Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Digital Systems and Design Automation Group

HS 2022

Prof. L. Josipović / J. Xu
based on Prof. L. Thiele's material

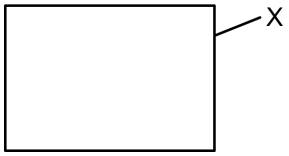
Discrete Event Systems

Solution to Exercise Sheet 10

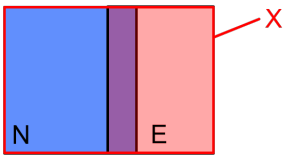
1 Sets Representation

1.1 Warm-up

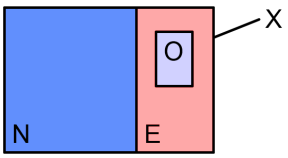
a) $\psi_X = 1$



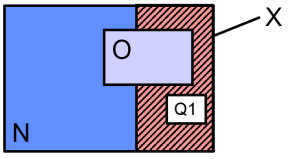
b) $N \cup E = X \quad \Leftrightarrow \quad \psi_N + \psi_E = 1$



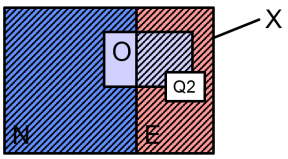
c) $N \cap O = \emptyset \quad \Leftrightarrow \quad \psi_N \cdot \psi_O = 0$



d) $Q_1 = E \backslash O \quad \Leftrightarrow \quad \psi_{Q_1} = \psi_E \cdot \overline{\psi_O}$



e) $Q_2 = (O \cap E) \cup \overline{O} \quad = (O \cup \overline{O}) \cap (E \cup \overline{O}) \quad \Leftrightarrow \quad \psi_{Q_2} = \psi_E + \overline{\psi_O}$
$= X \cap (E \cup \overline{O})$
$= E \cup \overline{O}$



1.2 Specification Composition

a) The specification for **C1**, **C2** and **C3** are the following:

C1 $\psi_{C1} = (x_1 + x_2 + x_3) \rightarrow x_s$

$\psi_{C1} = (x_1 + x_2 + x_3)x_s + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} = x_s + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3}$

C2 $\psi_{C2} = x_1 \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot \overline{x_3} + \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3}$

C3 $\psi_{C3} = x_b \rightarrow (x_s \cdot \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3})$

$\psi_{C3} = x_b \cdot x_s \cdot \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_b} = x_s \cdot \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_b}$

b) The specification consists in satisfying all constraints at all times:

$$\psi_N = \psi_{C1} \cdot \psi_{C2} \cdot \psi_{C3}$$

2 Binary Decision Diagrams

2.1 Verification using BDDs

a) $f_2 : y = \overline{\overline{x_1 + x_2 + x_3} + \overline{x_1 + \overline{x_2} + \overline{x_3}} + \overline{\overline{x_1} + \overline{x_2} + x_3}}$

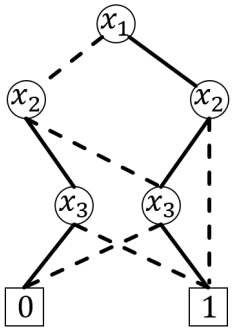
b) for f_1, we have

- case $x_1 = 0$:
  $y_{|x_1=0} = \overline{x_2}x_3 + x_2\overline{x_3}$
  - case $x_2 = 0$:
    $y_{|x_1=0,x_2=0} = x_3$
  - case $x_2 = 1$:
    $y_{|x_1=0,x_2=1} = \overline{x_3}$
- case $x_1 = 1$:
  $y_{|x_1=1} = \overline{x_2} + x_3 + \overline{x_2}x_3$
  - case $x_2 = 0$:
    $y_{|x_1=1,x_2=0} = 1$
  - case $x_2 = 1$:
    $y_{|x_1=1,x_2=1} = x_3$

for f_2, we have

- case $x_1 = 0$:
  $y_{|x_1=0} = \overline{\overline{x_2 + x_3} + \overline{\overline{x_2} + \overline{x_3}}}$
  - case $x_2 = 0$:
    $y_{|x_1=0,x_2=0} = \overline{\overline{x_3} + \overline{1 + \overline{x_3}}} = x_3$
  - case $x_2 = 1$:
    $y_{|x_1=0,x_2=1} = \overline{\overline{1} + \overline{\overline{x_3}}} = \overline{x_3}$
- case $x_1 = 1$:
  $y_{|x_1=1} = \overline{\overline{1} + \overline{1} + \overline{\overline{x_2} + x_3}} = \overline{x_2} + x_3$
  - case $x_2 = 0$:
    $y_{|x_1=1,x_2=0} = 1$
  - case $x_2 = 1$:
    $y_{|x_1=1,x_2=1} = x_3$

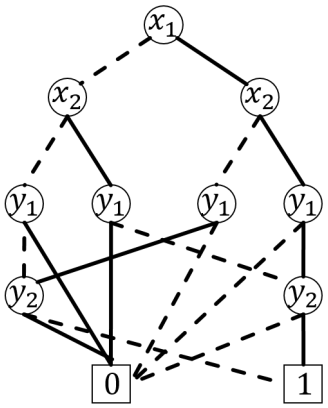
The two ROBDDs have identical falls, therefore they are equivalent.



2.2 BDDs with Respect to Different Orderings

a) $g = x_1\Big\{x_2\big[y_1(y_2)+\overline{y_1}(0)\big]+\overline{x_2}[y_1(\overline{y_2})+\overline{y_1}(0)]\Big\}+\overline{x_1}\Big\{x_2\big[y_1(0)+\overline{y_1}(y_2)\big]+\overline{x_2}\big[y_1(0)+\overline{y_1}(\overline{y_2})\big]\Big\}$

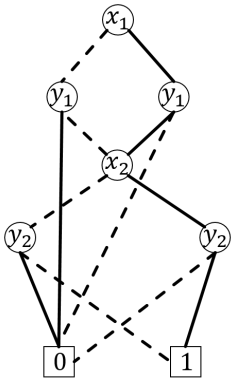
b) The ROBDD for g is the following:



c) Using the new ordering π', the Boole-Shannon decomposition becomes

$$g = x_1\Big\{y_1\big[x_2(y_2) + \overline{x_2}(\overline{y_2})\big] + \overline{y_1}[0]\Big\} + \overline{x_1}\Big\{y_1[0] + \overline{y_1}\big[x_2(y_2) + \overline{x_2}(\overline{y_2})\big]\Big\}.$$

This is a better ordering as it leads to a ROBDD with fewer nodes with respect to π (6 instead of 9).



3 Inductive Invariant

Let us consider a finite automation $M(Q, \delta, q_0, p)$, where Q is the set of states, δ is the transition relation, q_0 is the initial state, $p : Q \times \{1, 0\}$ is the output function, which maps a state to either good (1) or bad (0). We are interested in checking whether all reachable states are in "good state".

a) Consider our automation satisfies the following properties:

$$p(q_0) := 1. \tag{1}$$

$$\forall\ q, q' \in Q, p(q) \wedge \psi_\delta(q, q') \rightarrow p(q'). \tag{2}$$

Please show that, all reachable states are in "good state".

b) Does this hold vice versa? Consider that our automation has all its reachable states being "good states", can we use this to prove the above two properties? If not, try to sketch a counter example.

Solution

a) Intuition for the proof:

(1) The system always starts with a "good state".

(2) The system always moves from a "good state" to a "good state".

Formally speaking: We prove by induction
Base step: All states in Q_0 are good states

$$\forall q \in Q_0 : p(q) = 1. \tag{3}$$

Induction step: Assume that all states in Q_i are good states, show that all states in Q_{i+1} are good states.
From assumption, we have all states in Q_i are good states:

$$\forall q \in Q_i : p(q) = 1. \tag{4}$$

Using the transition relation ($Q_{i+1} := \{q' : \exists\ q,\ \psi_{Q_i}(q) \wedge \psi_\delta(q, q')\}$), we can get

$$\forall q' \in Q_{i+1} : \exists q \in Q_i,\ p(q) \wedge\ \psi_\delta(q, q'). \tag{5}$$

Using Eq. 2, we get all the states in Q_{i+1} are good states.

$$\forall q' \in Q_{i+1} : p(q') = 1. \tag{6}$$

By induction, we can conclude that $\forall i, \forall q \in Q_i, p(q) = 1$. Since computation of the set of states Q_i eventually will reach fix-point, then we conclude that all reachable states are "good states"..

b) Vice-versa is not true.

A property that holds in all reachable states is called an invariant property (sometime also referred to as a "safety property"). A straightforward way of proving invariant property is simply by performing reachability analysis. An invariant that holds these properties is called an *inductive invariant*. Proving the inductive hypotheses is much cheaper than the complete reachability analysis, i.e., it only requires to check the labels of the set of initial states, and the transition relation. This trick often fails to prove a correct property (i.e., it reports false negative) in practice due to a phenomenon called *induction leaking*.

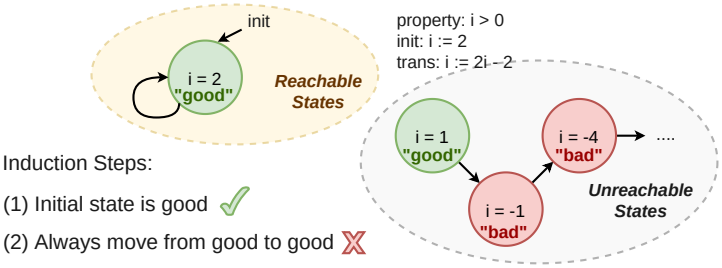
To illustrate this, consider a software program as the following:

$$i \in \mathcal{Z} \tag{7}$$

$$\text{init} : i \leftarrow 2 \tag{8}$$

$$\text{next} : i \leftarrow 2i - 2 \tag{9}$$

From here we can infer an invariant property (i.e. it holds in all reachable states of the



program) that i is always greater than 0. This property, however, is not an inductive invariant: if we plug in $i = 1$, then the value of i in the next state would be 0, which fails the induction hypothesis (transition from a "good state" to a "bad state").