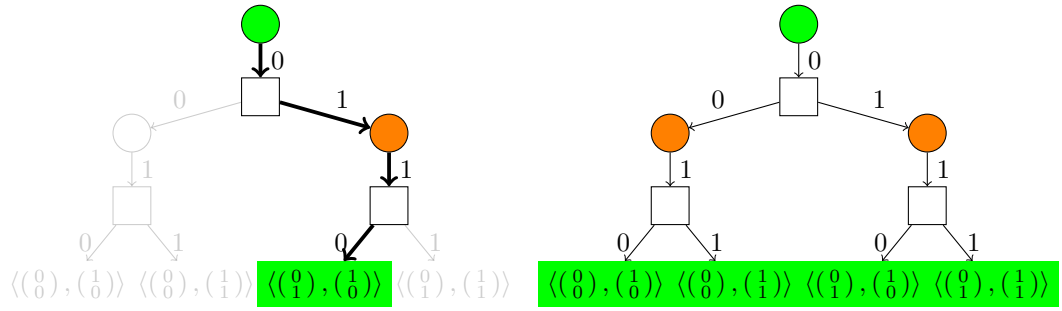
3 Distributed control of sampled switched systems

The invariant-based approach to controller synthesis (depicted in Section 2.1) generates a *centralized* controller, that is, a unique global strategy for the whole system, selecting a global mode u at each time interval τ. This approach assumes *full control* and *full observability* of the whole system. This is due to the structure of the synthesized control algorithm, where the mode switching depends on the local tile to which the system state belongs. Furthermore, we assume that at any switching instant, the controller can choose an arbitrary u from the set of modes U. In many applications, these assumptions are not realistic or would result in an overly complex communication and control infrastructure.

In our running example of a heating system, the controller computed in Example 1 using MINIMATOR selects the mode of the thermostats in both rooms, based on the global state of the system. Such a centralized controller might be *fragile*, in the sense that it is only (or at least was only proven) correct in the case where both thermostats obey the controller strategy. If for some reason the mode selected in one of the rooms is not the expected one (imprecision of the temperature sensor) or is not applied correctly (failure of some component), we have no guarantee about the behavior of the rest of the system.

We address these problems by combining the invariant analysis implemented in MINIMATOR with a game-based view, in order to generate (whenever possible) individual controllers that are correct even if the other components of the system do not behave as expected (but still achieve their objectives[2]), hence adding *robustness* to the whole system. This approach still assumes full observation of the system, and requires some technical restrictions that we explain below. We will then propose a second approach, which assumes partial observation of the system, ending up with robust and fully distributed controllers. Notice that both approaches

[2] Notice that if we do not require the other components to meet their obligations, there is no way of ending up in the target set R.



(a) Single global pattern **(b)** Individual pattern correct in any completion

Figure 3 Finding successful patterns

mainly amount to modifying the notion of being successful for a tile of the decomposition.

Figure 3 illustrates the difference between our approach and the classical approach of MINIMATOR. The left-hand part of the figure represents the behavior of MINIMATOR: it looks for a pattern (of length 2 in this example) that maps the current set of the decomposition into R (represented in green), and such that at all intermediary steps remain in S (in orange). In this example, a valid pattern is $\langle\binom{0}{1},\binom{1}{0}\rangle$, since the corresponding branch ends up in a green state (representing R) and visits an orange state (corresponding to S) at the intermediate position.

The right-hand part depicts what our algorithm does for checking if pattern $\pi = \langle 0, 1\rangle$ is correct: it has to check that, for any completions of the pattern π with modes for the other components, the image of the original set by this completion is in R, and that it is in S at all intermediary steps. The pattern $\langle 0, 1\rangle$ is then a correct pattern here, as all the completions lead to green configurations (meaning that the taregt zone of the considered player is reached), while all intermediary configurations are orange (corresponding to states in the safety set of the considered player).

3.1 Problem Statement

Consider a sampled switched system as defined in Section 2.1. Distributed control of such a system involving m agents is based on m sets of local modes U_p, with $1 \leq p \leq m$, which are related to the global modes U by means of a function $\gamma\colon U_1 \times \ldots \times U_m \to U$. In its most simple form, this setting can be implemented by considering that $U = \prod_{1\leq p\leq m} U_i$, and that $\gamma(u_1, ..., u_m) = (u_1, ..., u_m)$. Since each of the agents has only limited control over the system's behavior, we define local objectives that need to be fulfilled. In this work, we only consider projections on sub-spaces of the problem domain. For this purpose, we assume that the set Var of variables is divided among the players: $\mathsf{Var} = \bigcup_{1\leq p\leq m} G_p$. We write Γ_p for the set $\mathbb{R}^{G_p}$ of valuations of the variables of agent p. We denote the projection of X on the dimensions in Γ_p by $X \downarrow \Gamma_p$. Each agent p then has to take care of the variables in their set G_p, maintaining them in $S \downarrow \Gamma_p$ and visiting $R \downarrow \Gamma_p$ infinitely often. Notice that since R is a rectangular set, and since for each variable in Var is in some G_p, the following holds: whenever a set X satisfies $(X \downarrow \Gamma_p) \in (R \downarrow \Gamma_p)$ for all $1 \leq p \leq m$, then $X \subseteq R$. The same holds of S.

3.2 Robust Local Control with Global Observation

In this first approach, the *control* will be localized, while *each controller still has the ability to measure the global system state.* A straightforward solution would be the simple decomposition of a standard controller: this boils down to synthesizing a global controller using the approach

of [12], which gives a decomposition and global patterns for each tile of this decomposition. Then each pattern can be projected into m local patterns. Since all agents are acting simultaneously, this results again in a valid controller.

As already explained, this results in a very fragile solution: each controller depends on all other agents in the system. If one agent deviates from this strategy, no guarantees can be made on the global and local objectives (even if the new strategy of the deviating agent is a valid one). A central goal in the synthesis of distributed control is *robustness*: each agent should be able to enforce its own local objective, regardless (to a certain extent) of what the other agents are doing. Thus, we assume that each agent has no knowledge of the strategies of the other agents.

In order to test if a local pattern π is robust, we need to take into account *any possible behavior* of all other agents. For this purpose, the notion of *completion* of a local pattern is used:

▶ **Definition 1.** Given a player p and a local mode $u \in U_p$, the completion of u is defined as

$$\mathsf{Compl}_p(u) = \{w \in U \mid \exists u_1, ..., u_m \text{ s.t. } u_i \in U_i,\ u_p = u, \text{ and } w = \gamma(u_1, \ldots, u_m)\}$$

This notion can easily be extended to patterns. Given a local pattern $\pi \in U_p^+$ with $|\pi| = k$, it is completed to the set of global patterns

$$\mathsf{Compl}_p(\pi) = \{\phi \in U^k \mid \forall 1 \leq j \leq k.\ \phi_j \in \mathsf{Compl}_p(\pi_j)\}.$$

In fact, computing the completion of a local pattern naturally corresponds to exploring a game tree, as explained in the previous section. The leaf nodes of the tree in Fig. 3b correspond to the completion of the local pattern $\langle 0, 1\rangle$ of Player 1. Now, using this definition, we can state what it means for a local pattern π to be robust for a tile W:

▶ **Definition 2.** Player p has a robust strategy for a tile $W \subseteq R$ if there exists a pattern $\pi \in U_p^+$ such that, for all global patterns $\psi \in \mathsf{Compl}_p(\pi)$, the following holds:

1. $\mathsf{Post}_\psi(W) \downarrow \Gamma_p \subseteq R \downarrow \Gamma_p$,
2. $\mathsf{Interm}_\psi(W) \downarrow \Gamma_p \subseteq S \downarrow \Gamma_p$.

We then say that a tile W is successful if all the agents have a robust strategy for W.

A procedure to compute a robust pattern is shown in Algorithm 1. It can be used to compute a distributed control of a sampled switched system based on the basic procedure from [12]. In Algorithm 1, the outer loop searches for a tuple of patterns of *uniform length* ℓ. It does so by enumerating all valid local patterns and checking them for robustness. The procedure terminates successfully if for all agents, a robust pattern of some uniform length ℓ has been found. In order to find a successful decomposition, Algorithm 1 is embedded in the top-level procedure `Decomposition`, which upon success returns a decomposition $(W_i)_{i\in I}$ of R and, for each $i \in I$ and each $1 \leq p \leq m$, a pattern π_i^p. Due to space limitations, this procedure is not shown here, and we refer to [12] for more details.

The fact that we are searching for patterns with uniform length needs to be explained. If we consider the resulting distributed control system in action, then each agent will behave as follows: at some time instant t, it will measure the global system state $X \in R$. According to its local control table and to the tile containing X, each agent will select a pattern of some length ℓ. After $\ell \cdot \tau$ time units, this procedure will be repeated. Now, due to the robustness property of each local control, it is guaranteed that the resulting *global system state* at $t + \ell \cdot \tau$ will again be in R, hence in some tile W of the decomposition, and thus each agent will find a suitable entry in its control table. However, consider a situation where some agent p would play a pattern π of length $\ell' < \ell$ (say). Then, after $\ell' \cdot \tau$ time units, when agent p

Algorithm 1: `Find_Pattern_Simul`$(W, R, S, K, m, (\Gamma_p)_p)$

Input: Sets W,R,S; maximum length K; number of players m; sub-spaces $(\Gamma_p)_p$
Output: Patterns $(\pi_1, \ldots, \pi_m)$ of uniform length, where each pattern π_p robustly satisfies the objective of agent p, or $\bot$ if no such patterns exist

for $\ell = 1 \ldots K$ **do**
  for $p = 1 \ldots m$ **do**
    $\Pi := U_p^\ell$; `// set of local patterns of length` ℓ
    $\pi_p := \bot$;
    for $\pi \in \Pi$ **do**
      $\Psi := \mathsf{Compl}_p(\pi)$; `// set of global completions of` π
      for $\psi \in \Psi$ **do**
        if $\mathit{Post}_\psi(W) \downarrow \Gamma_p \not\subseteq R \downarrow \Gamma_p$ **then next** π;
        ;
        if $\mathit{Interm}_\psi(W) \downarrow \Gamma_p \not\subseteq S \downarrow \Gamma_p$ **then next** π;
        ;
      $\pi_p := \pi$; **break**; `// valid pattern for agent` p
    if $\pi_p = \bot$ **then next** ℓ ; `// no pattern of length` ℓ `for agent` p
    ;
  return $(\pi_1, \ldots, \pi_m)$;
return $\bot$;

measures the system state again in order to find the next pattern to play, it may happen that $X \in S \setminus R$, since the other agents have not finished applying their patterns. Thus, agent p would not be able to choose a new pattern, because the control is limited to R.

Overall, Algorithm 1 computes local patterns that are successful in the sense of Def. 2. Regarding the global controller obtained by this approach, we can definitely assert that it is correct (in the sense of Equation (2)): indeed, the individual patterns computed by our algorithm above can be combined (as they have the same length), and the global pattern obviously satisfies Equation (1), by construction. Given our construction, we would like to assert that each individual pattern is correct against any deviation of the other agents. However, for the same reasons as above, this is only correct w.r.t deviations that use the same pattern lengths, and as long as they achieve their objectives:

▶ **Proposition 3.** Assume that our procedure returns a successful decomposition $\Delta = (W_i, (\pi_i^p)_{1 \le p \le m})_{i \in I}$ of R. Then

- for any agent p, for any $i \in I$, any $X \in W_i$, and any completed pattern $\phi \in \mathsf{Compl}_p(\pi_i^p)$, it holds

$$\mathsf{Post}_\phi(X) \downarrow \Gamma_p \in R \downarrow \Gamma_p \qquad \mathsf{Interm}_\phi(X) \downarrow \Gamma_p \subseteq S \downarrow \Gamma_p$$

- for any $i \in I$ and any $X \in W_i$, for the pattern ϕ obtained by combining the patterns π_i^p of all the agents (which is a valid completion of each individual patterns, as they all have the same length), it holds

$$\mathsf{Post}_\phi(X) \in R \qquad \mathsf{Interm}_\phi(X) \subseteq S.$$

- Finally,

$$\mathsf{Post}_\Delta(R) \subseteq R \qquad \mathsf{Interm}_\Delta(R) \subseteq S.$$

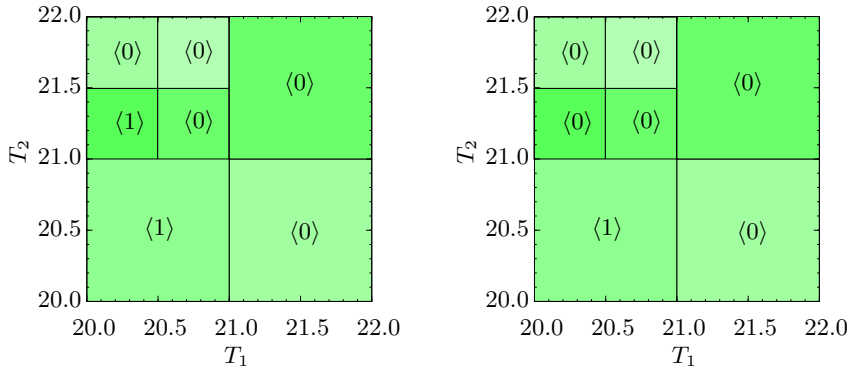


Figure 4 Distributed robust controllers for Example 1 for Players 1 (left) and 2 (right)

▶ **Example 1** (Contd). Consider again the heated rooms from Example 1. A distributed control could be synthesized by separating the problem into a two-player game: the first player controls variable u_1, and has $G_1 = \{T_1\}$; the second player controls u_2, and his objective is on $G_2 = \{T_2\}$. Possible controllers computed by our algorithm are shown in Fig. 4.

3.3 Local Observation

We now extend the above approach to local observation. We assume that each agent can only observe a dedicated sub-space, and can make his decisions only on the basis of this local observations. Compared to the previous approach, this will have two advantages:

- each agent measures only a sub-space, which is more realistic in many cases and allows for simpler (low-dimensional) controllers;
- as a side effect, the patterns can now be desynchronized (and have different lengths), which allows for more admissible controllers.

Some changes are necessary with respect to Algorithm 1. First of all, the local observations allow us to decouple the computation of valid patterns and the decomposition of the target set R: instead of one common decomposition for all m agents, the procedure will compute, for each agent, a successful decomposition of its observed space. In each run, the target set R will only be decomposed along the observed dimensions, according to the following definition:

▶ **Definition 4.** Given a variable $w \in \mathsf{Var}$ and a rectangular set $R \subseteq \mathbb{R}^{\mathsf{Var}}$, writing $R(v) = [a_v, b_v]$ for all $v \in \mathsf{Var}$, the *split of R along variable w* is the set $\mathsf{Split}_{\mathbb{R}^w}(R) = \{R_{\text{left}}, R_{\text{right}}\}$, where

$$R_{\text{left}}(v) = \begin{cases} [a_v, b_v] & \text{if } v \neq w \\ [a_v, \frac{a_v+b_v}{2}] & \text{otherwise} \end{cases} \qquad R_{\text{right}}(v) = \begin{cases} [a_v, b_v] & \text{if } v \neq w \\ [\frac{a_v+b_v}{2}, b_v] & \text{otherwise} \end{cases}$$

This notion is easily extended to a set of variables $V \subseteq \mathsf{Var}$, resulting in a set $\mathsf{Split}_{\mathbb{R}^V}(R)$ of $2^{|V|}$ boxes covering R.

For each agent p, we define its set $O_p \subseteq \mathsf{Var}$ of *observed* variables. ***We require that the set O_p of observed variables be included in his set G_p of variables defining his objective***: for all $1 \leq p \leq m$, we must have $O_p \subseteq G_p$. This condition is needed for the correctness of our procedure: as we explain below, this is precisely the condition that allows us to drop the uniform-length requirement. We write Ω_p for the set $\mathbb{R}^{O_p}$. We also

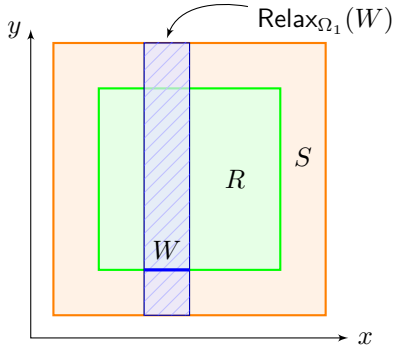


Figure 5 The set $\mathsf{Relax}_{\Omega_1}(W)$ (hatched area), where agent 1 observes only variable x

write $\Omega_{\bar{p}} = \mathbb{R}^{\mathsf{Var}\setminus O_p}$. As previously, we write $X \downarrow \Omega_p$ for the projection of the set X on Ω_p. Our algorithm will precisely try to compute a successful decomposition of $R \downarrow \Omega_p$. In order to reconstruct the set of possible states that correspond to a given observation, we define the converse of the projection on Ω_p, as follows:

▶ **Definition 5.** Consider a tile $W \subseteq R \downarrow \Omega_p$ observed by an agent p. Its relaxation is the set

$$\mathsf{Relax}_{\Omega_p}(W) = W \times (S \downarrow \Omega_{\bar{p}}).$$

The above definition is visualized in Fig. 5. By relaxing all non-observed dimensions to the invariant set S, we guarantee that the local controller can start a new pattern even if one or several of the other agents have not finished their current pattern (provided their patterns enforce their safety constraints). With these modifications, the patterns that we are looking for can be characterized as follows:

▶ **Definition 6.** Agent p has a strongly-robust strategy for a tile $W \subseteq R \downarrow \Omega_p$ if there exists a pattern $\pi \in U_p^+$ such that, for all global patterns $\psi \in \mathsf{Compl}_p(\pi)$, the following holds:

1. $\mathsf{Post}_\psi(\mathsf{Relax}_{\Omega_p}(W)) \downarrow \Gamma_p \subseteq R \downarrow \Gamma_p$,
2. $\mathsf{Interm}_\psi(\mathsf{Relax}_{\Omega_p}(W)) \downarrow \Gamma_p \subseteq S \downarrow \Gamma_p$.

In this setting, we say that a tile W is successful if all the agents have a strongly robust strategy for W.

The procedure for finding a strongly-robust pattern for some tile W and agent p is shown in Algorithm 2. The top-level procedure, which computes a successful decomposition of a tile W for some agent p, is shown in Algorithm 3. It tries to find a pattern for the whole tile W by calling `Find_Pattern_Local`. If no such pattern can be found, it recurses by splitting the tile W wrt. the observed dimensions. When invoked at some level D of decomposition, the next finer decomposition is called with level $D-1$, and the recursion stops as soon as the finest decomposition has been reached at $D = 0$ without finding a valid pattern. Computing local controllers for m agents boils down to calling $\texttt{Decomposition}(R, R, S, K, D, p, \Gamma_p)$ for each agent $p \in \{1, ..., m\}$.

In comparison to the strategy described in the previous section, we can establish stronger guarantees for the overall system's robustness, while slightly relaxing the guarantees wrt. to the target set. Each agent p only measures variables in O_p, while guaranteeing the local objective that the system will return to the projection of R on the variables in G_p. Since $O_p \subseteq G_p$, the subsequent measure of the variables in O_p will be in $R \downarrow \Omega_p$. The only assumption on the other dimensions is their continuous containment inside S. Thus, the

Algorithm 2: `Find_Pattern_Local`$(W, R, S, K, p, \Gamma_p, \Omega_p)$

Input: Sets W,R,S; maximum length K; agent p; sub-spaces Γ_p, Ω_p
Output: Pattern π robustly satisfying objective of agent p; $\perp$ if no such pattern exists

$W' := \mathsf{Relax}_{\Omega_p}(W, S)$
for $\ell = 1 \dots K$ **do**
  $\Pi := U_p^\ell$; `// set of local patterns of length` ℓ
  for $\pi \in \Pi$ **do**
    $\Psi := \mathsf{Compl}_p(\pi)$; `// set of global completions of` π
    for $\psi \in \Psi$ **do**
      if $\mathit{Post}_\psi(W') \downarrow \Gamma_p \not\subseteq R \downarrow \Gamma_p$ **then next** π;
      ;
      if $\mathit{Interm}_\psi(W') \downarrow \Gamma_p \not\subseteq S \downarrow \Gamma_p$ **then next** π;
      ;
    return π;
return $\perp$;

global control resulting from the cooperation of the local controllers will remain valid even if any of the controllers are replaced by any strategy that guarantees containment in S.

On the other hand, since the patterns of the distributed controllers can now be played in a decoupled manner, we can no longer guarantee that the global state will return to R. However, a slightly weaker property can be established for each infinite run of the global control system: for each player p, the local objective—the state viewed in the player's sub-space returns to the projection of R—will hold infinitely often.

▶ **Proposition 7.** Assume that our procedure returns successful decompositions $\Delta_p = (W_i^p, \pi_i^p)_{i \in I_p}$ of $R \downarrow \Omega_p$, for each agent p. Then

- for any agent p, for any $i \in I_p$, any $X \in \mathsf{Relax}_{\Omega_p}(W_i^p)$, and any completed pattern $\phi \in \mathsf{Compl}_p(\pi_i^p)$, it holds

$$\mathsf{Post}_\phi(X) \downarrow \Gamma_p \in R \downarrow \Gamma_p \qquad\qquad \mathsf{Interm}_\phi(X) \downarrow \Gamma_p \subseteq S \downarrow \Gamma_p$$

- fix an agent p, an index $i \in I_p$, and some state $X \in \mathsf{Relax}_{\Omega_p}(W_{i_p}^p)$. We define the tree $\mathcal{T}_{p,X}$ inductively as follows:
  - its root is labelled with X, and with the (non-empty) pattern π_i^p;
  - pick a node n with no descendant in the currently-constructed tree; assume that it is labelled with some state Y, and with some non-empty pattern $\rho = u \cdot \rho'$, where u is the first mode of ρ. We then extend the tree by adding sons to n as follows: for each completion w of u, we add a son m_w. We label m_w with $Z = \mathsf{Post}_w(Y)$. We also label it with a pattern, selected as follows:
    * if $Z \notin S$, we label m_w with the empty pattern ϵ;
    * if $Z \in S$ and ρ' is not empty, m_w is labelled with ρ';
    * if $Z \in S$ and ρ' is empty, and if $Z \downarrow \Omega_p \subseteq W_j^p$ for some $j \in I_p$, then we label m_w with π_j^p;
    * finally, if $Z \in S$ and ρ' is empty, but $Z \downarrow \Omega_p \not\subseteq R \downarrow \Omega_p$, m_w is labelled with the empty pattern ϵ.

  We claim that this tree is infinite (i.e., it contains infinite branches), and any infinite branch visits only states in S, and it visits $R \downarrow \Gamma_p$ infinitely many times.

Algorithm 3: `Decomposition`$(W, R, S, K, D, p, \Gamma_p, \Omega_p)$

Input: Sets W,R,S; maximum length K; depth D; agent p; sub-spaces Γ_p, Ω_p
Output: Successful decomposition Δ, or $\bot$ if no such decomposition exists

```
π := Find_Pattern_Local(W, R, S, K, p, Γ_p, Ω_p);
if π ≠ ⊥ then
    return (W, π);
else
    if D = 0 then
        return ⊥;                                  // finest decomposition reached
    else
        Dec := Split_{Γ_p}(W); Δ := ∅;             // decompose and recursive call
        for V ∈ Dec do
            Δ_V = Decomposition(W, R, S, K, D − 1, p, Γ_p, Ω_p);
            if Δ_V = ⊥ then return ⊥ ;
            else Δ := Δ ∪ Δ_V;
            ;
        return Δ;
```

Proof. The first claim is straightforward. The second claim can be proven by noticing that when ρ' becomes empty, agent p has completed his pattern, and provided that the other agents have maintained their variables in their safety sets, the corresponding state Z is in S and is such that $Z \downarrow \Gamma_p \subseteq R \downarrow \Gamma_p$. Since $O_p \subseteq G_p$, it follows[3] that $Z \downarrow \Omega_p \subseteq R \downarrow \Omega_p$, so that the node m_w will be labelled with a non-empty pattern, and the construction can continue. ◀

In the end, let $\Delta = (W_i, (\pi_i^p)_{1 \leq p \leq m})_{i \in I}$ be the decomposition obtained by merging the individual decompositions $(\Delta_p)_{1 \leq p \leq m}$. From Prop. 7, we deduce that if all the agents follow the strategy given by decomposition Δ, then the outcome from any state X will be infinite, it will visit only safe states in S, and each individual target set $R \downarrow \Gamma_p$ will be visited infinitely many times. Again notice that since patterns may have incompatible lengths, we cannot ensure that R itself is visited infinitely many times.

▶ **Example 1** (Contd). We consider our example of the heating system in this setting of local observation, with $O_p = G_p$ for $1 \leq p \leq 2$. The controllers obtained in this setting are depicted on Fig. 6.

4 A more realistic case study

The distributed local controllers obtained for our running example (Fig 6) are very simple: following the natural intuition, their strategy amounts to turning the heater on when the temperature is too low, using only patterns of length one. The only interesting information we get is the exact temperature at which we should turn the heater on or off. In this section, we develop this example a bit further, by assuming that the heaters are reacting slowly.

[3] If some variable v were in O_p but not in G_p (then it would be in $G_{p'}$ for another agent p'), we would need that agents p and p' play patterns of the same length, in order to ensure $Z \downarrow \Omega_p \subseteq R \downarrow \Omega_p$ when agent p terminates his pattern.

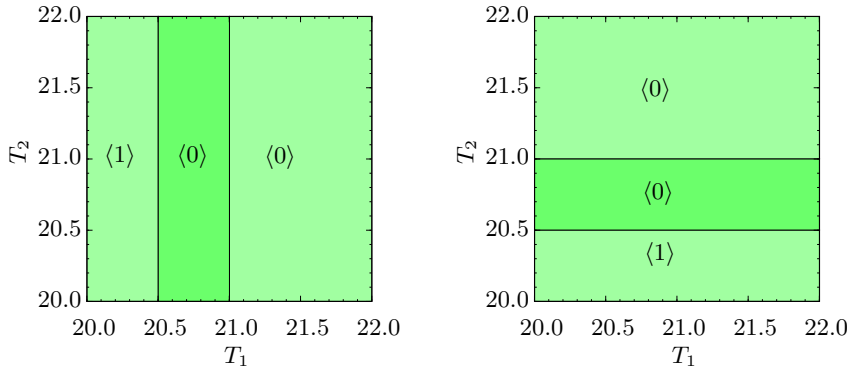


Figure 6 Local controllers for Example 1 for Players 1 (left) and 2 (right)

▶ **Example 2.** Consider a water underfloor heating system: hot water circulates in pipes under the floor, and the controller can open or close the valves. Hot water will first heat the floor, which then in turn transfers heat to the room. The heaters will start to heat up to temperature T_f when switched on. The state $X = (H_1, T_1, H_2, T_2)^T$ of this model is formed by the temperatures of the two rooms (T_1, T_2) and the heaters (H_1, H_2). The dynamics of the model can be described[4] by the equation $\dot{X} = A_{\binom{u_1}{u_2}} X + B_{\binom{u_1}{u_2}}$ with

$$A_{\binom{u_1}{u_2}} = \begin{pmatrix} -\beta_1 - u_1\alpha_f & \beta_1 & 0 & 0 \\ \gamma_1 & -\alpha_{e1} - \gamma_1 - \alpha_{21} & 0 & \alpha_{21} \\ 0 & 0 & -\beta_2 - u_2\alpha_f & \beta_2 \\ 0 & \alpha_{12} & \gamma_2 & -\alpha_{e2} - \gamma_2 - \alpha_{12} \end{pmatrix} \quad B_{\binom{u_1}{u_2}} = \begin{pmatrix} u_1\alpha_f T_f \\ \alpha_{e1} T_e \\ u_2\alpha_f T_f \\ \alpha_{e2} T_e \end{pmatrix}$$

where $u_1, u_2 \in \{0, 1\}$ indicate the state of the heaters ($0 =$ off, $1 =$ on). By discretization with sample time τ, we obtain a switched system $X(t + \tau) = \hat{A}_u \cdot X(t) + \hat{B}_u$.

The global objective of a controller is to keep both rooms at a temperature between 20° and 22°, and the heaters in the comfort zone between 20° and 30°. There are safety margins for the room temperature of 1°. The heaters should not be colder that 15° and should not exceed the maximum of 40°. In other words, the target set is given by $R = ([20, 30] \times [20, 22])^2$, while the safety set is given by $S = ([15, 40] \times [19, 23])^2$. Obviously, $R \subseteq S$.

In order to construct a distributed control for the two rooms, the global state space is projected to the respective dimensions $G_1 = O_1 = \{H_1, T_1\}$ for the first room and $G_2 = O_2 = \{H_2, T_2\}$ for the second room. For all experiments, we used a maximum pattern length of $K = 6$ and a maximum decomposition depth of $D = 3$.

The original implementation of MINIMATOR computes a global controller with a decomposition into 16 tiles (corresponding to a single split in all four dimensions) and patterns of up to three steps. The computation time is 10.45 s, where most of the time is spent on the (failing) attempt to find a single pattern for the whole target set. The approach described in Section 3.2 results in two controllers of similar complexity: 16 tiles with pattern length up to 3. The computation time is slightly higher (13.43 s) due to the more complex exploration of the completed local patterns. Finally, using the approach based on local observations described in Section 3.3, we obtain two simple controllers, each with four (2-dimensional) tiles, as shown in Fig. 8. The computation time was 27.75 s.

[4] For this example, we use the following parameters: $T_e = 10$, $T_f = 40$, $\alpha_{e1} = 0.005$, $\alpha_{e2} = 0.0033$, $\alpha_f = 0.12$, $\alpha_{12} = \alpha_{21} = 0.006$, $\beta_1 = \beta_2 = 0.083$, $\gamma_1 = \gamma_2 = 0.0083$, and $\tau = 5$.

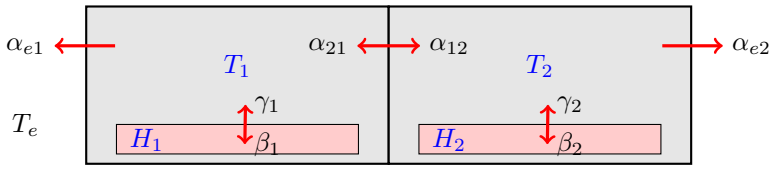


Figure 7 Two-room water underfloor heating system

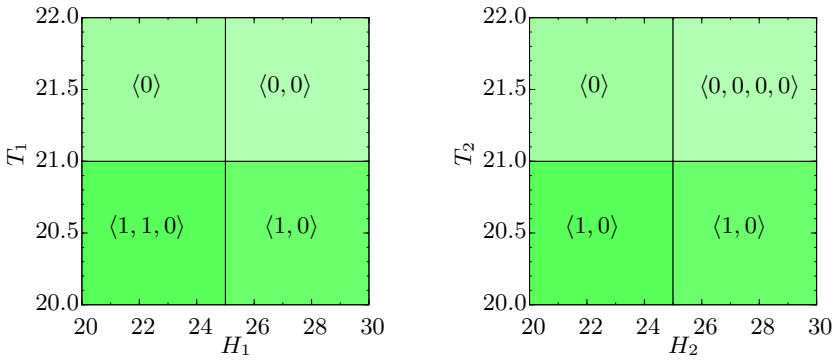


Figure 8 Local controllers for 4-dimensional case study (left: Player 1; right: Player 2)

5 Conclusion

We proposed an extension of the decomposition method for the control of sampled switched systems. The improvement is two-fold: first, we synthesize robust, distributed controllers, which are able to cope with changes in the behaviors of the other controllers of the system; second, our approach can deal with partially observable systems, where controllers may only observe (and base their decisions on) part of the system. We illustrated our approach on two examples of heating systems, for which we were able to effectively synthesize individual controllers, each having partial observation of the whole system.

This works opens many directions for future research: following classical results in game theory, it would be natural to make the controller *reconstruct information* about the global state of the system from the evolution of the variables it can observe. This would require that we introduce *memory* in our strategies, and seems to be more than a simple extension of our current approach. Another relevant direction would be to try to use *the same controller* (with partial observation) in all the rooms. This would preserve the partial-observation part of our present approach, but would drop the robustness aspect as the controllers would certainly make use of the fact that all components follow the same strategy. Finally, we would like to extend our approach with *costs*, in order to look for cheap controllers. Notice that just considering the cheapest pattern would only optimize "locally", while it might be more profitable to take a more expensive pattern in order to reach a zone from which control might be cheaper.

Acknowledgments This work has been supported by French network DIGITEO, FP7 project Cassting (FP7-ICT-601148), and iCODE Institute project (ANR-11-IDEX-0003-02).

References

1 R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

2 K. Apt and E. Grädel. *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, 2011.

3 E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective Synthesis of Switching Controllers for Linear Systems. *Proc. of the IEEE*, 88(7):1011–1025, 2000.

4 P. Bouyer, N. Markey, and S. Vester. Nash equilibria in symmetric games with partial observation. In *SR'14*, EPTCS 146, p. 49–55, Grenoble, France, 2014.

5 R. Brenguier. *Nash equilibria in concurrent games – Applications to timed games*. Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, 2012.

6 R. W. Brockett. Asymptotic stability and feedback stabilization. *Differential geometric control theory*, 27:181–191, 1983.

7 F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR'05*, LNCS 3653, p. 66–80. Springer, 2005.

8 F. Cassez, J. J. Jessen, K. G. Larsen, J. Raskin, and P. Reynier. Automatic synthesis of robust and optimal controllers - an industrial case study. In *HSCC'09*, LNCS 5469, p. 90–104. Springer, 2009.

9 K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Algorithms for omega-regular games with imperfect information. In *CSL'06*, LNCS 4207, p. 287–302. Springer, 2006.

10 C. Dima and F. L. Ţiplea. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. Research Report 1102.4225, arXiv, 2011.

11 G. Feld, L. Fribourg, D. Labrousse, B. Revol, and R. Soulat. Correct-by-design control synthesis for multilevel converters using state space decomposition. In *FSFMA'14*, EPTCS 156, p. 5–16, 2014.

12 L. Fribourg, U. Kühne, and R. Soulat. Finite controlled invariants for sampled switched systems. *Formal Methods in System Design*, 45(3):303–329, 2014.

13 L. Fribourg and R. Soulat. *Control of Switching Systems by Invariance Analysis: Application to Power Electronics*. Wiley-ISTE, 2013. 144 pages.

14 A. Girard. Synthesis using approximately bisimilar abstractions: state-feedback controllers for safety specifications. In *HSCC'10*, p. 111–120. ACM Press, 2010.

15 A. Girard, G. Pola, and P. Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Trans. on Automatic Control*, 55:116–126, 2010.

16 T. A. Henzinger. The theory of hybrid automata. In *LICS '96*, p. 278–292. IEEE, 1996.

17 W. Kühn. Zonotope dynamics in numerical quality control. In *Mathematical Visualization*, p. 125–134. Springer, 1998.

18 D. Liberzon and A. S. Morse. Basic problems in stability and design of switched systems. *IEEE Control Systems Magazine*, 19:59–70, 1999.

19 X. Liu and S. A. Smolka. Simple linear-time algorithms for minimal fixed points (extended abstract). In *ICALP'98*, LNCS 1443, p. 53–66, London, UK, 1998. Springer.

20 J. Lygeros, D. N. Godbole, and S. Sastry. Multiagent hybrid system design using game theory and optimal control. In *CDC'96*, p. 1190–1195, 1996.

21 Minimator Web page. `https://bitbucket.org/ukuehne/minimator/wiki/Home`.

22 Octave Web page. `http://www.gnu.org/software/octave/`.

23 P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proc. of the IEEE*, 77(1):81–98, 1989.

24 J. H. Reif. The complexity of two-player games of incomplete information. *J. Computer and System Sciences*, 29(2):274–301, 1984.

25 P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer Publishing Company, Incorporated, 1st edition, 2009.

26 W. Thomas. On the synthesis of strategies in infinite games. In *STACS'95*, LNCS 900, p. 1–13. Springer, 1995.

27 C. J. Tomlin, J. Lygeros, and S. S. Sastry. A game theoretic approach to controller design for hybrid systems. *Proc. IEEE*, 88:949–970, 2000.