

Efficient Equality Checking for Non-Shared Binary Decision Diagrams

Steffan Christ Sølvesten, Jaco van de Pol

6th of September, 2021



$$f \leftrightarrow g \equiv \top$$

$$f \leftrightarrow g \equiv \top$$

$$\underbrace{O(\text{sort}(N^2))}_{\text{Apply}} + \underbrace{O(\text{sort}(N^2))}_{\text{Reduce}} + \underbrace{O(1)}_{\text{check is } \top} = O(\text{sort}(N^2))$$

Theorem (Bryant '86)

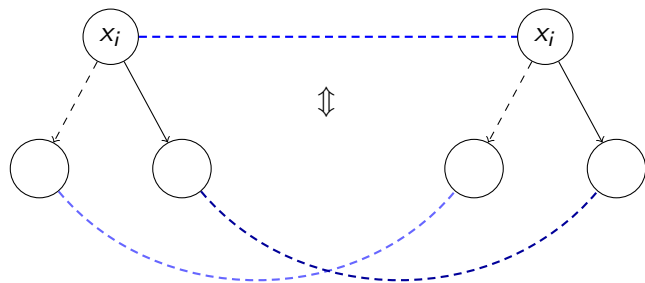
Let π be a variable order and $f : \mathbb{B}^n \rightarrow \mathbb{B}$ then there exists a unique (up to isomorphism) Reduced Ordered Binary Decision Diagram representing f with ordering π .

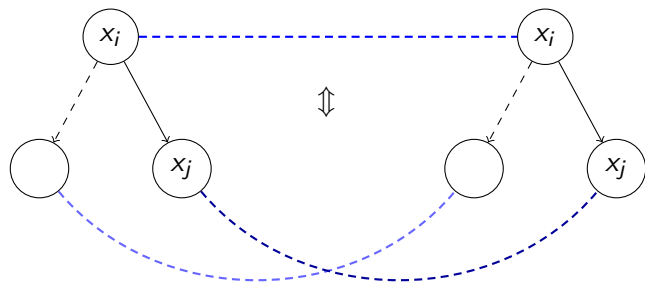
Theorem (Bryant '86)

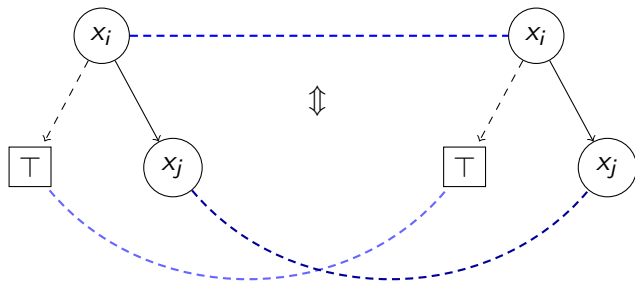
Let π be a variable order and $f : \mathbb{B}^n \rightarrow \mathbb{B}$ then there exists a unique (up to isomorphism) Reduced Ordered Binary Decision Diagram representing f with ordering π .

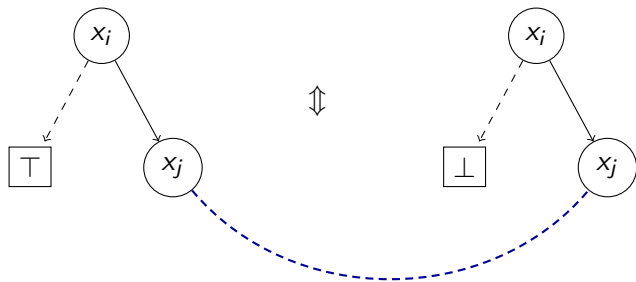
Trivial cases: $f \not\equiv g$ if there is a mismatch in

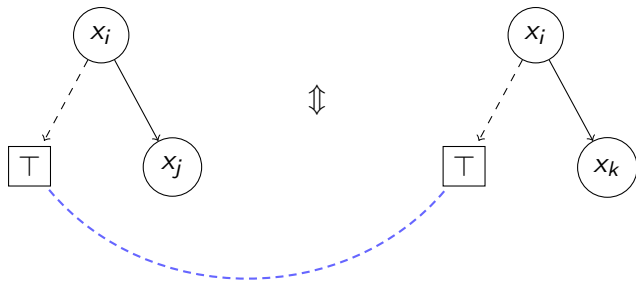
- $N_f \neq N_g$ Number of nodes $O(1)$ I/Os
- $L_f \neq L_g$ Number of levels $O(1)$ I/Os
- $N_{f,i} \neq N_{g,i}$ Number of nodes on a level $O(L/B)$ I/Os
- $L_{f,i} \neq L_{g,i}$ Label of an i th level $O(L/B)$ I/Os

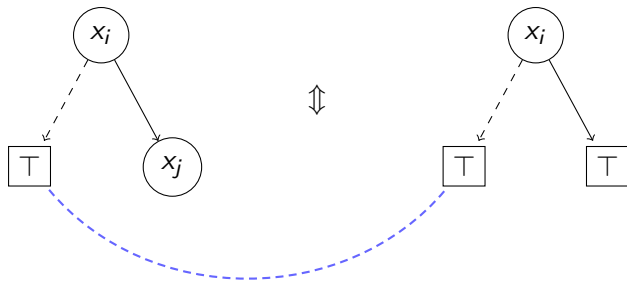












`IsIsomorphic(f , g)`

- Check whether root v_f of f and root v_g of g have a local violation.
- Check $low(v_f) \sim low(v_g)$ and $high(v_f) \sim high(v_g)$ “recursively”.

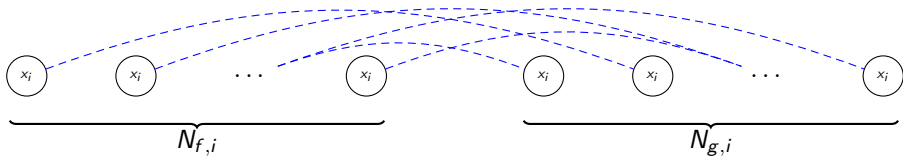
Return `false` on first violation. If there are no violations then return `true`.

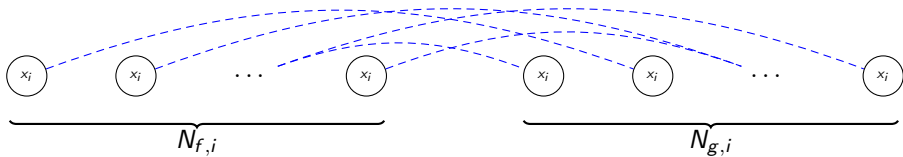
IsIsomorphic(f , g)

- Check whether root v_f of f and root v_g of g have a local violation.
- Check $low(v_f) \sim low(v_g)$ and $high(v_f) \sim high(v_g)$ “recursively”.

Return false on first violation. If there are no violations then return true.

$$\underbrace{O(\text{sort}(N^2))}_{\text{Apply'}} + \underbrace{\cancel{O(\text{sort}(N^2))}}_{\text{Reduce}} + \underbrace{\cancel{O(1)}}_{\text{check is T}} = O(\text{sort}(N^2))$$





Return false if more than $N_{f,i} = N_{g,i}$ pairs of nodes (v_f, v_g) are checked on level i .

$$O(\text{sort}(N))$$

Observation

The output of Reduce has the following properties

- Nodes on level i have their identifiers *consecutively* numbered

$$MAX - N_{f,i} + 1, \dots, MAX - 1, MAX .$$

- Nodes on level i are output sorted by their children

$$((i_1, id_1), low_1, high_1) <_{lex(i, low, high)} ((i_2, id_2), low_2, high_2) ,$$

where

$$\forall (i, id) : (i, id) < \perp < \top^1 .$$

¹Assuming the BDD is not negated. If that is the case then $(i, id) < \top < \perp$.

Theorem

If G_f and G_g are outputs of Reduce.

$G_f \sim G_g \iff \text{For all } i \in [0; N) \text{ the node } G_f[i] \text{ matches } G_g[i] \text{ numerically.}$

Proof.

\Leftarrow : Must describe the exact same graph.

\Rightarrow : Strong induction on BDD levels bottom-up ...



Theorem

If G_f and G_g are outputs of Reduce.

$$G_f \sim G_g \iff \text{For all } i \in [0; N) \text{ the node } G_f[i] \text{ matches } G_g[i] \text{ numerically.}$$

Proof.

\Leftarrow : Must describe the exact same graph.

\Rightarrow : Strong induction on BDD levels bottom-up ...



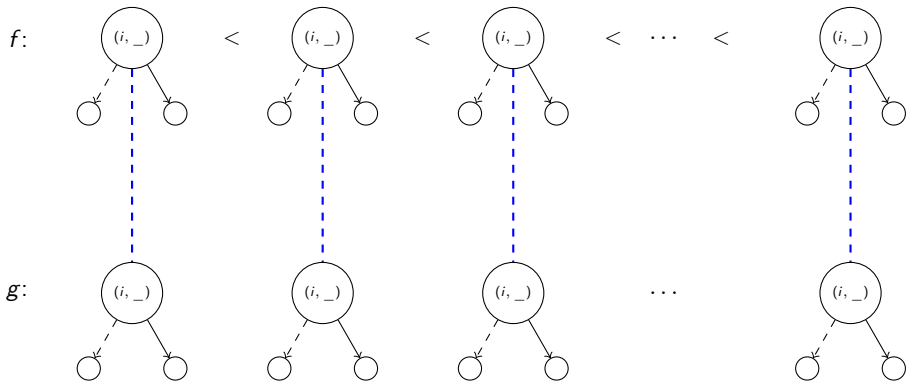
Corollary

If G_f and G_g are outputs of Reduce then $f \equiv g$ is computable using $2 \cdot N/B$ I/Os.²

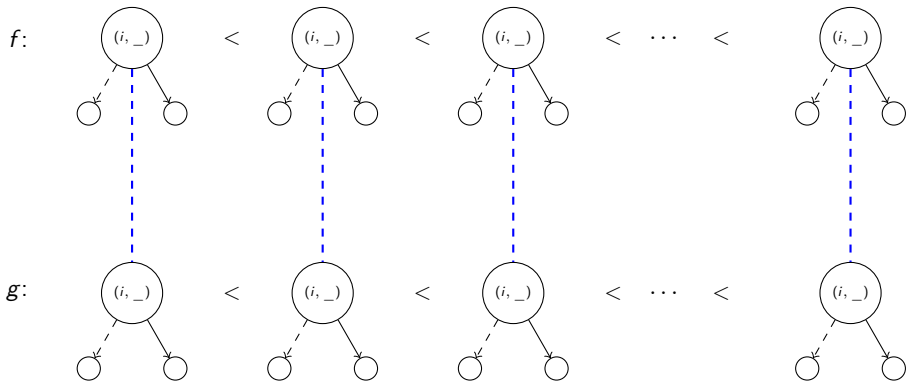
²Assuming they are both unnegated (or both negated).



Base case: n



Induction case: $i < n$



Induction case: $i < n$

Algorithm	Time (s)
$f \leftrightarrow g \equiv \top$	0.38
$O(\text{sort}(N))$	0.058
$2 \cdot N/B$	0.006

Checking the (EPFL Benchmark) *voter* circuit's single output gate ($|N_f| = |N_g| = 5.76$ MiB).