Knowledge Representation and Reasoning

Exercise Sheet 13

UNI FREIBURG

Albert-Ludwigs-Universität Freiburg

Bernhard Nebel, Gregor Behnke, Thorsten Engesser, Rolf-David Bergdoll, Leonardo Mieschendahl, Johannes Herrmann

February 4th, 2022

Exercise 13.1

Reactive Synthesis

UNI FREIBURG

Exercise 13.1 (Reactive Synthesis, 2+2+2)

We want to use LTL to model a pedestrian crossing with traffic lights, using the following atoms:

- p: the pedestrian light is set to green
- t: the traffic light is set to green
- c: the pedestrian call button was pressed

In this scenario the control of the lights is the output of the system, while the pressing of the call button is the input: $\mathcal{A}_{in} = \{c\}$, $\mathcal{A}_{out} = \{p, t\}$. The control system should (1) start with the traffic light set to green, (2) always set exactly one of the two lights to green and (3) set the pedestrian light to green if and only if the call button was pressed and the light is not green already. This translates to the following LTL-formula φ:

$$\varphi = t \wedge G(p \leftrightarrow \neg t) \wedge G((c \wedge \neg p) \leftrightarrow Xp)$$

(a) Construct a deterministic parity automaton for φ. You do not have to use a specific algorithm or construct an intermediate Büchi automaton. However, you should make sure that your transition function is complete; in every state every possible truth assignment over the atoms should be covered by an outgoing edge.

Hint: Your DPA should not require more than three states.

(b) Transform your DPA into a parity game, using the construction presented in the lecture.

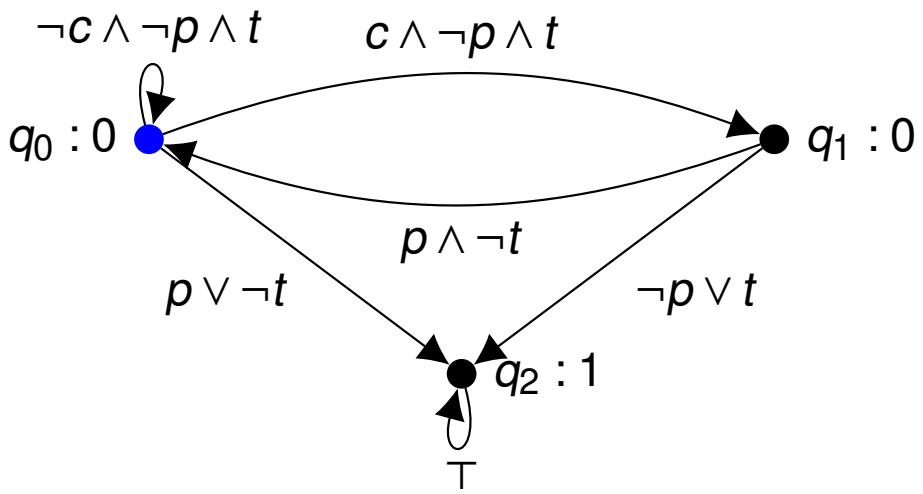
(c) Apply Zielonka's algorithm to your parity game to find its winning sets. Finally, extract the solution strategy for the reactive synthesis problem of φ over $\mathcal{A}$ from those sets.

Exercise 13.1 (a)

DPA Construction

$$\varphi = t \wedge G(p \leftrightarrow \neg t) \wedge G((c \wedge \neg p) \leftrightarrow Xp)$$

- We use two accepting states between which we jump, depending on whether c was read.
- Setting the output variables in a way that violates φ leads to a non-accepting terminal state.

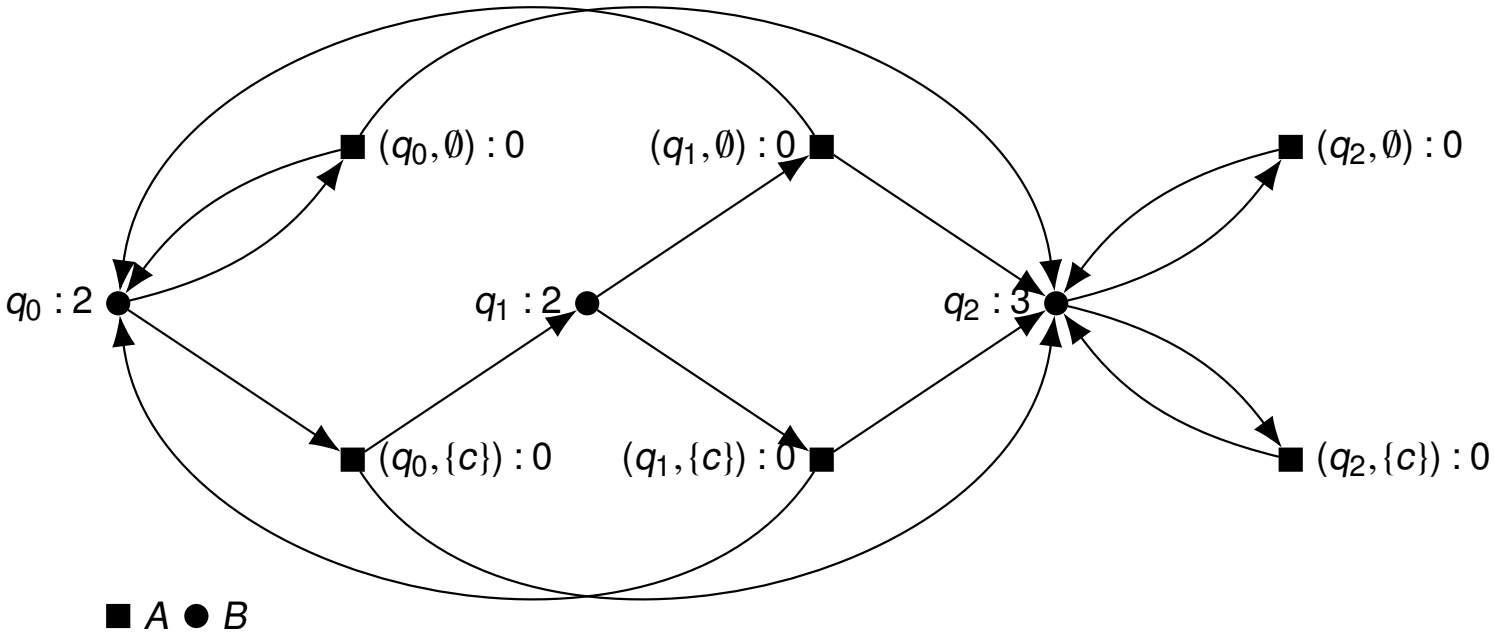


Each state q is labeled $q : \alpha(q)$

Exercise 13.1 (b)

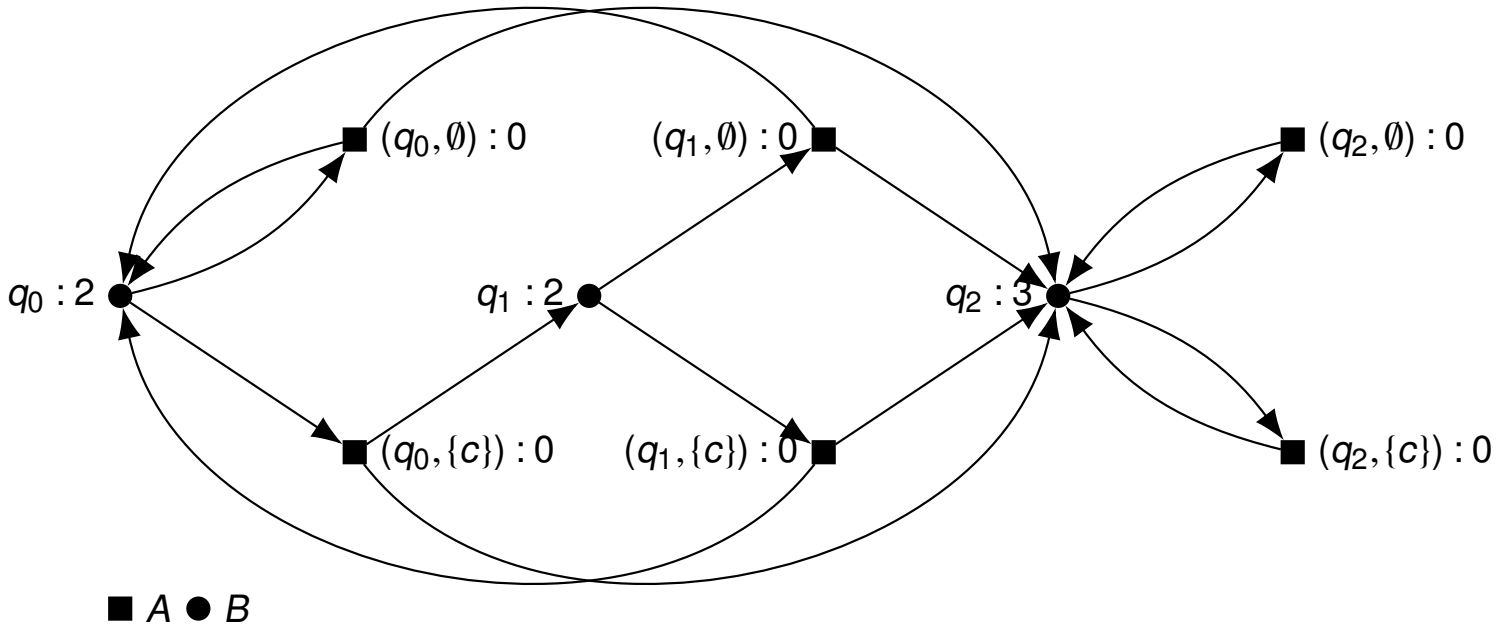
DPA to Parity Game

The DPA from (a) translates to the following parity game $G, alpha$:



Exercise 13.1 (c)

Zielonka's Algorithm



$Sol_A(G, \alpha)$:

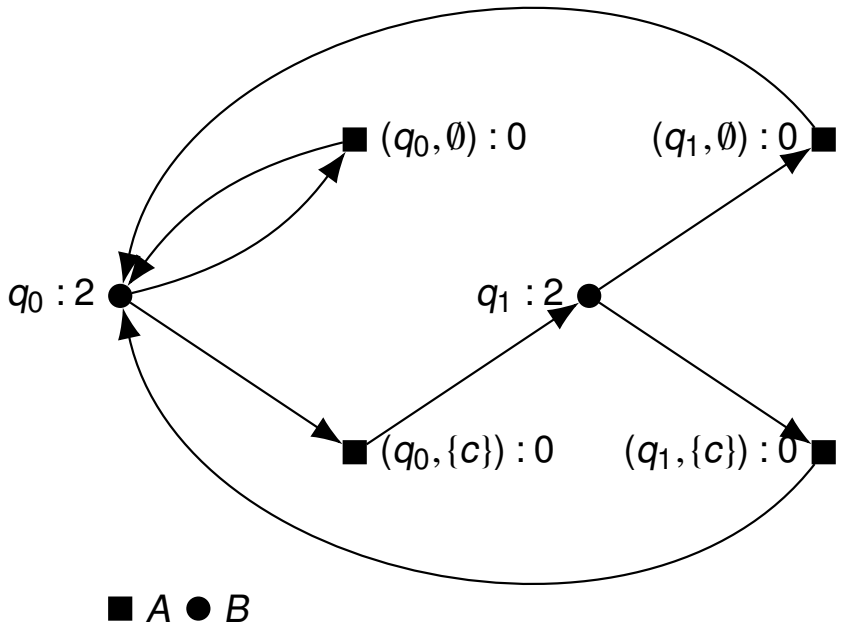
- $h = 3 \Rightarrow$ Compute $G \setminus Sol_B(G, \alpha)$

$Sol_B(G, \alpha)$:

- $N_h = \{q_2\}$
- $\mathfrak{A}_B(N_h) = \{q_2, (q_2, \emptyset), (q_2, \{c\})\}$

Exercise 13.1 (c)

Zielonka's Algorithm – Recursion



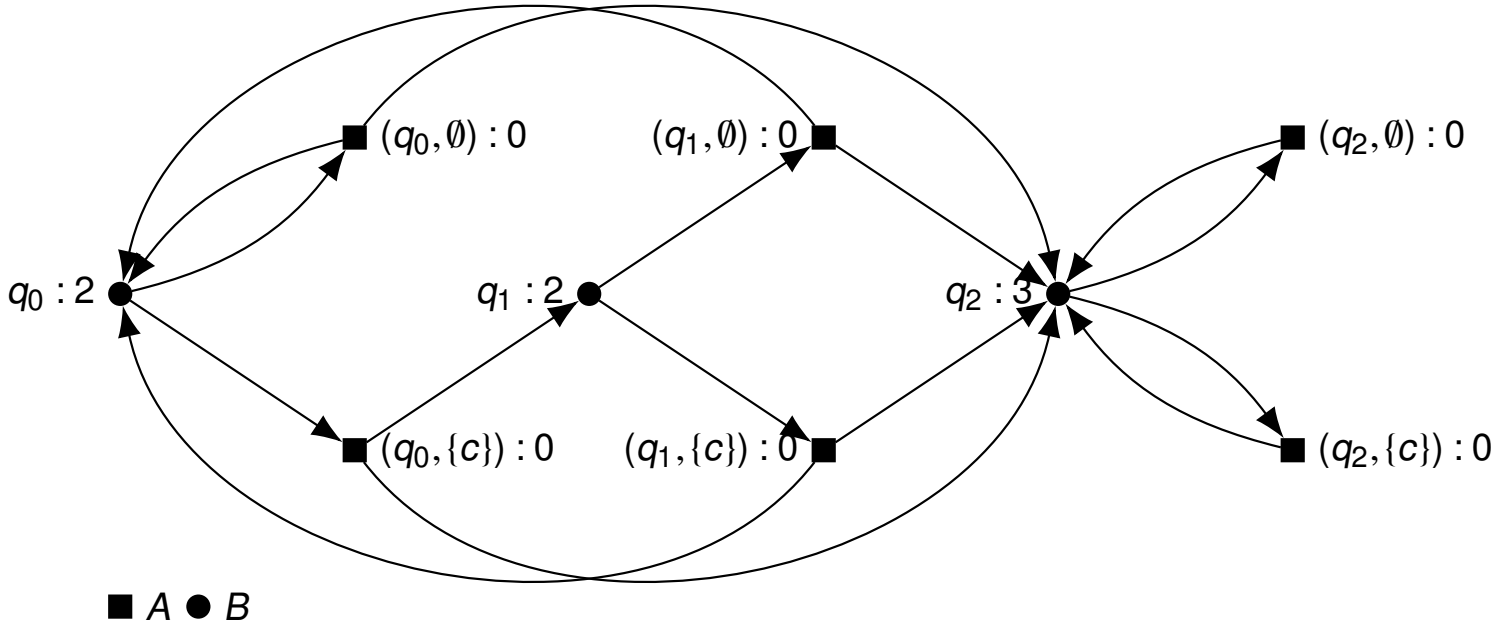
$H = G \setminus \mathfrak{A}_B(N_h)$
$Sol_A(H = (V', E'), \alpha)$:

- $h' = 2, N_{h'} = \{q_0, q_1\}$
- $\mathfrak{A}_A(N_{h'}) = V' \Rightarrow H' = (\emptyset, \emptyset) \Rightarrow W'_B = \emptyset$

$\Rightarrow$ return $W_A = \{q_0, (q_0, \emptyset), (q_0, \{c\}), q_1, (q_1, \emptyset), (q_1, \{c\})\}$

Exercise 13.1 (c)

Zielonka's Algorithm – Back from the recursion



$Sol_B(G, \alpha)$:

- $W_A = \{q_0, (q_0, \emptyset), (q_0, \{c\}), q_1, (q_1, \emptyset), (q_1, \{c\})\}$
- $\mathfrak{A}_A(W_A) = W_A$

$\Rightarrow$ return $Sol_B(G \setminus W_A, \alpha) = \{q_2, (q_2, \emptyset), (q_2, \{c\})\}$

$\Rightarrow Sol_A(G, \alpha) = \{q_0, (q_0, \emptyset), (q_0, \{c\}), q_1, (q_1, \emptyset), (q_1, \{c\})\}$

Exercise 13.1 (c)

Zielonka's Algorithm – Winning Strategy

$W_A = \{q_0, (q_0, \emptyset), (q_0, \{c\}), q_1, (q_1, \emptyset), (q_1, \{c\})\}$

- The winning region of A consists of all states associated with q_0 and q_1. (For this small example, you were probably able to guess that without the parity game detour.)
- Since we start in q_0 the winning strategy just has to stay within W_A.
- Looking back at the DPA we construct the recursive strategy σ:
  - $\sigma(i_1) = \{t\}$
  - $\sigma(i_1, \cdots, i_j) = \begin{cases} \{p\} & \text{if } i_{j-1} = \{c\} \text{ and } \sigma(i_1, \cdots, i_{j-1}) \models \neg p \\ \{t\} & \textit{else} \end{cases}$