Studies in Systems, Decision and Control 89

Calin Belta
Boyan Yordanov
Ebru Aydin Gol

Formal Methods for Discrete-Time Dynamical Systems

Springer

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

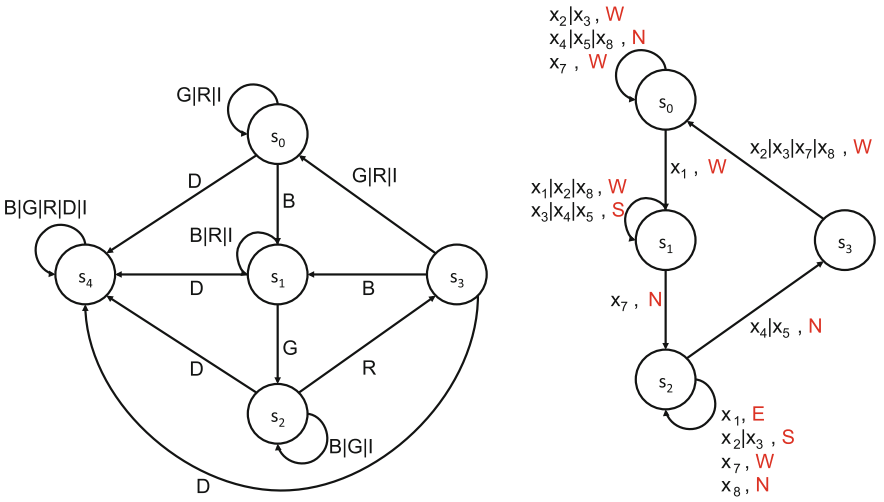
$$BIRG \text{ or } BIRIG,$$

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) $\phi = \square\diamondsuit B \wedge \square\diamondsuit G \wedge \square\diamondsuit R \wedge \square\neg D$ (b) Control automaton C

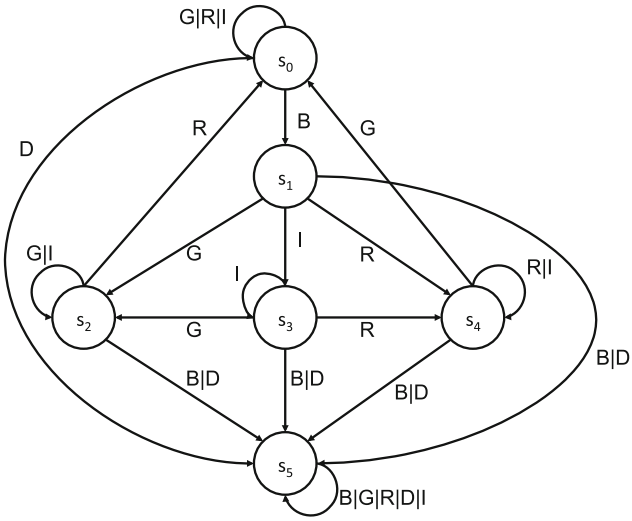
Fig. 5.7 Rabin automaton representation of the specification formula ϕ (**a**) and the control automaton (**b**) from Example 5.8. For the Rabin automaton, s_0 is the initial state. There is a single pair in the accepting condition: $F = \{(G, B)\}$, where $G = \{s_3\}$, and $B = \{s_4\}$. For the control 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

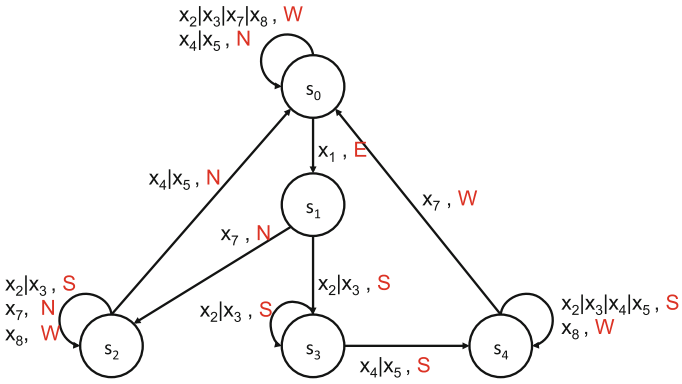
1: Translate ϕ to deterministic Büchi automaton $B = (S, S_0, O, \delta_B, F)$
2: Build a product automaton $P = T \otimes B$
3: Solve a Büchi game
4: Map the solution to a control strategy for the original transition system T

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

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

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 $\mathcal{O}(|S_P|^2|\Sigma|)$.

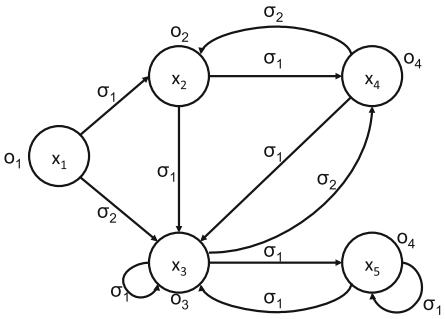
Remark 5.1 A Büchi automaton B can be interpreted as a Rabin automaton with a single pair $\{(G_1, B_1)\}$ in its accepting condition, where $G_1 = \overline{F}_P$ and $B_1 = \emptyset$. Consequently, Algorithm 10 for the Rabin game can be used for the Büchi automaton to solve Problem 5.3. In this particular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

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.

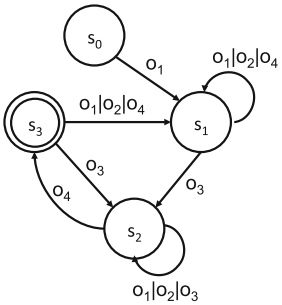
Example 5.9 Consider the nondeterministic transition system T shown in Fig. 5.9a and the following LTL formula over its set of observations:

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

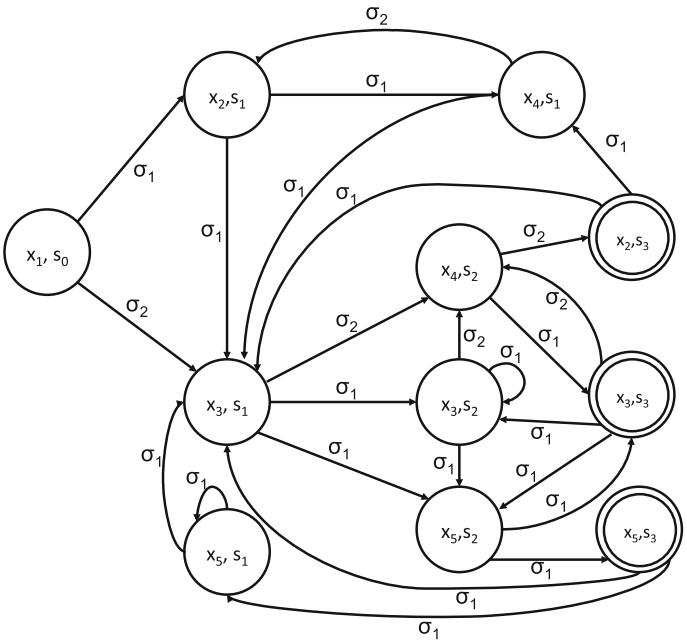
We follow Algorithm 11 to find the control strategy (X_T^Φ, Ω) that solves Problem 5.1 for the transition system T and formula ϕ.



(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:

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

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:

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

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.

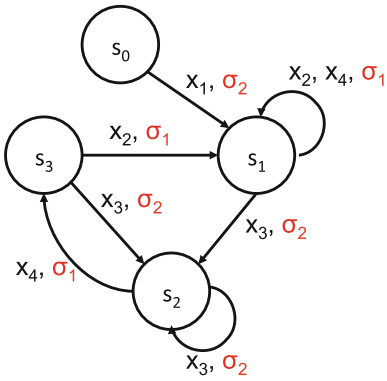
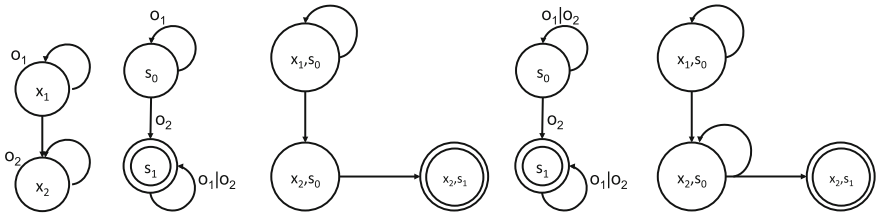


Fig. 5.10 The control automaton obtained by solving the Büchi game on the product automaton shown in Fig. 5.9c



(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 $x_2 x_2 x_2 \ldots$ 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 $x_1 x_1 x_1 \ldots$ originating from x_1 produces the word $o_1 o_1 o_1 \ldots$ 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 nondeterministic 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 $\rho_P = (x_1, s_1)(x_2, s_2) \dots (x_n, s_n)$ of the product automaton P can be projected into a trajectory $x_1 x_2 \dots x_n$ of T, such that the word $o(x_1)o(x_2) \dots o(x_n)$ is accepted by A (i.e., all words that contain the prefix $o(x_1)$ $o(x_2) \dots o(x_n)$ 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 $\mathcal{O}(|S_P||\Sigma|)$, since an attractor set is computed in maximum $\mathcal{O}(|S_P||\Sigma|)$ 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:

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

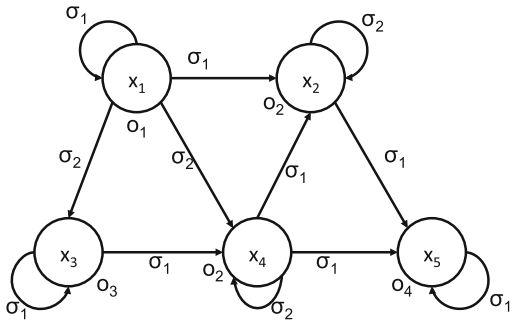
The control strategy (W_{P0}, π_P) solves Problem 5.4 for P, where $W_{P0} = \{(x_2, s_0), (x_4, s_0)\}$ and π_P is as defined above. The final step is the transformation of (W_{P0}, π_P) into a control strategy (X_T^Φ, Ω) for T. The set of initial states is $X_T^\Phi = \{x_2, x_4\}$, and the feedback 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:

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

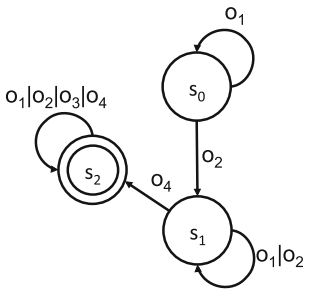
5.4 Notes

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.

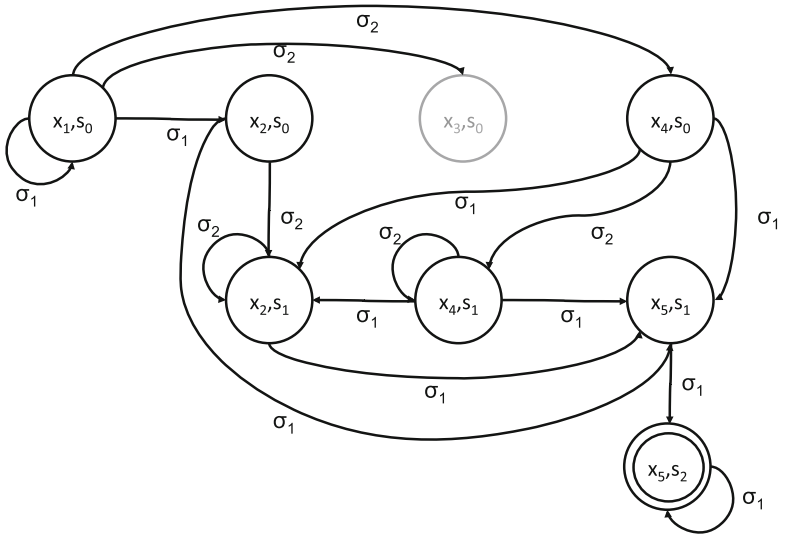
In this chapter, we focused on the case when the transition system is nondeterministic. We showed that, in the most general case, the problem can be reduced to a Rabin game [146]. There are various approaches to solve Rabin games [55, 90,



(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].