4. Computation Tree Logic (2)

Huixing Fang

School of Information Engineering
Yangzhou University

Outline

1. Counterexamples and Witnesses
2. Symbolic CTL Model Checking

1 Counterexamples and Witnesses

In the case of LTL

A counterexample for $TS \not\models \varphi$ is a **sufficiently long prefix of a path** π that indicates why π **refutes** φ.

1 Counterexamples and Witnesses

In the case of LTL

A counterexample for $TS \not\models \varphi$ is a **sufficiently long prefix of a path** π that indicates why π **refutes** φ.

Example 1

1. A counterexample for the LTL formula $\Diamond a$ is a finite prefix of just $\neg a$-states that ends with a single cycle traversal,i.e., there is a $\Box\neg a$-path
2. A counterexample for $\bigcirc a$ consists of a path π for which $\pi[1]$ violates a

1 Counterexamples and Witnesses

In the case of CTL

1. **Counterexamples** indicate the refutation of universally quantified path formulae
2. **Witnesses** indicate the satisfaction of existentially quantified path formulae

From a path-based view (for a path π)

1. a sufficiently long prefix of π with $\pi \not\models \varphi$ is a **counterexample** for the CTL state formula $\forall\varphi$
2. a sufficiently long prefix of π with $\pi \models \varphi$ is a **witness** of the CTL state formula $\exists\varphi$

1 Counterexamples and Witnesses

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a finite transition system without terminal states.

- **The Next Operator, CTL path formula $\varphi = \bigcirc\Phi$**
  1. **Counterexample**: (s, s') with $s \in I$ and $s' = Post(s)$ such that
  $$s' \not\models \Phi$$
  2. **Witness**: (s, s') with $s \in I$ and $s' = Post(s)$ such that
  $$s' \models \Phi$$
- **The Until Operator, CTL path formula $\varphi = \Phi \mathbf{U} \Psi$**
  1. **Witness**: an initial path fragment $s_0 s_1 \ldots s_n$ for which
  $$s_n \models \Psi \quad \text{and} \quad s_i \models \Phi \text{ for } 0 \leq i < n.$$
  2. **Counterexample**: an initial path fragment that indicates a path π:
  $$\pi \models \Box(\Phi \wedge \neg\Psi) \quad \text{or} \quad \pi \models (\Phi \wedge \neg\Psi)\mathbf{U}(\neg\Phi \wedge \neg\Psi)$$

Counterexample for $\varphi = \Phi \, \mathbf{U} \, \Psi$:

1. For $\pi \models \Box(\Phi \wedge \neg\Psi)$, an initial path fragment:

$$\underbrace{s_0 s_1 \ldots s_{n-1} \underbrace{s_n s'_1 \ldots, s'_r}_{\text{cycle}}}_{\text{satisfy } \Phi \wedge \neg\Psi} \quad \text{with} \quad s_n = s'_r.$$

2. For $\pi \models (\Phi \wedge \neg\Psi) \, \mathbf{U} \, (\neg\Phi \wedge \neg\Psi)$, an initial path fragment:

$$\underbrace{s_0 s_1 \ldots s_{n-1}}_{\text{satisfy } \Phi \wedge \neg\Psi} \; s_n \quad \text{with} \quad s_n \models \neg\Phi \wedge \neg\Psi$$

1 Counterexamples and Witnesses

Counterexamples can be determined by an analysis of the digraph $G = (S, E)$ where

$$E = \{(s, s') \in (S \times S) \mid s' \in Post(s) \land s \models \Phi \land \neg\Psi\}.$$

1. Each path in G that starts in an initial state $s_0 \in S$ and leads to a nontrivial SCC C in G provides a counterexample of the form

$$s_0 s_1 \ldots \underbrace{s_n s'_1 \ldots s'_r}_{\in C} \quad \text{with} \quad s_n = s'_r$$

2. Each path in G that leads from an initial state s_0 to a trivial terminal SCC

$$C = \{s'\} \quad \text{with} \quad s' \models \neg\Phi \land \neg\Psi$$

provides a counterexample of the form $s_0 s_1 \ldots s_n$ with $s_n \models \neg\Phi \land \neg\Psi$.

1 Counterexamples and Witnesses

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a finite transition system without terminal states.

- **The Always Operator, CTL path formula $\varphi = \Box\Phi$**
  1. Counterexample: an initial path fragment $s_0 s1 ... s_n$ such that

     $$s_i \models \Phi \quad \text{for } 0 \leq i < n \text{ and } s_n \not\models \Phi$$

     Counterexamples may be determined by a backward search starting in $\neg\Phi$-states
  2. Witness: an initial path fragment of the form

     $$\underbrace{s_0 s_1 ... s_n s'_1 ... s'_r}_{\text{satisfy } \Phi} \quad \text{with} \quad s_n = s'_r.$$

     Witnesses can be determined by a simple cycle search in the digraph $G = (S, E)$, with $E = \{(s, s') \mid s' \in Post(s) \wedge s \models \Phi\}$

Theorem 2 (Time Complexity of Counterexample Generation)

Let TS be a transition system with N states and K transitions and φ a CTL path formula.

1. *If $TS \not\models \forall\varphi$, then a counterexample for φ in TS can be determined in time $O(N+K)$.*
2. *The same holds for a witness for φ, provided that $TS \models \exists\varphi$.*

Outline

2 Symbolic CTL Model Checking

To attack the state explosion problem, the CTL model-checking procedure can be reformulated in a **symbolic** way:

1. **Binary encoding** of the states, which permits identifying subsets of the state space
2. Symbolic approach: operates on **sets** of states rather than single states and
3. relay on a **representation of transition systems** by **switching functions**

2.1 Switching Functions

Evaluation

One **evaluation** is a function:

$$\eta : Var \to \{0, 1\},$$

in which, $Var = \{z_1, ..., z_m\}$, z_i are **Boolean variables**.

1. $Eval(z_1, ..., z_m)$ denotes the set of evaluations for Boolean variables $z_1, ..., z_m$
2. Evaluations are written as $[z_1 = b_1, ..., z_m = b_m]$ for $b_i \in \{0, 1\}$
3. $[\bar{z} = \bar{b}]$ is short for the evaluation $[z_1 = b_1, ..., z_m = b_m]$ when $\bar{z} = (z_1, ..., z_m)$ and $\bar{b} = (b_1, ..., b_m)$

2.1 Switching Function

Switching function

A **switching function** for Var is a function

$$f : Eval(Var) \to \{0, 1\}.$$

The special case $Var = \emptyset$ is allowed. The switching functions for $\emptyset$ are just constants 0 or 1.

1. Write $f(z)$ or $f(z_1, ..., z_m)$ to indicate the underlying set of variables
2. Write $f(b_1, ..., b_m)$ or $f(b)$ instead of $f([z_1 = b_1, ..., z_m = bm])$ when z_i is clear from the context
3. Boolean connectives can be defined for switching functions

If f_1, f_2 are switching functions, then

$$\begin{aligned}&(\boldsymbol{f_1} \vee \boldsymbol{f_2})([z_1 = b_1, ..., z_k = b_k]) \\ &\qquad = max\{f_1([z_1 = b_1, ..., z_m = b_m]), f_2([z_n = b_n, ..., z_k = b_k])\}.\end{aligned}$$

2.1 Switching Function

Projection function

We often simply write z_i for the **projection function**:

$$pr_{z_i} : Eval(\bar{z}) \rightarrow \{0, 1\},$$

$pr_{z_i}([\bar{z} = b]) = b_i$ and 0 or 1 for the constant switching functions.

1. Switching functions can be represented by Boolean connections of the variables z_i (viewed as projection functions) and constants
2. For instance, $z_1 \vee (z_2 \wedge \neg z_3)$ stands for a switching function

2.1 Switching Function

Positive cofactor

Let $f : Eval(z, y_1, ..., y_m) \rightarrow \{0, 1\}$ be a switching function. The **positive cofactor** of f for variable z is the switching function

$$\boldsymbol{f|_{z=1} : Eval(z, y_1, ..., y_m) \rightarrow \{0, 1\}}$$

given by

$$f|_{z=1}(\zeta, b_1, ..., b_m) = f(1, b_1, ..., b_m)$$

where $(\zeta, b_1, ..., b_m) \in \{0, 1\}^{m+1}$ is short for $[z = \zeta, y_1 = b_1, ..., y_m = b_m]$

Negative cofactor

Let $f : Eval(z, y_1, ..., y_m) \to \{0, 1\}$ be a switching function. The **negative cofactor** of f for variable z is the switching function

$$\boldsymbol{f|_{z=0} : Eval(z, y_1, ..., y_m) \to \{0, 1\}}$$

given by

$$f|_{z=0}(\zeta, b_1, ..., b_m) = f(0, b_1, ..., b_m)$$

where $(\zeta, b_1, ..., b_m) \in \{0, 1\}^{m+1}$ is short for $[z = \zeta, y_1 = b_1, ..., y_m = b_m]$

2.1 Switching Function

Cofactor

Let $f : Eval(z_1, ..., z_k, y_1, ..., y_m) \to \{0, 1\}$ be a switching function. The **iterated cofactor** (also simply called **cofactor**) of f is the switching function

$$\boldsymbol{f|_{z_1=b_1,...,z_k=b_k} : (...(f|_{z_1=b_1})|_{z_2=b_2} \cdots)|_{z_k=b_k}}$$

given by

$$f|_{z_1=b_1,...,z_k=b_k}(\zeta_1, ..., \zeta_k, a_1, ..., a_m) = f(b_1, ..., b_k, a_1, ..., a_m).$$

2.1 Switching Function

Cofactor

Let $f : Eval(z_1, ..., z_k, y_1, ..., y_m) \to \{0, 1\}$ be a switching function. The **iterated cofactor** (also simply called **cofactor**) of f is the switching function

$$f|_{z_1=b_1,...,z_k=b_k} : (...(f|_{z_1=b_1})|_{z_2=b_2}\cdots)|_{z_k=b_k}$$

given by

$$f|_{z_1=b_1,...,z_k=b_k}(\zeta_1, ..., \zeta_k, a_1, ..., a_m) = f(b_1, ..., b_k, a_1, ..., a_m).$$

Essential Variable

Variable z is called **essential** for switching function f if $f|_{z=0} \neq f|_{z=1}$.

1. Variable z is **not essential** for the cofactors $f|_{z=0}$ and $f|_{z=1}$
2. At most the variables in $Var \setminus \{z_1, ..., z_k\}$ are essential for $f|_{z_1=b_1,...,z_k=b_k}$ (f is a switching function for Var)

2.1 Switching Function

1. Let switching function $f(z_1, z_2, z_3) = (z_1 \vee \neg z_2) \wedge z_3$,

$$f|_{z_1=1} = z_3, \quad f|_{z_1=0} = \neg z_2 \wedge z_3,$$

variable z_1 is essential for f

2.1 Switching Function

1. Let switching function $f(z_1, z_2, z_3) = (z_1 \vee \neg z_2) \wedge z_3$,
$$f|_{z_1=1} = z_3, \quad f|_{z_1=0} = \neg z_2 \wedge z_3,$$
variable z_1 is essential for f
2. Let $Var = \{z_1, z_2, z_3\}$, z_2 and z_3 are not essential for the projection function $pr_{z_1} = z_1$
$$z_1|_{z_2=0} = z_1|_{z_2=1} = z_1,$$
but z_1 is essential for the projection function z_1:
$$z_1|_{z_1=1} = 1 \neq z_1|_{z_1=0} = 0.$$

2.1 Switching Function

1. Let switching function $f(z_1, z_2, z_3) = (z_1 \vee \neg z_2) \wedge z_3$,
$$f|_{z_1=1} = z_3, \quad f|_{z_1=0} = \neg z_2 \wedge z_3,$$
variable z_1 is essential for f
2. Let $Var = \{z_1, z_2, z_3\}$, z_2 and z_3 are not essential for the projection function $pr_{z_1} = z_1$
$$z_1|_{z_2=0} = z_1|_{z_2=1} = z_1,$$
but z_1 is essential for the projection function z_1:
$$z_1|_{z_1=1} = 1 \neq z_1|_{z_1=0} = 0.$$
3. Variables z_1 and z_2 are essential for
$$f(z_1, z_2, z_3) = z_1 \vee \neg z_2 \vee (z_1 \wedge z_2 \wedge \neg z_3),$$
but z_3 is not,
$$f|_{z_3=1} = z_1 \vee \neg z_2 = f|_{z_3=0} = z_1 \vee \neg z_2 \vee (z_1 \wedge z_2)$$

Theorem 3 (Shannon Expansion)

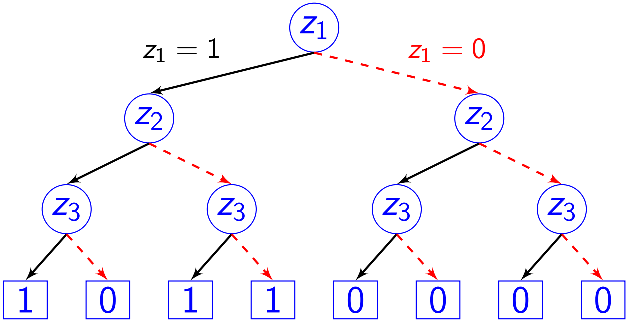
If f is a switching function for Var, then for each variable $z \in Var$:

$$f = (\neg z \wedge f|_{z=0}) \vee (z \wedge f|_{z=1}).$$

- *The Shannon expansion is inherent in the representation of switching functions by* ***binary decision trees***

2.1 Switching Function

Binary decision tree for $f(z_1, z_2, z_3) = z_1 \wedge (\neg z_2 \vee z_3)$:



1. The paths from the root to a leaf represent the evaluations
2. The leaves stand for the function values 0 or 1 of f
3. The subtree of node v yields a representation of the iterated cofactor $f|_{z_1=b_1,\ldots,z_m=b_m}$

2.1 Switching Function

Existential and Universal Quantification

Let f be a switching function for Var and $z \in Var$.

- $\exists z.\ f$ is the switching function give by:

$$\exists z.\ f = f|_{z=0} \vee f|_{z=1}$$

  and $\exists \bar{z}.\ f = \exists z_1 . \exists z_2 .\ ... \exists z_k .\ f$

Existential and Universal Quantification

Let f be a switching function for Var and $z \in Var$.

- $\exists z.\ f$ is the switching function give by:

$$\exists z.\ f = f|_{z=0} \vee f|_{z=1}$$

  and $\exists \bar{z}.\ f = \exists z_1.\exists z_2.\ ...\exists z_k.\ f$

- The universal quantification is defined by

$$\forall z.\ f = f|_{z=0} \wedge f|_{z=1}$$

  and $\forall \bar{z}.\ f = \forall z_1.\forall z_2.\ ...\forall z_k.\ f$

2.1 Switching Function

Existential and Universal Quantification

Let f be a switching function for Var and $z \in Var$.

- $\exists z.\ f$ is the switching function give by:

$$\exists z.\ f = f|_{z=0} \vee f|_{z=1}$$

  and $\exists \bar{z}.\ f = \exists z_1.\exists z_2.\ ...\exists z_k.\ f$

- The universal quantification is defined by

$$\forall z.\ f = f|_{z=0} \wedge f|_{z=1}$$

  and $\forall \bar{z}.\ f = \forall z_1.\forall z_2.\ ...\forall z_k.\ f$

- Let $f(z, y_1, y_2) = (z \vee y_1) \wedge (\neg z \vee y_2)$, then

$$\exists z.\ f = f|_{z=0} \vee f|_{z=1} = y_1 \vee y_2,$$
$$\forall z.\ f = f|_{z=0} \wedge f|_{z=1} = y_1 \wedge y_2$$

1. Counterexamples and Witnesses

2. Symbolic CTL Model Checking
   - Switching Functions
   - **Encoding Transition Systems by Switching Functions**
   - Odered Binary Decision Diagrams

2.2 Encoding Transition Systems by Switching Functions

Encoding State of Transition Systems

Let $TS = (S, \rightarrow, I, AP, L)$ and $\bar{x} = (x_1, ..., x_n)$, for each state $s \in S$, the encoding of s is an evaluation

$$[x_1 = b_1, ..., x_n = b_n] \in Eval(\bar{x}),$$

x_i are Boolean variables, $b_i \in \{0, 1\}$

Characteristic Function

The characteristic function χ_B is the switching function give by

$$\chi_B : Eval(\bar{x}) \rightarrow \{0, 1\}, \quad \chi_B(s) = \begin{cases} 1 & \text{if } s \in B \\ 0 & \text{otherwise.} \end{cases}$$

2.2 Encoding Transition Systems by Switching Functions

Encoding of Satisfaction Set

For any $a \in AP$, the satisfaction set $Sat(a) = \{s \in S \mid s \models a\}$ can be represented by the switching function

$$f_a = \chi_{Sat(a)} \ .$$

Encoding of Labeling Function

A family $(f(a))_{a \in AP}$ of switching functions can be used as a symbolic representation of the labeling function L.

Encoding of Transition Relation

The transition relation $\rightarrow$ of TS can be represented by the switching function

$$\Delta : Eval(\bar{x}, \bar{x}') \rightarrow \{0, 1\}, \quad \Delta(s, t\{\bar{x}'/\bar{x}\}) = \begin{cases} 1 & \text{if } s \rightarrow t \\ 0 & \text{otherwise} \end{cases}$$

1. s and t are elements of the state space $S = Eval(\bar{x})$
2. $\bar{x} = (x_1, \ldots, x_n)$ and $\bar{x}' = (x_1', \ldots, x_n')$
3. Unprimed variables x_i serve to encode the current state
4. Primed variables x_i' serve to encoding the next state
5. $t\{\bar{x}'/\bar{x}\}$ is the evaluation after applying the replacement of x_i by x_i'

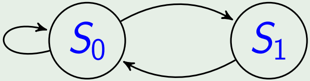
2.2 Encoding Transition Systems by Switching Functions

Example 4 (Symbolic Representation of Transition Relation)

Just one Boolean variable x can be used for encoding, the transition relation $\rightarrow$ is represented by $\Delta : Eval(x, x') \rightarrow \{0, 1\}$:

$$\Delta = \neg x \vee \neg x'$$

TS :



1. Satisfying assignment $[x = 0, x' = 0]$ denotes the transition $s_0 \rightarrow s_0$
2. Satisfying assignment $[x = 0, x' = 1]$ denotes the transition $s_0 \rightarrow s_1$
3. Satisfying assignment $[x = 1, x' = 0]$ denotes the transition $s_1 \rightarrow s_0$

2.2 Encoding Transition Systems by Switching Functions

Example 5 (Symbolic Representation of a Ring)

Consider a transition system TS with states $\{s_0, \ldots, s_{k-1}\}$ where $k = 2^n$ that are organized in a ring, TS has the transitions

$$s_i \longrightarrow s_{(i+1) \bmod k}, \quad \text{for } 0 \le i < k$$

Encoding any state s_i with the binary encoding of its index i:

1. if $k = 16$, then $n = 4$ and state s_1 is identified with 0001
2. n Boolean variables $x_1, \ldots, x_n$ are used for encoding, evaluation $[x_1 = b_1, \ldots., x_n = b_n]$ stands for state $\sum_{1 \le i \le n} b_i 2^{i-1}$
3. The switching function is

$$\sum_{1 \le i \le n} x_i' \cdot 2^{i-1} \equiv \left((1 + \sum_{1 \le i \le n} x_i \cdot 2^{i-1}) \mod k \right)$$

Switching Function $\chi_{Post(s)}$

If $s = [x_1 = b_1, ..., x_n = b_n]$, then a switching function $\chi_{Post(s)}$ for the successor set $Post(s) = \{s' \in S \mid s \to s'\}$ is obtained from Δ by building the cofactor for the variables $x_1, ..., x_n$:

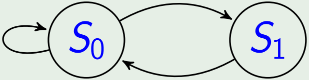
$$\chi_{Post(s)} = \Delta|_s\{\bar{x}/\bar{x}'\}$$

1. $\Delta|_s$ stands for the iterated cofactor $\Delta|_{x_1=b_1,...,x_n=b_n}$
2. The renaming operator $\{\bar{x}/\bar{x}'\}$ yields a representation of $Post(s)$ by the variables $x_1, ..., x_n$

2.2 Encoding Transition Systems by Switching Functions

Example 6

TS :



Symbolic encoding of the successor set $Post(s_0) = \{s_0, s_1\}$ for TS:

$$\Delta|_{x=0}\{x/x'\} = \underbrace{(\neg x \vee \neg x')|_{x=0}}_{=1}\{x/x'\} = 1 \ .$$

For state $s_1 = [x = 1]$, $Post(s_1) = \{s_0\} = \{[x = 0]\}$, a symbolic representation of $Post(s_1)$ is:

$$\Delta|_{x=1}\{x/x'\} = \underbrace{(\neg x \vee \neg x')|_{x=1}}_{=\neg x'}\{x/x'\} = \neg x \ .$$

Computing $Pre^*(B)$

We can now describe the backward BFS-based reachability analysis to compute all states in $Pre^*(B) = \{s \in S \mid s \models \exists\Diamond B\}$.

1. Start with the switching function $f_0 = \chi_B$ that characterizes the set $T_0 = B$;
2. Then, compute the characteristic functions $f_{j+1} = \chi_{T_{j+1}}$ of

$$T_{j+1} = T_j \cup \{s \in S \mid \exists s' \in S.\ s' \in Post(s) \wedge s' \in T_j\}$$

3. The set of states s where the condition $\exists s' \in S.\ s' \in Post(s) \wedge s' \in T_j$ holds is given by the switching function:

$$\exists \bar{x}'.\ (\underbrace{\Delta(\bar{x}, \bar{x}')}_{s' \in Post(s)} \wedge \underbrace{f_i(\bar{x}')}_{s' \in T_j}), \quad \text{where } f_j(\bar{x}') = f_j\{\bar{x}'/\bar{x}\}.$$

2.2 Encoding Transition Systems by Switching Functions

This BFS-based technique can easily be adapted to treat constrained reachability properties $\exists(C\,\mathbf{U}\,B)$ for subsets B, C of S:

Algorithm 1: Symbolic computation of $Sat(\exists(C\,\mathbf{U}\,B))$

$f_0(\bar{x}) := \chi_B(\bar{x})$;
$j := 0$;
repeat
  $f_{j+1}(\bar{x}) := f_j(\bar{x}) \vee (\chi_C(\bar{x}) \wedge \exists \bar{x}'.\ (\Delta(\bar{x}, \bar{x}') \wedge f_j(\bar{x}')))$;
  $j := j + 1$;
until $f_j(\bar{x}) = f_{j-1}(\bar{x})$;
return $f_j(\bar{x})$;

$$Sat(\exists \bigcirc B) = \exists \bar{x}'.\ (\underbrace{\Delta(\bar{x}, \bar{x}')}_{s' \in Post(s)} \wedge \underbrace{\chi_B(\bar{x}')}_{s' \in B})$$

2.2 Encoding Transition Systems by Switching Functions

The set $Sat(\exists\Box B)$ of all states s that have an infinite path consisting of states in a given set B can be computed symbolically

1. $T_0 = B$
2. $T_{j+1} = T_j \cap \{s \in S \mid \exists s' \in S.\ s' \in Post(s) \wedge s' \in T_j\}$

Algorithm 2: Symbolic computation of $Sat(\exists\Box B)$

$f_0(\bar{x}) := \chi_B(\bar{x})$;
$j := 0$;
repeat
  $f_{j+1}(\bar{x}) := f_j(\bar{x}) \wedge \exists \bar{x}'.\ (\Delta(\bar{x}, \bar{x}') \wedge f_j(\bar{x}'))$;
  $j := j + 1$;
until $f_j(\bar{x}) = f_{j-1}(\bar{x})$;
return $f_j(\bar{x})$;

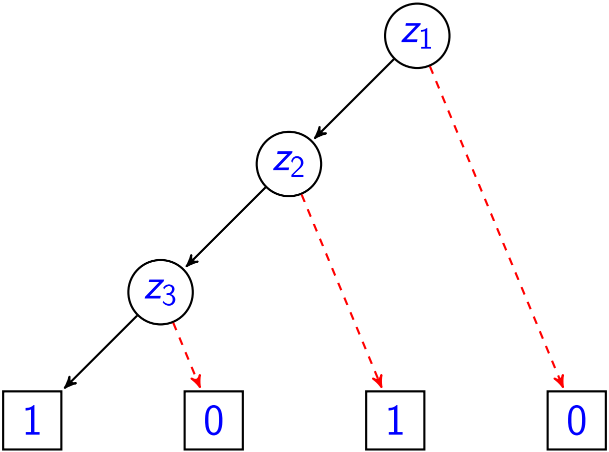
2.3 Odered Binary Decision Diagrams

Ordered Binary Decision Diagrams

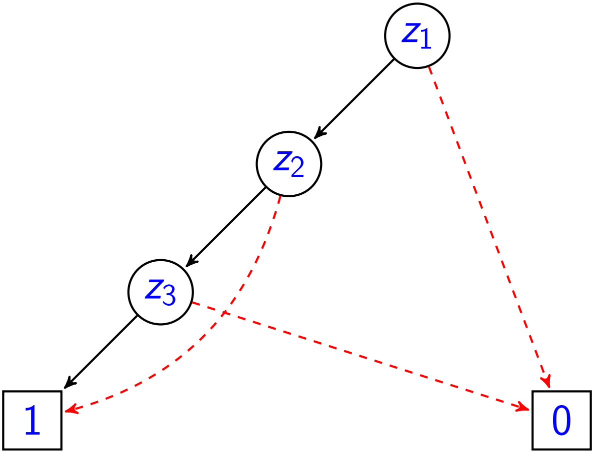
Ordered binary decision diagrams (OBDD), a data structure for switching functions that relies on a compactification of binary decision trees

1. Collapsing constant subtrees (i.e., subtrees where all terminal nodes have the same value) into a single node
2. Identifying nodes with isomorphic subtrees

Binary decision diagram for $z_1 \wedge (\neg z_2 \vee z_3)$:

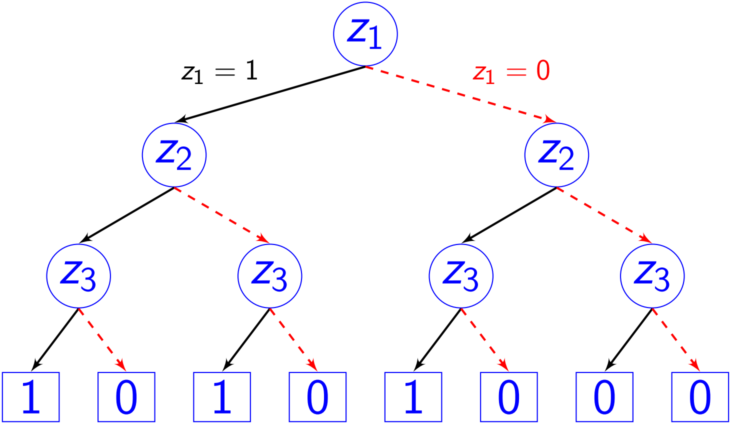


Binary decision diagram for $z_1 \wedge (\neg z_2 \vee z_3)$:

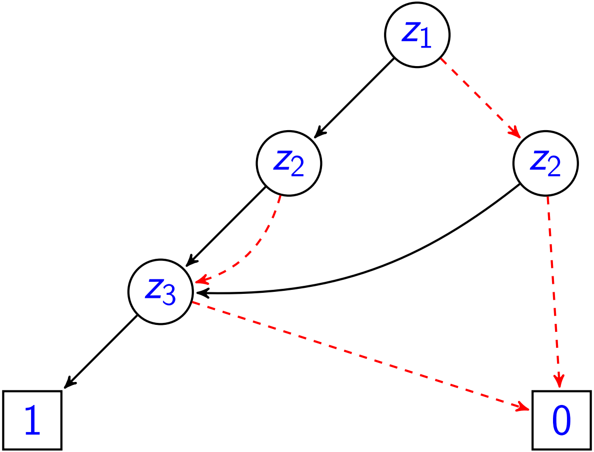


2.3 Odered Binary Decision Diagrams

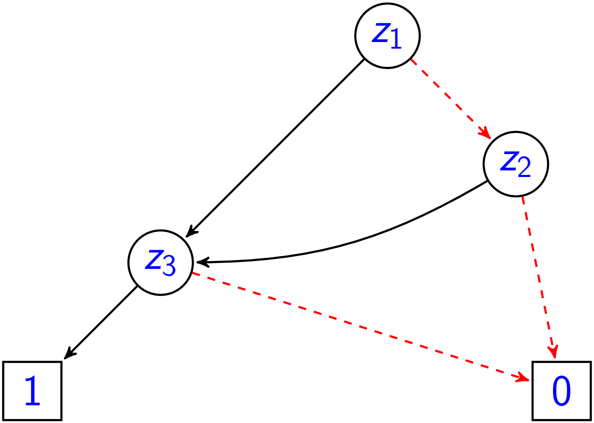
Binary decision tree for $f = (z_1 \wedge z_3) \vee (\neg z_2 \vee z_3)$: The three subtrees of the $z3$-nodes for the cofactors $f|_{z_1=0,z_2=1}$, $f|_{z_1=1,z_2=0}$ and $f|_{z_1=1,z_2=1}$ are isomorphic and can thus be collapsed



Binary decision diagram for $f = (z_1 \wedge z_3) \vee (\neg z_2 \vee z_3)$: now the z_2-node for the cofactor $f|_{z_1=1}$ becomes redundant, as regardless whether $z_2 = 0$ or $z_2 = 1$,



Binary decision diagram for $f = (z_1 \wedge z_3) \vee (\neg z_2 \vee z_3)$:



Variable Ordering

Let *Var* be a finite set of variables. A **variable ordering** for *Var* denotes any tuple $\wp = (z_1, \ldots, z_m)$ such that $Var = \{z_1, \ldots, z_m\}$ and $z_i \neq z_j$ for $1 \leq i < j \leq m$.

$$z_i <_\wp z_j \quad \text{iff} \quad i < j$$
$$z_i \leq_\wp z_j \quad \text{iff} \quad i < j \text{ or } i = j$$

Definition 7 (Ordered Binary Decision Diagram (OBDD))

Let $\wp$ be a variable ordering for Var. A $\wp$-ordered binary decision diagram ($\wp$-OBDD for short) is a tuple

$$\mathfrak{B} = (V, V_I, V_T, succ_0, succ_1, var, val, v_0)$$

1. $V = V_I \cup V_T$: a finite set V of **nodes**, V_I : set of **inner** nodes, V_T: set of **terminal** nodes
2. $succ_0, succ_1 : V_I \to V$, assign to each inner node v a **0-successor** $succ_0(v) \in V$ and **1-successor** $succ_1(v) \in V$
3. $var : V_I \to Var$, assigns to each **inner** node v a **variable** $var(v) \in Var$
4. $val : V_T \to \{0, 1\}$, assigns to each **terminal** node a **function value**
5. a **root node** $v_0 \in V$

For each path $v_0 v_1 ... v_n$ in $\mathfrak{B}$ we have $v_i \in V_I$ for $1 \leq i < n$ and

$$var(v_0) <_{\wp} var(v_1) <_{\wp} ... <_{\wp} var(v_n),$$

where for the terminal nodes we put $var(v) = \perp$(undefined) and extend $<_{\wp}$ by $z <_{\wp} \perp$ for all variables $z \in Var$, and

$$var : V \rightarrow Var \cup \{\perp\}$$

This yields that the underlying graph of an OBDD is acyclic.

2.3 Odered Binary Decision Diagrams

Semantics of OBDDs

Let $\mathfrak{B}$ be an $\wp$-OBDD. The semantics of $\mathfrak{B}$ is the switching function $f_{\mathfrak{B}}$ for *Var*:

- $f_{\mathfrak{B}}([z_1 = b_1, ..., z_m = b_m])$ is the value of the terminal node that will be reached from the root

Sub-OBDD, Switching Function for the Nodes

Let $\mathfrak{B}$ be a $\wp$-OBDD,

1. If v is a node in $\mathfrak{B}$, then the **sub-OBDD** induced by v, denoted $\mathfrak{B}_v$ arises from $\mathfrak{B}$ by declaring v as the root node and removing all nodes that are not reachable from v
2. The **switching function** for node, denoted f_v, is the switching function for *Var* that is given by the sub-OBDD $\mathfrak{B}_v$.

Lemma 8 (Bottom-up Characterization of the Functions f_v)

Let $\mathfrak{B}$ be a $\wp$-OBDD. The switching functions f_v for the nodes $v \in V$ are given as follows:

1. *If v is a terminal node, then f_v is the constant switching function with value $val(v)$.*
2. *If v is a z-node ($var(v) = z$), then*

$$f_v = \underbrace{\left(\neg z \wedge f_{succ_0(v)}\right) \vee \left(z \wedge f_{succ_1(v)}\right)}_{\textbf{Shannon Expansion}}$$

3. $f_{\mathfrak{B}} = f_{v_0}$ *for the root v_0 of $\mathfrak{B}$*

This yields that

$$f_v = f_{\mathfrak{B}}|_{z_1=b_1,\ldots,z_i=b_i}$$

where $[z_1 = b_1, \ldots, z_i = b_i]$ leads from the root v_0 of $\mathfrak{B}$ to node v.

2.3 Odered Binary Decision Diagrams

$\wp$-Consistent Cofactor

Let f be a switching function for Var and $\wp = (z_1, ..., z_m)$ a variable ordering for Var. A switching function f' for Var is called a $\wp$-consistent cofactor of f if such that $f' = f|_{z_1=b_1,...,z_i=b_i}$ for some $i \in \{0, 1, ..., m\}$.

Example 9

If $f = z_1 \wedge (z_2 \vee \neg z_3)$ and $\wp = (z_1, z_2, z_3)$

1. f, $f|_{z_1=1} = z_2 \vee \neg z_3$, $f|_{z_1=1,z_2=0} = \neg z_3$ and constants 0 and 1 are $\wp$-consistent cofactors of f

$\wp$-Consistent Cofactor

Let f be a switching function for Var and $\wp = (z_1, ..., z_m)$ a variable ordering for Var. A switching function f' for Var is called a $\wp$-consistent cofactor of f if such that $f' = f|_{z_1=b_1,...,z_i=b_i}$ for some $i \in \{0, 1, ..., m\}$.

Example 9

If $f = z_1 \wedge (z_2 \vee \neg z_3)$ and $\wp = (z_1, z_2, z_3)$

1. f, $f|_{z_1=1} = z_2 \vee \neg z_3$, $f|_{z_1=1,z_2=0} = \neg z_3$ and constants 0 and 1 are $\wp$-consistent cofactors of f
2. The cofactors $f|_{z_3=0} = z_1$ and $f|_{z_2=0} = z_1 \wedge \neg z_3$ are not $\wp$-consistent

$\wp$-Consistent Cofactor

Let f be a switching function for Var and $\wp = (z_1, \ldots, z_m)$ a variable ordering for Var. A switching function f' for Var is called a $\wp$-consistent cofactor of f if such that $f' = f|_{z_1=b_1,\ldots,z_i=b_i}$ for some $i \in \{0, 1, \ldots, m\}$.

Example 9

If $f = z_1 \wedge (z_2 \vee \neg z_3)$ and $\wp = (z_1, z_2, z_3)$

1. f, $f|_{z_1=1} = z_2 \vee \neg z_3$, $f|_{z_1=1,z_2=0} = \neg z_3$ and constants 0 and 1 are $\wp$-consistent cofactors of f
2. The cofactors $f|_{z_3=0} = z_1$ and $f|_{z_2=0} = z_1 \wedge \neg z_3$ are not $\wp$-consistent
3. The cofactor $f|_{z_2=0,z_3=1}$ is $\wp$-consistent as it agrees with the cofactors $f|_{z_1=0}$ or $f|_{z_1=1,z_2=0,z_3=1}$

Lemma 10 (Nodes in OBDDs and $\wp$-Consistent Cofactors)

1. *For each node v of an $\wp$-OBDD $\mathfrak{B}$, the switching function f_v is a $\wp$-consistent cofactor of $f_{\mathfrak{B}}$*
2. *Vice versa, for each $\wp$-consistent cofactor f' of $f_{\mathfrak{B}}$ there is at least one node v in $\mathfrak{B}$ such that $f_v = f'$.*

- Given a $\wp$-OBDD $\mathfrak{B}$ and a $\wp$-consistent cofactor f' of $f_{\mathfrak{B}}$ there **could be more than one node** in $\mathfrak{B}$ representing f'
- e.g., $f|_{z_1=0} = f|_{z_1=0,z_2=b} = f|_{z_1=0,z_2=n,z_3=c} = 0$ for all $b, c \in \{0, 1\}$ (for $f = z_1 \wedge (\neg z_2 \vee z_3)$)