Chapter 5
Finite Temporal Logic Control

In this chapter, we treat the general problem of controlling non-deterministic finite transition systems from specifications given as LTL formulas over their sets of observations. We show that, in general, this control problem can be mapped to a Rabin game. For the particular case when the LTL formula translates to a deterministic Büchi automaton, we show that a more efficient solution to the control problem can be found via a Büchi game. Finally, for specifications given in the syntactically co-safe fragment of LTL, we show that the control problem maps to a simple reachability problem. For all three cases, we present all the details of the involved algorithms and several illustrative examples. In Part III, we combine these algorithms with abstractions to derive LTL control strategies for systems with infinitely many states. The problem that we consider in this chapter can be formally stated as follows:

Definition 5.1 (*Control strategy*) A (history dependent) *control function*[1] $\Omega : X^+ \to \Sigma$ for control transition system $T = (X, \Sigma, \delta, O, o)$ maps a finite, non-empty sequence of states to an input of T. A control function Ω and a set of initial states $X_0 \subseteq X$ provide a *control strategy* for T.

We denote a control strategy by (X_0, Ω), the set of all trajectories of the closed loop system T under the control strategy by $T(X_0, \Omega)$, and the set of all words produced by the closed loop T as $\mathscr{L}_T(X_0, \Omega)$. For any trajectory $x_1 x_2 x_3 \ldots \in T(X_0, \Omega)$ we have $x_1 \in X_0$ and $x_{k+1} \in \delta(x_k, \sigma_k)$, where $\sigma_k = \Omega(x_1, \ldots, x_k)$, for all $k \geq 1$.

Definition 5.2 (*Largest Controlled Satisfying Region*) Given a transition system $T = (X, \Sigma, \delta, O, o)$ and an LTL formula ϕ over O, the largest controlled satisfying region $X_T^\phi \subseteq X$ is the largest set of states for which there exists a control function $\Omega : X^+ \to \Sigma$ such that all trajectories $T(X_T^\phi, \Omega)$ of the closed loop system satisfy ϕ (i.e., $\mathscr{L}_T(X_T^\phi, \Omega) \subseteq \mathscr{L}_\phi$).

The LTL control problem is analogous to LTL analysis problem (Problem 4.1), and can be formulated as:

[1] In general, the control function Ω is a partial function, i.e. not every finite sequence of states is mapped to an input.

C. Belta et al., *Formal Methods for Discrete-Time Dynamical Systems*,
Studies in Systems, Decision and Control 89, DOI 10.1007/978-3-319-50763-7_5

Problem 5.1 (*Largest Controlled Satisfying Region Problem*) Given a finite transition system $T = (X, \Sigma, \delta, O, o)$ and an LTL formula ϕ over O, find a control strategy (X_T^ϕ, Ω) such that X_T^ϕ is the largest controlled satisfying region and $\mathcal{L}_T(X_T^\phi, \Omega) \subseteq \mathcal{L}_\phi$.

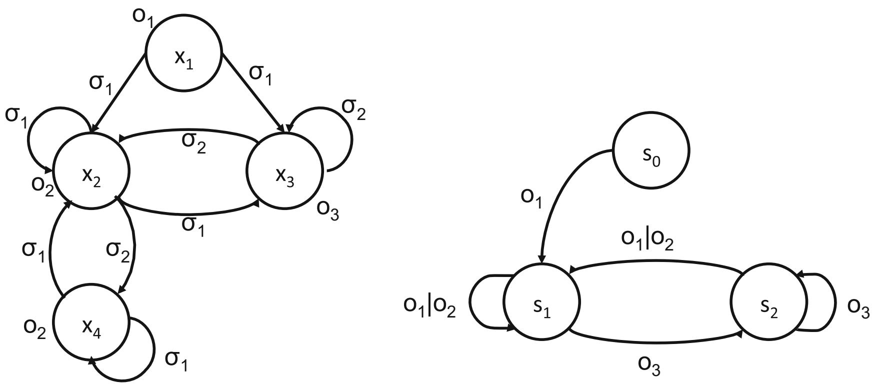
The control problem for transition systems from LTL specifications is stated in most general form in Problem 5.1, i.e., for nondeterministic transition systems and full LTL specifications. In the following section, we present an algorithm to solve this problem and discuss the related complexity. In the presented algorithm, the control synthesis problem is treated as a game played on a finite graph and approached using automata theoretic methods. Such game semantics are introduced due to the nondeterminism of the transition system and the accepting condition of a Rabin automaton. However, if the transition system is deterministic, the control problem can be solved through model checking techniques in a more efficient way. In the subsequent sections, we focus on particular cases of this problem, e.g., when the LTL formula can be translated to a deterministic Büchi automaton (a dLTL specification), and when the LTL formula can be translated to an FSA (an scLTL formula), and present more efficient solutions to the control problem and discuss the associated complexities.

5.1 Control of Transition Systems from LTL Specifications

In this section, we provide a solution to the general problem of controlling finite, nondeterministic systems from LTL specifications (Problem 5.1). The procedure involves the translation of the LTL formula into a deterministic Rabin automaton, the construction of the product automaton of the transition system and the Rabin automaton, followed by the solution of a Rabin game on this product. The solution of the Rabin game is a control strategy for the product automaton, and finally this solution is transformed into a control strategy for the transition system. The resulting control strategy takes the form of a *feedback control automaton*, which reads the current state of T and produces the control input to be applied at that state. The overall control procedure is summarized in Algorithm 9. In the rest of this section, we provide the details of this procedure.

Algorithm 9 LTL CONTROL(T, ϕ): Control strategy (X_T^ϕ, Ω) such that all trajectories in $T(X_T^\phi, \Omega)$ satisfy ϕ

1: Translate ϕ into a deterministic Rabin automaton $R = (S, S_0, O, \delta_R, F)$.
2: Build a product automaton $P = T \otimes R$
3: Transform P into a Rabin game
4: Solve the Rabin game
5: Map the solution to the Rabin game into a control strategy for the original transition system T



(a) Transition system T (b) $\phi = o_1 \wedge (\diamond\Box(o_1 \vee o_2) \vee \diamond\Box o_3)$

Fig. 5.1 Graphical representations of transition system (**a**) and the Rabin automaton (**b**) from Example 5.1. For the automaton, s_0 is the initial state and the acceptance condition is defined by $F = \{(G_1, B_1), (G_2, B_2)\}$, where $G_1 = B_2 = \{s_2\}$ and $B_1 = G_2 = \{s_1\}$

Step 1: Construction of the Rabin Automaton

The first step is to translate the LTL specification ϕ into a deterministic Rabin automaton R. Note that there are readily available off-the-shelf tools for such translations (see Sect. 5.4).

Example 5.1 Consider the nondeterministic transition system $T = (X, \Sigma, \delta, O, o)$ from Example 1.1 shown in Fig. 1.1, and reproduced for convenience in Fig. 5.1a. We consider the following specification "a trajectory of T originates at a state where o_1 is satisfied, and it eventually reaches and remains in a region where either o_1 or o_2 are satisfied, or o_3 is satisfied". The specification is formally defined as the LTL formula

$$\phi = o_1 \wedge (\diamond\Box(o_1 \vee o_2) \vee \diamond\Box o_3).$$

A Rabin automaton representation of the formula ϕ is shown in Fig. 5.1b.

Step 2: Construction of the Product Automaton

The second step is the construction of a product automaton between the transition system T and the Rabin automaton R, which is formally defined as:

Definition 5.3 (*Controlled Rabin Product Automaton*) The *controlled Rabin product automaton* $P = T \otimes R$ of a finite (control) transition system $T = (X, \Sigma, \delta, O, o)$ and a Rabin automaton $R = (S, S_0, O, \delta_R, F)$ is defined as $P = (S_P, S_{P0}, \Sigma, \delta_P, F_P)$, where

- $S_P = X \times S$ is the set of states,
- $S_{P0} = X \times S_0$ is the set of initial states,
- Σ is the input alphabet,
- $\delta_P : S_P \times \Sigma \to 2^{S_P}$ is the transition map, where $\delta_P((x, s), \sigma) = \{(x', s') \in S_P \mid x' \in \delta(x, \sigma),\ \text{and}\ s' = \delta_R(s, o(x))\}$, and
- $F_P = \{(X \times G_1, X \times B_1), \ldots, (X \times G_n, X \times B_n)\}$ is the Rabin acceptance condition.

This product automaton is a nondeterministic Rabin automaton with the same input alphabet Σ as T. Each accepting run $(x_1, s_1)(x_2, s_2)(x_3, s_3)\ldots$ of a product automaton $P = T \otimes R$ can be projected into a trajectory $x_1 x_2 x_3 \ldots$ of T, such that the word $o(x_1)o(x_2)o(x_3)\ldots$ is accepted by R (i.e., satisfies ϕ) and vice versa. This allows us to reduce Problem 5.1 to finding a control strategy for P. We define a control strategy for a Rabin automaton, and therefore for a product automaton constructed as in Definition 5.3, similarly as for a transition system. However, instead of history dependent control strategy, we introduce a memoryless strategy. As we will present later in this section, control strategies obtained by solving Rabin games (step 4 of Algorithm 9) are memoryless.

Definition 5.4 (*Control strategy for a Rabin automaton*) A memoryless *control function* $\pi : S \to O$ for a Rabin automaton $R = (S, S_0, O, \delta_R, F)$ maps a state of R to an input of R. A control function π and a set of initial states $W_0 \subseteq S_0$ provide a *control strategy* (W_0, π) for R.

A run $s_1 s_2 s_3 \ldots$ *under strategy* (W_0, π) is a run satisfying the following two conditions: (1) $s_1 \in W_0$ and (2) $s_{k+1} \in \delta_R(s_k, \pi(s_k))$, for all $k \geq 1$.

The product automaton P allows us to reduce Problem 5.1 to the following problem:

Problem 5.2 Given a controlled Rabin product automaton $P=(S_P, S_{P0}, \Sigma, \delta_P, F_P)$ find the largest set of initial states $W_{P0} \subseteq S_{P0}$ for which there exists a control function $\pi_P : S_P \to \Sigma$ such that each run of P under the strategy (W_{P0}, π_P) satisfies the Rabin acceptance condition F_P.

Example 5.2 The product automaton $P = (S_P, S_{P0}, \Sigma, \delta_P, F_P)$ of the transition system and the Rabin automaton from Example 5.1 (Fig. 5.1) is shown in Fig. 5.2. Note that the blocking states that are not reachable from the non-blocking initial state $p_0 = (x_1, s_0)$ are removed from P and are not shown in Fig. 5.2.

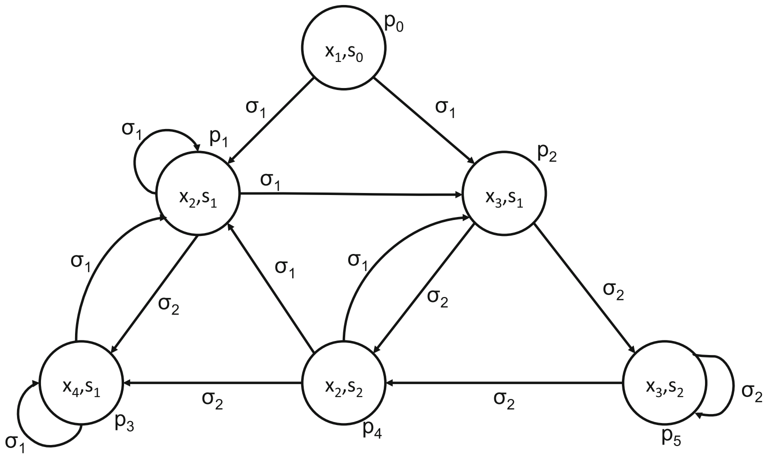


Fig. 5.2 Graphical representation of the product between the transition system from Fig. 5.1a and the Rabin automaton from Fig. 5.1b. The initial state is $p_0 = (x_1, s_0)$. The accepting condition is defined by $F_P = \{(G_1, B_1), (G_2, B_2)\}$, where $G_1 = B_2 = \{p_4, p_5\}$ and $G_2 = B_1 = \{p_1, p_2, p_3\}$

Step 3: Translation to a Rabin Game

A Rabin game consists of a finite graph (V, E) containing a token. The token is always present in one of the states and can move along the edges. There are two players: a protagonist and an adversary. V is partitioned into protagonist's states $V_\mathbf{P}$ and adversary's states $V_\mathbf{A}$. The owner of the state containing a token chooses the edge along which the token moves. A Rabin game is formally defined as:

Definition 5.5 (*Rabin Game*) A Rabin game played by two players (a protagonist and an adversary) on a graph (V, E) is a tuple $\mathbf{G} = (V_\mathbf{P}, V_\mathbf{A}, E, F_\mathbf{G})$, where

- $V_\mathbf{P}$ is the set of protagonist's states,
- $V_\mathbf{A}$ is the set of adversary's states,
- $V_\mathbf{P} \cup V_\mathbf{A} = V$, $V_\mathbf{P} \cap V_\mathbf{A} = \emptyset$,
- $E \subseteq V \times V$ is the set of possible actions,
- $F_\mathbf{G} = \{(G_1, B_1), \ldots, (G_n, B_n)\}$ is the winning condition for the protagonist, where $G_i, B_i \subseteq V$ for all $i \in \{1, \ldots, n\}$.

A play p is an infinite sequence of states visited by the token. Each play is winning either for the protagonist or the adversary. The protagonist wins if $inf(p) \cap G_i \neq \emptyset \wedge inf(p) \cap B_i = \emptyset$ for some $i \in \{1, \ldots, n\}$, where $\inf(p)$ denotes the set of states that appear in the play p infinitely often. The adversary wins in the rest of the cases. The winning region for the protagonist is defined as the set of states $W_\mathbf{P} \subseteq V$ such that there exists a control function $\pi_P : W_\mathbf{P} \cap V_\mathbf{P} \to E$, and all plays starting in the winning region and respecting the winning strategy are winning for the protagonist regardless of the adversary's choices. A solution to a Rabin game is a winning region and winning strategy for the protagonist.

The third step of Algorithm 9 is the construction of a Rabin game from the product automaton, which is performed as follows.

Definition 5.6 (*Rabin game of a Rabin automaton*) A *Rabin game* $\mathbf{G} = (V_{\mathbf{P}}, V_{\mathbf{A}}, E, F_{\mathbf{G}})$ of a Rabin automaton $P = (S_P, S_{P0}, \Sigma, \delta_P, F_P)$ is defined as:

- $V_{\mathbf{P}} = S_P$ is the protagonist's states,
- $V_{\mathbf{A}} = S_P \times \Sigma$ is the adversary's states,
- $E \subseteq \{V_{\mathbf{P}} \times V_{\mathbf{A}} \cup V_{\mathbf{A}} \times V_{\mathbf{P}}\}$ is the set of edges, which is defined as
  - $(q_{\mathbf{P}}, q_{\mathbf{A}}) \in E$ if $q_{\mathbf{P}} \in V_{\mathbf{P}}$, $q_{\mathbf{A}} \in V_{\mathbf{A}}$, and $q_{\mathbf{A}} = (q_{\mathbf{P}}, \sigma)$, where $\sigma \in \Sigma^{q_{\mathbf{P}}}$ (i.e., if $\delta_P(q_{\mathbf{P}}, \sigma) \neq \emptyset$),
  - $(q_{\mathbf{A}}, q_{\mathbf{P}}) \in E$ if $q_{\mathbf{A}} \in V_{\mathbf{A}}$, $q_{\mathbf{P}} \in V_{\mathbf{P}}$, and $q_{\mathbf{A}} = (q'_{\mathbf{P}}, \sigma)$, and $q_{\mathbf{P}} \in \delta_P(q'_{\mathbf{P}}, \sigma)$,
- $F_{\mathbf{G}} = F_P$ is the protagonist's winning condition.

Intuitively, the protagonist chooses action σ, whereas the adversary resolves nondeterminism. Note that in a Rabin game constructed from a Rabin automaton, the protagonist's (adversary's) states can be reached in one step only from the adversary's (protagonist's) states. We will show later in this section that a solution to the Rabin game $\mathbf{G}$ can be easily transformed into a solution to Problem 5.2.

Example 5.3 The Rabin game of the product automaton from Example 5.2 is shown in Fig. 5.3, where protagonist's states are represented as circles and adversary's states are represented as rectangles.

Step 4: Solving the Rabin Game

We present Horn's algorithm for solving Rabin games. The main idea behind the algorithm is as follows. The protagonist wins if they can infinitely often visit G_i and avoid B_i for some $i \in \{1, \ldots, n\}$. Conversely, the protagonist can not win if the adversary can infinitely often visit B_i for each $i \in \{1, \ldots, n\}$. Since it is sufficient for the protagonist to satisfy one of the conditions (G_i, B_i) from $F_{\mathbf{G}}$, the protagonist chooses a condition and tries to avoid visits to B_i and enforce visits to G_i. In turn the adversary tries to avoid G_i. By removing the states where the protagonist (or the adversary) can enforce a visit to a desired set, a smaller game is defined and the algorithm is applied to this game recursively. If the computation ends favorably for the adversary, then the protagonist chooses a different condition (G_j, B_j) from $F_{\mathbf{G}}$ and tries to win the game by satisfying this condition. For a given set $V' \subset V$, the set of states from which the protagonist (or the adversary) can enforce a visit to V' is called an *attractor set*, which is formally defined as follows:

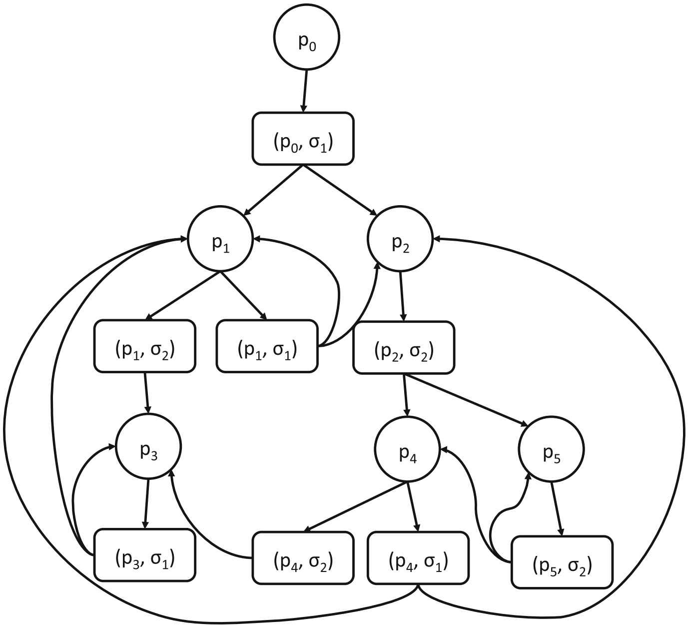


Fig. 5.3 Graphical representation of the Rabin game constructed from the Rabin automaton from Fig. 5.2. An example play is $p = p_0(p_0, \sigma_1)p_2(p_2, \sigma_2)(p_5(p_5, \sigma_2))^{\omega}$

Definition 5.7 (*Protagonist's direct attractor*) The protagonist's direct attractor of a set of states V', denoted by $\mathsf{A}^1_{\mathbf{P}}(V')$, is the set of all states $v_{\mathbf{P}} \in V_{\mathbf{P}}$, such that there exists an edge $(v_{\mathbf{P}}, v_{\mathbf{A}})$, where $v_{\mathbf{A}} \in V'$ together with the set of all states $v_{\mathbf{A}} \in V_{\mathbf{A}}$, such that for all $v_{\mathbf{P}} \in V_{\mathbf{P}}$ it holds that $(v_{\mathbf{A}}, v_{\mathbf{P}}) \in E$ implies $v_{\mathbf{P}} \in V'$:

$$\mathsf{A}^1_{\mathbf{P}}(V') := \{v_{\mathbf{P}} \in V_{\mathbf{P}} | (v_{\mathbf{P}}, v_{\mathbf{A}}) \in E, v_{\mathbf{A}} \in V'\} \bigcup \{v_{\mathbf{A}} \in V_{\mathbf{A}} | \{v_{\mathbf{P}} | (v_{\mathbf{A}}, v_{\mathbf{P}}) \in E\} \subseteq V'\}.$$

Definition 5.8 (*Adversary's direct attractor*) The adversary's direct attractor of V', denoted by $\mathsf{A}^1_{\mathbf{A}}(V')$, is the set of all states $v_{\mathbf{A}} \in V_{\mathbf{A}}$, such that there exists an edge $(v_{\mathbf{A}}, v_{\mathbf{P}})$, where $v_{\mathbf{P}} \in V'$ together with the set of all states $v_{\mathbf{P}} \in V_{\mathbf{P}}$, such that for all $v_{\mathbf{A}} \in V_{\mathbf{A}}$ it holds that $(v_{\mathbf{P}}, v_{\mathbf{A}}) \in E$ implies $v_{\mathbf{A}} \in V'$:

$$\mathsf{A}^1_{\mathbf{A}}(V') := \{v_{\mathbf{A}} \in V_{\mathbf{A}} | (v_{\mathbf{A}}, v_{\mathbf{P}}) \in E, v_{\mathbf{P}} \in V'\} \bigcup \{v_{\mathbf{P}} \in V_{\mathbf{P}} | \{v_{\mathbf{A}} | (v_{\mathbf{P}}, v_{\mathbf{A}}) \in E\} \subseteq V'\}.$$

In other words, the protagonist can enforce a visit to V' from each state $v \in \mathsf{A}^1_{\mathbf{P}}(V')$, regardless of the adversary's choice. Similarly, the adversary can enforce a visit to V' from each state $v \in \mathsf{A}^1_{\mathbf{A}}(V')$, regardless of the protagonist's choice.

Example 5.4 Consider the Rabin game shown in Fig. 5.3. The protagonist's direct attractor set of $\{p_5\}$, $\mathsf{A}^1_{\mathbf{P}}(\{p_5\})$ is empty, since p_5 can be reached from (p_2, σ_2) and (p_5, σ_2), and for both these states the adversary can choose an edge incident to p_4 instead of p_5. On the other hand

$$\mathsf{A}^1_{\mathbf{P}}(\{p_4, p_5\}) = \{(p_2, \sigma_2), (p_5, \sigma_2)\},$$

since at (p_2, σ_2) (and similarly at (p_5, σ_2)), the adversary can either choose the edge $((p_2, \sigma_2), p_4)$ or $((p_2, \sigma_2), p_5)$ and both lead to $\{p_4, p_5\}$.

The adversary's direct attractor set of $\{p_5\}$ is $\mathsf{A}^1_{\mathbf{A}}(\{p_5\}) = \{(p_2, \sigma_2), (p_5, \sigma_2)\}$, since the adversary can enforce a visit to $\{p_5\}$ only from (p_2, σ_2) and (p_5, σ_2). As there are no other adversary states that have an edge to a state from the set $\{p_4, p_5\}$, we have:

$$\mathsf{A}^1_{\mathbf{A}}(\{p_4\}) = \mathsf{A}^1_{\mathbf{A}}(\{p_5\}) = \mathsf{A}^1_{\mathbf{A}}(\{p_4, p_5\}) = \{(p_2, \sigma_2), (p_5, \sigma_2)\}.$$

The protagonist's attractor set $\mathsf{A}_{\mathbf{P}}(V')$ is the set of all states from which a visit to V' can be enforced by the protagonist in zero or more steps. $\mathsf{A}_{\mathbf{P}}(V')$ can be computed iteratively via computation of the converging sequence

$$\mathsf{A}^*_{\mathbf{P}0}(V') \subseteq \mathsf{A}^*_{\mathbf{P}1}(V') \subseteq \ldots,$$

where $\mathsf{A}^*_{\mathbf{P}0}(V') = V'$ and

$$\mathsf{A}^*_{\mathbf{P}i+1}(V') = \mathsf{A}^1_{\mathbf{P}}(\mathsf{A}^*_{\mathbf{P}i}(V')) \cup \mathsf{A}^*_{\mathbf{P}i}(V').$$

The sequence is indeed converging because there are at most $|V_{\mathbf{P}} \cup V_{\mathbf{A}}|$ different sets in the sequence. Intuitively $\mathsf{A}^*_{\mathbf{P}i}(V')$ is the set from which a visit to the set V' can be enforced by the protagonist in at most i steps.

Example 5.5 Consider the Rabin game shown in Fig. 5.3. The protagonist's attractor set for $V' = \{p_4, p_5\}$ is recursively computed as follows:

$$\mathsf{A}^*_{\mathbf{P}1}(V') = \mathsf{A}^*_{\mathbf{P}0}(V') \cup \mathsf{A}^1_{\mathbf{P}}(\{p_4, p_5\}) = \{p_4, p_5, (p_2, \sigma_2), (p_5, \sigma_2)\},$$
$$\mathsf{A}^*_{\mathbf{P}2}(V') = \mathsf{A}^*_{\mathbf{P}1}(V') \cup \mathsf{A}^1_{\mathbf{P}}(\mathsf{A}^*_{\mathbf{P}1}(V')) = \{p_2, p_4, p_5, (p_2, \sigma_2), (p_5, \sigma_2)\},$$
$$\mathsf{A}^*_{\mathbf{P}}(V') = \mathsf{A}^*_{\mathbf{P}3}(V') = \mathsf{A}^*_{\mathbf{P}2}(V') \cup \mathsf{A}^1_{\mathbf{P}}(\mathsf{A}^*_{\mathbf{P}2}(V')) = \{p_2, p_4, p_5, (p_2, \sigma_2), (p_5, \sigma_2)\}.$$

The adversary's attractor set of V' is computed similarly. This computation converges at the fifth iteration, and the resulting set is

$$\mathsf{A}^*_{\mathbf{A}}(V') = \{p_0, p_2, p_4, p_5, (p_0, \sigma_1), (p_1, \sigma_1), (p_2, \sigma_2), (p_4, \sigma_1), (p_5, \sigma_2)\}. \tag{5.1}$$

Attractor strategy $\pi_{\mathsf{A}_{\mathbf{P}}(V')}$ for the protagonist's attractor set determines how to ensure a visit to set V' from attractor set $\mathsf{A}_{\mathbf{P}}(V')$. For all $v \in \mathsf{A}^*_{\mathbf{P}i+1}(V') \setminus \mathsf{A}^*_{\mathbf{P}i}(V')$, the attractor strategy is defined as $\pi_{\mathsf{A}_{\mathbf{P}}(V')}(v) = (v, v')$, where v' is an arbitrary $v' \in \mathsf{A}^*_{\mathbf{P}i}(V')$. The adversary's attractor $\mathsf{A}_{\mathbf{A}}(V')$ and attractor strategy $\pi_{\mathsf{A}_{\mathbf{A}}(V')}$ are computed analogously. The protagonist's and adversary's attractors of V' in a game $\mathbf{G}$ are denoted by $\mathsf{A}^{\mathbf{G}}_{\mathbf{P}}(V')$ and $\mathsf{A}^{\mathbf{G}}_{\mathbf{A}}(V')$, respectively.

Let (V, E) denote the graph of a Rabin game $G = (V_{\mathbf{P}}, V_{\mathbf{A}}, E, F_{\mathbf{G}})$, where $V = V_{\mathbf{P}} \cup V_{\mathbf{A}}$. For simplicity, for a set $Q \subseteq V$, we denote $\mathbf{G} \setminus Q$ the graph $(V \setminus Q, E \setminus E')$ (and the corresponding game), where E' is the set of all edges incident with states from Q.

Horn's algorithm is summarized in Algorithm 10. First the protagonist chooses a condition (G_i, B_i) (line 1). As the protagonist needs to avoid B_i, a sub game $\mathbf{G}^0_i$ is defined by removing the adversary's attractor set for B_i. Then, a sub game $\mathbf{G}^j_i$ is defined iteratively by removing winning regions for the adversary (line 7). The iterative process terminates when no winning region is found for the adversary, i.e., $\mathbf{G}^j_i = \mathbf{G}^{j+1}_i$. In this case, either $\mathbf{G}^j_i$ is empty, or it is winning for the protagonist. If $\mathbf{G}^j_i$ is not empty, then the protagonists attractor of $\mathbf{G}^j_i$ in game $\mathbf{G}$ (line 11) is also winning for the protagonist. By removing the winning region for the protagonist (line 14), a new smaller game is defined and the algorithm is run on this game.

Algorithm 10 RabinGame $(G = (V_{\mathbf{P}}, V_{\mathbf{A}}, E, F_{\mathbf{G}}))$: Winning region $W_{\mathbf{P}} \subseteq (V_{\mathbf{P}} \cup V_{\mathbf{A}})$ and winning strategy $\pi_{\mathbf{P}}$ for the protagonist, winning region $W_{\mathbf{A}} \subseteq (V_{\mathbf{P}} \cup V_{\mathbf{A}})$ for the adversary

1: **for all** $(G_i, B_i) \in F_{\mathbf{G}}$ **do**
2: $j = 0$
3: $\mathbf{G}_i^j = \mathbf{G} \setminus \mathsf{A}_{\mathbf{A}}^{\mathbf{G}}(B_i)$ {remove all states in $\mathsf{A}_{\mathbf{A}}^{\mathbf{G}}(B_i)$ and transitions adjacent to them from $\mathbf{G}$}
4: **repeat**
5: $\mathbf{H}_i^j = \mathbf{G}_i^j \setminus \mathsf{A}_{\mathbf{P}}^{\mathbf{G}_i^j}(G_i)$ {*note that* (G_i, B_i) *is not present in* $\mathbf{H}_i^j$ *any more*}
6: $(W'_{\mathbf{P}}, \pi'_{\mathbf{P}}, W'_{\mathbf{A}}) =$ RabinGame$(\mathbf{H}_i^j)$ {*recursive call*}
7: $\mathbf{G}_i^{j+1} = \mathbf{G}_i^j \setminus \mathsf{A}_{\mathbf{A}}^{\mathbf{G}_i^j}(W'_{\mathbf{A}})$
8: j++
9: **until** $\mathbf{G}_i^j = \mathbf{G}_i^{j+1}$ {$\mathbf{G}_i^j$ *is guaranteed to be winning for the protagonist*}

10: **if** $\mathbf{G}_i^j \neq \emptyset$ **then**
11: $W_{\mathbf{P}} = W_{\mathbf{P}} \cup \mathsf{A}_{\mathbf{P}}^{\mathbf{G}}(\mathbf{G}_i^j)$ {The protagonist's attractor of $\mathbf{G}_i^j$ in $\mathbf{G}$ is winning}
12: $\pi_{\mathbf{P}} = \pi_{\mathbf{P}} \cup \pi'_{\mathbf{P}} \cup \pi''_{\mathbf{P}}$, {$\pi'_{\mathbf{P}}$ is the protagonist's attractor strategy computed in line 6}
13: {$\pi''_{\mathbf{P}}$ is the protagonist's attractor strategy for $\mathsf{A}_{\mathbf{P}}^{\mathbf{G}}(\mathbf{G}_i^j)$}
14: $\mathbf{G}^s = \mathbf{G} \setminus W_{\mathbf{P}}$
15: $(W^s_{\mathbf{P}}, \pi^s_{\mathbf{P}}, W^s_{\mathbf{A}}) =$ RabinGame$(\mathbf{G}^s)$ {*run the algorithm on a smaller graph; consider all pairs in the acceptance condition over again*}
16: $W_{\mathbf{P}} = W_{\mathbf{P}} \cup W^s_{\mathbf{P}}$
17: $\pi_{\mathbf{P}} = \pi_{\mathbf{P}} \cup \pi^s_{\mathbf{P}}$
18: **break** {*break the whole for-cycle 1–18*}
19: **end if**
20: **end for**
21: $W_{\mathbf{A}} = \mathbf{G} \setminus W_{\mathbf{P}}$

Example 5.6 We illustrate Algorithm 10 on the Rabin game shown in Fig. 5.3. At the first iteration, we consider Rabin pair (G_1, B_1), where $G_1 = \{p_4, p_5\}$ and $B_1 = \{p_1, p_2, p_3\}$. The adversary's attractor $\mathsf{A}_{\mathbf{A}}^{\mathbf{G}}(B_1)$ is $V_{\mathbf{P}} \cup V_{\mathbf{A}}$, therefore, on line 10 of Algorithm 10, the graph $\mathbf{G}_1^0$ is empty. As we do not find any states winning for the protagonist, we continue with the next Rabin pair.

In the second iteration of Algorithm 10, we consider Rabin pair (G_2, B_2), where $G_2 = \{p_1, p_2, p_3\}$ and $B_2 = \{p_4, p_5\}$. We eliminate $A_{\mathbf{A}}^{\mathbf{G}}(B_2)$ from the graph on line 3. The remaining graph is $\mathbf{G}_2^0$. We compute $A_{\mathbf{P}}^{\mathbf{G}_2^0}(G_2)$, and find out that it is equal to $\mathbf{G}_2^0$. This means that $\mathbf{H}_2^0$ is empty, $\mathbf{G}_2^1$ is equal to $\mathbf{G}_2^0$, and $\mathbf{G}_2^0$ is guaranteed to be a part of the protagonist's winning region. $A_{\mathbf{A}}^{\mathbf{G}}(B_2)$ and $\mathbf{G}_2^0$ are shown in Fig. 5.4. The protagonist's attractor of $\mathbf{G}_2^0$ in game $\mathbf{G}$ is

$$W_{\mathbf{P}} = A_{\mathbf{P}}^{\mathbf{G}}(\mathbf{G}_2^0) = \{p_1, p_3, p_4, (p_1, \sigma_2), (p_3, \sigma_1), (p_4, \sigma_2)\},$$

and the corresponding winning strategy for the protagonist is (lines 11 and 12)

$$\pi_{\mathbf{P}}(p_1) = (p_1, (p_1, \sigma_2)),$$
$$\pi_{\mathbf{P}}(p_3) = (p_3, (p_3, \sigma_1)),$$
$$\pi_{\mathbf{P}}(p_4) = (p_4, (p_4, \sigma_2)).$$

As we find a winning region for the protagonist, we rerun the algorithm for a smaller game (line 15) as illustrated in Fig. 5.5. Note that the algorithm is run from the beginning on the subgame and all Rabin acceptance pairs are considered again.

At the first iteration of Algorithm 10 on the subgame $\mathbf{G}^s$ shown in Fig. 5.5, we consider Rabin pair (G_1^s, B_1^s), where $G_1^s = \{p_5\}$ and $B_1^s = \{p_2\}$. The adversary's attractor of B_1^s is $\{p_0, p_2, (p_0, \sigma_1), (p_1, \sigma_1), (p_4, \sigma_1)\}$, and the protagonist's attractor of G_1^s on $\mathbf{G}_1^0 = \mathbf{G}^s \setminus A_{\mathbf{A}}^{\mathbf{G}^s}(B_1)$ is $\mathbf{G}_1^0$. $\mathbf{H}_1^0$ is empty, and the protagonists wins everywhere in $\mathbf{G}_1^0$ and its attractor in $\mathbf{G}^s$. The attractor of $\mathbf{G}_1^0$ in $\mathbf{G}^s$ covers $\mathbf{G}^s$. Therefore, we find that the protagonist wins everywhere in $\mathbf{G}^s$ with the following strategy:

$$\pi_{\mathbf{P}}^s(p_0) = (p_0, (p_0, \sigma_1)),$$
$$\pi_{\mathbf{P}}^s(p_2) = (p_2, (p_2, \sigma_2)),$$
$$\pi_{\mathbf{P}}^s(p_5) = (p_5, (p_5, \sigma_2)).$$

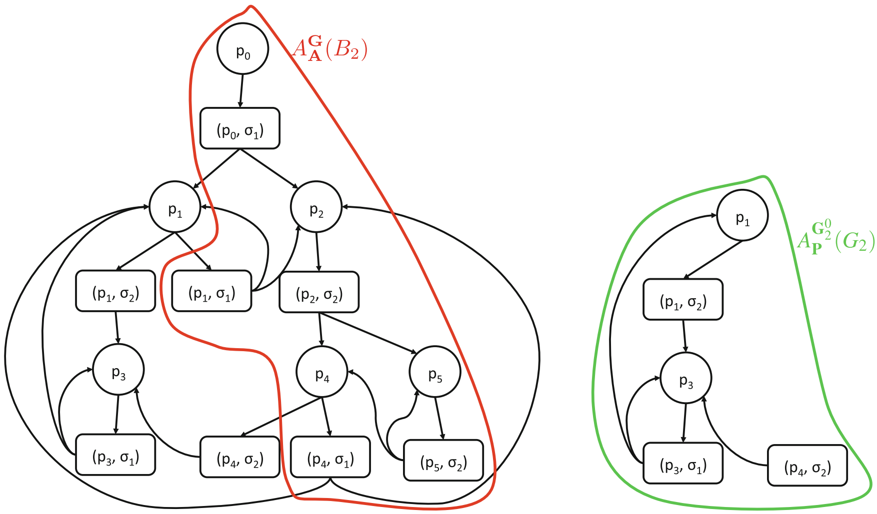
As $W_{\mathbf{P}}^s$ covers $\mathbf{G}^s$, the algorithm (recursive call on the sub-game $\mathbf{G}^s$) terminates with $W_{\mathbf{P}}^s$ and strategy $\pi_{\mathbf{P}}^s$. Finally, the winning region for the protagonist $W_{\mathbf{P}}$ on the initial game $\mathbf{G}$ covers $V_{\mathbf{P}} \cup V_{\mathbf{A}}$, and the protagonist wins everywhere in $\mathbf{G}$ with the strategy $\pi_{\mathbf{P}}$ computed in line 17.

Complexity The complexity of Algorithm 10 is $\mathcal{O}(|V|^{2n} n!)$. Intuitively, the first part ($\mathcal{O}(|V|^{2n})$) comes from the two recursions and the second part ($n!$) comes from the protagonist's ability to change the condition. For a Rabin game of a Rabin automaton, the complexity of the algorithm is $\mathcal{O}((|S_P| + |S_P||\Sigma|)^{2n} n!)$, since $V = V_{\mathbf{P}} \cup V_{\mathbf{A}}$, $V_{\mathbf{P}} = S_P$, and $V_{\mathbf{A}} = S_P \times \Sigma$.

Step 5: Mapping the Rabin Game Solution to a Control Strategy

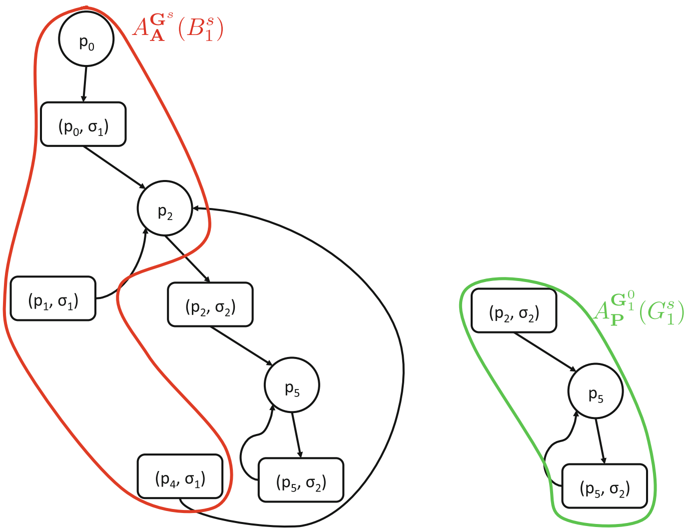
In order to complete the solution to Problem 5.1, we transform a solution to a Rabin game $\mathbf{G} = (V_{\mathbf{P}}, V_{\mathbf{A}}, E, F_{\mathbf{G}})$ of the product automaton $P = T \otimes R$ into a control strategy (X_T^{ϕ}, Ω) for T. The solution to the Rabin game is given as a winning region $W_{\mathbf{P}} \subseteq V_{\mathbf{P}}$ and a winning strategy $\pi_{\mathbf{P}} : W_{\mathbf{P}} \to E$.

We first transform the solution into a memoryless strategy for the product P, and present the solution to Problem 5.2. Clearly, the winning region for P is $W_P = W_{\mathbf{P}}$. The initial winning region is the subset of initial states that belong to W_P, i.e., $W_{P0} =$



(a) $A_{\mathbf{A}}^{\mathbf{G}}(B_2)$ (b) Sub-game G_2^0

Fig. 5.4 Adversary's attractor of B_2 in game $\mathbf{G}$ is shown with a red frame in (**a**). The sub-game $\mathbf{G}_2^0$ obtained by removing $A_{\mathbf{A}}^{\mathbf{G}}(B_2)$ from $\mathbf{G}$ is shown in (**b**). The protagonist's attractor of G_2 in game $\mathbf{G}_2^0$ covers $\mathbf{G}_2^0$



(a) $A_{\mathbf{A}}^{\mathbf{G}^s}(B_1^s)$ (b) $A_{\mathbf{P}}^{\mathbf{G}_1^0}(G_1^s)$

Fig. 5.5 Adversary's attractor set of B_1^s in game $\mathbf{G}^s$ is shown with a red frame in (**a**). The sub-game $\mathbf{G}_1^0$ is shown in (**b**). The protagonist's attractor of G_1 in game $\mathbf{G}_1^0$ covers $\mathbf{G}_1^0$

$W_P \cap S_{P0}$. The strategy π_P is obtained as follows. For all $v \in W_{\mathbf{P}}$, $\pi_P(v) = \sigma$, such that $\pi_{\mathbf{P}}(v) = (v, v')$, and $v' = (v, \sigma)$.

The remaining task is to adapt (W_{P0}, π_P) as a control strategy (X_T^{ϕ}, Ω) for T. Although the control function π_P was memoryless, Ω is history dependent and takes the form of a feedback control automaton:

Definition 5.9 Given a product automaton $P = T \otimes R$, where $T = (X, \Sigma, \delta, O, o)$ and $R = (S, S_0, O, \delta_R, F)$, a winning region W_P for P, and a control strategy (W_{P0}, π_P) for P, a feedback control automaton $C = (S_C, S_{C0}, X, \tau, \Sigma, \pi)$ is defined as

- $S_C = S$ is the set of states,
- $S_{C0} = S_0$ is the set of initial states,
- X is the set of inputs (the set of states of T),
- $\tau : S_C \times X \to S_C$ is the memory update function defined as:

$$\tau(s, x) = \delta_R(s, o(x)) \text{ if } (x, s) \in W_P, \tau(s, x) = \perp \text{ otherwise}$$

- Σ is the set of outputs (the set of inputs of T),
- $\pi : S_C \times X \to \Sigma$ is the output function:

$$\pi(s, x) = \pi_P((x, s)) \text{ if } (x, s) \in W_P, \pi(s, x) = \perp \text{ otherwise.}$$

The set of initial states X_T^{ϕ} of T is given by $\alpha(W_{P0})$, where $\alpha : S_P \to X$ is the projection from states of P to X. The control function Ω is given by C as follows: for a sequence $x_1 \ldots x_n$, $x_1 \in X_T^{\phi}$, we have $\Omega(x_1 \ldots x_n) = \sigma$, where $\sigma = \pi(s_n, x_n)$, $s_{i+1} = \tau(s_i, x_i)$, and $x_{i+1} \in \delta(x_i, \pi(s_i, x_i))$, for all $i \in \{1, \ldots, n-1\}$. It is easy to see that the product automaton of T and C will have the same states as P but contains only transitions of P closed under π_P. Then, all trajectories in $T(X_T^{\phi}, \Omega)$ satisfy ϕ and therefore (X_T^{ϕ}, Ω) is a solution to Problem 5.1. Note that if $p = (x, s) \notin W_{\mathbf{P}}$, then the adversary wins all the plays starting from p regardless of the protagonists choices, which implies that there is always a run starting from the product automaton state $(x, s) \in S_P$ that does not satisfy the Rabin acceptance condition F_P regardless of the applied control function. Therefore, Algorithm 9 finds the largest controlled satisfying region.

Example 5.7 We transform the winning region $W_{\mathbf{P}}$ and the winning strategy $\pi_{\mathbf{P}}$ found in Example 5.6 into a control strategy (X_T^{ϕ}, Ω) for the transition system T and formula Φ from Example 5.1. The memoryless control strategy (W_{P0}, π_P) for the product P (Fig. 5.2) is defined as $W_{P0} = \{p_0\}$, $\pi_P(p_0) = \sigma_1, \pi_P(p_1) = \sigma_2, \pi_P(p_2) = \sigma_2, \pi_P(p_3) = \sigma_1, \pi_P(p_4) = \sigma_2$, and $\pi_P(p_5) = \sigma_2$.

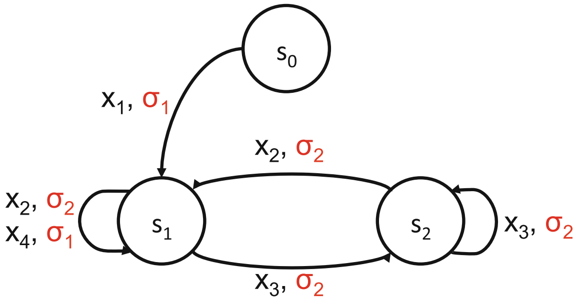


Fig. 5.6 The control automaton from Example 5.7. The initial state is s_0. The *arrows* between states are labeled with the states of the transition system depicting the memory update function. The corresponding control actions are shown in *red*

The set of initial states is $X_T^{\Phi} = \{x_1\}$, and the feedback control automaton $C = (S_C, S_{C0}, X, \tau, \Sigma, \pi)$, that defines the history dependent control function Ω, is constructed as in Definition 5.9. The control automaton is shown in Fig. 5.6 and is formally defined as:

$$S_C = \{s_0, s_1, s_2\},$$
$$S_{C0} = \{s_0\},$$
$$X = \{x_1, x_2, x_3, x_4\},$$
$$\tau(s_0, x_1) = s_1, \tau(s_1, x_2) = s_1, \tau(s_1, x_3) = s_2, \tau(s_1, x_4) = s_1, \tau(s_2, x_2) = s_1, \tau(s_2, x_3) = s_2,$$
$$\Sigma = \{\sigma_1, \sigma_2\},$$
$$\pi(s_0, x_1) = \sigma_1, \pi(s_1, x_2) = \sigma_2, \pi(s_1, x_3) = \sigma_2, \pi(s_1, x_4) = \sigma_1, \pi(s_2, x_2) = \sigma_2, \pi(s_2, x_3) = \sigma_2.$$

Example 5.8 Consider the robot transition system described in Example 1.4, and the motion planning task ϕ described in Example 2.2. The Rabin automaton representation of the formula ϕ is shown in Fig. 5.7a. The Rabin automaton, and therefore the product of the robot transition system and the Rabin automaton, has a single pair (G, B) in its accepting condition. We follow Algorithm 9 and synthesize a control strategy for the robot from the formula ϕ. The robot satisfies the motion planning task if it starts from any region except the dangerous region, i.e., $X_T^{\phi} = \{x_1, x_2, x_3, x_4, x_5, x_7, x_8\}$, and chooses its directions according to the control automaton C depicted in Fig. 5.7b.

When the robot starts from x_1 (B), the control automaton outputs $\pi(s_0, x_1) = W$, and updates its memory from s_0 to $\tau(s_0, x_1) = s_1$. The robot moves West and ends in x_7 (G). The next action is $\pi(s_1, x_7) = N$, and the next control automaton state is $\tau(s_1, x_7) = s_2$. The robot moves North and ends in x_4 (R). Then, the robot moves North again and ends in x_2 (I) as the control automaton outputs $\pi(s_2, x_4) = N$ and updates its memory as $\tau(s_2, x_4) = s_3$. Then, the control automaton outputs $\pi(s_3, x_2) = W$, and updates its memory as $\tau(s_3, x_2) = s_0$. The robot moves West and ends in x_0 (B). Since the robot and the control automaton both are in their initial conditions and all the applied actions are deterministic, the robot continues by applying the same series of actions, and produces the satisfying word:

$$(BGRI)^{\omega}$$

Next, we consider the second motion planning task ψ described in Example 2.2. Again, we apply Algorithm 9 and synthesize a control strategy for the specification formula ψ. The Rabin automaton representation of ψ and the control automaton generated by the algorithm are shown in Fig. 5.8. The set of satisfying initial states are $X_T^{\psi} = \{x_1, x_2, x_3, x_4, x_5, x_7, x_8\}$. When the robot starts from x_1, and chooses its directions according to the control automaton, it produces

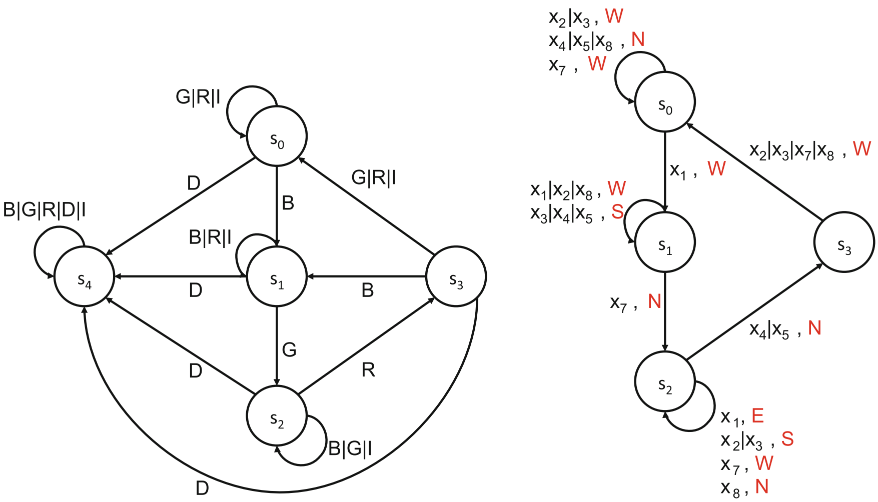
$$BIRG \text{ or } BIRIG,$$

before it returns to x_0, and the control automaton state set to s_0 again. As both the robot and the control automaton are in their initial states, the robot repeatedly produces either $BIRG$ or $BIRIG$. The corresponding word is represented as

$$(BIRG \mid BIRIG)^{\omega}.$$

5.2 Control of Transition Systems from dLTL Specifications

In this section, we present a slightly more efficient and intuitive solution to Problem 5.1 for the case when the LTL specification formula can be translated to a deterministic Büchi automaton. The solution follows the main lines of the method presented in Sect. 5.1 for arbitrary LTL specifications. Instead of the Rabin automaton, we construct a deterministic Büchi automaton, and take its product with the transition system. In this case, the product is a nondeterministic Büchi automaton. We find a control strategy for the product by solving a Büchi game and then transform it to a strategy for the original transition system. This procedure is summarized in Algorithm 11. The details are presented in the rest of this section.



(a) $\phi = \Box\Diamond B \wedge \Box\Diamond G \wedge \Box\Diamond R \wedge \Box\neg D$ (b) Control automaton C

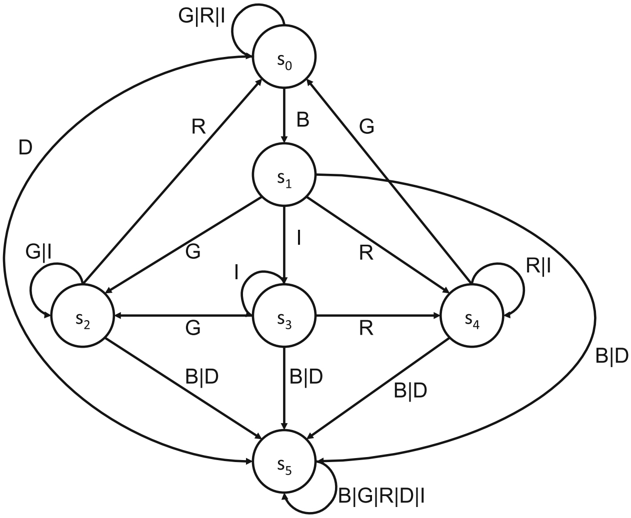
Fig. 5.7 Rabin automaton representation of the specification formula ϕ (**a**) and the control automaton (**b**) from Example 5.8. For the Rabin automaton, s_0 is the initial state. There is a single pair in the accepting condition: $F = \{(G, B)\}$, where $G = \{s_3\}$, and $B = \{s_4\}$. For the control automaton C, s_0 is the initial state. The arrows between states are labeled with the states of the robot transition system depicting the memory update function. The corresponding control actions are shown in *red*. For example $\tau(s_0, x_2) = \tau(s_0, x_3) = s_0$ and the corresponding action is defined as $\pi(s_0, x_2) = \pi(s_0, x_3) = W$. State s_4, which is not reachable from the initial state s_0, is not shown

Algorithm 11 DLTL CONTROL(T, ϕ) : Control strategy (X_T^ϕ, Ω) such that all trajectories in $T(X_T^\phi, \Omega)$ satisfy ϕ

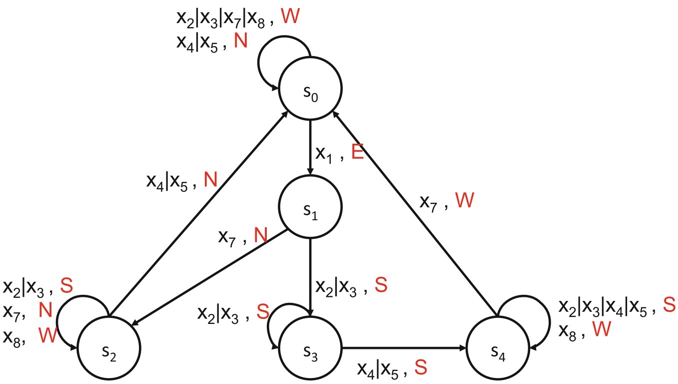
1: Translate ϕ to deterministic Büchi automaton $B = (S, S_0, O, \delta_B, F)$
2: Build a product automaton $P = T \otimes B$
3: Solve a Büchi game
4: Map the solution to a control strategy for the original transition system T

The first step of Algorithm 11 is to translate the dLTL specification Φ into a deterministic Büchi automaton $B = (S, S_0, O, \delta_B, F)$. The second step is the construction of a product automaton P of the transition system $T = (X, \Sigma, \delta, O, o)$ and B. The product automaton $P = (S_P, S_{P0}, \Sigma, \delta_P, F_P)$ is constructed as described in Definition 5.3 with the exception that the set of accepting states of P is defined as $F_P = X \times F$. The product automaton P is a nondeterministic Büchi automaton if T is nondeterministic, otherwise it is a deterministic Büchi automaton.

Each accepting run $\rho_P = (x_1, s_1)(x_2, s_2)(x_3, s_3)\ldots$ of a product automaton $P = T \otimes B$ can be projected into a trajectory $x_1 x_2 x_3 \ldots$ of T, such that the word $o(x_1)o(x_2)o(x_3)\ldots$ is accepted by B (i.e., satisfies ϕ) and vice versa. Similar to the solution proposed in the previous section, this allows us to reduce Problem 5.1 to finding a control strategy for P.



(a) $\psi = \Box\Diamond B \wedge \Box\neg D \wedge \Box(B \Rightarrow \bigcirc(\neg B U G)) \wedge \Box(B \Rightarrow \bigcirc(\neg B U R))$



(b) Control automaton C

Fig. 5.8 Rabin automaton representation of the specification formula ψ (**a**) and the control automaton (**b**) from Example 5.8. For the Rabin automaton, s_0 is the initial state. There is a single pair in the accepting condition: $F = \{(G, B)\}$, where $G = \{s_1\}$, and $B = \{s_5\}$. For the control automaton C, s_0 is the initial state. The arrows between states are labeled with the states of the robot transition system depicting the memory update function. The corresponding control actions are shown in *red*. State s_5, which is not reachable from the initial state s_0, is not shown

Problem 5.3 Given a controlled Büchi product automaton $P=(S_P, S_{P0}, \Sigma, \delta_P, F_P)$, find the largest set of initial states $W_{P0} \subseteq S_{P0}$ for which there exists a control function $\pi_P : S_P \to \Sigma$ such that each run of P under strategy (W_{P0}, π_P) satisfies the Büchi accepting condition F_P.

The solution to Problem 5.3 is summarized in Algorithm 12. The main idea behind the algorithm is to first compute a subset $\overline{F}_P$ of F_P such that a visit to $\overline{F}_P$ can be enforced from $\overline{F}_P$ in a finite number of steps. Then, what remains is to compute the set of all states W_P and a control function π_P such that all runs originating from W_P in closed loop with π_P can reach $\overline{F}_P$ in a one or more steps. By the definition of $\overline{F}_P$, it holds that $\overline{F}_P \subseteq W_P$, and hence these runs satisfy the Büchi condition. To compute $\overline{F}_P$ and π_P, we first define direct and proper attractor sets of a set $S \subseteq S_P$ for a Büchi automaton P:

Definition 5.10 (*Direct attractor*) The direct attractor of a set S, denoted by $\mathsf{A}^1(S)$, is defined as the set of all $s \in S_P$ from which there can be enforced a visit to S in one step:

$$\mathsf{A}^1(S) = \{s \in S_P \mid \exists \sigma \in \Sigma, \delta_P(s, \sigma) \subseteq S\}.$$

The direct attractor set induces a strategy $\pi_P^{1,S} : \mathsf{A}^1(S) \to \Sigma$ such that

$$\delta_P(s, \pi_P^1(s)) \subseteq S.$$

Definition 5.11 (*Proper attractor*) The proper attractor of a set S, denoted by $\mathsf{A}^+(S)$, is defined as the set of all $s \in S_P$ from which there can be enforced a visit to S in one or more steps.

The proper attractor set $\mathsf{A}^+(S)$ of a set S can be computed iteratively via the converging sequence

$$\mathsf{A}^1(S) \subseteq \mathsf{A}^2(S) \subseteq \dots,$$

where $\mathsf{A}^1(S)$ is the direct attractor of S, and

$$\mathsf{A}^{i+1}(S) = A^1(A^i(S) \cup S) \cup \mathsf{A}^i(S).$$

Intuitively, $\mathsf{A}^i(S)$ is the set from which a visit to S in at most i steps can be enforced by choosing proper controls. The *attractor strategy* $\pi_P^{+,S}$ for $\mathsf{A}^+(S)$ is defined from the direct attractor strategies computed through the converging sequence as follows:

$$\pi_P^{+,S} = \pi_P^{1,\mathsf{A}^i(S)}(s), \text{ for all} s \in \mathsf{A}^{i+1}(S) \setminus \mathsf{A}^i(S).$$

A *recurrent set* of a given set A is the set of states from which infinitely many revisits to A can be enforced. In Algorithm 12, first the recurrent set $\overline{F}_P$ of F_P is computed with an iterative process (lines 3–6). Note that we start with $\overline{F}_P = F_P$ and iteratively remove the states from which a revisit to $\overline{F}_P$ can not be guaranteed. This loop terminates after a finite number of iterations since F_P is a finite set. The

Algorithm 12 BÜCHIGAME ($P = (S_P, S_{P0}, \Sigma, \delta_P, F_P)$): Winning region $W_P \subseteq S_P$ and winning strategy π_P.

1: $\overline{F}_P = \emptyset$
2: $\overline{F}_P^{new} = F_P$
3: **while** $\overline{F}_P \neq \overline{F}_P^{new}$ **do**
4: $\quad \overline{F}_P = \overline{F}_P^{new}$
5: $\quad \overline{F}_P^{new} = \mathsf{A}^+(\overline{F}_P) \cap \overline{F}_P$
6: **end while**
7: $W_P = \mathsf{A}^+(\overline{F}_P)$, compute the corresponding attractor strategy π_P

termination guarantees $\overline{F}_P \subseteq \mathsf{A}^+(\overline{F}_P)$, and hence infinitely many revisits to $\overline{F}_P$ from $\overline{F}_P$ can be enforced. In the final step of the algorithm the proper attractor of $\overline{F}_P$ and the corresponding attractor strategy is computed. As $\overline{F}_P \subseteq \mathsf{A}^+(\overline{F}_P)$, π_P is an attractor strategy that solves the Büchi game for all $s \in W_P$.

Complexity The time complexity of Algorithm 12 is $\mathscr{O}(|S_P|^2|\Sigma|)$.

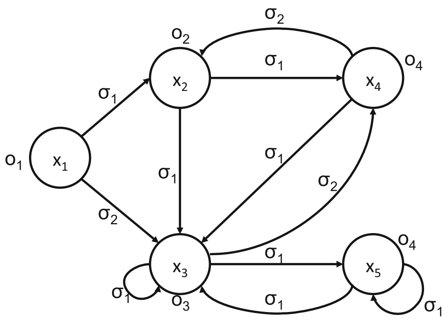
Remark 5.1 A Büchi automaton B can be interpreted as a Rabin automaton with a single pair $\{(G_1, B_1)\}$ in its accepting condition, where $G_1 = \overline{F}_P$ and $B_1 = \emptyset$. Consequently, Algorithm 10 for the Rabin game can be used for the Büchi automaton to solve Problem 5.3. In this particular case, $n = 1$ and the time complexity of Algorithm 10 is $\mathscr{O}((|S_P| + |S_P||\Sigma|)^2)$.

The final step of the dLTL control algorithm is to translate the control strategy (W_{P0}, π_P) obtained from Algorithm 12 into a control strategy (X_T^ϕ, Ω) for T, where $W_{P0} = W_P \cap S_{P0}$. As in the solution presented for LTL specifications in the previous section, although the control function π_P is memoryless, Ω is history dependent and takes the form of a feedback control automaton. The control automaton is constructed from P and π_P as in Definition 11, and the control function Ω is defined by the control automaton. Finally, the set of initial states X_T^ϕ of T is given by $\alpha(W_{P0})$, where $\alpha : S_P \to X$ is the projection from states of P to X.

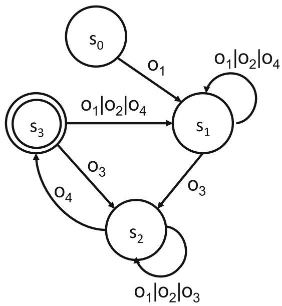
Example 5.9 Consider the nondeterministic transition system T shown in Fig. 5.9a and the following LTL formula over its set of observations:

$$\phi = o_1 \wedge \Box(\Diamond o_3 \wedge \Diamond o_4).$$

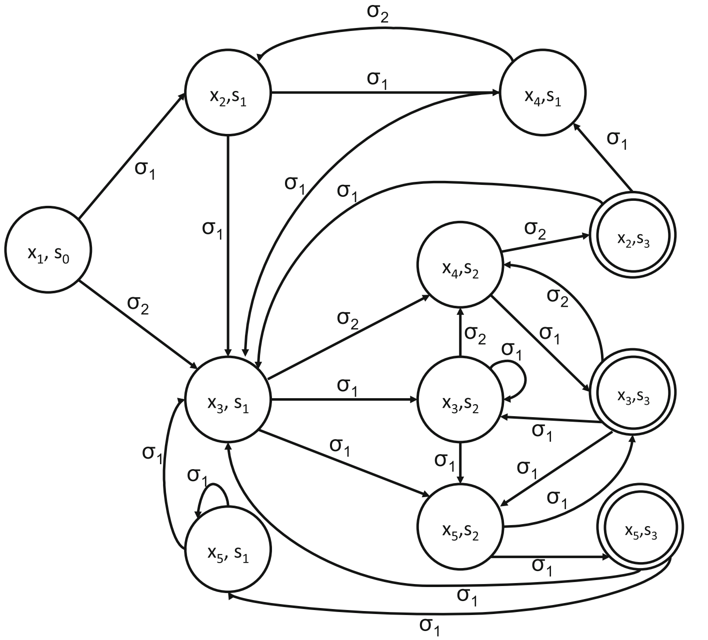
We follow Algorithm 11 to find the control strategy (X_T^Φ, Ω) that solves Problem 5.1 for the transition system T and formula ϕ.



(a) Transition system T



(b) $\phi = o_1 \wedge \Box(\diamond o_3 \wedge \diamond o_4)$



(c) The product of T (a) and B (b)

Fig. 5.9 Transition system (**a**), Büchi automaton (**b**), and their product (**c**) from Example 5.9. For the Büchi automaton, s_0 is the initial state and s_3 is the accepting state. For the product automaton, (x_1, s_0) is the initial state, and $\{(x_2, s_3), (x_3, s_3), (x_5, s_3)\}$ is the set of accepting states. The states that are not reachable from the non-blocking initial state (x_1, s_0) are not shown in (**c**)

We first construct a deterministic Büchi automaton B (Fig. 5.9b) that accepts the language satisfying the formula. Then, we construct the product of the system and the automaton. The product automaton P, which is shown in Fig. 5.9c, is a non-deterministic Büchi automaton since T is nondeterministic. Note that the states that are not reachable from non-blocking initial states are removed from P and are not shown in Fig. 5.9c.

To find a control strategy for P, we follow Algorithm 12. In the first iteration, $\overline{F}_P = \{(x_2, s_3), (x_3, s_3), (x_5, s_3)\}$ (F_P) and we compute the proper attractor of F_P as follows:

$$\begin{aligned}
\mathsf{A}^1(F_P) &= \{(x_4, s_2), (x_5, s_2)\},\\
\mathsf{A}^2(F_P) &= \{(x_3, s_1), (x_3, s_2), (x_3, s_3), (x_4, s_2), (x_5, s_2)\},\\
\mathsf{A}^3(F_P) &= \{(x_1, s_0), (x_4, s_1), (x_3, s_1), (x_3, s_2), (x_3, s_3), (x_4, s_2), (x_5, s_2)\}\\
\mathsf{A}^4(F_P) &= \{(x_2, s_1), (x_2, s_3), (x_1, s_0), (x_4, s_1), (x_3, s_1), (x_3, s_2),\\
&\qquad (x_3, s_3), (x_4, s_2), (x_5, s_2)\}
\end{aligned}$$

The sequence converges at iteration 4 and $\mathsf{A}^+(F_P) = \mathsf{A}^4(F_P)$. In the first iteration of the main loop of Algorithm 12, $\overline{F}_P^{new} = \{(x_2, s_3), (x_3, s_3)\}$, and (x_5, s_3) is eliminated. The main loop terminates after the second iteration as

$$\mathsf{A}^+(\overline{F}_P) \cap \overline{F}_P = \overline{F}_P, \text{ where } \overline{F}_P = \{(x_2, s_3), (x_3, s_3)\}.$$

As the last step of Algorithm 12, we compute $W_P = \mathsf{A}^+(\{(x_2, s_3), (x_3, s_3)\})$, and the corresponding attractor strategy as follows:

$$\begin{aligned}
&\mathsf{A}^1(\overline{F}_P) = \{(x_4, s_2)\}, && \pi_P((x_4, s_2)) = \sigma_1,\\
&\mathsf{A}^2(\overline{F}_P) = \{(x_3, s_3), (x_3, s_2), (x_3, s_1)\} \cup \mathsf{A}^1(\overline{F}_P),\\
& && \pi_P((x_3, s_3)) = \sigma_2, \pi_P((x_3, s_2)) = \sigma_2, \pi_P((x_3, s_1)) = \sigma_2,\\
&\mathsf{A}^3(\overline{F}_P) = \{(x_1, s_0), (x_4, s_1)\} \cup \mathsf{A}^2(\overline{F}_P), && \pi_P((x_1, s_0)) = \sigma_2, \pi_P((x_4, s_1)) = \sigma_1,\\
&\mathsf{A}^4(\overline{F}_P) = \{(x_2, s_1), (x_2, s_3)\} \cup \mathsf{A}^3(\overline{F}_P), && \pi_P((x_2, s_1)) = \sigma_1, \pi_P((x_2, s_3)) = \sigma_1,\\
&\mathsf{A}^4(\overline{F}_P) = \mathsf{A}^5(\overline{F}_P) = \mathsf{A}^+(\overline{F}_P).
\end{aligned}$$

The control strategy (W_{P0}, π_P) solves Problem 5.3 for P, where $W_{P0} = \{(x_1, s_0)\}$ and π_P is as defined above. The final step is the transformation of (W_{P0}, π_P) into a control strategy (X_T^{Φ}, Ω) for T. The set of initial states is $X_T^{\Phi} = \{x_1\}$, and the feedback control automaton $C = (S_C, S_{C0}, X, \tau, \Sigma, \pi)$ (shown in Fig. 5.10), which defines the history dependent control function Ω, is constructed as in Definition 5.9, and formally defined as:

$S_C = \{s_0, s_1, s_2, s_3\}$,
$S_{C0} = \{s_0\}$,
$X = \{x_1, x_2, x_3, x_4, x_5\}$,
$\tau(s_0, x_1) = s_1, \tau(s_1, x_2) = s_1, \tau(s_1, x_3) = s_2, \tau(s_1, x_4) = s_1, \tau(s_2, x_3) = s_2,$
$\tau(s_2, x_4) = s_3,\ \tau(s_3, x_2) = s_1,\ \tau(s_3, x_3) = s_2$
$\Sigma = \{\sigma_1, \sigma_2\}$,
$\pi(s_0, x_1) = \sigma_2, \pi(s_1, x_2) = \sigma_1, \pi(s_1, x_3) = \sigma_2, \pi(s_1, x_4) = \sigma_1, \pi(s_2, x_3) =$
$\sigma_2, \pi(s_2, x_4) = \sigma_1, \pi(s_3, x_2) = \sigma_1, \pi(s_3, x_3) = \sigma_2.$

Remark 5.2 In a Büchi game over a product automaton $P = T \otimes B$, a state (x, s) is added to W_P only if there is a control strategy guaranteeing that all runs ρ_P of P originating from (x, s) satisfy that the projection of ρ_P onto S (Büchi automaton states) is an accepting run of B. The condition is necessary to guarantee that each run of T that originate from x is satisfying. While the product is a non-deterministic Büchi automaton, this condition is stronger than the Büchi acceptance: a word is accepted by a non-deterministic Büchi automaton if there exists an accepting run. In other words, it is not necessary that all runs are accepting. Due to this difference in the notion of the satisfying TS states and non-deterministic Büchi acceptance, an algorithm similar to Algorithm 11 cannot be used for non-deterministic Büchi automaton, which is illustrated in Example 5.10.

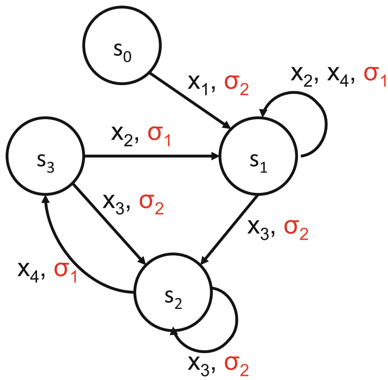
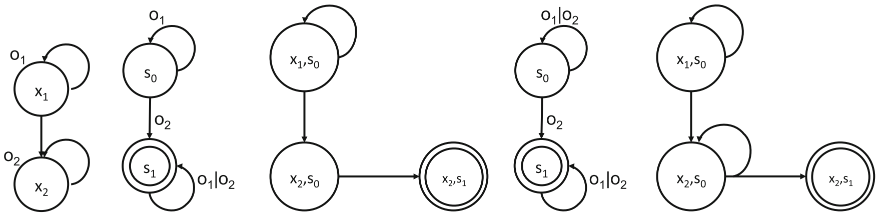


Fig. 5.10 The control automaton obtained by solving the Büchi game on the product automaton shown in Fig. 5.9c



(a) T (b) $\phi = \diamond o_2$, B_1 (c) $P_1 = T \otimes B_1$ (d) $\phi = \diamond o_2$, B_2 (e) $P_2 = T \otimes B_2$

Fig. 5.11 Transition system T (**a**), Büchi automata B_1 (**b**) and B_2 (**c**), the product of T and B_1 (**d**), and the product of T and B_2 (**e**) from Example 5.10

Example 5.10 Consider the transition system T shown in Fig. 5.11a and specification $\phi = \diamond o_2$ over its set of observations. A deterministic Büchi automaton and a non-deterministic Büchi automaton that accept the language satisfying the formula are shown in Figs. 5.11b and 5.11d, respectively. The corresponding product automata are shown in Figs. 5.11c and 5.11e. T has a single run $x_2x_2x_2\ldots$ originating from x_2 and it satisfies ϕ_1. Due to the non-determinism of T, there are multiple runs originating from x_1. The run $x_1x_1x_1\ldots$ originating from x_1 produces the word $o_1o_1o_1\ldots$ that violates the formula, and all other runs originating from x_1 satisfy the formula. Therefore, we have $X_T^\phi = \{x_2\}$. We can easily verify this observation by running Algorithm 12 on the product automaton P_1 with $\Sigma = \{\sigma\}$, i.e., a single control input labels all the transitions. The algorithm returns $W_P = \mathsf{A}^+(\overline{F}_P) = \{(x_2, s_0), (x_2, s_1)\}$, hence $W_{P0} = \{(x_2, s_0)\}$ and $X_T^\phi = \{x_2\}$.

Now, consider the product P_2 (Fig. 5.11e) of T and the non-deterministic Büchi automaton B_2 accepting the same language as B_1. It is not possible to differentiate (x_1, s_0) and (x_2, s_0) on P_2 via reachability analysis or recurrence set construction, since satisfying and violating runs originate from both (x_1, s_0) and (x_2, s_0).

5.3 Control of Transition Systems from scLTL Specifications

A solution to Problem 5.1 is found more efficiently when the specification ϕ is an scLTL formula. This is due to the simple FSA acceptance condition for scLTL formulas. The solution we present here resembles the one we presented in Sect. 5.1

for arbitrary LTL specifications. The procedure involves the construction of an FSA from the specification formula ϕ, the construction of the product automaton of the system and the FSA, solving a reachability problem on the product automaton to find a control strategy for the product automaton, and finally translation of this strategy to the transition system. While a control strategy for the product automaton was found by solving a Rabin game in Sect. 5.1 and a Büchi game in Sect. 5.2, the product of an FSA and a nondeterministic transition system is a nondeterministic finite state automaton (NFA), for which a control strategy can be found by solving a reachability problem. This step can be interpreted as finding the attractor set of the accepting states of the product automaton and the corresponding control strategy. Moreover, when T is deterministic, the product is an FSA and the largest controlled satisfying region can simply be found by traversing the graph of the product automaton starting from its set of accepting states. The procedure for determining control strategies for non-deterministic transition systems from scLTL formulas is summarized in Algorithm 13. The details are presented in the rest of this section.

Algorithm 13 SCLTL CONTROL(T, ϕ) : Control strategy (X_T^ϕ, Ω) such that all trajectories in $T(X_T^\phi, \Omega)$ satisfy scLTL formula ϕ

1: Translate ϕ into an FSA $A = (S, s_0, O, \delta_A, F)$
2: Build a product automaton $P = T \otimes R$
3: Solve a reachability problem on the graph of the product automaton
4: Map the solution to the reachability problem to a control strategy for the original transition system T

The first step of Algorithm 13 is to translate the scLTL specification ϕ into an FSA $A = (S, s_0, O, \delta_A, F)$. This can be done using off-the-shelf tool as discussed in Sect. 2.3. The second step is the construction of a product automaton P of the transition system $T = (X, \Sigma, \delta, O, o)$ and the FSA A. The product automaton $P = (S_P, S_{P0}, \Sigma, \delta_P, F_P)$ is constructed as described in Definition 5.3 with the exception that the set of accepting states of P is defined as $F_P = X \times F$. The product automaton P is a NFA if T is nondeterministic, and it is an FSA if T is deterministic.

Each accepting run $\rho_P = (x_1, s_1)(x_2, s_2) \ldots (x_n, s_n)$ of the product automaton P can be projected into a trajectory $x_1 x_2 \ldots x_n$ of T, such that the word $o(x_1)o(x_2) \ldots o(x_n)$ is accepted by A (i.e., all words that contain the prefix $o(x_1)$ $o(x_2) \ldots o(x_n)$ satisfies ϕ) and vice versa. Analogous to the solution for arbitrary LTL specifications presented in Sect. 5.1, this allows us to reduce Problem 5.1 to finding a control strategy (W_0, π) for P, which is defined as in Definition 5.4.

Problem 5.4 Given a product NFA $P = (S_P, S_{P0}, \Sigma, \delta_P, F_P)$, find the largest set of initial states $W_{P0} \subseteq S_{P0}$ for which there exists a control function $\pi_P : S_P \to \Sigma$ such that each run of P under the strategy (W_{P0}, π_P) reaches the set of accepting states F_P.

We use W_P to denote the set of states of P from which a visit to the set of accepting states can be enforced by a control function. This set and the corresponding control strategy can easily be computed with a single attractor computation:

$$W_P = F_P \cup \mathsf{A}^+(F_P),$$

where $\mathsf{A}^+(F_P)$ is the proper attractor of F_P and π_P is the corresponding attractor strategy, which are described in Definition 5.11.

This computation results in a control strategy (W_{P0}, π_P) that solves Problem 5.4, where $W_{P0} = W_P \cap S_{P0}$. The final step of Algorithm 13 is the transformation of the control strategy (W_{P0}, π_P) for the product P into a control strategy (X_T^Φ, Ω) for T. The control function Ω for T is history dependent and takes the form of a feedback control automaton $C = (S_C, S_{C0}, X, \tau, \Sigma, \pi)$, which is constructed from P, T and A as described in Definition 5.9. The set of initial states X_T^ϕ of T is given by $\alpha(W_{P0})$, where $\alpha : S_P \to X$ is the projection from states of P to X. The control function Ω is given by C as explained in Sect. 5.1. The product automaton of T and C will have the same states as P but contains only transitions of P closed under π_P. Then, all trajectories in $T(X_T^\phi, \Omega)$ satisfy ϕ. Moreover, if $(x_1, s_1) \notin W_P$, then $\delta_P((x_1, s_1), \sigma) \nsubseteq W_P$ for all $\sigma \in \Sigma$, which implies that there exists a run of P that originate at (x_1, s_1) and can not reach F_P regardless of the applied control function. Therefore, in the case when ϕ is an scLTL formula, X_T^ϕ is the largest controlled satisfying region and the strategy (X_T^ϕ, Ω) obtained from Algorithm 13 is a solution to Problem 5.1.

Complexity The complexity of finding the control strategy for the product automaton P (step 3 of Algorithm 13) is $\mathscr{O}(|S_P||\Sigma|)$, since an attractor set is computed in maximum $\mathscr{O}(|S_P||\Sigma|)$ iterations.

Example 5.11 Consider the nondeterministic transition system T shown in Fig. 5.12a and the scLTL formula over its set of observations:

$$\phi = \Diamond o_4 \wedge (\neg o_3 U o_4) \wedge (\neg o_4 U o_2).$$

We follow Algorithm 13 to find the control strategy (X_T^Φ, Ω) that solves Problem 5.1 for transition system T and formula ϕ. We first construct an FSA A (Fig. 5.12b) that accepts the good prefixes of the formula. Then, we construct the product of the system and the FSA. The product automaton P, which is shown in Fig. 5.12c, is an NFA since T is nondeterministic. Note that the states that are not reachable from non-blocking initial states are removed from P and are not shown in Fig. 5.12c.

To find a control strategy for P, we compute the converging sequence W_P^{i*} and control function π_P:

$$
\begin{aligned}
&W_P^{0*} = \{(x_5, s_2)\}, && \pi_P((x_5, s_2)) = \sigma_1\\
&W_P^{1*} = \{(x_5, s_1)\} \cup W_P^{0*}, && \pi_P((x_5, s_1)) = \sigma_1\\
&W_P^{2*} = \{(x_2, s_0), (x_2, s_1)\} \cup W_P^{1*}, && \pi_P((x_2, s_0)) = \sigma_1, \pi_P((x_2, s_1)) = \sigma_1\\
&W_P^{3*} = \{(x_4, s_0), (x_4, s_1)\} \cup W_P^{2*}, && \pi_P((x_4, s_0)) = \sigma_1, \pi_P((x_4, s_1)) = \sigma_1\\
&W_P^{4*} = W_P^{3*}.
\end{aligned}
$$

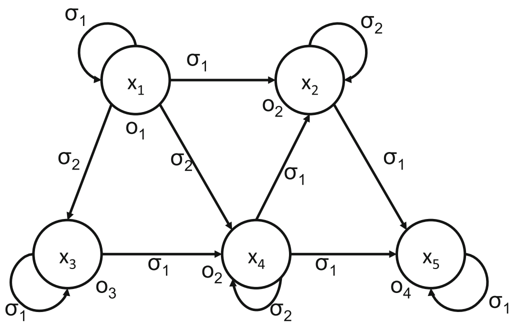
The control strategy (W_{P0}, π_P) solves Problem 5.4 for P, where $W_{P0} = \{(x_2, s_0), (x_4, s_0)\}$ and π_P is as defined above. The final step is the transformation of (W_{P0}, π_P) into a control strategy (X_T^{Φ}, Ω) for T. The set of initial states is $X_T^{\Phi} = \{x_2, x_4\}$, and the feedback control automaton $C = (S_C, S_{C0}, X, \tau, \Sigma, \pi)$, that defines the history dependent control function Ω, is constructed as in Definition 5.9, and formally defined as:

$$
\begin{aligned}
&S_C = \{s_0, s_1, s_2\},\\
&S_{C0} = \{s_0\},\\
&X = \{x_1, x_2, x_3, x_4, x_5\},\\
&\tau(s_0, x_2) = s_1, \tau(s_0, x_4) = s_1, \tau(s_1, x_2) = s_1, \tau(s_1, x_4) = s_1, \tau(s_1, x_5) = s_2,\\
&\tau(s_2, x_5) = s_2,\\
&\Sigma = \{\sigma_1, \sigma_2\},\\
&\pi(s_0, x_2) = \sigma_1, \pi(s_0, x_4) = \sigma_1, \pi(s_1, x_2) = \sigma_1, \pi(s_1, x_4) = \sigma_1, \pi(s_1, x_5) =\\
&\sigma_1, \pi(s_2, x_5) = \sigma_1.
\end{aligned}
$$

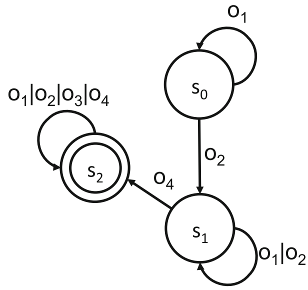
5.4 Notes

We presented a complete treatment of the LTL control problem for a finite transition system. If the transition system is deterministic, the problem can be solved through model-checking-based techniques (see Chap. 3). Indeed, an off-the-shelf model checker can be used to model check the system against the negation of the formula. If the negation of the formula is not satisfied at a state, i.e., there exists a run violating the negation of the formula, then it is returned as a certificate of violation. This run, which satisfies the formula, can be enforced in the deterministic transition system by choosing appropriate controls at the states in the run. This approach was used in [105] to develop a conservative solution to an LTL control problem for a continuous-time, continuous space linear system.

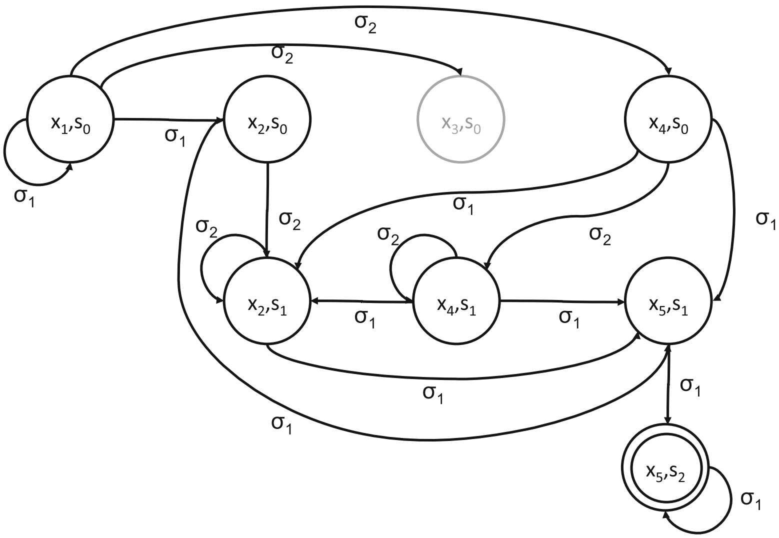
In this chapter, we focused on the case when the transition system is nondeterministic. We showed that, in the most general case, the problem can be reduced to a Rabin game [146]. There are various approaches to solve Rabin games [55, 90,



(a) Transition system T



(b) $\phi = \diamond o_4 \wedge (\neg o_3 U o_4) \wedge (\neg o_4 U o_2)$



(c) The product of T (a) and FSA (b)

Fig. 5.12 Transition system (**a**), the FSA (**b**), and the product of them (**c**) from Example 5.11. For the FSA, s_0 is the initial state and s_2 is the accepting state. For the product automaton, $\{(x_1, s_0), (x_2, s_0), (x_3, s_0), (x_4, s_0)\}$ is the set of initial states, and (x_5, s_2) is the accepting state. The blocking state (x_3, s_0) that is reachable from a non-blocking initial state (x_1, s_0) is shown in *grey*

141]. The solution we presented is based on [90]. The Rabin game based approach to the control problem from this chapter is based on [170]. For the particular case when the LTL formula can be translated to a deterministic Büchi automaton, we showed that the control problem reduced to a Büchi game [38], for which efficient solutions exist [167]. A treatment of the control problem for this case can be found in [104]. Finally, if the specification is given in the syntactically co-safe fragment of LTL, called scLTL [156], then the solution reduced to a reachability problem, for which we propose an efficient algorithm. In all three cases mentioned above, the control strategy for the original transition system takes the form of a feedback control automaton, which is easy to interpret and implement.

For simplicity of exposition, we only consider synthesis from LTL specifications. Readers interested in CTL and CTL* specifications are referred to [9, 57, 92]. There has also been some interest in combining optimality with correctness in formal synthesis. Examples include optimal LTL control for transition systems [51, 161, 171] and Markov decision processes [40, 52, 160], and optimization problems with costs formulated using the quantitative semantics of logics such as signal temporal logic (STL) and metric temporal logic (MTL) [12, 19, 54, 59, 94, 95, 107, 176].