Mathematical Foundation of Computer Sciences IV

Efficient Data Structures and Algorithms for FA

Guoqiang Li

School of Software, Shanghai Jiao Tong University

Ordered Binary Decision Diagram

What is OBDD

Practically efficient DAG structure that represents Boolean formulae.

Theoretically, complexity does not improve.

Used in logical validation, such as equivalence checking, satisfiability checking etc.

Used in complex algorithms on automata, say model checking.

Binary Decision Trees (BDT)

A Binary Decision Tree is a rooted, directed tree, consisting of two types of vertices, terminal, and nonterminal ones

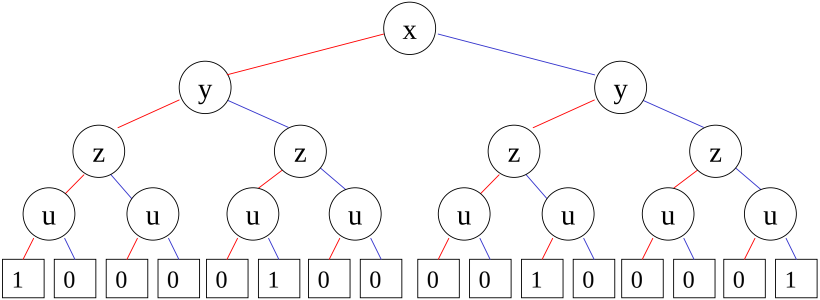
Each nonterminal vertex v has two successors: $low(v)$ and $high(v)$

- $low(v)$ corresponds to the case where v is assigned 0.
- $high(v)$ corresponds to the case where v is assigned 1.

Each terminal vertex v is labeled by $value(v)$ which is either 0 or 1.

Example: BDT

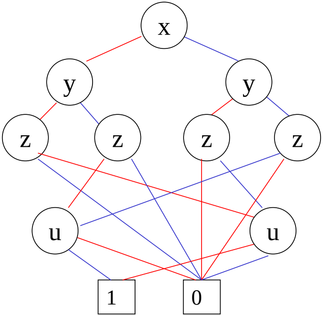
$$f(x, y, z, u) = (x \leftrightarrow z) \wedge (y \leftrightarrow u)$$



Ordered Binary Decision Diagram (OBDD)

BDTs are essentially the same size as truth tables.

A BDD is obtained by merging isomorphic subtrees of a BDT.



Ordered Binary Decision Diagram (OBDD)

BDTs are essentially the same size as truth tables.

A BDD is obtained by merging isomorphic subtrees of a BDT.

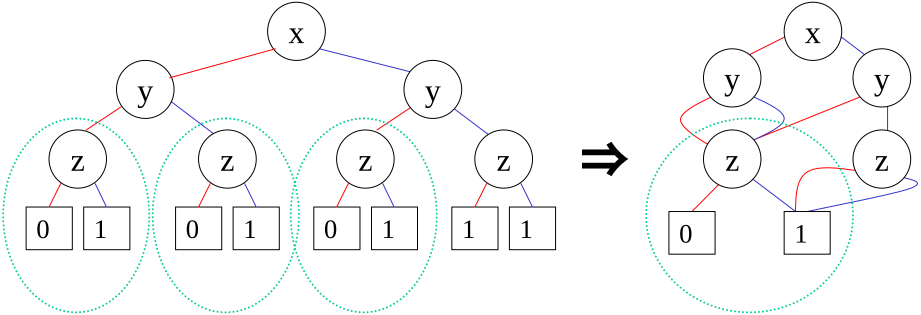
It is desirable to have a canonical representation for boolean functions. Two boolean functions are logically equivalent iff they have isomorphic representations.

- The boolean variables should appear in the same order along each path from the root to a terminal.
- There should be no isomorphic subtrees or redundant vertices.

Reduce to Canonical Representation: Step 1

Sharing isomorphic subtrees.

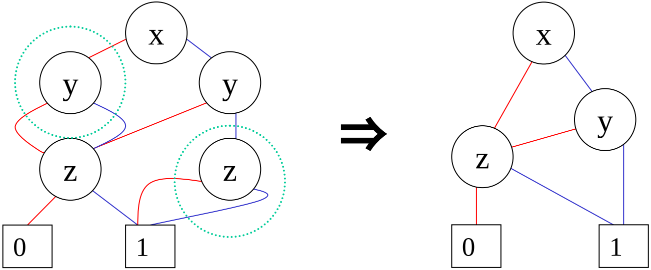
$$(x \land y) \lor z, x \prec y \prec z$$



Reduce to Canonical Representation: Step 2

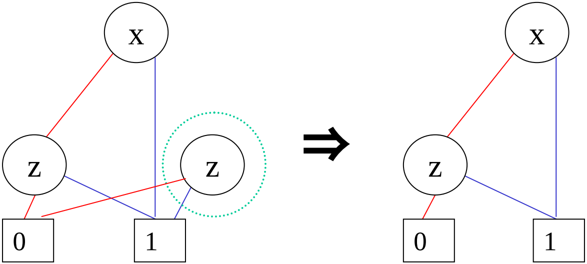
Removing tautological vertices ($low(v) = high(v)$).

$$(x \wedge y) \vee z, x \prec y \prec z$$



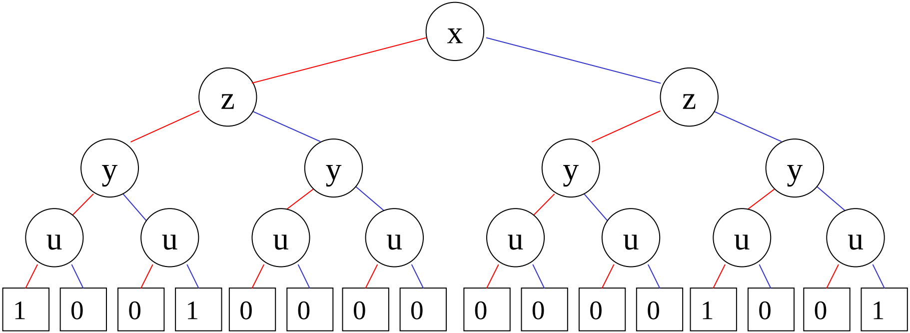
Reduce to Canonical Representation: Step 3

Removing unreachable vertices from the root.



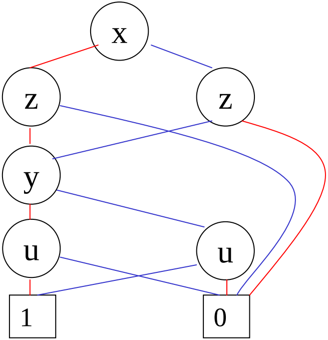
Example: OBDD with $x \prec z \prec y \prec u$

$$f(x, y, z, u) = (x \leftrightarrow z) \wedge (y \leftrightarrow u)$$



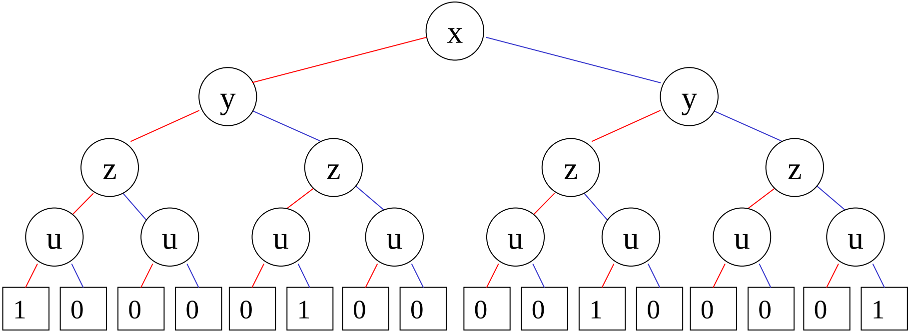
Example: OBDD with $x \prec z \prec y \prec u$

$$f(x, y, z, u) = (x \leftrightarrow z) \wedge (y \leftrightarrow u)$$



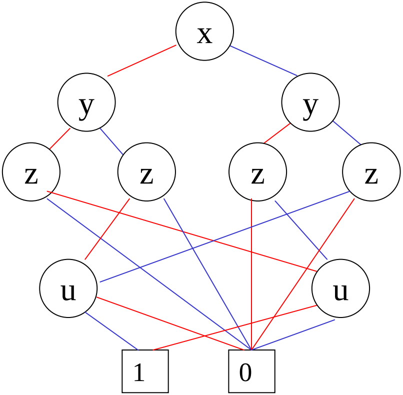
Example: OBDD with $x \prec y \prec z \prec u$

$$f(x, y, z, u) = (x \leftrightarrow z) \wedge (y \leftrightarrow u)$$



Example: OBDD with $x \prec y \prec z \prec u$

$$f(x, y, z, u) = (x \leftrightarrow z) \land (y \leftrightarrow u)$$



Properties of OBDD

Fix an order among variables, OBDD of a formula in canonical form is unique.

This implies efficient tests for

- validity: whether OBDD is only terminal
- equivalence: compare pointers (assuming sharing).

Number of nodes may explode exponentially (though usually quite small).

Number of nodes heavily depends on choice of an order (many heuristics).

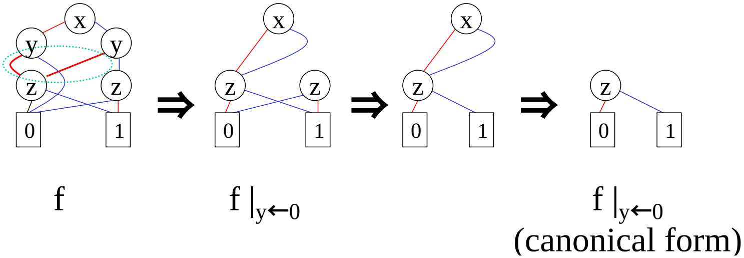
- e.g. $(a_1 \leftrightarrow b_1) \wedge \ldots \wedge (a_n \leftrightarrow b_n)$
- when $a_1 \prec b_1 \prec \ldots \prec a_n \prec b_n$, $3n + 2$ nodes.
- when $a_1 \prec \ldots a_n \prec b1 \prec \ldots \prec b_n$, $3 \times 2^n - 1$ nodes.

$$f|_{x\leftarrow b}(\ldots,x,\ldots) = f(\ldots,b,\ldots)$$

As OBDD operations,

- $f|_{x\leftarrow 0}$ corresponds to "replace subtree at x with $low(x)$ "
- $f|_{x\leftarrow 1}$ corresponds to "replace subtree at x with $high(x)$ "

$f|_{y\leftarrow 0}$



Logical Operation: Assignment

Shannon Expansion

$$f = (\neg x \wedge f|_{x \leftarrow 0}) \vee (x \wedge f|_{x \leftarrow 1})$$

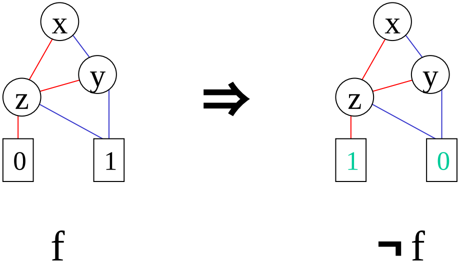
Quantified Boolean Formula (QBF)

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

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

Logical Operation: Negation

$\neg f$: alternate value at terminal vertices.



Efficient Logical Operations

Assuming efficient $\wedge$ operations,

- $f \vee g = \neg(\neg f \wedge \neg g)$
- $\exists x.f = f|_{x \leftarrow 0} \vee f|_{x \leftarrow 1}$
- $\forall x.f = f|_{x \leftarrow 0} \wedge f|_{x \leftarrow 1}$

Logical Operation: Conjunction

Top-down recursive operation for $f \wedge g$

When f or g is a constant (e.g. g)

$$f \wedge 0 = 0, f \wedge 1 = f$$

When neither f nor g is a constant, use Shannon expansion for the topmost variable.

$$f \wedge g = (\neg x \wedge f|_{x \leftarrow 0} \wedge g|_{x \leftarrow 0}) \vee (x \wedge f|_{x \leftarrow 1} \wedge g|_{x \leftarrow 1})$$

Logical Operation: Conjunction

Shannon expansion for the topmost variable.

$$f \wedge g = (\neg x \wedge f|_{x \leftarrow 0} \wedge g|_{x \leftarrow 0}) \vee (x \wedge f|_{x \leftarrow 1} \wedge g|_{x \leftarrow 1})$$

Case 1. $f.top = g.top = x$

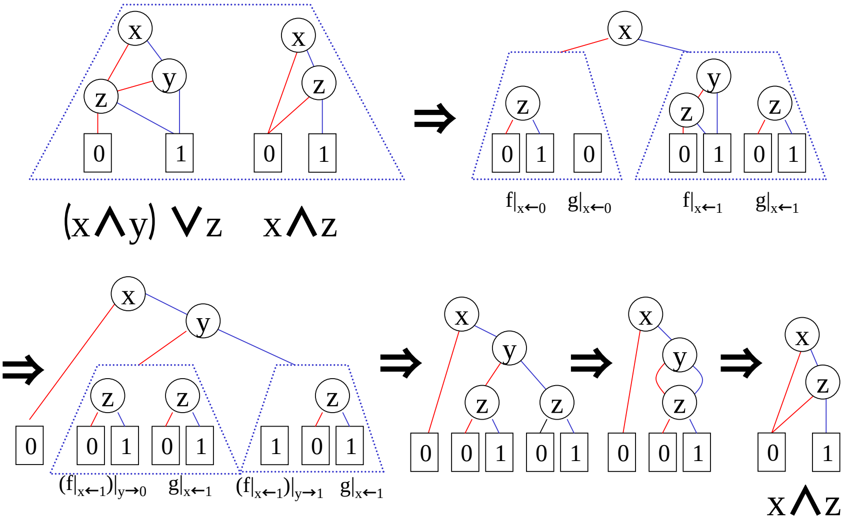
$$low(x) = f|_{x \leftarrow 0} \wedge g|_{x \leftarrow 0}, high(x) = f|_{x \leftarrow 1} \wedge g|_{x \leftarrow 1}$$

Case 2. $f.top = x \prec g.top$

$$low(x) = f|_{x \leftarrow 0} \wedge g, high(x) = f|_{x \leftarrow 1} \wedge g$$

$O(|f| \cdot |g|)$

Logical Operation: Conjunction



Representation Kripke Structure by OBDD

$$M = (S, S_0, R, L)$$

For states S, numbering states by binary coding.

- $s_0 = (\ldots, 0, 0)$ i.e., $v_0 = 0, v_1 = 0, \ldots$
- $s_1 = (\ldots, 0, 1)$ i.e., $v_0 = 1, v_1 = 0, \ldots$
- $s_2 = (\ldots, 1, 0)$ i.e., $v_0 = 0, v_1 = 1, \ldots$

Transitions $(s_i, s_j) \in R$, represented as state $s_i \wedge s_j$. R is represented as disjunction of all transitions.

For an atom proposition φ, defining $L_\varphi = \{s \mid \varphi \in L(s)\}$.

Antichain Algorithm

The Usual Solution for Inclusion Problem

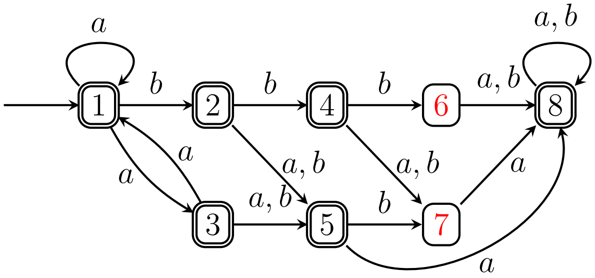
$$L(\mathcal{A}) \subseteq L(\mathcal{B})$$

- Complementation: $\mathcal{B}^c$
    - Cost: deterministic: low
    - Cost: nondeterministic $\Longrightarrow$ determination: high ($O(2^n)$)
- Intersection: $\mathcal{A} \cap \mathcal{B}^c$
    - Cost: low
- Emptiness: $L(\mathcal{A} \cap \mathcal{B}^c) = \emptyset$
    - Cost: low

Determination is the major fact to consume the execution time.

Determination for NFA: Subset Construction

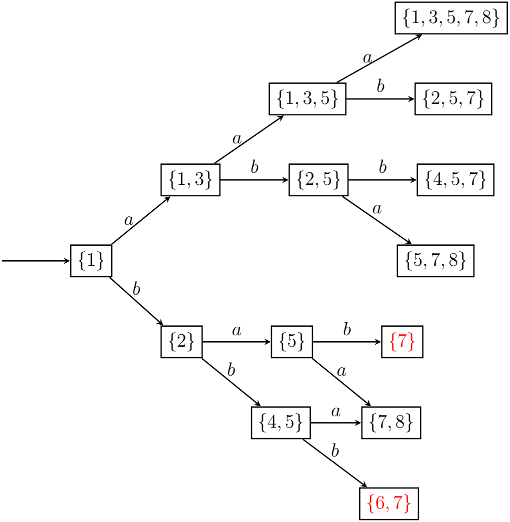
NFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$



DFA $\mathcal{A}' = (Q', \Sigma, \delta', q_0', F')$

- $Q' = 2^Q$
- $\delta'(P, a) = \{q \in Q \mid \exists p \in P_1, q \in \delta(p, a)\}$
- $q_0' = \{q_0\}$
- $F' = \{P \in Q' \mid P \cap F \neq \emptyset\}$

Determination for NFA: Subset Construction



Determination for NFA: Subset Construction

For büchi automata, DBA $\subset$ NBA.

Thus, this technique does not work. ($O(2^{n log n})$)

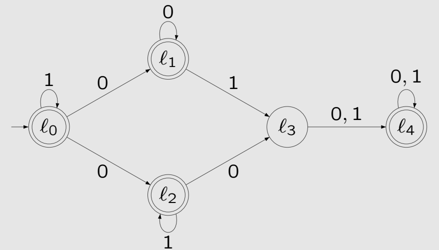
An Alternative Solution for Inclusion Problem

$$L(\mathcal{A}) \subseteq L(\mathcal{B})$$

- Complementation: $\mathcal{A}^c$
  - Cost: deterministic: low
- Union: $\mathcal{A}^c \cup \mathcal{B}$
  - Cost: low
- Universality: $L(\mathcal{A}^c \cup \mathcal{B}) = \Sigma^*$
  - Cost: high
  - The usual way to solve the universality is determination.
  - A new algorithm without determination. antichain
- Determination VS. Antichain

Universality: A Game Approach

- Consider two players: protagonist, antagonist
- Protagonist wants to show that $\mathcal{A}$ is not universal.
- Protagonist has to provide a finite word w such that no matter how the antagonist reads it on $\mathcal{A}$, the automaton ends up in a rejecting location.



- Example:
  - Protagonist: 101
  - Antagonist: $l_0 \xrightarrow{1} l_0 \xrightarrow{0} l_2 \xrightarrow{1} l_2$
  - Antagonist wins.
  - One-shot game.

Universality: A Game Approach

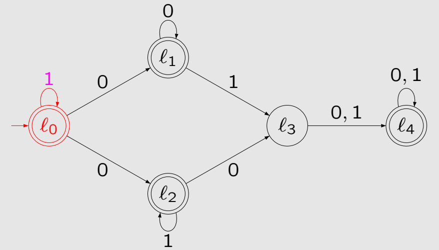
protagonist has a strategy to win the game

iff

$\mathcal{A}$ is not universal

Universality: A Game Approach

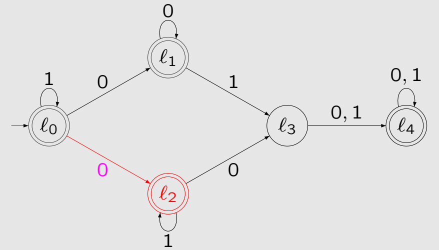
- Consider two players: protagonist, antagonist
- Protagonist provides a letter from w a time.
- Antagonist updates control locations based on $\mathcal{A}$ accordingly.



- Example:
  - protagonist: 1
  - antagonist: $l_0 \xrightarrow{1} l_0$

Universality: A Game Approach

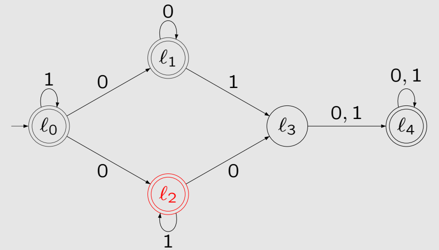
- Consider two players: protagonist, antagonist
- Protagonist provides a letter from w a time.
- Antagonist updates control locations based on $\mathcal{A}$ accordingly.



- Example:
  - protagonist: 10
  - antagonist: $l_0 \xrightarrow{1} l_0 \xrightarrow{0} l_2$

Universality: A Game Approach

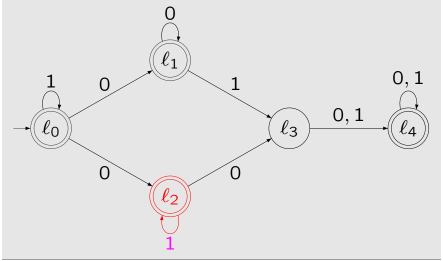
- Consider two players: protagonist, antagonist
- Protagonist provides a letter from w a time.
- Antagonist updates control locations based on $\mathcal{A}$ accordingly.



- Example:
  - protagonist: 10
  - antagonist: $l_0 \xrightarrow{1} \{l_0\} \xrightarrow{0} \{l_1, l_2\}$

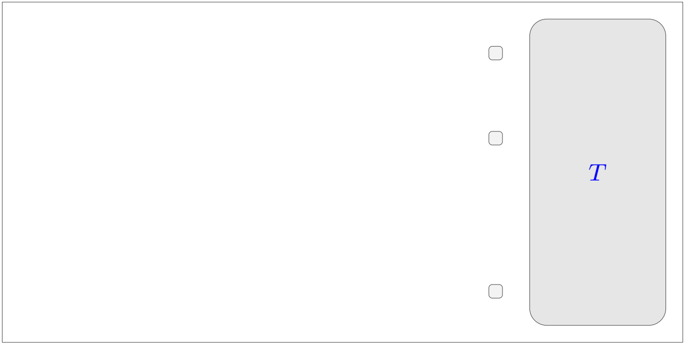
Universality: A Game Approach

- Consider two players: protagonist, antagonist
- Protagonist provides a letter from w a time.
- Antagonist updates control locations based on $\mathcal{A}$ accordingly.



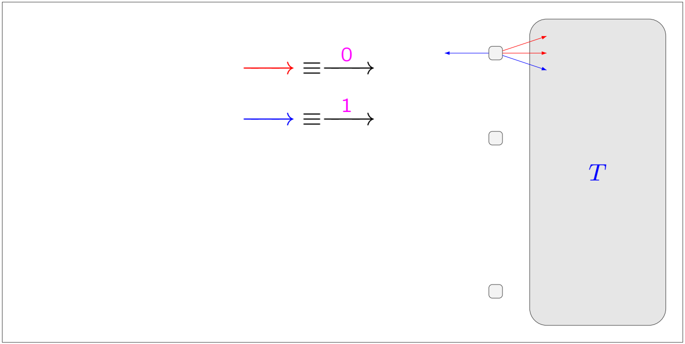
- Example:
  - protagonist: 101
  - antagonist: $l_0 \xrightarrow{1} \{l_0\} \xrightarrow{0} \{l_1, l_2\} \xrightarrow{1} \{l_2\}$
  - Turn-based blind game (game with null information).

- let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$
- to solve a blind reachability game G_T with the target $T = Q/F$



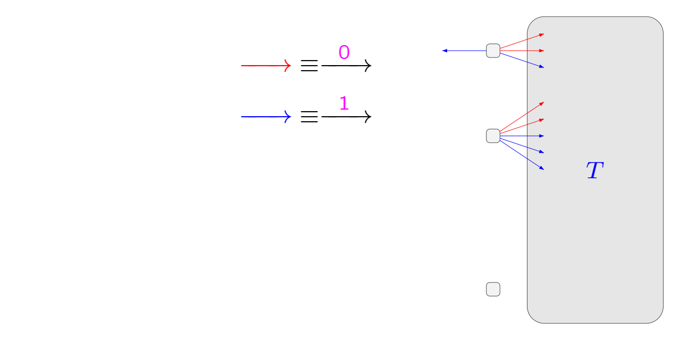
Solution of the Game

- let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$
- to solve a blind reachability game G_T with the target $T = Q/F$



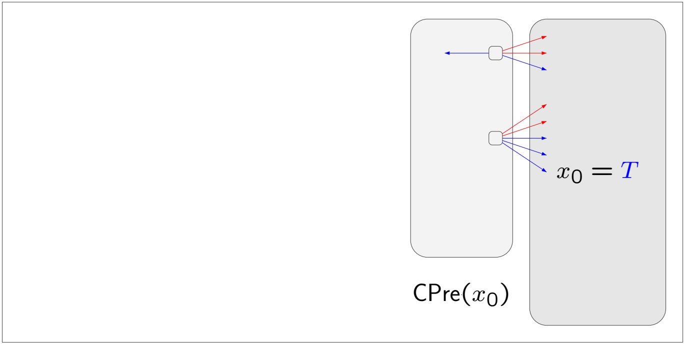
Solution of the Game

- let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$
- to solve a blind reachability game G_T with the target $T = Q/F$



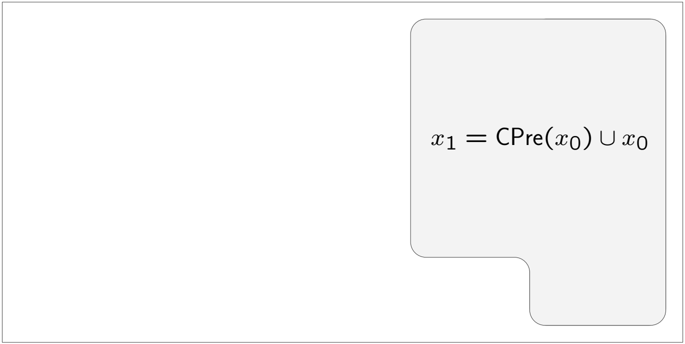
Solution of the Game

- let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$
- to solve a blind reachability game G_T with the target $T = Q/F$



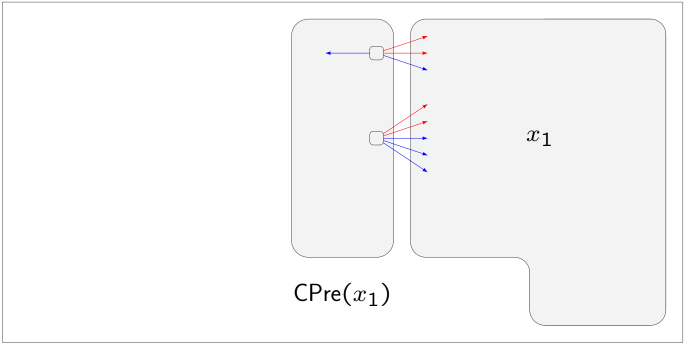
Solution of the Game

- let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$
- to solve a blind reachability game G_T with the target $T = Q/F$



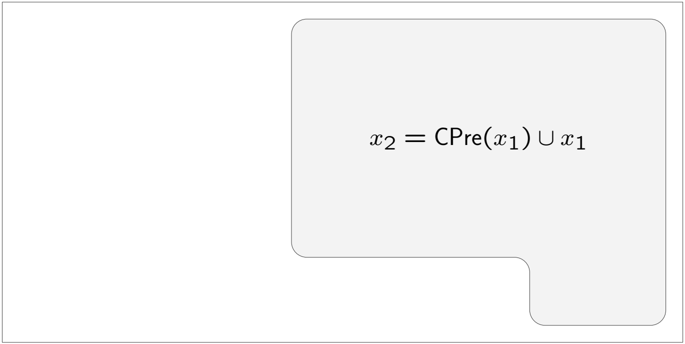
Solution of the Game

- let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$
- to solve a blind reachability game G_T with the target $T = Q/F$



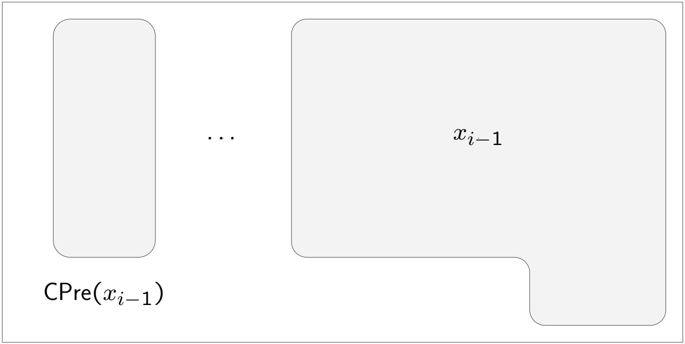
Solution of the Game

- let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$
- to solve a blind reachability game G_T with the target $T = Q/F$

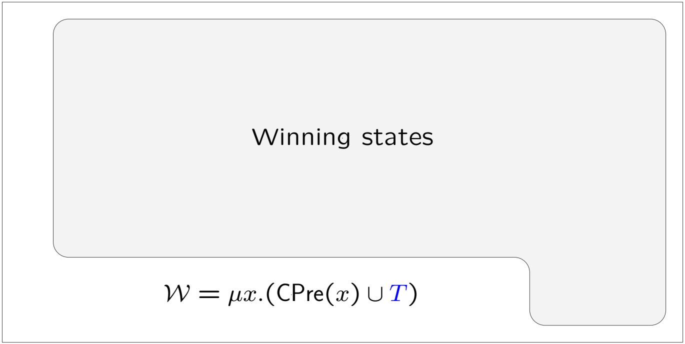


Solution of the Game

- let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$
- to solve a blind reachability game G_T with the target $T = Q/F$



- let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$
- to solve a blind reachability game G_T with the target $T = Q/F$



Solution of the Game

- let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$
- to solve a blind reachability game G_T with the target $T = Q/F$

- Recipe for solving reachability games
    - Compute the set of control locations that are winning in one move $CPre^{\mathcal{A}}(T)$
    - Iterate $Cpre^{\mathcal{A}}(.)$, compute $\mathcal{W} = \mu x.(CPre^{\mathcal{A}}(x) \cup T)$
    - Check whether $q_0 \in \mathcal{W}$

Small Formalization: Antichains

- let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$
- an antichain over Q is a set $\mathtt{q} \subseteq 2^Q$, such that $\forall s, s' \in \mathtt{q}. s \not\subset s'$. That is, a finite set of pairwise incomparable elements.
- define a monotone function:
  $CPre^{\mathcal{A}}(\mathtt{q}) = \lceil(\{s \mid \exists s' \in \mathtt{q},\ \exists \sigma \in \Sigma,\ s = cpre_{\sigma}^{\mathcal{A}}(s')\})\rceil$, where
  - $cpre_{\sigma}^{\mathcal{A}}(s) = \{q \in Q \mid \forall q' \in Q, \delta(q, \sigma, q') \rightarrow q' \in s\}$
  - $\lceil p \rceil$ denotes the maximal elements in p
  - $\lfloor p \rfloor$ denotes the minimal elements in p
- To check whether the initial state contained in the greatest fixed point of a lattice of the antichain.

The Algorithm

$$q_0 \subseteq \mu x.(CPre^{\mathcal{A}}(x) \cup T)$$

iff

$\mathcal{A}$ is not universal

The Lattice of Antichains

- For two antichains $p, p' \in L$, let $p \sqsubseteq p'$ iff $\forall s' \in p', \exists s \in p : s \subseteq s'$
- Given two antichains $p, p' \in L$, the $\sqsubseteq_{\texttt{lub}}$ is the antichain $p \sqcup p' = \lfloor \{ s \cup s' \mid s \in p \wedge s' \in p' \} \rfloor$
- Given two antichains $p, p' \in L$, the $\sqsubseteq_{\texttt{glb}}$ is the antichain $p \sqcap p' = \lfloor \{ s \mid s \in p \vee s \in p' \} \rfloor$
- The partial order $\sqsubseteq$ yields a complete lattice on the set L of antichains: $\mathit{Latt} = \langle L, \sqsubseteq, \sqcup, \sqcap, \emptyset, \{Q\} \rangle$
- It is possible to use least fixed point.
    - It is easy to get a counterexample.
    - we need not compute the final fixed point.

Least Fixed Point and Forward Search

- let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$
- define a monotone function:
  $CPost^{\mathcal{A}}(\mathrm{q}) = \lfloor(\{s \mid \exists\, s' \in \mathrm{q},\, \exists\, \sigma \in \Sigma,\, s = cpost_{\sigma}^{\mathcal{A}}(s')\})\rfloor$, where
  - $cpost_{\sigma}^{\mathcal{A}}(s) = \{q \in Q \mid \exists q' \in Q, \delta(q', \sigma, q)\}$
  - $\lfloor p \rfloor$ denotes the minimal elements in p

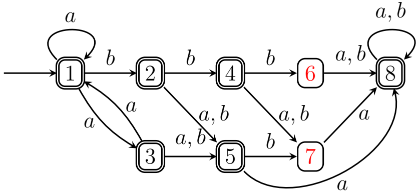
$$\exists s \in \mu x.(Cpost^{\mathcal{A}}(x) \sqcap \{q_0\}), s \cap F = \emptyset$$

iff

$\mathcal{A}$ is not universal

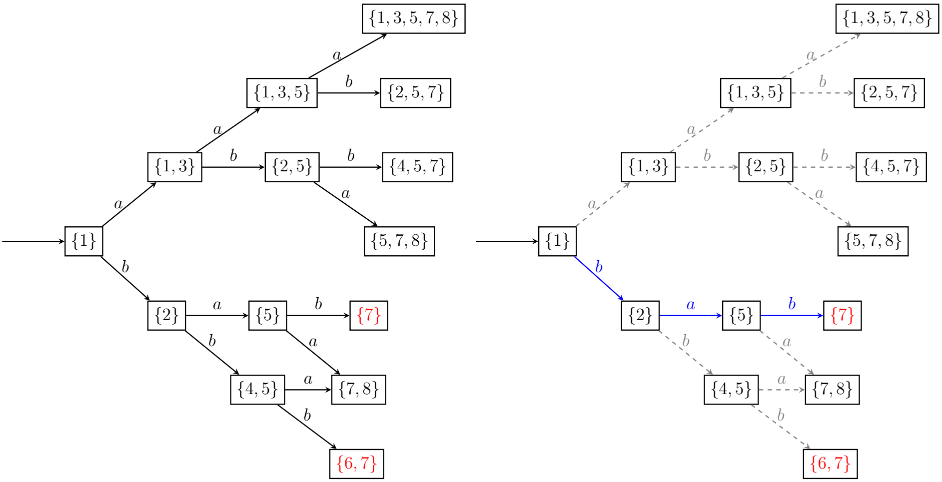
- When automaton is not universal, the computation can be stopped as soon as one of the sets does not intersect with F.

Examples



- $X_0 = \{\{1\}\}$
- $X_1 = cpost(X_0) \sqcap p_0 = \{\{1\}, \{2\}, \{1,3\}\} \sqcap \{\{1\}\} = \lfloor\{\{1\}, \{2\}, \{1,3\}\}\rfloor = \{\{1\}, \{2\}\}$
- $X_2 = cpost(X_1) \sqcap p_0 = \{\{1\}, \{2\}, \{1,3\}, \{5\}, \{4,5\}\} \sqcap \{\{1\}\} = \lfloor\{\{1\}, \{2\}, \{1,3\}, \{5\}, \{4,5\}\}\rfloor = \{\{1\}, \{2\}, \{5\}\}$
- $X_3 = cpost(X_2) \sqcap p_0 = \{\{1\}, \{2\}, \{1,3\}, \{5\}, \{4,5\}, \{7,8\}, \{7\}\} \sqcap \{\{1\}\} = \lfloor\{\{1\}, \{2\}, \{1,3\}, \{5\}, \{4,5\}, \{7,8\}, \{7\}\}\rfloor = \{\{1\}, \{2\}, \{5\}, \{7\}\}$
- $\{7\} \cap F = \emptyset$

Examples



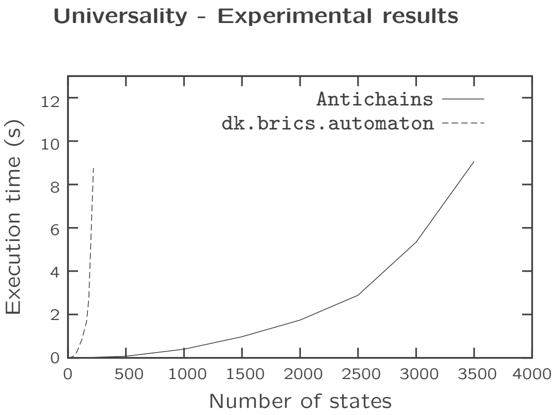
Two Equivalent Algorithms

Two algorithms,

- Backwards, greatest fixed point,
- Forwards, least fixed point,

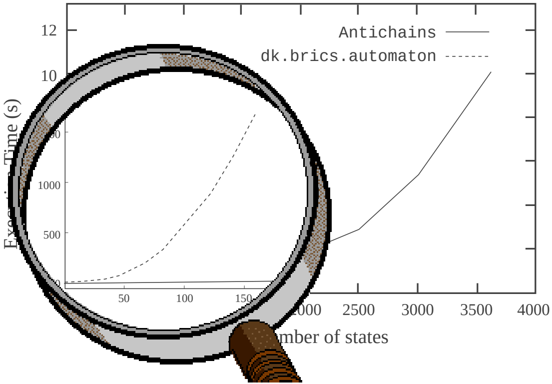
are equivalent.

Experimental Result (CAV'06)



Experimental Result (CAV'06)

Universality - Experimental results



Inclusion Problem of two NFAs

Let $\mathcal{A} = (Q_{\mathcal{A}}, \Sigma, \delta_{\mathcal{A}}, q^0_{\mathcal{A}}, F_{\mathcal{A}})$ and $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, \delta_{\mathcal{B}}, q^0_{\mathcal{B}}, F_{\mathcal{B}})$

The language inclusion can be checked using an antichain algorithm based on a richer lattice.

An antichain q over $Q_A \times 2^{Q_B}$ is a set such that for all
$\forall (q_1, s_1), (q_2, s_2) \in p.(q_1 = q_2) \wedge (s_1 \neq s_2) \rightarrow s_1 \not\subseteq s_2 \wedge s_2 \not\subseteq s_1$

Conclusion

Antichain algorithm does not improve the complexity in theory, it is significant in practice.

- e.g. there are no implementations of complementing Büchi automata, but now ALASKA

In a model checking view, antichain algorithm adopts "lazy determination"

- In lazy model checking, a model is expanded only when needed.
- In antichain algorithm, an automaton is determinized only when needed.