ETH zürich

dynamo.ethz.ch

Exercise: Specification and Verification Using Set Operations and BDDs

Jiahui Xu
DYNAMO research group

Computer Engineering and Networks Laboratory

About Myself

Jiahui Xu

- PhD student in the DYNAMO group since April 2022

Research topics that I am interested:
- *High-level synthesis* (HLS): compile C/C++ code into digital circuits
- Optimization and formal verification of HLS-produced circuits

Email: jxu@ethz.ch Office: ETZ G75 Website: jiahui17.github.io

Event Reminder: Student Meets Lab

- Introduce to you our theses and projects
- Next Tuesday, 05 December 2023
- From 17:30 to 19:00 in the ETZ Foyer (ETZ E90.1)

Schedule

First half of today's lecture

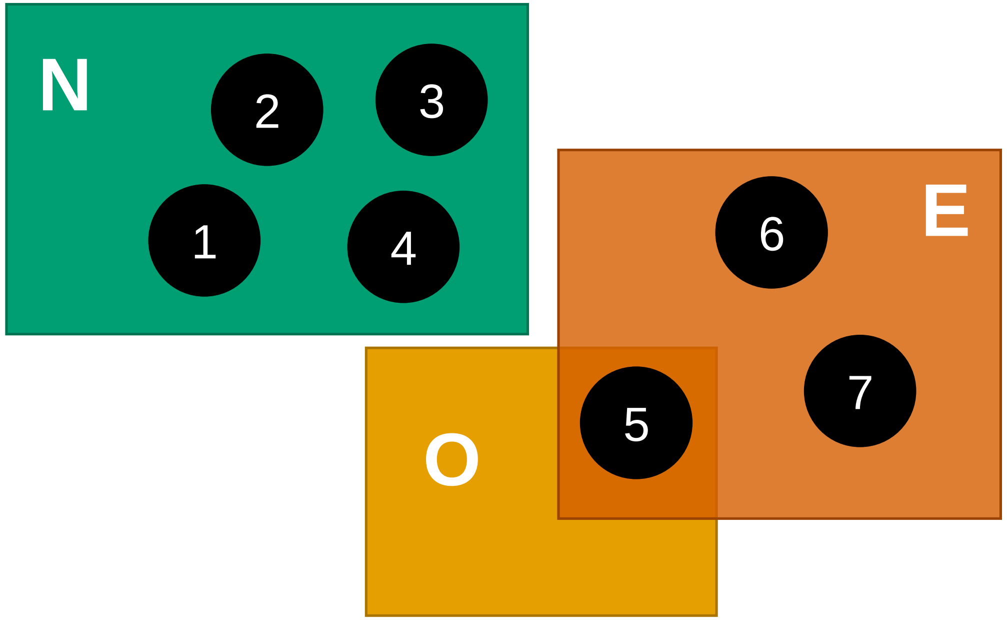
We have four exercise sessions:

- 30.11.2023: set operations, characteristic functions, BDDs
- 07.12.2023: reachability and temporal logic
- 14.12.2023: Petri nets
- 21.12.2023: time Petri nets

Q1.1 Set Operations and Characteristic Functions

Program states classified in sets

- **X: Set of all states**
- **N: Set of normal states**
- **E: Set of error states**
- **O: Set of states with memory overflow**

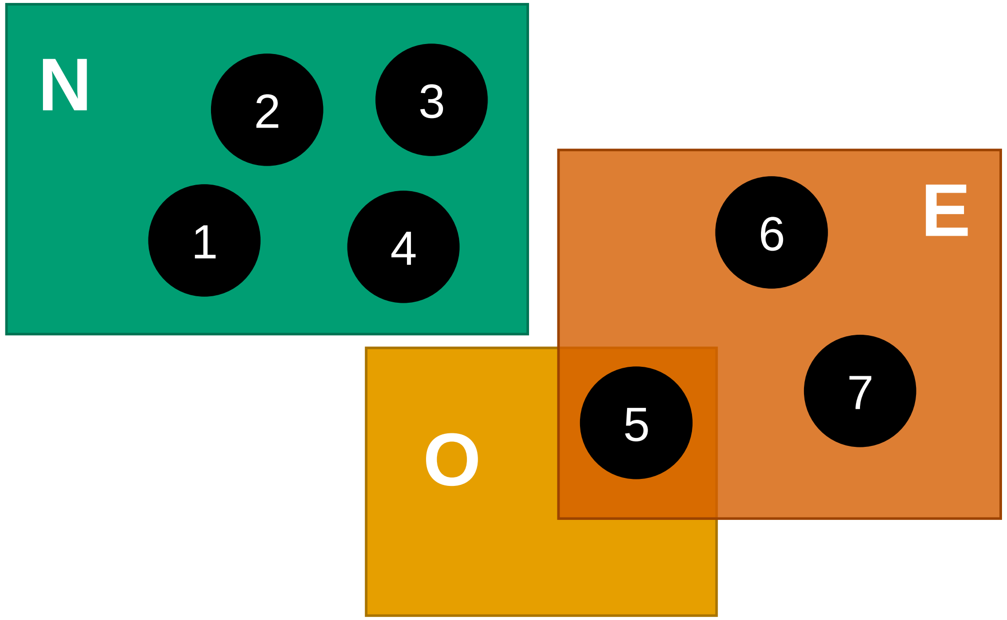


*Characteristic function $\Psi_N(\sigma(s))$: evaluates to 1 if $s \in N$, and 0 otherwise.

Q1.1 Set Operations and Characteristic Functions

Program states classified in sets

- **X: Set of all states**
- **N: Set of normal states**
- **E: Set of error states**
- **O: Set of states with memory overflow**



(a) What is the characteristic function* Ψ_X of the set of *all states X*?

$$\forall s \in X, \Psi_X(\sigma(s)) = 1$$

Because state s is always in the set of all states.

*Characteristic function $\Psi_N(\sigma(s))$: evaluates to 1 if $s \in N$, and 0 otherwise.

Q1.1 Set Operations and Characteristic Functions

Program states classified in sets

- **X: Set of all states**
- **N: Set of normal states**
- **E: Set of error states**
- **O: Set of states with memory overflow**

Sets and Relations

characteristic function of subset A

- Representation of a subset $A \subseteq E$:
  - Binary encoding $\sigma(e)$ of all elements $e \in E$
  - Subset A is represented by $a \in A \Leftrightarrow \psi_A(\sigma(a))$
  - Stepwise construction of the BDD corresponding to some subsets.

$$\begin{aligned}
c \in A \cap B &\Leftrightarrow \psi_A(\sigma(c)) \cdot \psi_B(\sigma(c)) \\
c \in A \cup B &\Leftrightarrow \psi_A(\sigma(c)) + \psi_B(\sigma(c)) \\
c \in A \setminus B &\Leftrightarrow \psi_A(\sigma(c)) \cdot \overline{\psi_B(\sigma(c))} \\
c \in E \setminus A &\Leftrightarrow \overline{\psi_A(\sigma(c))}
\end{aligned}$$

The union of two sets *A* and *B* is the set of elements which are in *A*, in *B*, or in both *A* and *B*.[2] In set-builder notation,

$$A \cup B = \{x : x \in A \text{ or } x \in B\}.$$ [3]

The intersection of two sets A and B, denoted by $A \cap B$,[3] is the set of all objects that are members of both the sets A and B. In symbols:

$$A \cap B = \{x : x \in A \text{ and } x \in B\}.$$

(b) **"Each state (in the set of all states) is either a normal or an error state or both"**.
Express this property in terms of sets and characteristic functions

$$X = N \cup E \qquad \Psi_X = 1 = \Psi_N + \Psi_E$$

*Characteristic function $\Psi_N(\sigma(s))$: evaluates to 1 if $s \in N$, and 0 otherwise.

Q1.1 Set Operations and Characteristic Functions

Sets and Relations

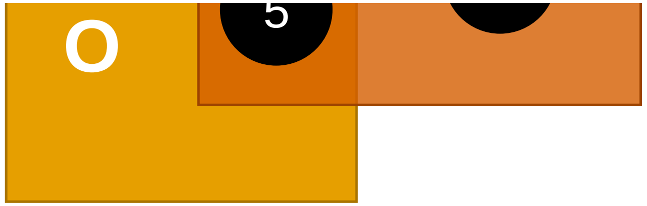
characteristic function of subset A

- Representation of a subset $A \subseteq E$:
  - Binary encoding $\sigma(e)$ of all elements $e \in E$
  - Subset A is represented by $a \in A \Leftrightarrow \psi_A(\sigma(a))$
  - Stepwise construction of the BDD corresponding to some subsets.

$$
\begin{aligned}
c \in A \cap B &\Leftrightarrow \psi_A(\sigma(c)) \cdot \psi_B(\sigma(c)) \\
c \in A \cup B &\Leftrightarrow \psi_A(\sigma(c)) + \psi_B(\sigma(c)) \\
c \in A \setminus B &\Leftrightarrow \psi_A(\sigma(c)) \cdot \overline{\psi_B(\sigma(c))} \\
c \in E \setminus A &\Leftrightarrow \overline{\psi_A(\sigma(c))}
\end{aligned}
$$

Program states classified in sets

- **X: Set of all states**
- **N: Set of normal states**
- **E: Set of error states**
- **O: Set of states with memory overflow**



(c) ”If a state (in the set of all states) is in the set of overflow states, then it is not a normal state”.
Express this property in terms of sets and characteristic functions.

(d) Describe Q1, the set of error states which are not an overflow, in terms of sets and characteristic functions.

Your turn!

(e) Describe Q2, the set of states that satisfies O ⇒ E, i.e., the set of states for which this property holds, in terms of sets and characteristic functions.
Hint: O ⇒ E reads O implies E, in other words, if a state is in O, then it is in E.

*Characteristic function $\Psi_N(\sigma(s))$: evaluates to 1 if s ∈ N, and 0 otherwise.

Q1.1 Set Operations and Characteristic Functions

Sets and Relations

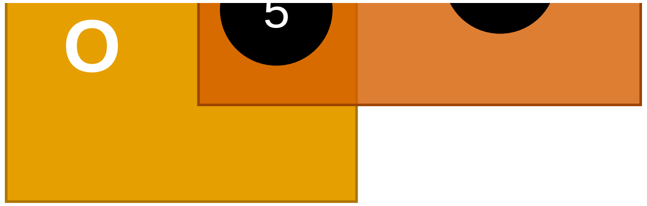
- Representation of a subset $A \subseteq E$:
  - Binary encoding $\sigma(e)$ of all elements $e \in E$
  - Subset A is represented by $a \in A \Leftrightarrow \psi_A(\sigma(a))$
  - Stepwise construction of the BDD corresponding to some subsets.

characteristic function of subset A

$$
\begin{array}{lcl}
c \in A \cap B & \Leftrightarrow & \psi_A(\sigma(c)) \cdot \psi_B(\sigma(c)) \\
c \in A \cup B & \Leftrightarrow & \psi_A(\sigma(c)) + \psi_B(\sigma(c)) \\
c \in A \setminus B & \Leftrightarrow & \psi_A(\sigma(c)) \cdot \overline{\psi_B(\sigma(c))} \\
c \in E \setminus A & \Leftrightarrow & \overline{\psi_A(\sigma(c))}
\end{array}
$$

Program states classified in sets

- **X: Set of all states**
- **N: Set of normal states**
- **E: Set of error states**
- **O: Set of states with memory overflow**



(c) "If a state (in the set of all states) is in the set of overflow states, it is not a normal state".
Express this property in terms of sets and characteristic functions.

For every state s: if "s is in set O" then "s is not in set N"

*Characteristic function $\Psi_N(\sigma(s))$: evaluates to 1 if $s \in N$, and 0 otherwise.

Let’s agree with the following convention…

When we test a specification “if A then B”, we are actually executing the following program:

```
// if A is true, then
if (A){
    // check if B is true
    assert(B);
}
// the execution is ok
return(0);
```

Conclusion: “if A then B” is the same as “(not A) or B”

*assert(B): the program crushes with B is false, and the program ignores the statement if B is true.

Convention: if A then B

Material implication (rule of inference)

2 languages

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

For other uses, see Material implication (disambiguation).
Not to be confused with Material inference.

This article **may be too technical for most readers to understand**. Please help improve it to make it understandable to non-experts, without removing the technical details. *(December 2018)* *(template removal help)*

In propositional logic, **material implication**[1][2] is a valid rule of replacement that allows for a conditional statement to be replaced by a disjunction in which the antecedent is negated. The rule states that *P implies* Q is logically equivalent to *not*-P *or* Q and that either form can replace the other in logical proofs. In other words, if P is true, then Q must also be true, while if Q is *not* true, then P cannot be true either; additionally, when P is not true, Q may be either true or false.

$$P \to Q \Leftrightarrow \neg P \lor Q$$

Where "$\Leftrightarrow$" is a metalogical symbol representing "can be replaced in a proof with," *P* and *Q* are any given logical statements, and $\neg P \lor Q$ can be read as "(not *P*) or *Q*". To illustrate this, consider the following statements:

- P: Sam ate an orange for lunch
- Q: Sam ate a fruit for lunch

Then, to say, "Sam ate an orange for lunch" *implies* "Sam ate a fruit for lunch" ($P \to Q$). Logically, if Sam did not eat a fruit for lunch, then Sam also cannot have eaten an orange for lunch (by contraposition). However, merely saying that Sam did not eat an orange for lunch provides no information on whether or not Sam ate a fruit (of any kind) for lunch.

Material implication

Type	Rule of replacement
Field	Propositional calculus
Statement	*P implies* Q is logically equivalent to *not*-P *or* Q. Either form can replace the other in logical proofs.
Symbolic statement	$P \to Q \Leftrightarrow \neg P \lor Q$

Transformation rules

Propositional calculus

Rules of inference

Implication introduction / elimination (*modus ponens*)
Biconditional introduction / elimination
Conjunction introduction / elimination
Disjunction introduction / elimination
Disjunctive / hypothetical syllogism

I didn’t invent this concept!

Q1.1 Set Operations and Characteristic Functions

Program states classified in sets

- **X: Set of all states**
- **N: Set of normal states**
- **E: Set of error states**
- **O: Set of states with memory overflow**

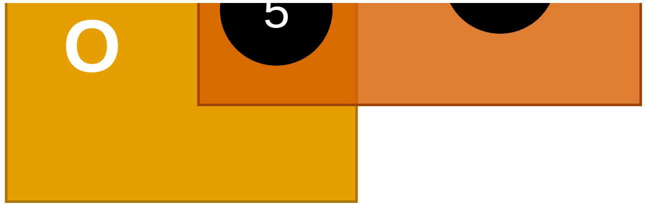
Sets and Relations

- Representation of a subset $A \subseteq E$:
  - Binary encoding $\sigma(e)$ of all elements $e \in E$
  - Subset A is represented by $a \in A \Leftrightarrow \psi_A(\sigma(a))$
  - Stepwise construction of the BDD corresponding to some subsets.

$$
\begin{aligned}
c \in A \cap B &\Leftrightarrow \psi_A(\sigma(c)) \cdot \psi_B(\sigma(c)) \\
c \in A \cup B &\Leftrightarrow \psi_A(\sigma(c)) + \psi_B(\sigma(c)) \\
c \in A \setminus B &\Leftrightarrow \psi_A(\sigma(c)) \cdot \overline{\psi_B(\sigma(c))} \\
c \in E \setminus A &\Leftrightarrow \overline{\psi_A(\sigma(c))}
\end{aligned}
$$

characteristic function of subset A

“if A then B” is the same as “(not A) or B”



(c) ”If a state (in the set of all states) is in the set of overflow states, it is not a normal state”.
Express this property in terms of sets and characteristic functions.

For every state s: if “s is in set O” then “s is not in set N”

For every state s: not (“s is in set O” and “s is in set N”)

Set of states violates this property: $\overline{X} = O \cap N$ $\qquad \Psi_X = \overline{\Psi_O \cdot \Psi_N}$

Set of all states: $X = \overline{O \cap N}$

*Characteristic function $\Psi_N(\sigma(s))$: evaluates to 1 if s ∈ N, and 0 otherwise.

Q1.1 Set Operations and Characteristic Functions

Program states classified in sets

- **X: Set of all states**
- **N: Set of normal states**
- **E: Set of error states**
- **O: Set of states with memory overflow**

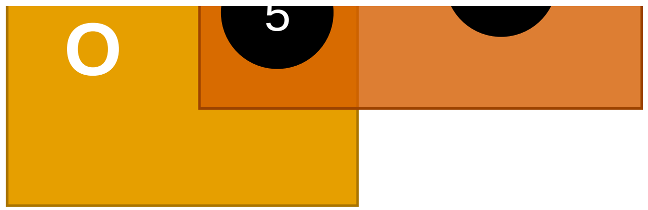
Sets and Relations

- Representation of a subset $A \subseteq E$:
  - Binary encoding $\sigma(e)$ of all elements $e \in E$
  - Subset A is represented by $a \in A \Leftrightarrow \psi_A(\sigma(a))$
  - Stepwise construction of the BDD corresponding to some subsets.

characteristic function of subset A

$$
\begin{aligned}
c \in A \cap B &\Leftrightarrow \psi_A(\sigma(c)) \cdot \psi_B(\sigma(c)) \\
c \in A \cup B &\Leftrightarrow \psi_A(\sigma(c)) + \psi_B(\sigma(c)) \\
c \in A \setminus B &\Leftrightarrow \psi_A(\sigma(c)) \cdot \overline{\psi_B(\sigma(c))} \\
c \in E \setminus A &\Leftrightarrow \overline{\psi_A(\sigma(c))}
\end{aligned}
$$

"if A then B" is the same as "(not A) or B"



(d) Describe Q1, the set of error states which are not an overflow, in terms of sets and characteristic functions.

For each state x in Q_1: "x is not in set O" and "x is in set E"

Set of states satisfies this property: $Q_1 = E \setminus O$

$$\Psi_{Q_1} = \Psi_E \cdot \overline{\Psi_O}$$

*Characteristic function $\Psi_N(\sigma(s))$: evaluates to 1 if $s \in N$, and 0 otherwise.

Q1.1 Set Operations and Characteristic Functions

Program states classified in sets

- **X: Set of all states**
- **N: Set of normal states**
- **E: Set of error states**
- **O: Set of states with memory overflow**

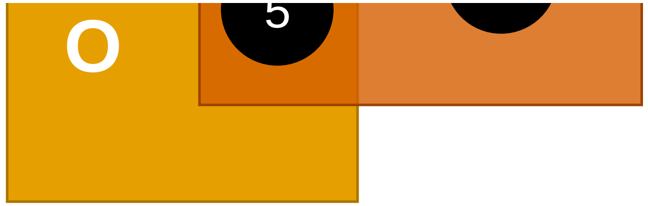
Sets and Relations

characteristic function of subset A

- Representation of a subset $A \subseteq E$:
  - Binary encoding $\sigma(e)$ of all elements $e \in E$
  - Subset A is represented by $a \in A \Leftrightarrow \psi_A(\sigma(a))$
  - Stepwise construction of the BDD corresponding to some subsets.

$$
\begin{array}{lcl}
c \in A \cap B & \Leftrightarrow & \psi_A(\sigma(c)) \cdot \psi_B(\sigma(c)) \\
c \in A \cup B & \Leftrightarrow & \psi_A(\sigma(c)) + \psi_B(\sigma(c)) \\
c \in A \setminus B & \Leftrightarrow & \psi_A(\sigma(c)) \cdot \overline{\psi_B(\sigma(c))} \\
c \in E \setminus A & \Leftrightarrow & \overline{\psi_A(\sigma(c))}
\end{array}
$$

“if A then B” is the same as “(not A) or B”

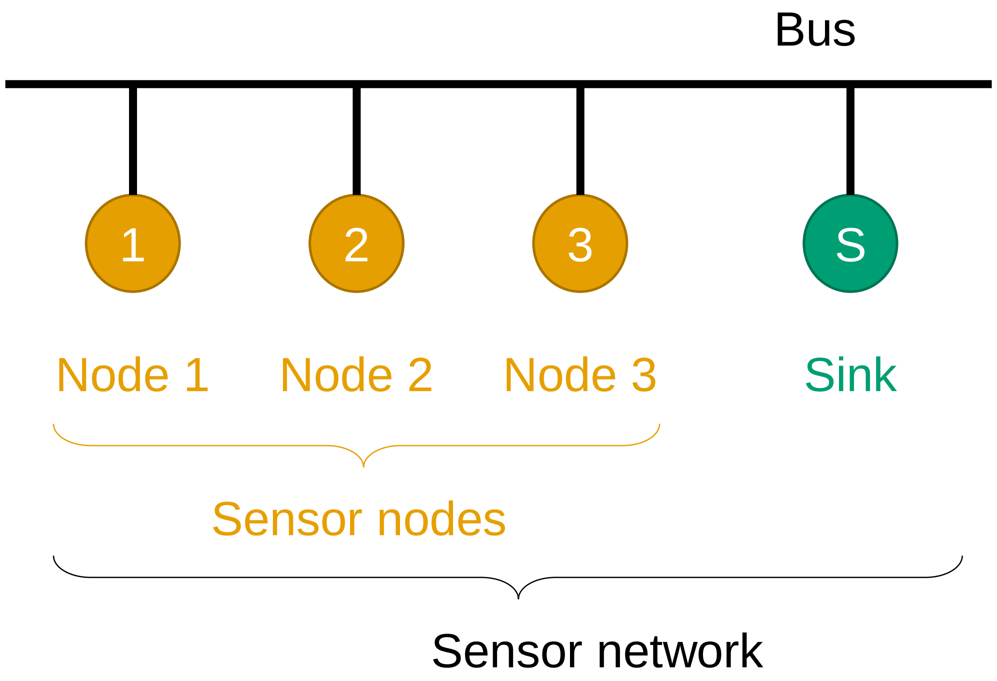


(e) Describe Q2, the set of states that satisfies O ⇒ E, i.e., the set of states for which this property holds, in terms of sets and characteristic functions.
Hint: O ⇒ E reads O implies E, in other words, if a state is in O, then it is in E.

The same problem as (c).
What happened for all states that are not in O?

*Characteristic function $\Psi_N(\sigma(s))$: evaluates to 1 if s ∈ N, and 0 otherwise.

Q1.2 Specifications in Boolean Encoding



"if A then B" is the same as "(not A) or B"

We can use binary variables to indicate the state of the system:

- $x_1 = 1$: node 1 is using the bus
- $x_2 = 1$: node 2 is using the bus
- $x_3 = 1$: node 3 is using the bus
- $x_s = 1$: the sink is awake
- $x_b = 1$: the system is bootstrapping

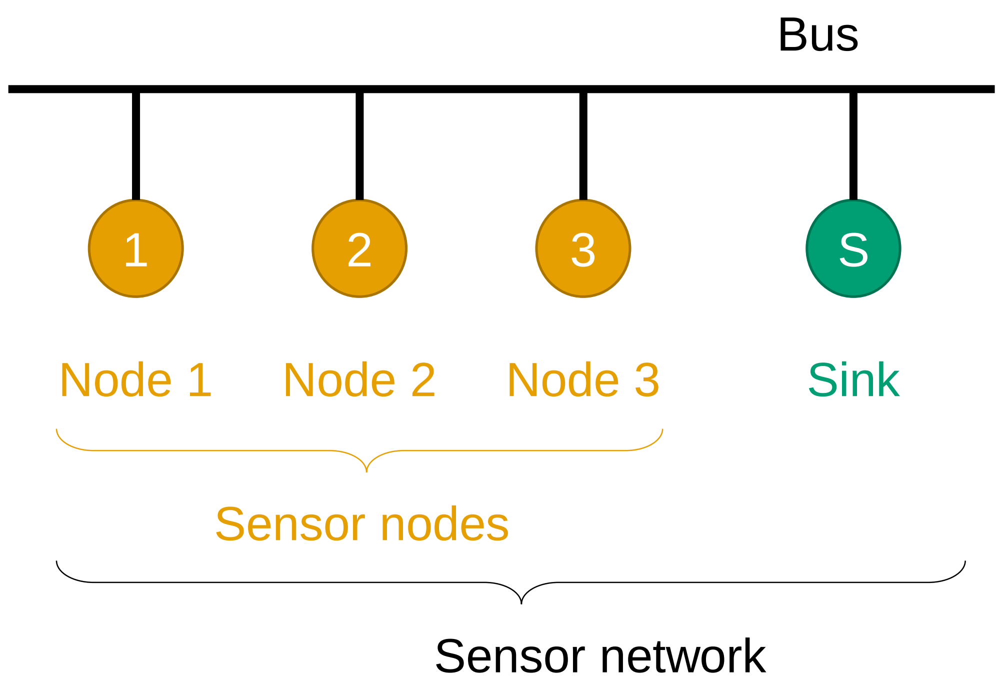
(C1) When one or more nodes are using the bus, the sink must be awake to receive data.

(C2) No more than one node can use the bus at the same time.

(C3) When the network is in bootstrapping mode, then the sink must be awake, and the nodes cannot use the bus.

Your turn! Describe (1) the 3 constraints using the binary variables, and (2) the complete specification

Q1.2 Specifications in Boolean Encoding



"if A then B" is the same as "(not A) or B"

We can use binary variables to indicate the state of the system:

- $x_1 = 1$: node 1 is using the bus
- $x_2 = 1$: node 2 is using the bus
- $x_3 = 1$: node 3 is using the bus
- $x_s = 1$: the sink is awake
- $x_b = 1$: the system is bootstrapping

(C1) When one or more nodes are using the bus, the sink must be awake to receive data.

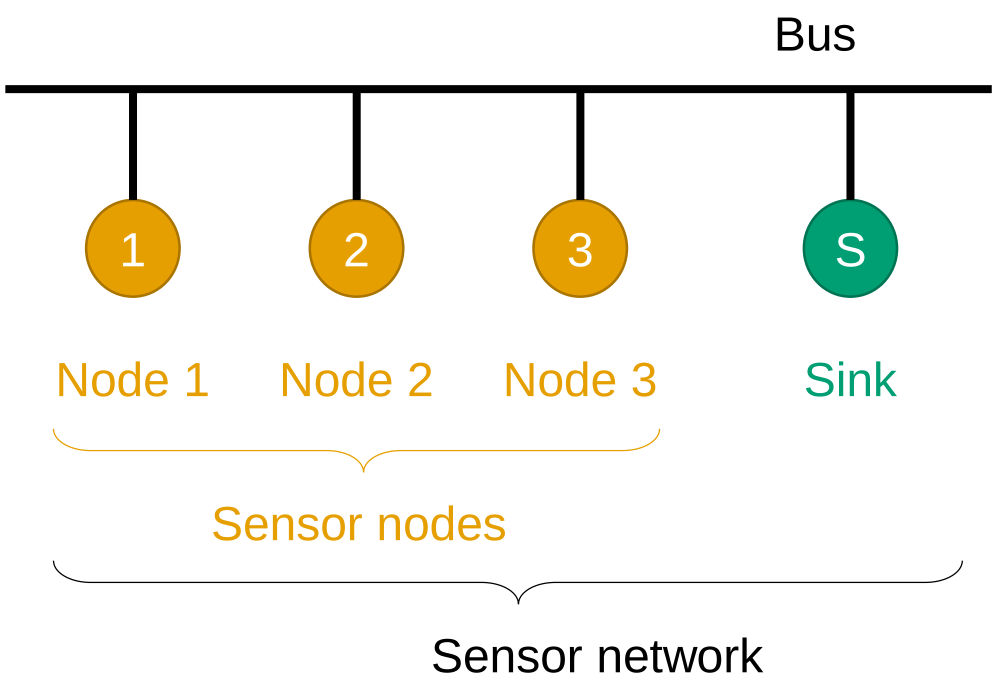
If "**one or more nodes are using the bus**" then "**the sink is awake**"

Not ("**not (no node is using the bus)**" and "**the sink is not awake**")

$$\overline{\overline{\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3}} \cdot \overline{x_s}}$$

Q1.2 Specifications in Boolean Encoding

“if A then B” is the same as “(not A) or B”



We can use binary variables to indicate the state of the system:

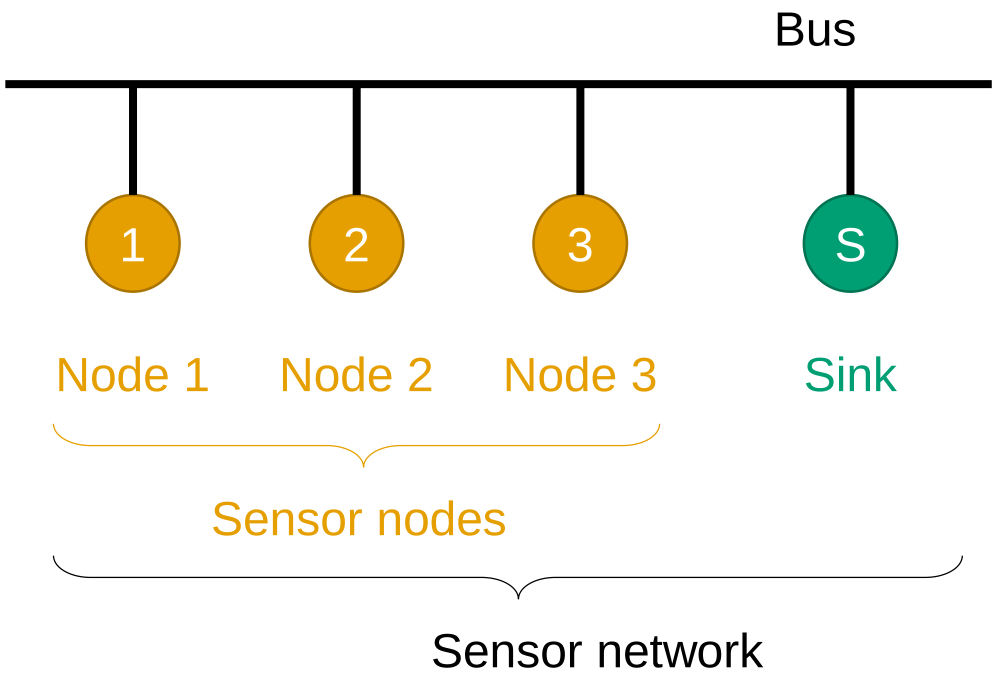
- $x_1 = 1$: node 1 is using the bus
- $x_2 = 1$: node 2 is using the bus
- $x_3 = 1$: node 3 is using the bus
- $x_s = 1$: the sink is awake
- $x_b = 1$: the system is bootstrapping

(C2) No more than one node can use the bus at the same time.

“**exactly one node is using the bus**” or “**no node is using the bus**”

$$x_1 \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot \overline{x_3} + \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3}$$

Q1.2 Specifications in Boolean Encoding



"if A then B" is the same as "(not A) or B"

We can use binary variables to indicate the state of the system:

- x_1 = 1: node 1 is using the bus
- x_2 = 1: node 2 is using the bus
- x_3 = 1: node 3 is using the bus
- x_s = 1: the sink is awake
- x_b = 1: the system is bootstrapping

(C3) When the network is in bootstrapping mode, then the sink must be awake, and the nodes cannot use the bus.

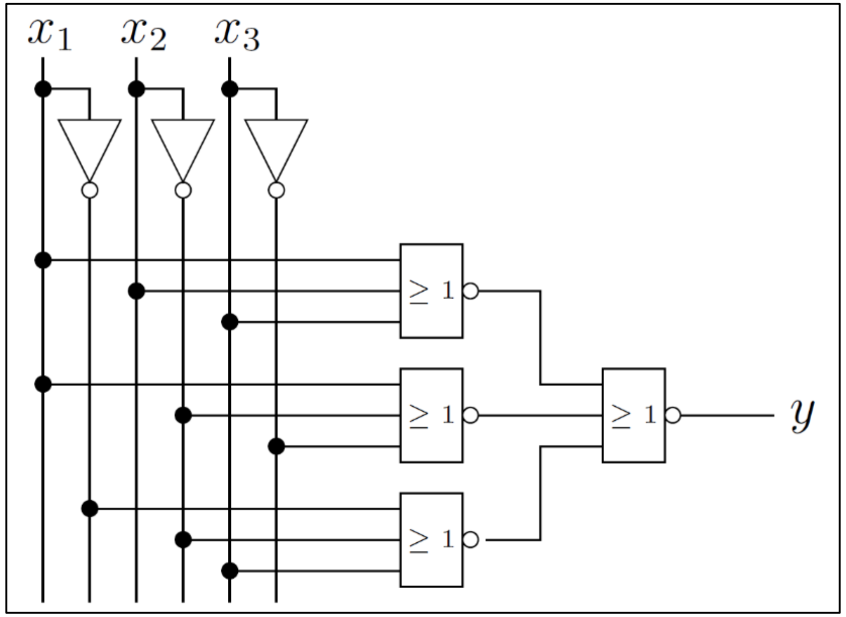
If "**the system is bootstrapping**" then ("**the sink is awake**" and "**no nodes is using the bus**")

Not ("**the system is bootstrapping**" and ("**the sink is not awake**" or "**one or more nodes is using the bus**"))

Q2.1: Combinational Equivalence Checking (CEC) Using BDDs

We want to check that the circuit implements the functionality of F1.

$$F_1 := x_1\overline{x_2} + x_1 x_3 + \overline{x_2} x_3 + \overline{x_1} x_2 \overline{x_3}.$$



Can we just test the two circuits with all possible inputs?
Not scalable for large designs!

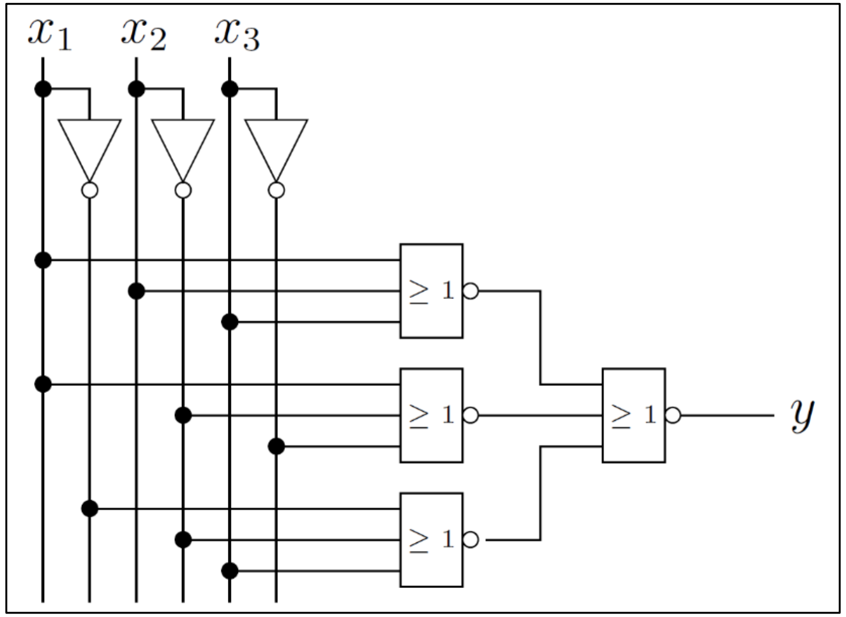
For a 64-bit input circuit. It takes about **~70 years to verify the design**, assuming we have a 2GHz CPU, that checks one input in every clock cycle

ROBDD-based method has **linear complexity** (in the best case) with respect to the number of variables

Q2.1: Combinational Equivalence Checking (CEC) Using BDDs

We want to check that the circuit implements the functionality of F1.

$$F_1 := x_1\overline{x_2} + x_1 x_3 + \overline{x_2}x_3 + \overline{x_1}x_2\overline{x_3}.$$



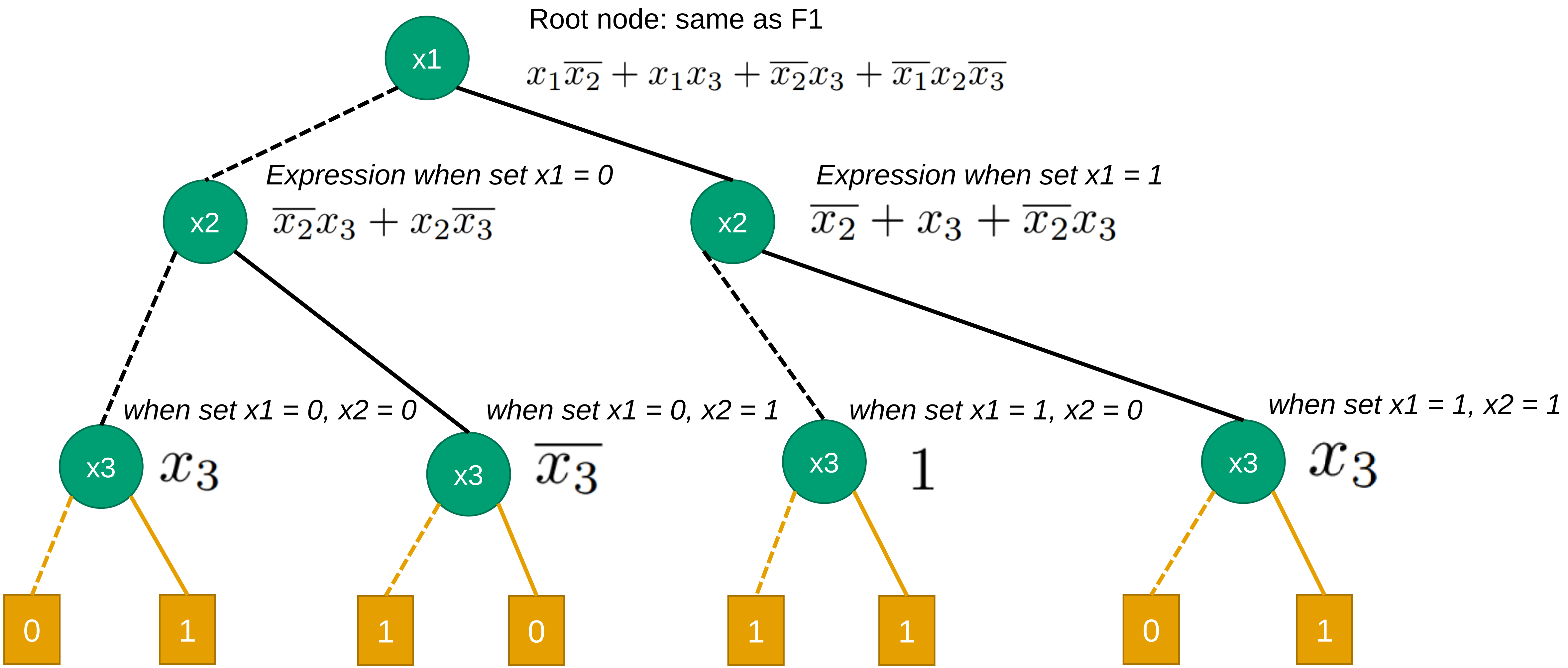
(a) Derive the Boolean function of the circuit diagram:

$$\overline{\overline{x_1 + x_2 + x_3} + \overline{x_1 + \overline{x_2} + \overline{x_3}} + \overline{\overline{x_1} + \overline{x_2} + x_3}}$$

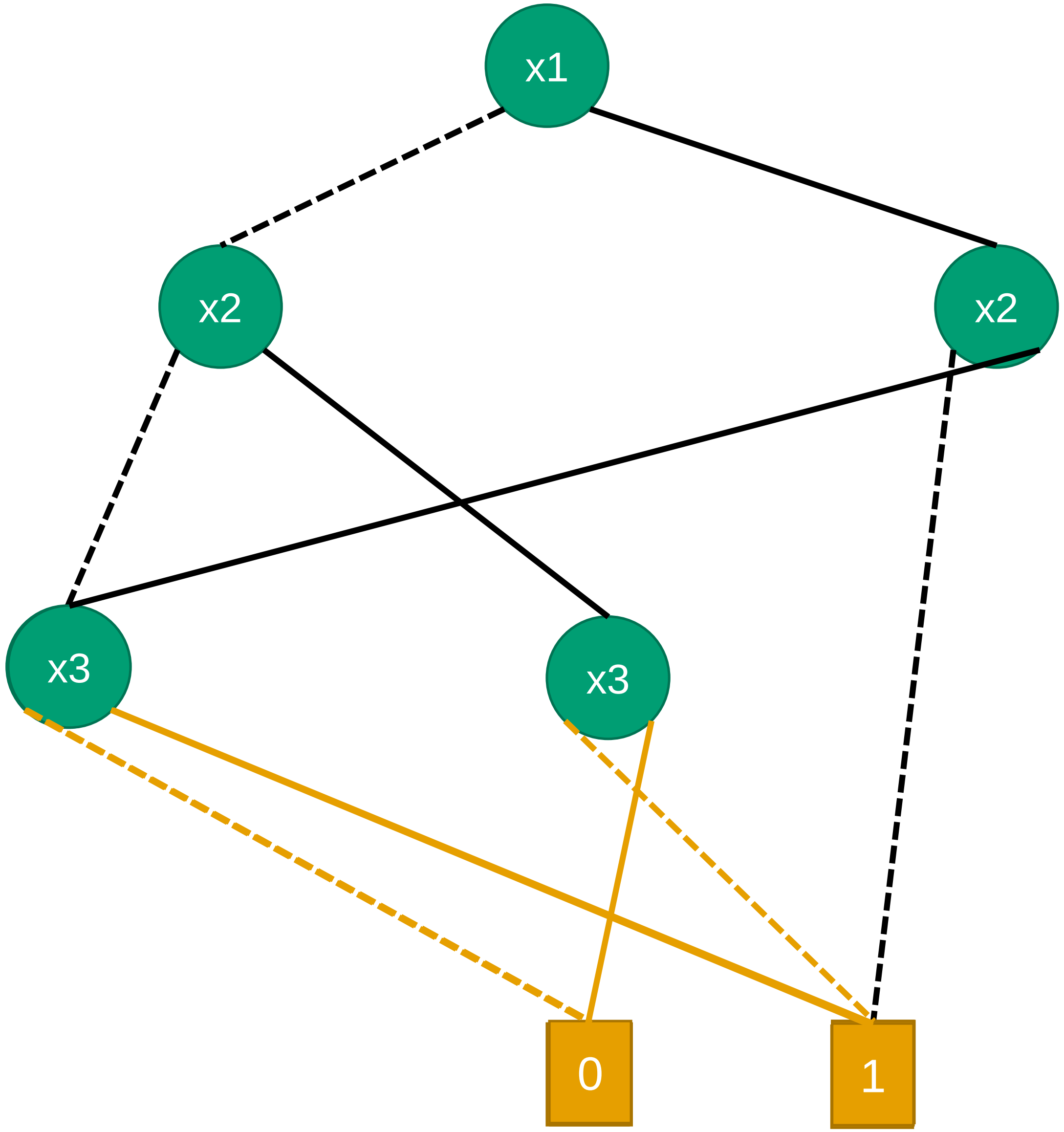
(b) Draw the ROBDD of F1 (evaluation order x1 < x2 < x3)*.

*x1 < x2 means we evaluate x1 before x2

Q2.1: Combinational Equivalence Checking (CEC) Using BDDs



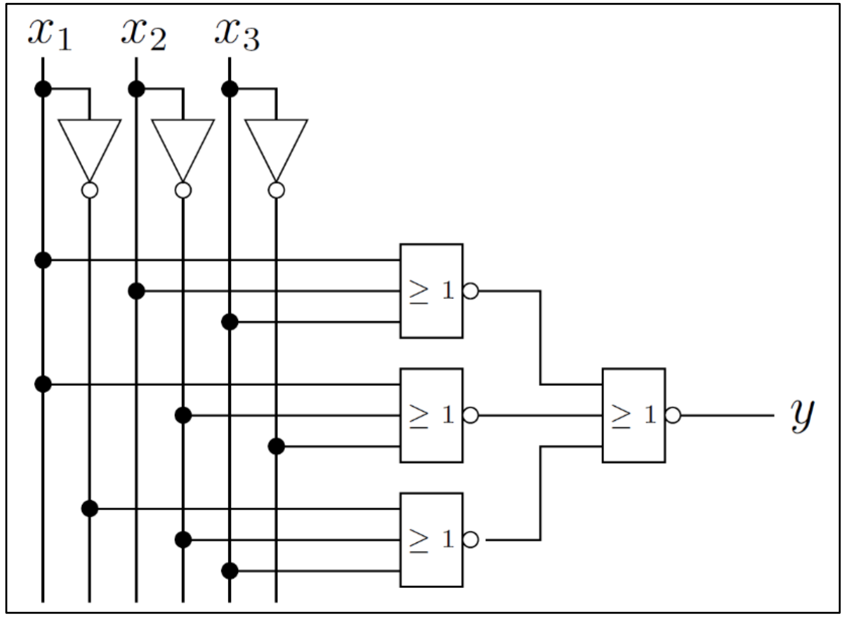
Q2.1: Combinational Equivalence Checking (CEC) Using BDDs



Q2.1: Combinational Equivalence Checking (CEC) Using BDDs

We want to check that the circuit implements the functionality of F1.

$$F_1 := x_1\overline{x_2} + x_1 x_3 + \overline{x_2} x_3 + \overline{x_1} x_2 \overline{x_3}.$$



(a) Derive the Boolean function of the circuit diagram:

$$\overline{\overline{x_1 + x_2 + x_3} + \overline{x_1 + \overline{x_2} + \overline{x_3}} + \overline{\overline{x_1} + \overline{x_2} + x_3}}$$

(b) Draw the ROBDD of F1 (evaluation order x1 < x2 < x3).

(c) Draw the ROBDD of F2, are the two functions identical?

Q2.2: ROBDDs with Different Orderings

$$G(x_1, x_2, y_1, y_2) := (x_1 \leftrightarrow y_1) \cdot (x_2 \leftrightarrow y_2)$$

(a) Compute the Boole-Shannon expansion of G, with respect to the following ordering:

$$x_1 < x_2 < y_1 < y_2$$

(b) Draw the ROBDD of the function above, with the following two variable orderings:

$$x_1 < x_2 < y_1 < y_2 \qquad x_1 < y_1 < x_2 < y_2$$

Which of the following have fewer decision nodes?

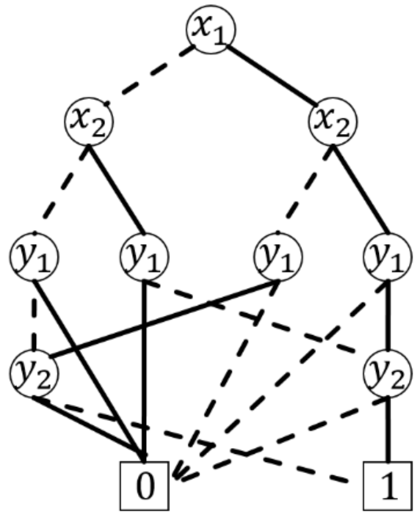
Q2.2: ROBDDs with Different Orderings

$$G(x_1, x_2, y_1, y_2) := (x_1 \leftrightarrow y_1) \cdot (x_2 \leftrightarrow y_2)$$

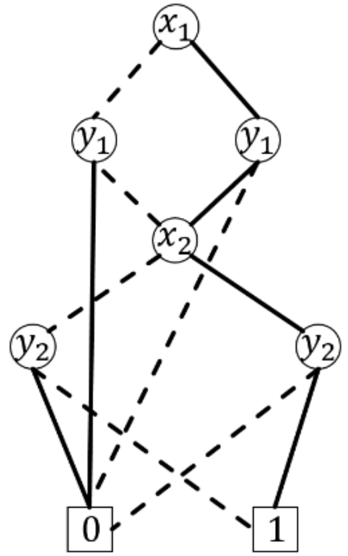
(a) Boole-Shannon expansion of G:

$$g = x_1 \left\{ y_1 \left[x_2(y_2) + \overline{x_2}(\overline{y_2}) \right] + \overline{y_1}[0] \right\} + \overline{x_1} \left\{ y_1[0] + \overline{y_1} \left[x_2(y_2) + \overline{x_2}(\overline{y_2}) \right] \right\}$$

(b) ROBDDs:



$$x_1 < x_2 < y_1 < y_2$$



$$x_1 < y_1 < x_2 < y_2$$