Studies in Systems, Decision and Control 89

Calin Belta
Boyan Yordanov
Ebru Aydin Gol

Formal Methods for Discrete-Time Dynamical Systems

Springer

Chapter 9
Temporal Logic Control

In Chaps. 7 and 8 we considered autonomous PWA systems. We showed that finite quotients of such systems can be computed and, based on this, developed methods for analysis and parameter synthesis from temporal logic specifications. Unlike the systems we discussed in previous chapters, which evolved autonomously, in this chapter we consider PWA control systems, which can be affected externally by applying a control signal. Then, it is possible to guarantee the satisfaction of a specification by trajectories of a PWA control system if an appropriate control signal is applied. More specifically, in this chapter we focus on fixed-parameter, PWA control systems and consider the following problem:

Problem 9.1 (*PWA Control*) Given a fixed-parameter PWA control system $\mathscr{W}$ (Definition 6.2) and an LTL formula ϕ over $L \cup \{\text{Out}\}$, find a control strategy such that all trajectories of the closed loop system satisfy ϕ.

Using Definition 6.6, Problem 9.1 becomes an LTL control problem, where we seek a feedback control strategy $(\mathbf{X}_0, \Omega)$ for the infinite, deterministic embedding transition system $T_{\mathscr{W}}$. In this chapter, we assume that the state of the system cannot be measured precisely or the applied inputs are corrupted by noise and, therefore, seek control strategies that are robust both with respect to measured state and applied input. As a result, the set of initial states $\mathbf{X}_0$ and control function Ω from the feedback control strategy $(\mathbf{X}_0, \Omega)$ (see Definition 5.1) are defined in terms of the initial regions of $\mathscr{W}$ and no state refinement is performed (such approaches will be considered in subsequent chapters). In Chap. 5 we discussed the problem of controlling a finite, possibly nondeterministic transition system from LTL specifications. To apply the methods presented there to Problem 9.1, we develop an approach based on the construction of a finite abstraction for $T_{\mathscr{W}}$, which we refer to as the control transition system T_c such that a control strategy generated for T_c using the algorithms from Chap. 5 can be adapted for $T_{\mathscr{W}}$.

More specifically, we construct T_c through a two step process. In the first step, by using the state equivalence relation induced by the polytopes from the definition of the PWA system, we construct a quotient $T_{\mathscr{W}}/_{\sim}$, which has finitely many states but an infinite set of inputs. This part of the procedure is similar to the methods we used

C. Belta et al., *Formal Methods for Discrete-Time Dynamical Systems*,

in Chaps. 7 and 8, with the exception that fixed-parameter, PWA control systems are considered, resulting in the construction of quotient transition systems with inputs. Thus, these results provide an extension of the methods for the construction of finite abstraction of autonomous PWA systems from Chap. 7 to a control framework. Once the quotient $T_{\mathscr{W}}/_{\sim}$ is constructed, we then define an equivalence relation in the control space, which leads to the construction of the finite control transition system T_c, which is suitable for the methods from Chap. 5. The solution to Problem 9.1 is obtained by implementing the control strategy for T_c as a feedback control automaton for the initial PWA system that reads the index of the region visited at each step and supplies the next input. As it will become clear later, our approach is robust in the sense that the closed loop system is guaranteed to satisfy the specification, even when state measurements and applied inputs are perturbed.

The remainder of this chapter is organized as follows. In Sect. 9.1 we define the control transition system T_c and outline an algorithm for its computation. In Sect. 9.2 we show how the methods from Chap. 5 are applied to T_c to formulate a solution to Problem 9.1. In Sect. 9.3 we discuss a strategy for reducing the conservatism of the overall method by characterizing the *stuttering behavior* (self transitions at a state of T_c that can be taken infinitely in T_c but do not correspond to real trajectories of $T_{\mathscr{W}}$) inherent in the construction of the control transition system. This approach is related to the characterization of transient regions and parameters from Chap. 8 as well as the well known Zeno behavior and addresses a major source of conservatism with the abstraction procedure from Chap. 7.

9.1 Control Abstraction

In this section, we define the finite control transition system $T_c = (X_c, \Sigma_c, \delta_c, O_c, o_c)$ for the (infinite) embedding $T_{\mathscr{W}} = (X_{\mathscr{W}}, \Sigma_{\mathscr{W}}, \delta_{\mathscr{W}}, O_{\mathscr{W}}, o_{\mathscr{W}})$ (Definition 6.6) and present an algorithm for its computation. In Sect. 9.2 we will show how the control methods from Chap. 5 are applied to T_c to solve Problem 9.1.

9.1.1 Definition

As in Chap. 7, the observation map $o_{\mathscr{W}}$ of $T_{\mathscr{W}}$ induces an observational equivalence relation $\sim$ over the set of states $X_{\mathscr{W}}$. However, the systems we considered before were autonomous and here, the quotient transition system $T_{\mathscr{W}}/_{\sim} = (X_{\mathscr{W}}/_{\sim}, \Sigma_{\mathscr{W}}, \delta_{\mathscr{W},\sim}, O_{\mathscr{W}}, o_{\mathscr{W},\sim})$ induced by $\sim$ has an infinite set of inputs $\Sigma_{\mathscr{W}} = \mathbf{U}$, which is preserved from $T_{\mathscr{W}}$. The set of transitions of $T_{\mathscr{W}}/_{\sim}$ is defined as $l' \in \delta_{\mathscr{W},\sim}(l, u)$ if and only if there exist $u \in \mathbf{U}$, $x \in \mathbf{X}_l$ and $x' \in \mathbf{X}'$ such that $x' = \delta_{\mathscr{W}}(x, u)$. Note that in general $T_{\mathscr{W}}/_{\sim}$ is nondeterministic, even though $T_{\mathscr{W}}$ is deterministic. Indeed, for a state of the quotient $l \in X_{\mathscr{W}}/_{\sim}$ it is possible that different states $x, x' \in \mathbf{X}_l$ have transitions in $T_{\mathscr{W}}$ to states from different equivalence classes under the same input. The set of observations $O_{\mathscr{W}} = L$ of $T_{\mathscr{W}}/_\sim$ is preserved from $T_{\mathscr{W}}$ and the observation map $o_{\mathscr{W},\sim}$ is identity.

The transition map $\delta_{\mathscr{W},\sim}$ can be related to the transitions of $T_{\mathscr{W}}$ by using the *Post* operator defined in Eq. (7.3):

$$\delta_{\mathscr{W},\sim}(l,u) = \{l' \in X_{\mathscr{W}}/_\sim \mid Post(\mathbf{X}_l, \{u\}) \cap \mathbf{X}_{l'} \neq \emptyset\}, \tag{9.1}$$

for all $l \in X_{\mathscr{W}}/_\sim$ and $u \in \Sigma_{\mathscr{W}}$. For each state $l \in X_{\mathscr{W}}/_\sim$, we define an equivalence relation $\approx_l$ over the set of inputs $\Sigma_{\mathscr{W}}$ as

$$(u_1, u_2) \in \approx_l \text{ iff } \delta_{\mathscr{W},\sim}(l, u_1) = \delta_{\mathscr{W},\sim}(l, u_2). \tag{9.2}$$

In other words, inputs u_1 and u_2 are equivalent at state l if they produce the same set of transitions in $T_{\mathscr{W}}/_\sim$. Let U_l^C, $l \in L$, $C \in 2^{X_{\mathscr{W}}/\sim}$ denote the equivalence classes of $\Sigma_{\mathscr{W}}$ in the partition induced by the equivalence relation $\approx_l$:

$$U_l^C = \{u \in \Sigma_{\mathscr{W}} \mid \delta_{\mathscr{W},\sim}(l,u) = C\} \tag{9.3}$$

Let $c(U_l^C)$ be an input in U_l^C such that

$$\forall u \in \Sigma_{\mathscr{W}}, d(c(U_l^C), u) < \varepsilon \Rightarrow u \in U_l^C, \tag{9.4}$$

where $d(u, u')$ denotes the distance between inputs $u, u' \in \Sigma_{\mathscr{W}}$ and ε is a predefined parameter specifying the robustness of the control strategy. As it will become clear in Sect. 9.1.2, $d(u, u')$ is the Euclidian distance in $\mathbb{R}^M$ and $c(U_l^C)$ can be computed as the center of a sphere inscribed in U_l^C.

Initially, the states of T_c are the observations of $T_{\mathscr{W}}$ (i.e., $X_c = L$). The set of inputs available at a state $l \in L$ is $\Sigma_c^l = \{c(U_l^C) \mid C \in 2^{X_{\mathscr{W}}/\sim}\}$ and the transition map is $\delta_c(l, c(U_l^C)) = C$. In general, it is possible that at a given state l, $\Sigma_c^l = \emptyset$, in which case state l is blocking. As it will become clear in Sect. 9.1.2, such states are removed from the system in a recursive procedure together with their incoming transitions and therefore $X_c \subseteq L$. The set of X_c observations and observation map of T_c are preserved from $T_{\mathscr{W}}/_\sim$, which completes the construction of the control transition system.

Following from the construction described so far, the control transition system T_c is a finite transition system and a control strategy for it can be generated using the methods presented in Chap. 5. In Sect. 9.2, we will show that a control strategy for T_c can be adapted as a robust control strategy (with respect to knowledge of exact state and applied input) for the infinite $T_{\mathscr{W}}$. This will allow us to use T_c as part of our solution to Problem 9.1.

9.1.2 Computation

Initially, the states of the control transition system T_c are simply the labels L of the polytopes from the definition of the PWA system (Definition 6.2). To complete its construction, we need to be able to compute the set of inputs Σ_c^l available at each state $l \in X_c$ and the transition map δ_c, while eliminating the states that are unreachable in order to guarantee that T_c remains non-blocking.

Given a polytope $\mathbf{X}_l$ from the definition of the PWA system, let

$$\Sigma^l = \{u \in \Sigma_{\mathscr{W}} \mid Post_{T_{\mathscr{W}}}(\mathbf{X}_l, u) \subseteq \mathbf{X}\} \tag{9.5}$$

be the set of all inputs guaranteeing that all states from $\mathbf{X}_l$ transit inside $\mathbf{X}$ (i.e., Σ^l is the set of all inputs allowed at l). In other words, regardless which $u \in \Sigma^l$ and $x \in \mathbf{X}_l$ are selected, x will transit inside $\mathbf{X}$ under u in $T_{\mathscr{W}}$. Then, in order to guarantee that $\mathbf{X}$ is an invariant for all trajectories of the system (an assumption that we made in the formulation of Problem 9.1) it is sufficient to restrict the set of inputs Σ_c^l available at each state $l \in X_c$ to $\Sigma_c^l \subseteq \Sigma^l$.

Proposition 9.1 *Let $\boldsymbol{X} = \{x \in \mathbb{R}^N \mid Hx < K\}$ be the H-representation of the polytope $\boldsymbol{X}$ from the definition of the PWA system (Definition 6.2). Then, Σ^l is a polytope with the following H-representation:*

$$\Sigma^l = \{u \in \boldsymbol{U} \mid \forall v \in V(\boldsymbol{X}_l), HB_l u < K - H(A_l v + c_l)\}, \tag{9.6}$$

where $V(\boldsymbol{X}_l)$ denotes the set of vertices of $\boldsymbol{X}_l$.

Proof Note that the set defined in Eq. (9.5) can be equivalently written as

$$\Sigma^l = \{u \in \mathbf{U} \mid \forall x \in \mathbf{X}_l, A_l x + B_l u + c_l \in \mathbf{X}\} \tag{9.7}$$

Let $u \in \mathbf{U}$ such that $\forall x \in \mathbf{X}_l.\ A_l x + B_l u + c_l \in \mathbf{X}$. Then,

$$\begin{aligned}\forall x \in \mathbf{X}_l.\ H(A_l x + B_l u + c_l) < K \Rightarrow \forall x \in \mathbf{X}_l.\ HB_l u < K - H(A_l x + c_l) \Rightarrow \\ \Rightarrow \forall v \in V(X_l).\ HB_l u < K - H(A_l v + c_l)\end{aligned}$$

Let $u \in \mathbf{U}$ such that $\forall v \in V(\mathbf{X}_l).\ HB_l u < K - H(A_l v + c_l)$. Then, $\forall v \in V(\mathbf{X}_l).\ A_l v + B_l u + c_l \in \mathbf{X}$. Let $m = |V(\mathbf{X}_l)|$, $x = \Sigma_{i=1}^m \lambda_i v_i$, where $v_i \in V(\mathbf{X}_l)$, $0 < \lambda_i < 1$ for all $i = 1, \ldots, m$ and $\Sigma_{i=1}^m \lambda_i = 1$. Then,

$$\begin{aligned}A_l x + B_l u + c_l = A_l \Sigma_{i=1}^m \lambda_i v_i + B_l u + c_l = \Sigma_{i=1}^m \lambda_i (A_l v_i + B_l u + c_l) \in \mathbf{X} \Rightarrow \\ \Rightarrow \forall x \in \mathbf{X}_l.\ A_l x + B_l u + c_l \in \mathbf{X}\end{aligned}$$

■

The set of states reachable from state l in $T_{\mathscr{W}}/_\sim$ under the allowed inputs is

$$Post_{T_{\mathscr{W}}/\sim}(l, \Sigma^l) = \{l' \in X_{\mathscr{W}}/\sim \mid Post_{T_{\mathscr{W}}}(\mathbf{X}_l, \Sigma^l) \cap \mathbf{X}_{l'} \neq \emptyset\} \tag{9.8}$$

and can be computed using polyhedral operations, since

$$Post_{T_{\mathscr{W}}}(\mathbf{X}_l, \Sigma^l) = A_l\mathbf{X}_l + B_l\Sigma^l + c_l. \tag{9.9}$$

Given a polytope $\mathbf{X}_l$ from the definition of the PWA system (Definition 6.2) and an arbitrary polytope $\mathbf{X}'$, let

$$U^{\mathbf{X}_l \to \mathbf{X}'} = \{u \in \Sigma_{\mathscr{W}} \mid Post_{T_{\mathscr{W}}}(\mathbf{X}_l, u) \cap \mathbf{X}' \neq \emptyset\} \tag{9.10}$$

denote the set of all inputs under which $T_{\mathscr{W}}$ can make a transition from a state in $\mathbf{X}_l$ to a state inside $\mathbf{X}'$. Equivalently, applying any input $u \in \mathbf{U}$, $u \notin U^{\mathbf{X}_l \to \mathbf{X}'}$ guarantees that $T_{\mathscr{W}}$ will not make a transition inside $\mathbf{X}'$, from any state in $\mathbf{X}_l$. The following proposition states that $U^{\mathbf{X}_l \to \mathbf{X}'}$ is a polyhedral set that can be computed from the V- (vertex) and H- (hyperplane) representations of $\mathbf{X}_l$ and $\mathbf{X}'$:

Proposition 9.2 *Let H and K be the matrices in the H-representation of the following polytope:*

$$\{\hat{x} \in \mathbb{R}^N \mid \exists x \in \boldsymbol{X}_l,\, A_l x + \hat{x} + c_l \in \boldsymbol{X}'\} \tag{9.11}$$

Then $U^{X_l \to X'}$ is a polytope with the following H-representation:

$$U^{X_l \to X'} = \{u \in \boldsymbol{U} \mid HB_l u < K\} \tag{9.12}$$

Proof The set defined in Eq. (9.11) is a polytope with the following V-representation:

$$\text{hull}\{v' - (Av + c) \mid v \in V(\mathbf{X}_l),\ v' \in V(\mathbf{X}')\} \tag{9.13}$$

Let $x \in \mathbf{X}_l$ such that $A_l x + \hat{x} + c_l \in \mathbf{X}'$. Let $m = |V(\mathbf{X}_l)|$ and $x = \Sigma_{i=1}^m \lambda_i v_i$, where $0 < \lambda_i < 1$ for all $i = 1, \ldots, m$ and $\Sigma_{i=1}^m \lambda_i = 1$. Let $n = |V(\mathbf{X}')|$ and $x' = \Sigma_{j=1}^n \mu_j v'_j$, where $0 < \mu_j < 1$ for all $j = 1, \ldots, n$ and $\Sigma_{j=1}^n \mu_j = 1$. Then,

$$A_l \Sigma_{i=1}^m \lambda_i v_i + \hat{x} + c_l = \Sigma_{j=1}^n \mu_j v'_j \Rightarrow \hat{x} = \Sigma_{j=1}^n \mu_j v'_j - A_l \Sigma_{i=1}^m \lambda_i v_i - c_l =$$
$$= \Sigma_{i=1}^m \Sigma_{j=1}^n \lambda_i \mu_j (v'_j - (A_l v_i + c_l)) \Rightarrow \hat{x} \in \text{hull}\{v' - (Av + c) | v \in V(\mathbf{X}_l),\ v' \in V(\mathbf{X}')\}$$

Let $\hat{x} = \Sigma_{i=1}^m \Sigma_{j=1}^n \nu_{ij}(v'_j - (A_l v_i + c_l))$, where $0 < \nu_{ij} < 1, i = 1, \ldots, m, j = 1, \ldots, n$ and $\Sigma_{i=1}^m \Sigma_{j=1}^n \nu_{ij} = 1$. Let $\lambda_i = \Sigma_{j=1}^m \nu_{ij}$ and $\mu_j = \Sigma_{i=1}^n \nu_{ij}$. Of course, $0 < \lambda_i < 1$ for all $i = 1, \ldots, m$, $0 < \mu_j < 1$ for all $j = 1, \ldots, n$ and $\Sigma_{i=1}^m \lambda_i = \Sigma_{j=1}^n \mu_j = \Sigma_{i=1}^m \Sigma_{j=1}^n \nu_{ij} = 1$. Then, for $x = \Sigma_{i=1}^m \lambda_i v_i$ and $x' = \Sigma_{j=1}^n \mu_j v'_j$ we have $A_l x + \hat{x} + c_l = x'$ and therefore $\exists x \in \mathbf{X}_l$ such that $A_l x + \hat{x} + c_l \in \mathbf{X}'$. To conclude the proof of Proposition 9.2, let H, K be the matrices in the H-representation of the set defined in Eq. (9.11) and note that the set defined in Eq. (9.10) can be equivalently written as

$$U^{\mathbf{X}_l \to \mathbf{X}'} = \{u \in \mathbf{U} \mid \exists x \in \mathbf{X}_l,\ A_l x + B_l u + c_l \in \mathbf{X}'\} \tag{9.14}$$

■

Proposition 9.3 *Given a state $l \in X_c$ and a set of states $C \in 2^{X_c}$, the set U_l^C from Eq. (9.3) can be computed as follows:*

$$U_l^C = \bigcap_{l' \in C} U^{\mathbf{X}_l \to \mathbf{X}_{l'}} \setminus \bigcup_{l'' \notin C} U^{\mathbf{X}_l \to \mathbf{X}_{l''}} \tag{9.15}$$

Proof From Eqs. (9.1) and (9.3) we have

$$\begin{aligned} U_l^C &= \{u \in \Sigma_{\mathscr{W}} \mid \forall l' \in C,\ Post_{T_{\mathscr{W}}}(\mathbf{X}_l, u) \cap \mathbf{X}_{l'} \neq \emptyset, \\ &\quad \forall l'' \notin C,\ Post_{T_{\mathscr{W}}}(\mathbf{X}_l, u) \cap \mathbf{X}_{l''} = \emptyset\} = \\ &= \{u \in \Sigma_{\mathscr{W}} \mid \forall l' \in C,\ Post_{T_{\mathscr{W}}}(\mathbf{X}_l, u) \cap \mathbf{X}_{l'} \neq \emptyset\} \setminus \\ &\quad \{u \in \Sigma_{\mathscr{W}} \mid \exists l'' \notin C,\ Post_{T_{\mathscr{W}}}(\mathbf{X}_l, u) \cap \mathbf{X}_{l''} \neq \emptyset\} = \\ &= \bigcap_{l' \in C} U^{\mathbf{X}_l \to \mathbf{X}_{l'}} \setminus \bigcup_{l'' \notin C} U^{\mathbf{X}_l \to \mathbf{X}_{l''}} \end{aligned}$$

■

We can guarantee that if a state l' is not reachable from state l in $T_{\mathscr{W}}/_{\sim}$ (i.e., $l' \notin Post_{T_{\mathscr{W}}/_{\sim}}(l, \Sigma^l)$) then $U^{\mathbf{X}_l \to \mathbf{X}_{l'}} = \emptyset$ and therefore, $U_l^C = \emptyset$ if $C \nsubseteq Post_{T_{\mathscr{W}}/_{\sim}}(l, \Sigma^l)$ and otherwise the computation in Eq. (9.15) reduces to

$$U_l^C = \bigcap_{l' \in C} U^{\mathbf{X}_l \to \mathbf{X}_{l'}} \setminus \bigcup_{l'' \in Post_{T_{\mathscr{W}}/_{\sim}}(l, \Sigma^l) \setminus C} U^{\mathbf{X}_l \to \mathbf{X}_{l''}} \tag{9.16}$$

A non-empty input region U_l^C is in general nonconvex but can always be represented as a finite union of open polytopes (see Eq. (9.16)). In order to guarantee the robustness of the control strategy (as described in Sect. 9.1.1) we only include input sets that are "large enough" (i.e., $r(U_l^C) > \varepsilon$, where ε is a predefined robustness parameter and $r()$ is the radius of the Chebyshev ball (Definition A.9). Note that in general this approach might be conservative, since a sphere inscribed in a union of polytopes from U_l^C might have a larger radius. Following from the results presented in this section, the control transition system T_c can be computed using polyhedral operations only (the computation is summarized in Algorithm 17).

Remark 9.1 It is possible to reduce the size of T_c after it is initially constructed without sacrificing solutions. More "nondeterminism" available at a state does not result in more winning strategies for Algorithm 17, while at the same time unnecessarily increases the complexity of the method. Formally, let $u_1 = c(U_l^{C_1})$ and $u_2 = c(U_l^{C_2})$ where $C_1, C_2 \in 2_c^X$, $C_1 \subseteq C_2$ be inputs of T_c available at state $l \in X_c$ (i.e., $\{u_1, u_2\} \subseteq \Sigma_c^l$). If input u_2 is used in a control strategy, then the specification

Algorithm 17 T_c = CONTROL- TS($\mathscr{W}, \varepsilon$): Construct control transition system T_c

```
1: $X_c := L$
2: for each $l \in X_c$ do
3:   $\Sigma_c^l := \Sigma^l$ [Eq. (9.5)]
4:   compute $Post_{T_{\mathscr{W}}/\sim}(l, \Sigma_c^l)$ [Eq. (9.8)]
5:   for each $C \subseteq Post_{T_{\mathscr{W}}/\sim}(l, \Sigma_c^l)$ do
6:     compute $U_l^C$ [Eq. (9.16)]
7:     if $r(U_l^C) > \varepsilon$ then
8:       include input $c(U_l^C)$ in $\Sigma_c^l$
9:       include transition $\delta_c(l, c(U_l^C)) = C$
10:    end if
11:  end for
12:  if $\Sigma_c^l = \emptyset$ then
13:    recursively make state $l$ unreachable and set $X_c := X_c \setminus l$
14:  end if
15: end for
16: $\Sigma_c = \bigcup_{l \in X_c} \Sigma_c^l$
17: return $T_c$
```

is satisfied regardless of which state $l' \in C_2$ is visited in the next step. Clearly, the same holds for input u_1 since C_1 is a subset of C_2 but keeping both inputs is unnecessary. Therefore, at each state $l \in X_c$ we set $\Sigma_c^{ls} = \Sigma_c^{ls} \setminus u_2$ if $u_1, u_2 \in \Sigma_c^{ls}$ or $\Sigma_c^{lu} = \Sigma_c^{lu} \setminus u_2$ if $u_1, u_2 \in \Sigma_c^{lu}$ when the property described above holds.

9.2 LTL Control of PWA Systems

In Sect. 9.1 we defined the control transition system T_c as a finite abstraction of the infinite $T_{\mathscr{W}}$ and showed that it can be computed using polyhedral operations. In Sect. 5.1 of Chap. 5, we presented an approach for controlling finite transition systems (such as T_c) from specifications given as LTL formulas. In this section, we show that a control strategy generated for T_c can be adapted to the infinite $T_{\mathscr{W}}$, while the satisfaction of LTL formulas by the closed loop systems is preserved, which completes the solution to Problem 9.1.

Definition 9.1 (*PWA control strategy*) A control strategy (X_0^c, Ω^c) for T_c can be translated into a control strategy (X_0, Ω) for $T_{\mathscr{W}}$ as follows. The initial set $X_0^c \subseteq X_c$ gives the initial set $X_0 = \bigcup_{l \in X_0^c} \mathbf{X}_l \subseteq X_{\mathscr{W}}$. Given a finite sequence of states $x(0) \dots x(k)$ where $x(0) \in X_0$, the control function is defined as $\Omega(x(0) \dots x(k)) = \Omega^c(o_{\mathscr{W}}(x(0)) \dots o_{\mathscr{W}}(x(k)))$.

Proposition 9.4 *Given a control strategy (X_0^c, Ω^c) for T_c translated as a control strategy (X_0, Ω) for $T_{\mathscr{W}}$, $\mathcal{L}_{T_{\mathscr{W}}}(X_0, \Omega) \subseteq \mathcal{L}_{T_c}(X_0^c, \Omega^c)$, which implies that if $T_c(Q_0^c, \Omega^c)$ satisfies an arbitrary LTL formula ϕ, then so does $T_{\mathscr{W}}(X_0, \Omega)$.*

Proof Transition systems T_c and $T_{\mathscr{W}}$ have the same set of observations and therefore the same LTL formula can be interpreted over both systems. Let $x(0) \in X_0$ be an initial state for $T_{\mathscr{W}}$. Its observation is $x^c(0) = o_{\mathscr{W}}(x(0))$, which is a satisfying initial state for T_c (i.e., $x^c(0) \in X_0^c$). The next input to be applied in T_c is given by the control function $(u^c(0) = \Omega(x^c(0)) \in \Sigma_c \subset \Sigma_{\mathscr{W}})$ and we can guarantee that regardless which input $u(0) \in \Sigma_{\mathscr{W}}$ such that $d(u(0), u^c(0)) < \varepsilon$ is applied in $T_{\mathscr{W}}$, we have $\delta_{\mathscr{W}}(x(0), u(0)) \in \delta_c(x^c(0), u^c(0))$. This shows that a finite fragment of a word in $T_{\mathscr{W}}(Q_0, \Omega)$ is also a finite fragment of a word in $T_c(Q_0^c, \Omega^c)$ and the rest of the proof follows by induction. ■

The overall solution to Problem 9.1 consists of constructing the control transition system T_c (Sect. 9.1), finding a satisfying control strategy for T_c and adapting it to the original $T_{\mathscr{W}}$, or equivalently PWA system (Definition 9.1), which from Proposition 9.4 guarantees the correctness of the solution. It is important to note that a control strategy generated using this approach is robust with respect to knowledge of the exact state of the system (i.e., the control strategy depends on the observation of a state rather than the state itself). In addition, the control strategy is robust with respect to perturbations in the applied input bounded by ε, which can be used as a tuning parameter.

9.3 Conservatism and Stuttering Behavior

In Chap. 5 we described a solution to the problem of controlling a finite and possibly nondeterministic transition system from LTL specifications. In order to generate a control strategy for an infinite transition system such as $T_{\mathscr{W}}$ (Problem 9.1) we described the construction of a finite control abstraction T_c in Sect. 9.1. However, due to *spurious trajectories* (i.e., trajectories of T_c not present in $T_{\mathscr{W}}$) we cannot guarantee that a control strategy will be found for T_c even if one exists for $T_{\mathscr{W}}$ and therefore, the overall method is conservative. In Chap. 7, we eliminated spurious trajectories through state refinement but the states of T_c cannot be refined. Indeed, the control strategies we consider in this chapter cannot differentiate between states $x_1, x_2 \in X_{\mathscr{W}}$ when $o(x_1) = o(x_2)$ and therefore the states of T_c must satisfy $o_c(l_1) = o_c(l_2)$ if and only if $l_1 = l_2$ for all $l_1, l_2 \in X_c$. In the following, we present an alternative approach for reducing this conservatism.

Similar to the approach for the characterization of transient parameters from Sect. 8.3, here we characterize *stutter* steps as a specific class of spurious trajectories, which we introduce through Example 9.1 and Fig. 9.1.

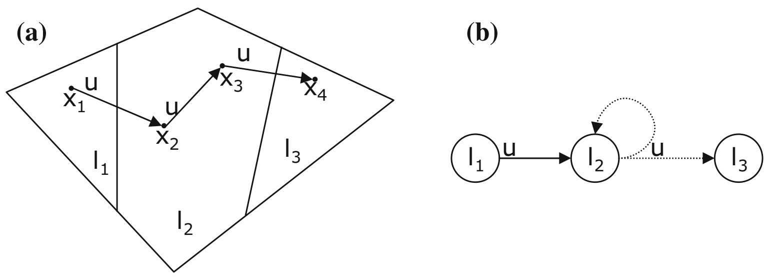


Fig. 9.1 A trajectory remaining forever in state l_2 exists in the finite abstraction (**b**), although such a behavior is not necessarily possible in the concrete system (**a**)

Example 9.1 Assume that a constant input $uuu\ldots$ produces a trajectory $x_1x_2x_3x_4\ldots$ in $T_{\mathscr{W}}$ where $o(x_1) = l_1, o(x_2) = o(x_3) = l_2, o(x_4) = l_3$ (Fig. 9.1a). The corresponding word $l_1l_2l_2l_3\ldots$ is a trajectory of T_c (i.e., $l_1, l_2, l_3 \in X_c$) and from the construction described in Sect. 9.1 it follows that $l_2 \in \delta_c(l_1, u)$ and $\{l_2, l_3\} \subseteq \delta_c(l_2, u)$ (Fig. 9.1b). Then, there exists a trajectory of T_c that remains infinitely in state $l_2 \in X_c$ under input u, which is not necessarily true for $T_{\mathscr{W}}$. Such spurious trajectories do not affect the correctness of a control strategy but increase the overall conservativeness of the method. We address this by characterizing *stuttering inputs*, which guarantee that the system will leave a state eventually, rather than in a single step, and using this additional information during the construction of the control strategy for T_c.

Definition 9.2 (*Stuttering inputs*) Given a state $l \in X_c$ and a set of states $C \in 2^{X_c}$, the set of inputs U_l^C is *stuttering* if and only if $l \in C$ and for all input words $u(0)u(1)\ldots$, where $u(i) \in U_l^C$, there exists a finite $k > 1$ such that the trajectory $x(0)x(1)\ldots$ produced in $T_{\mathscr{W}}$ by the input word satisfies $o(x(i)) = l$ for $i = 0, \ldots, k-1$ and $o(x(k)) = l' \in C, l' \neq l$.

Using Definition 9.2 we identify a stuttering subset $\Sigma_c^{ls} \subseteq \Sigma_c^l$ of the inputs available at a state $l \in X_c$. Let $u = c(U_l^C) \in \Sigma_c^l$ for some $C \in 2^{X_c}$ be an input of T_c computed as described in Sect. 9.1. Then $u \in \Sigma_c^{ls}$ if and only if U_l^C is stuttering. Note that a transition $\delta_c(l, u) = C$ from a state $l \in X_c$ where u is stuttering is always nondeterministic (i.e., $|C| > 1$) and contains a self loop (*i.e.*, $l \in C$) but the self loop cannot be taken infinitely in a row (i.e., a trajectory of $T_{\mathscr{W}}$ cannot remain infinitely in region $\mathbf{X}_l$ under input word $uuu\ldots$). An input $u \in \Sigma_c^{lu} = \Sigma_c^l \setminus \Sigma_c^{ls}$ induces a transition $\delta_c(l, u) = C$ where: (1) when $C = \{l\}$ trajectories of T_c and $T_{\mathscr{W}}$ produced by input word $uuu\ldots$ remain infinitely in state l and region $\mathbf{X}_l$, respectively, (2) when $l \notin C$ trajectories of T_c and $T_{\mathscr{W}}$ leave state l and region $\mathbf{X}_l$, respectively in one step under input u, (3) when $\{l\} \subset C$ trajectories of $T_{\mathscr{W}}$ produced by input word $uuu\ldots$ can potentially remain in region $\mathbf{X}_l$ infinitely. Although in case (3) it is also possible that trajectories of $T_{\mathscr{W}}$ produced by input word $uuu\ldots$ leave region $\mathbf{X}_l$ in finite time, we have to be conservative in order to guarantee the correctness of the control strategy.

Note that our definition of stuttering in Definition 9.2 requires that T_c leaves a state after a finite number of transitions are taken under the same stuttering input and therefore an infinite stutter cycle is never possible. Second, we identify a set of stuttering inputs rather than constructing T_c as a time abstract system. While we only characterize spurious infinite self loops (i.e., cycles of length 1), in general, it is possible that cycles of arbitrary length are spurious in T_c. Considering higher order cycles is computationally challenging and decreases the conservativeness of the approach only for very specific cases, while spurious self loops are commonly produced during the construction of T_c and can be identified or constructed through polyhedral operations as described in the following (Propositions 9.5 and 9.6).

Proposition 9.5 *Given a state $l \in X_c$ and a set of states $C \in 2^{X_c}$, input region U_l^C is stuttering if and only if $l \in C$ and $\mathbf{0} \notin hull\{(A_l - I)v_x + B_l v_u + c_l \mid \forall v_x \in V(\boldsymbol{X}_l), \forall v_u \in V(U_l^C)\}$, where hull denotes the convex hull, $V(.)$ is the set of vertices and I is the identity matrix.*

Proof ($\Rightarrow$) Let $\mathbf{0} \in \text{hull}\{(A_l - I)v_x + B_l v_u + c_l \mid \forall v_x \in V(\mathbf{X}_l), \forall v_u \in V(U_l^C)\}$. Then, there exists $x \in \mathbf{X}_l$, $u \in U_l^C$ such that $A_l x + B_l u + c_l = x$ and a trajectory of the system produced by applying input sequence $uuu\ldots$ and starting at x remains forever inside $\mathbf{X}_l$. Therefore, from Definition 9.2, U_l^C is not stuttering.

($\Leftarrow$) Let $\mathbf{0} \notin \text{hull}\{(A_l - I)v_x + B_l v_u + c_l \mid \forall v_x \in V(\mathbf{X}_l), \forall v_u \in V(U_l^C)\}$. From the separating hyperplane theorem it follows that there exists $a \in \mathbb{R}^N$ such that, for all $z \in \text{hull}\{(A_l - I)v_x + B_l v_u + c_l \mid \forall v_x \in V(\mathbf{X}_l)), a^T z > 0$. Then, any trajectory of the system originating in $\mathbf{X}_l$ and produced by input word $u_1 u_2 u_3 \ldots$, where $u_i \in U_l^C$ will have a positive displacement along the direction of a^T at every step. Since $\mathbf{X}_l$ is bounded, all trajectories will leave it in a finite number of steps and, therefore, U_l^C is stuttering. ■

The strategy from Proposition 9.5 provides a computational characterization of stuttering input regions. In general, however, it is possible that an input region U_l^C cannot be identified as stuttering but a stuttering subset $\hat{U}_l^C \subset U_l^C$ can be identified. Then, if such a subset is "large enough" (i.e., $r(\hat{U}_l^C) > \varepsilon$) it can be used in T_c and allow more general control strategies. In Proposition 9.6 we describe the computation of such stuttering subsets.

Proposition 9.6 *Given an arbitrary $a \in \mathbb{R}^N$, the input region $\hat{U}_l^C = \{u \in U_l^C \mid \forall v \in V(\boldsymbol{X}_l), a^T B_l u > -a^T(A_l - I_N)v - c_l\}$, where $l \in C$ is always stuttering.*

Proof

$$\begin{aligned}
\hat{U}_l^C &= \{u \in U_l^C \mid \forall v_x \in V(\mathbf{X}_l), a^T B_l u > -a^T(A_l - I)v_x - c_l\} = \\
&= \{u \in U_l^C \mid \forall v_x \in V(\mathbf{X}_l), a^T((A_l - I)v_x + B_l u + c_l) > 0\} \Rightarrow \\
&\Rightarrow \mathbf{0} \notin \text{hull}\{(A_l - I)v_x + B_l v_u + c_l \mid \forall v_x \in V(\mathbf{X}_l), \forall v_u \in V(\hat{U}_l^C)\},
\end{aligned}$$

which, from Proposition 9.5, guarantees that $\hat{U}_l^C$ is stuttering. ■

Although Proposition 9.6 is valid for an arbitrary $a \in \mathbb{R}^N$, the volume of the stuttering subset $\hat{U}_l^C \subset U_l^C$ depends on a. Since only "large enough" input regions are considered in T_c (see Algorithm 17), a should be chosen in such a way that the radius $r(\hat{U}_l^C)$ is maximized.

The control algorithms we discussed in Sect. 5.1 can be adapted to handle the additional information about stuttering inputs captured in T_c, while the correctness and completeness of the control strategy computation for the product automaton P is still guaranteed. P is constructed as in Sect. 5.1 and therefore it naturally inherits the partitioned input set $\Sigma_c^l = \Sigma_c^{ls} \cup \Sigma_c^{lu}$ for each state $l \in X_c$. Going back to the Rabin game interpretation of the control problem discussed in Sect. 5.1, we need to account for the fact that the adversary cannot take transitions under the same stuttering input infinitely many times in a row. As a result, the construction of the control strategy is still performed using the same algorithm and only the computations of the direct attractors from Definitions 5.7 and 5.8 are modified as follows (the notation used in the rest of this section was introduced in Chap. 5).

Let $l \in X_c$ and $u \in \Sigma_c^{ls}$ be a state and a stuttering input of T_c (Definition 9.2). We are interested in *edge* (s, u, s') of transition $\delta_P(s, u) = S'$, where $\alpha(s) = l$ and $s' \in S'$ (here $\alpha()$ is the projection of states from P to states of T_c—see Sect. 4.5). Edge (s, u, s') is called *u-nontransient* edge if $\alpha(s) = \alpha(s') = l$ and *transient* otherwise. Note that, even though (l, u, l) is a self loop in T_c, (s, u, s') is not necessarily a self loop in P. In addition, since there is at most one self loop at a state $l \in X_c$ and R is deterministic, there is at most one u-nontransient edge leaving state s.

We refer to a sequence of edges $(s_1, u_1, s_2)(s_2, u_2, s_3) \ldots (s_{n-1}, u_{n-1}, s_n)$, where $s_i \neq s_j$ for any $i, j \in \{1, \ldots, n\}$ as a *simple path*, and to a simple path $(s_1, u_1, s_2) \ldots (s_{n-1}, u_{n-1}, s_n)$ followed by (s_n, u_n, s_1) as a *cycle*. We can observe that any sequence of u-nontransient edges (i.e., a run of the product automaton, or its finite fragment) is of one of the following shapes: a cycle (called a *u-nontransient cycle*), a lasso shape (a simple path leading to a u-nontransient cycle), or a simple path ending at a state where the input u is not available at all. Informally, the existence of a stuttering self loop in a state l under input u in T_c means that this self loop cannot be followed infinitely many times in a row. Similarly, any u-nontransient cycle in the product graph cannot be followed infinitely many times in a row without leaving it. This leads us to the new definitions of protagonist's and adversary's direct attractor.

Definition 9.3 (*Modified protagonist's direct attractor*) The protagonist's direct attractor of S', denoted by $\mathsf{A}_P^1(S')$, is the set of all states $s \in S_P$, such that there exists an input u satisfying

(1) $\delta_P(s, u) \subseteq S'$, or
(2) s lies on a u-nontransient cycle, such that each state s' of the cycle satisfies that $s'' \in S'$ for all transient edges (s', u, s'').

In other words, the protagonist can enforce a visit to S' also by following a u-nontransient cycle finitely many times and eventually leaving it to S'.

Definition 9.4 (*Modified adversary's direct attractor*) The adversary's direct attractor of S', denoted by $\mathsf{A}^1_S(S')$, is the set of all states $s \in S_P$, such that for each input u there exists a state s' such that

(1) $s' \in \delta_P(s, u) \cap S'$, and
(2) s' does not lie on a u-nontransient cycle.

In other words, the adversary cannot enforce a visit to S' via an edge of a u-nontransient cycle. This edge can be taken only finitely many times in a row and eventually a different edge under input u has to be chosen.

By identifying stuttering inputs during the construction of the control transition system T_c (Propositions 9.5 and 9.6) and modifying the approach from Sect. 5.1 to handle this additional information during the construction of a control strategy for T_c (Definitions 9.3 and 9.4), we can reduce the conservatism associated with the overall method. Even so, our solution to Problem 9.1 remains conservative but it is important to note that the only source of conservativeness is the construction of T_c—the solution to the LTL control problem for T_c is complete.

Example 9.2 We consider the problem of controlling the two-tank system shown in Fig. 9.2. The system has two state variables ($N = 2$) that represent the water levels in the two tanks and range in $(0, 0.7)$. It has one control dimension ($M = 1$) representing the inflow rate, which ranges in $(0, 5e^{-4})$. The state space of the system is partitioned into 49 rectangular regions (i.e., $L = 1, \ldots, 49$) by 7 evenly spaced thresholds along each dimension. These thresholds signify that we can only detect whether the water level in each tank is above or below the marks at $0.1, 0.2, \ldots, 0.7$ (see Fig. 9.3a). Valve v_1 is opened only if submerged (i.e., if the water level in either tank is above 0.2—the height of the valve) and, therefore, the valve is closed when the system is in regions 1, 2, 8, and 9 and opened otherwise (see Figs. 9.2 and 9.3a). Discrete-time, linear equations describing the dynamics of the system in each mode are derived using a 5 sec. time step and $1.54e^{-2}\ m^2$, $1e^{-4}\ m^2$, and $2.125e^{-5}\ m^2$ as the cross sectional areas of the two tanks, valve v_1, and valve v_2, respectively. The dynamics of the system for each region $l \in L$ are given by

$$A_l = \begin{cases} \begin{bmatrix} 1 & 0 \\ 0 & 0.9635 \end{bmatrix} & \text{for } l = 1, 2, 8, 9 \\ \begin{bmatrix} 0.8281 & 0.1719 \\ 0.1719 & 0.7916 \end{bmatrix} & \text{otherwise,} \end{cases}$$

$$B_l = [324.6753, 0]^T, c_l = [0, 0]^T \text{ for } l = 1, \ldots, 49.$$

We seek a control strategy for the system guaranteeing the satisfaction of a specification, expressed informally as "whenever tank 2 is empty, it will eventually get filled up". To formalize this specification we define the sub-formulas ϕ_1 = "the level of tank 2 is below 0.1" (i.e., "tank 2 is empty") and ϕ_2 = "the level of tank 2 is above 0.4" (i.e., "tank 2 is full"), which can be expressed as disjunctions of regions from L as $\phi_1 = 1 \vee \ldots \vee 7$ and $\phi_2 = 29 \vee \ldots \vee 49$. The above specification translates to the following LTL formula:

$$\phi = \Box(\phi_1 \Rightarrow \Diamond\phi_2).$$

Satisfying control strategies were found from all regions expect 49 when the required robustness was set to $\varepsilon = 5e^{-6}$ (Fig. 9.3a). If stuttering inputs are not characterized as described in Sect. 9.3, satisfying control strategies are identified for regions $29, \ldots, 48$ only. A simulated trajectory of the closed loop system is shown in Fig. 9.3b.

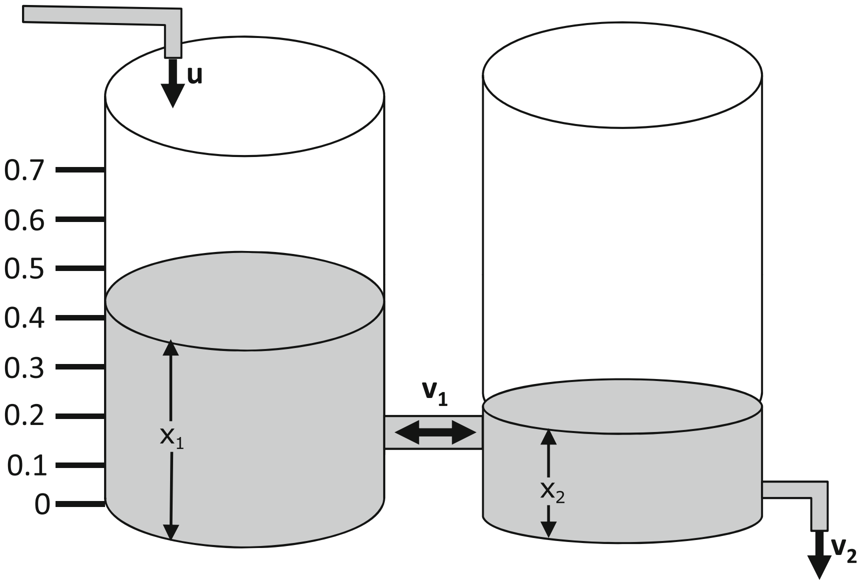
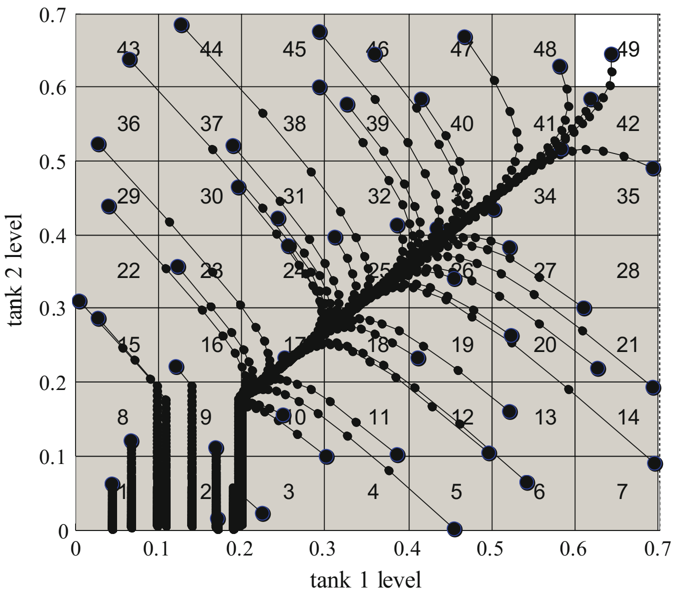
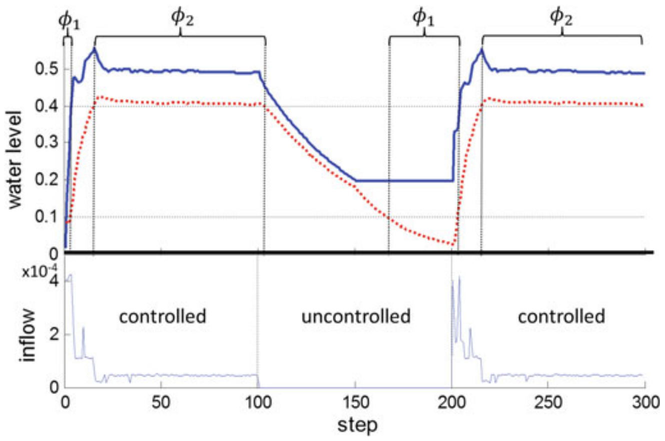


Fig. 9.2 A two-tank system. Water is drained at a constant rate from tank 2 though valve v_2, while tank 1 is filled at a rate that is controlled externally. Water can also flow in either direction, from the tank with more water to the one with less, through valve v_1, which is opened only if submerged



(a) Simulated trajectories of the uncontrolled system.



(b) A simulated trajectory of the closed-loop system.

Fig. 9.3 Simulated trajectories of the uncontrolled and closed-loop water tank system from Fig. 9.2. Initial conditions are shown as *blue circles* and regions are labeled only by their indexes in (**a**). Control strategies guaranteeing the satisfaction of specification $\phi = \Box(\phi_1 \Rightarrow \Diamond\phi_2)$ are found from all shaded regions in (**a**). The water levels of tanks 1 and 2 are shown respectively as a *blue (solid)* and a *red (dashed) line* in (**b**) and the trajectory is guaranteed to satisfy specification ϕ. See Example 9.2 for additional details

Example 9.3 We apply the method from this chapter to generate control strategies for a simple PWA system with two state variables ($N = 2$) ranging in $(0, 100)$. The state space is partitioned into 36 rectangular regions (i.e., $L = 1, \dots, 36$). The system has two control inputs ($M = 2$) ranging in $(-15, 15) \times (-16, 16)$. The dynamics of the system are defined as follows:

$$A_{1\dots4,9\dots12,25\dots28,33\dots36} = \begin{bmatrix} 0.9900 & 0.0 \\ 0.0 & 0.9800 \end{bmatrix} \quad A_{5\dots8,29\dots32} = \begin{bmatrix} 0.9900 & 0.0 \\ -0.0200 & 0.9800 \end{bmatrix}$$
$$A_{13\dots16,21\dots24} = \begin{bmatrix} 0.9900 & -0.0300 \\ 0.0 & 0.9800 \end{bmatrix} \quad A_{17\dots20} = \begin{bmatrix} 0.9900 & -0.0300 \\ -0.0200 & 0.9800 \end{bmatrix}$$
$$B_{1\dots36} = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$$
$$c_{1\dots4} = \begin{bmatrix} 0.9200 \\ 1.5300 \end{bmatrix} \quad c_{5\dots8} = \begin{bmatrix} 1.3900 \\ 1.5300 \end{bmatrix} \quad c_{9\dots12} = \begin{bmatrix} 0.1700 \\ 1.5300 \end{bmatrix}$$
$$c_{13\dots16} = \begin{bmatrix} 0.9200 \\ 2.9000 \end{bmatrix} \quad c_{17\dots20} = \begin{bmatrix} 1.3800 \\ 2.9000 \end{bmatrix} \quad c_{21\dots24} = \begin{bmatrix} 0.1700 \\ 2.9200 \end{bmatrix}$$
$$c_{25\dots28} = \begin{bmatrix} 0.9200 \\ 0.1500 \end{bmatrix} \quad c_{29\dots32} = \begin{bmatrix} 1.4100 \\ 0.1500 \end{bmatrix} \quad c_{33\dots36} = \begin{bmatrix} 0.1700 \\ 0.1500 \end{bmatrix}$$

The PWA model captures the characteristic bistability of the system, where trajectories of the uncontrolled system go towards one of two possible stable equilibria located in regions $\mathbf{X}_{10}$ and $\mathbf{X}_{27}$ (see Fig. 9.4).

We seek a control strategy that drives the system to low values of state variable 1 and high values of state variable 2, while avoiding intermediate values of both state variables. We define sub-formulas ϕ_1 = "state variable 1 is below 20 and state variable 2 is above 75" and ϕ_2 = "state variable 1 is above 40 and below 80 and state variable 2 is above 20 and below 50" which can be expressed as disjunctions of regions from L as $\phi_1 = 10$ and $\phi_2 := 17 \vee \dots \vee 20$. The control specification translates to the following LTL formula:

$$\phi = \Diamond\Box\phi_1 \wedge \Box\neg\phi_2,$$

which cannot be translated into a deterministic Büchi automaton (see Chap. 2).

A control transition system T_c with 36 states was constructed. Out of the total 949 nonempty input regions found (denoted by U_l^C in Sect. 9.1), 684 were "large enough" (the radii of their inscribed spheres were larger than $\varepsilon = 5e^{-2}$) to be considered for a robust control strategy and included in T_c. After reducing the size of T_c (i.e., removing unnecessary nondeterministic transitions as explained in Remark 9.1) only 222 input regions were included out of which 42 were deterministic and 109 were identified as stuttering. The specification ϕ translates into a deterministic Rabin automaton with 3 states and 1 pair in its acceptance condition.

Satisfying control strategies with the required robustness ($\varepsilon = 5e^{-2}$) were found from all regions expect for $13, \ldots, 20$ (shown in light gray in Fig. 9.5a). For this particular problem, the target region $\mathbf{X}_{10}$ of specification ϕ is reachable only through transitions under stuttering inputs in T_c. Therefore, a satisfying control strategy can be identified only from region $\mathbf{X}_{10}$, unless stuttering behavior is considered. The additional computation described in Sect. 9.3 allowed us to expand the satisfying initial set from $\mathbf{X}_{10}$ only to the entire region highlighted in Fig. 9.5.

Starting from random initial conditions, trajectories of the closed loop system were simulated (Fig. 9.5), where at each step applied inputs were corrupted by noise bounded by ε. All simulated trajectories avoid the unsafe regions $\mathbf{X}_{17} \ldots \mathbf{X}_{20}$ and satisfy the specification, thereby demonstrating the correctness and robustness of the control strategy.

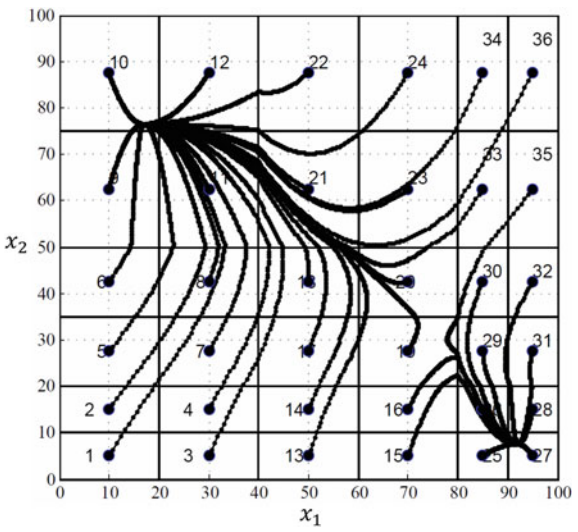
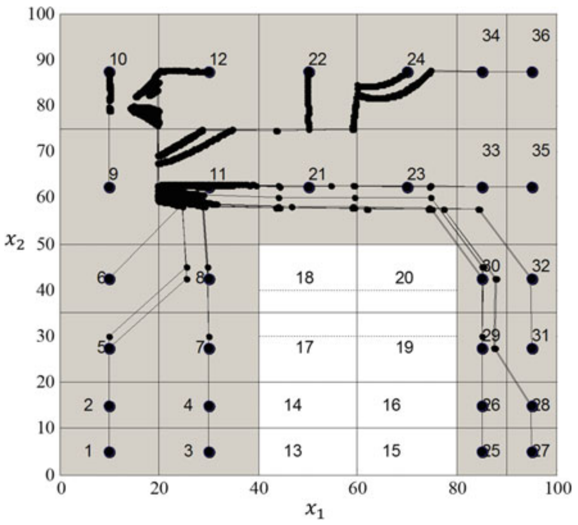
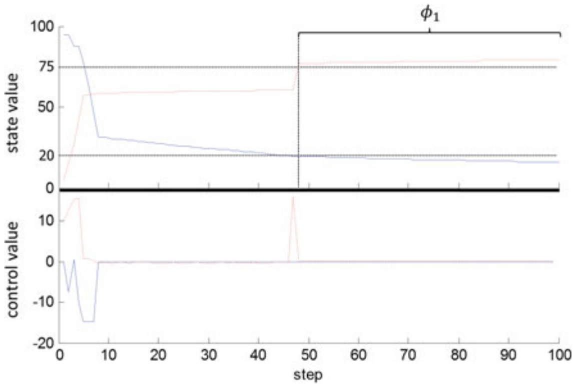


Fig. 9.4 Trajectories of the uncontrolled PWA system go towards one of two possible stable equilibria located in regions $\mathbf{X}_{10}$ and $\mathbf{X}_{27}$ (initial states are shown as *small circles* and regions are labeled only by their indexes). See Example 9.3 for additional details

Fig. 9.5 Control strategies guaranteeing the satisfaction of specification $\phi = \Diamond\Box\phi_1 \wedge \Box\neg\phi_2$ are found from all shaded regions (regions are labeled only by their indexes). Simulated trajectories from different regions satisfy the specification. A simulated trajectory of the closed loop system is guaranteed to satisfy specification ϕ—the values of state variables 1 and 2 and the respective external control values are shown as a *blue (solid)* and a *red (dashed) line*. See Example 9.3 for additional details



(a) Satisfying regions and simulated trajectories of the closed-loops system (in state space)



(b) Simulated trajectory and control values of the closed-loop system (over time)

Example 9.4 We seek a control strategy for the PWA system defined in Example 9.3 that forces the system to oscillate between states where the values of one of the state variables is high and the other is low and vice versa, while states where the values of both state variables are high or low are never reached. We define sub-formulas $\phi_1 =$ "state variable 1 is below 40 and state variable 2 is above 50", $\phi_2 :=$ "state variable 1 is above 80 and state variable 2 is below 20", $\phi_3 :=$ "state variable 1 is below 40 and state variable 2 is below 20", and $\phi_4 :=$ "state variable 1 is above 80 and state variable 2 is above 50", which can be expressed as disjunctions of regions from L as $\phi_1 = 9 \vee \ldots \vee 12$, $\phi_2 = 25 \vee \ldots \vee 28$, $\phi_3 = 1 \vee \ldots \vee 4$, and $\phi_4 = 33 \vee \ldots \vee 36$. The above specification translated to the following LTL formula:

$$\phi = \Box(\Diamond\phi_1 \wedge \Diamond\phi_2 \wedge \neg(\phi_3 \vee \phi_4)).$$

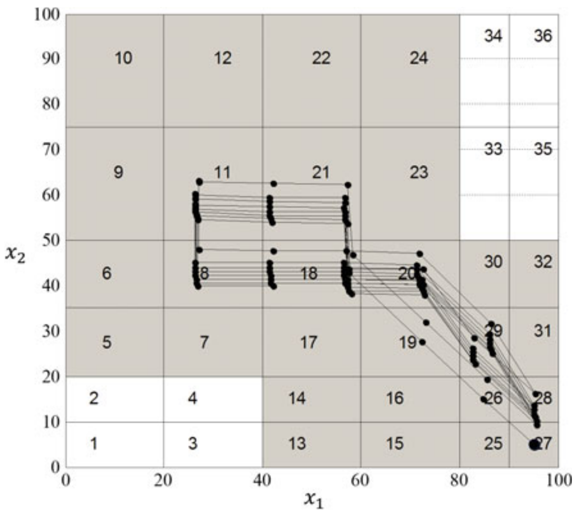
Satisfying control strategies where found from all regions expect $1, \ldots, 4$ and $33, \ldots, 36$ when the required robustness was set to $\varepsilon = 5e^{-2}$ (Fig. 9.6a). If stuttering inputs are not characterized as described in Sect. 9.3, no satisfying control strategies are identified. A simulated trajectory of the closed loop system is shown in Fig. 9.6b.

9.4 Notes

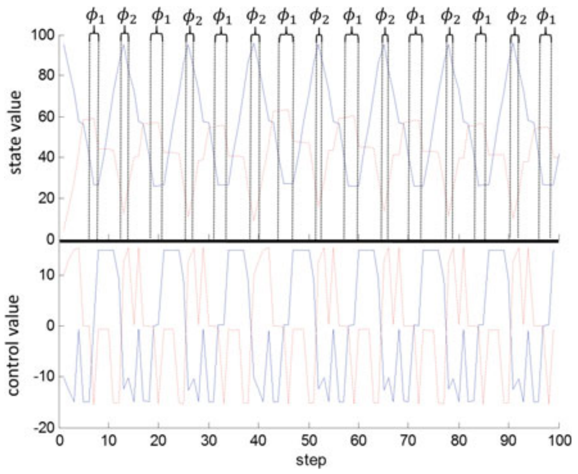
The material presented in this chapter is based on [170, 179, 184]. Related approaches can be found in [138, 164], where the existence of equivalent (bisimulation) quotients and control strategies under the assumption of controllability for discrete-time linear systems is characterized. The monotonicity property of a discrete-time piecewise system is exploited in [46] to develop an efficient abstraction algorithm. Algorithmic procedures for controlling continuous-time linear systems are given in [104, 105], where the constructed deterministic (nondeterministic) abstractions are not equivalent to the original system but instead capture only a restricted but controllable subset of its behavior. An alternative abstraction technique based on quantifier elimination for real closed fields and theorem proving has been proposed in [168].

To solve Rabin games, we implemented an algorithm from [90] (see details in Sect. 5.1) but extended it to deal with stuttering behavior, which leads to additional winning strategies. The concept of stuttering (see Sect. 9.3) has been established previously [15] and related work has focused on determining if a specification is closed under stuttering [137], in which case it can be expressed in the LTL$\setminus\bigcirc$ fragment (LTL without the next operator) [139] and less conservative stutter bisimulation quotients can be constructed [15]. The approach from this chapter does not require any special structure from either the specification or the quotient. Instead, we charac-

Fig. 9.6 Control strategies guaranteeing the satisfaction of specification $\phi = \Box(\Diamond\phi_1 \wedge \Diamond\phi_2 \wedge \neg(\phi_3 \vee \phi_4))$ are found from all shaded regions (regions are labeled only by their indexes). A sample satisfying simulated trajectory is shown. A simulated trajectory of the closed loop toggle switch system is guaranteed to satisfy specification ϕ—state and control variables are labeled as in Fig. 9.5b. See Example 9.4 for additional details



(a) Satisfying regions and simulated trajectories of the closed-loops system (in state space)



(b) Simulated trajectory and control values of the closed-loop system (over time)

terize individual transitions as stuttering while constructing the abstraction and use this additional information during the solution of the Rabin game, which reduces the conservatism of the overall method but does not restrict the expressivity of the specification. Another related approach is based on characterizing the sets of transient states (instead of an individual transition) and augmenting the transition relation with the corresponding transitions, which was applied to switched systems in [136]. While considering transient sets reduces conservatism, it is computationally expensive to characterize such sets.

As our proposed solution to Problem 9.1 consists of (1) the construction of the control transition system T_c and (2) the generation of a control strategy for T_c, the overall computational complexity is the cumulative complexity of the two parts. The computation of T_c involves enumerating all subsets of L at any element of L, which gives $\mathscr{O}(|L| \cdot 2^{|L|})$ iterations in the worst case, although in practice this can be reduced (see Eq. (9.16)). At each iteration, polyhedral operations are performed, which scale exponentially with N, the size of the continuous state space. The characterization of stuttering inputs described in this chapter checks each element from Σ_c through polyhedral operations.

The overall complexity of the control strategy synthesis for T_c is $\mathscr{O}(k!n^k)$, where n is the size of the product automaton and k is the number of pairs in the Rabin condition of the product automaton (see Chap. 5). The modifications in the computation of the direct attractor we made in order to adapt the algorithm to deal with stuttering behavior do not change the overall complexity. Note that, in general, Rabin games are NP-complete, so the exponential complexity with respect to k is not surprising. However, LTL formulas are usually translated into Rabin automata with very few tuples in their acceptance condition.

The method described in this chapter was implemented in MATLAB as the software package CONPAS2, where all polyhedral operations were performed using the MPT toolbox [113]. The tool takes as input a PWA system (Definition 6.2) and an LTL formula and produces a set of satisfying initial regions and a feedback control strategy for the system (Definition 9.1). The tool is freely downloadable from our web site at http://sites.bu.edu/hyness/conpas2/. As an alternative to CONPAS2, the MPT TOOLBOX [113] and the HYBRID TOOLBOX [27] for MATLAB can also be used to design piecewise affine control laws but neither can handle the richness of LTL specifications directly. The problem of controlling Mixed Logical Dynamical (MLD) systems from LTL specifications has been considered in [100] and is related to this problem, due to the equivalence between MLD and PWA systems [82]. Rather than relying on the construction of finite quotients as in this chapter, the approach taken in [100] involves representing LTL specifications as mixed-integer linear constraints but a finite horizon assumption is imposed.

Other temporal logic control tools include PESSOA [128], LTLMOP [61], TULIP [174] (http://tulip-control.sourceforge.net), LTLCON [105], and its extension BÜCON [104]. PESSOA is capable of generating control strategies for linear, nonlinear, and switched systems from temporal logic specifications. Abstractions based on the notions of approximate simulation and bisimulation are constructed. LTLMOP uses a fragment of LTL called GR(1) [109, 142], which accounts for the adversarial nature of the environment. Similar to LTLMOP, TULIP models the environment as an adversary and uses a receding horizon framework to alleviate the computational complexity of synthesis. Similar to CONPAS2, TULIP can handle discrete-time affine control systems. LTLCON addresses the control problem for linear systems but only deterministic abstractions are allowed, which leads to very conservative results. Its extension, BüCon, relaxes this but restricts the specification language to the fragment of LTL generated by deterministic Büchi automata (see Sect. 5.2). In contrast, conPAS2 constructs nondeterministic abstractions of piecewise affine systems and generates control strategies from full LTL specifications, while conservatism is further reduced by identifying stuttering behavior.