I/O-efficient Manipulation of Binary Decision Diagrams

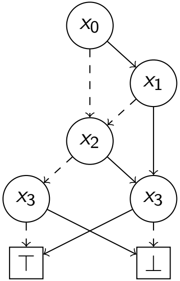
Steffan Christ Sølvsten

S. C. Sølvsten, J. van de Pol, A. B. Jakobsen, and M. W. B. Thomasen.
Adiar: Binary Decision Diagrams in External Memory. 2022

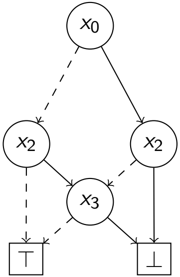
AARHUS
UNIVERSITY

Contents

Contents



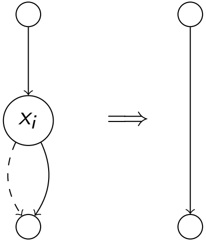
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$



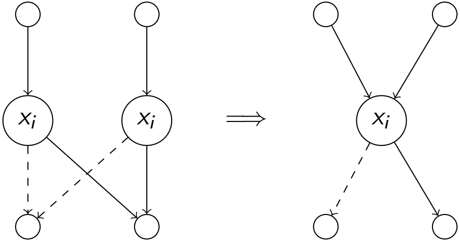
(b) $\neg(x_0 \ ? \ x_2 \wedge x_3 : x_2 \wedge x_3)$

Examples of (Reduced Ordered) Binary Decision Diagrams.

Theorem (Bryant '86)
For a fixed variable order, if one exhaustively applies the two rules below, then one obtains the Reduced OBDD, which is a unique canonical form of the function.



(1) Remove redundant nodes



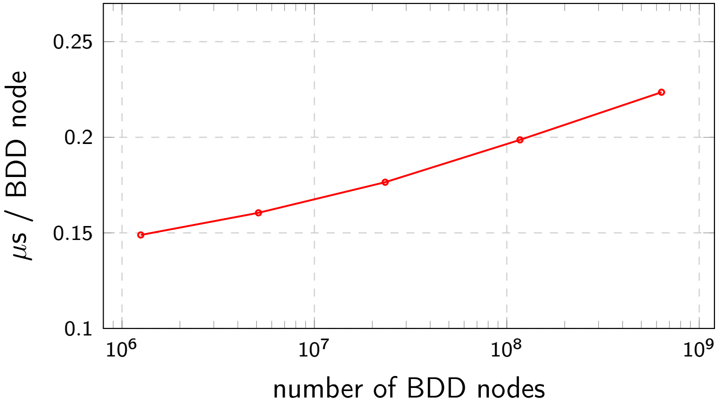
(2) Merge duplicate nodes

```
bdd_apply(f, g, ⊗):
  if  f, g ∈ {⊥, ⊤}
  then  f ⊗ g
  else let  i = top(f.(var), g.var)
            t = bdd_apply(f[x_i := ⊤], g[x_i := ⊤], ⊗)
            e = bdd_apply(f[x_i := ⊥], g[x_i := ⊥], ⊗)
       in make_node(i, t, e)
```

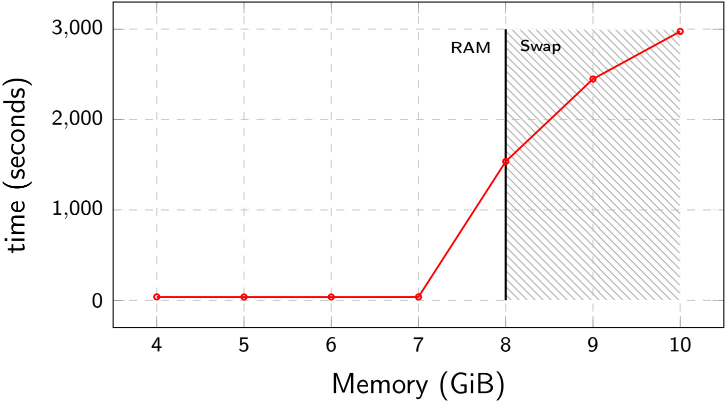
```
bdd_apply(f, g, ⊗):
  if f,g ∈ {⊥,⊤}
  then f ⊗ g
  else let i = top(f.(var), g.var)
           t = bdd_apply(f[x_i := ⊤], g[x_i := ⊤], ⊗)
           e = bdd_apply(f[x_i := ⊥], g[x_i := ⊥], ⊗)
       in make_node(i,t,e)
```

Theorem
`bdd_apply` *runs in* $O(N_f \cdot N_g)$ *time.*

- Memoisation (*Computation Cache*) ensures each recursion is computed only once.
- Reduction Rules can be maintained within `make_node(i,t,e)` in $O(1)$ time.
  1. Redundancy is resolved with an if-statement.
  2. Duplication is avoided with a hash table (*Unique Node Table*).

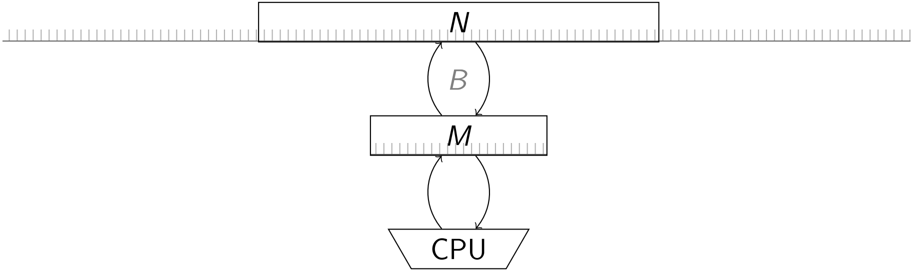


Running time of *BuDDy* for the *N*-Queens problem.



Running time of *BuDDy* for *Tic-Tac-Toe* with $N = 21$.

Contents



The I/O model by Aggarwal and Vitter '87

For any realistic values of N, M, and B we have that

$$N/B \quad < \quad \text{sort}(N) \triangleq N/B \cdot \log_{M/B} N/B \quad \ll \quad N \ ,$$

Theorem (Aggarwal and Vitter '87)
N elements can be sorted in $\Theta(\mathit{sort}(N))$ I/Os.

Theorem (Arge '95)
N elements can be inserted in and extracted from a Priority Queue in $\Theta(\mathit{sort}(N))$ I/Os.

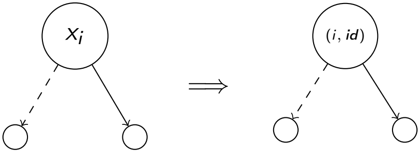
CountPaths : *Example*

Algorithm	Time Complexity
`bdd_pathcount`	$O(N_f)$
`bdd_not`	$O(N_f)$
`bdd_restrict`	$O(N_f)$
`bdd_apply`	$O(N_f \cdot N_g)$
`bdd_equal`	$O(1)$

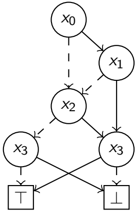
Algorithm	I/O-Complexity
`bdd_pathcount`	$O(N_f)$
`bdd_not`	$O(N_f)$
`bdd_restrict`	$O(N_f)$
`bdd_apply`	$O(N_f \cdot N_g)$
`bdd_equal`	$O(1)$

Contents

Contents

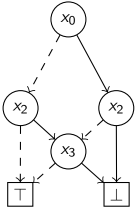


$$(i_1, id_1) < (i_2, id_2) \equiv i_1 < i_2 \vee (i_1 = i_2 \wedge id_i < id_j)$$



$$
\begin{array}{llr}
[& ((0,0),(2,0),(1,0)) & , \\
 & ((1,0),(2,0),(3,1)) & , \\
 & ((2,0),(3,0),(3,1)) & , \\
 & ((3,0),\top,\bot) & , \\
 & ((3,1),\bot,\top) &]
\end{array}
$$

(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$



$$
\begin{array}{llr}
[& ((0,0),(2,0),(2,1)) & , \\
 & ((2,0),\bot,(3,0)) & , \\
 & ((2,1),(3,0),\top) & , \\
 & ((3,0),\top,\bot) &]
\end{array}
$$

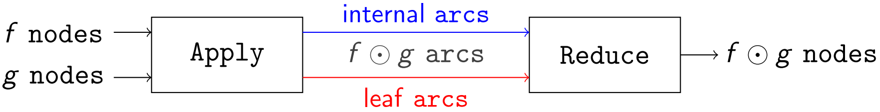
(b) $\neg(x_0 \ ?\ x_2 \wedge x_3 : x_2 \wedge x_3)$

Node-based representation of prior shown BDDs

CountPaths : *Example*

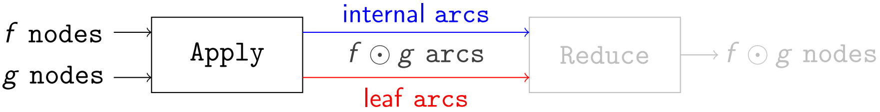
Contents

Apply



f nodes
g nodes
Apply
internal arcs
$f \odot g$ arcs
leaf arcs
Reduce
$f \odot g$ nodes

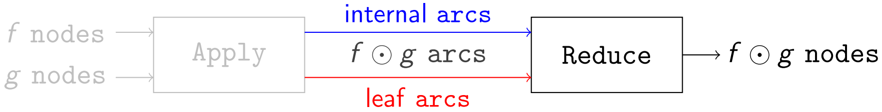
Apply



f nodes
g nodes
Apply
internal arcs
$f \odot g$ arcs
leaf arcs
Reduce
$f \odot g$ nodes

Apply : *Example*

Apply


f nodes
g nodes
Apply
internal arcs
$f \odot g$ arcs
leaf arcs
Reduce
$f \odot g$ nodes

Apply : *Example* ***(Continued)***

Algorithm	I/O-Complexity
`bdd_pathcount`	$O(\text{sort}(N_f))$
`bdd_not`	$O(N_f / B)$
`bdd_restrict`	$O(\text{sort}(N_f))$
`bdd_apply`	$O(\text{sort}(N_f \cdot N_g))$

Algorithm	I/O-Complexity
`bdd_pathcount`	$O(\text{sort}(N_f))$
`bdd_not`	$O(N_f/B)$
`bdd_restrict`	$O(\text{sort}(N_f))$
`bdd_apply`	$O(\text{sort}(N_f \cdot N_g))$
`bdd_equal`	?

Contents

Equality Checking

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

Equality Checking

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

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

Equality Checking

Theorem (Bryant '86)
Let π *be a variable order and* $f : \mathbb{B}^n \to \mathbb{B}$ *then there exists a unique (up to isomorphism) Reduced Ordered Binary Decision Diagram representing* f *with ordering* π*.*

Equality Checking

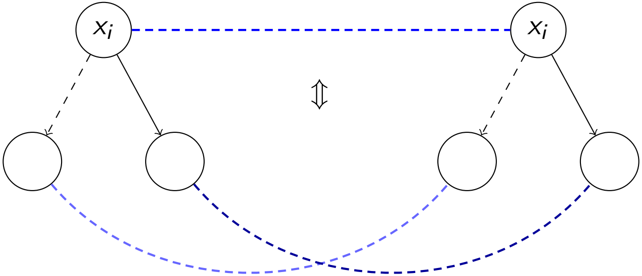
Theorem (Bryant '86)
Let π be a variable order and $f : \mathbb{B}^n \rightarrow \mathbb{B}$ then there exists a unique (up to isomorphism) Reduced Ordered Binary Decision Diagram representing f with ordering π.

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 ith level	$O(L/B)$ I/Os

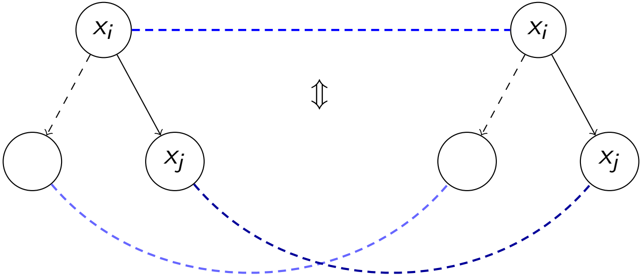
Equality Checking

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.



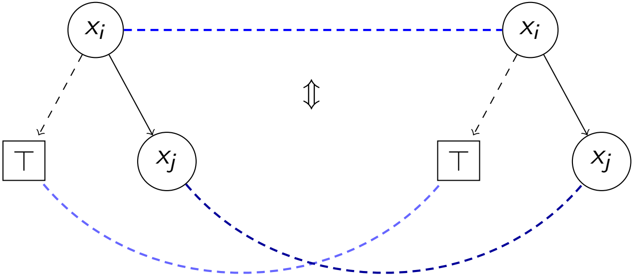
Equality Checking

Theorem (Bryant '86)
Let π be a variable order and $f : \mathbb{B}^n \to \mathbb{B}$ then there exists a unique (up to isomorphism) Reduced Ordered Binary Decision Diagram representing f with ordering π.



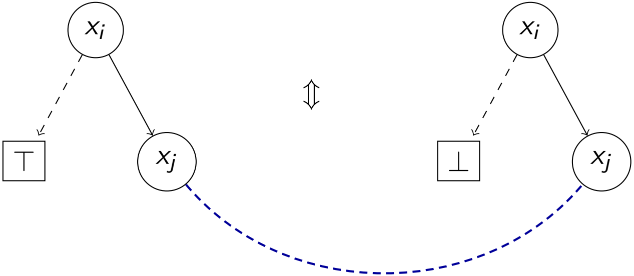
Equality Checking

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.



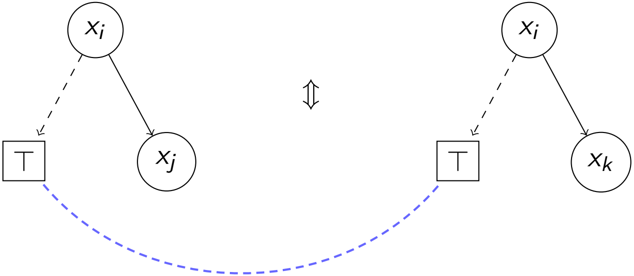
Equality Checking

Theorem (Bryant '86)
Let π be a variable order and $f : \mathbb{B}^n \to \mathbb{B}$ then there exists a unique (up to isomorphism) Reduced Ordered Binary Decision Diagram representing f with ordering π.



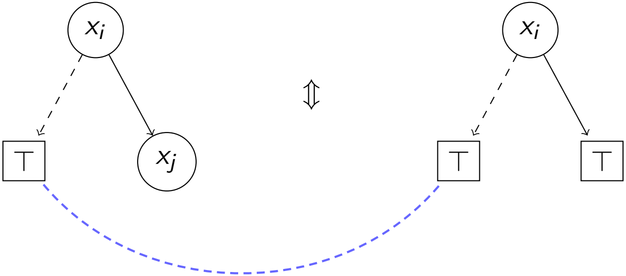
Equality Checking

Theorem (Bryant '86)
Let π be a variable order and $f : \mathbb{B}^n \to \mathbb{B}$ then there exists a unique (up to isomorphism) Reduced Ordered Binary Decision Diagram representing f with ordering π.



Equality Checking

Theorem (Bryant '86)
Let π be a variable order and $f : \mathbb{B}^n \to \mathbb{B}$ then there exists a unique (up to isomorphism) Reduced Ordered Binary Decision Diagram representing f with ordering π.



Equality Checking

Theorem (Bryant '86)
Let π be a variable order and $f : \mathbb{B}^n \to \mathbb{B}$ then there exists a unique (up to isomorphism) Reduced Ordered Binary Decision Diagram representing f with ordering π.

`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`.

Equality Checking

Theorem (Bryant '86)
Let π be a variable order and $f : \mathbb{B}^n \to \mathbb{B}$ then there exists a unique (up to isomorphism) Reduced Ordered Binary Decision Diagram representing f with ordering π.

`IsIsomorphic(`f`, `g`)`

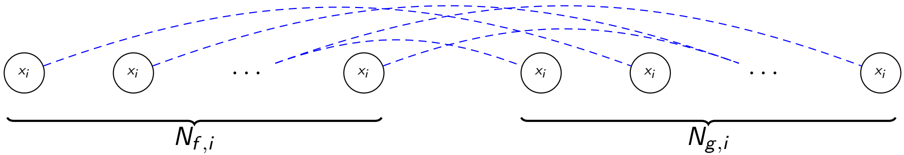
- Check whether root v_f of f and root v_g of g have a local violation.
- Check $low(v_f) \sim low(v_g)$ and $high(v_f) \sim high(v_g)$ "recursively".

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

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

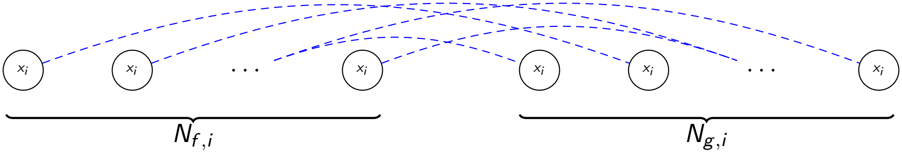
Equality Checking

Theorem (Bryant '86)
Let π be a variable order and $f : \mathbb{B}^n \to \mathbb{B}$ then there exists a unique (up to isomorphism) Reduced Ordered Binary Decision Diagram representing f with ordering π.



Equality Checking

Theorem (Bryant '86)
Let π be a variable order and $f : \mathbb{B}^n \to \mathbb{B}$ then there exists a unique (up to isomorphism) Reduced Ordered Binary Decision Diagram representing f with ordering π.



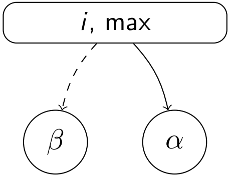
Return `false` if more than $N_{f,i} = N_{g,i}$ pairs of nodes are checked on level i.

$$\underbrace{O(\text{sort}(\Sigma_i\ N_{f,i}))}_{\texttt{Apply}''} = O(\text{sort}(N))$$

Equality Checking

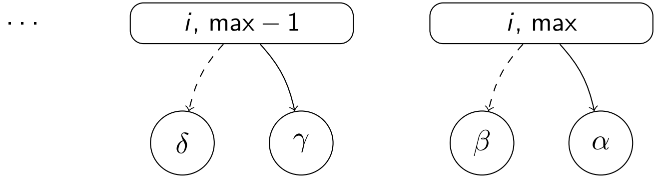
Observation
Each level output by the `Reduce` algorithm has the following properties:

Equality Checking



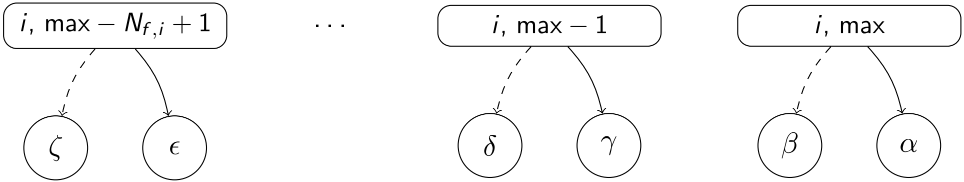
Observation
Each level output by the `Reduce` algorithm has the following properties:

Equality Checking



Observation
Each level output by the `Reduce` algorithm has the following properties:

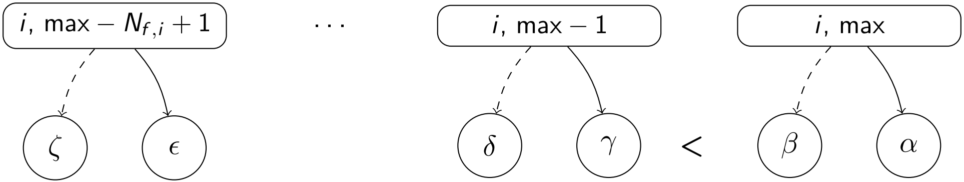
Equality Checking



Observation
Each level output by the `Reduce` algorithm has the following properties:

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

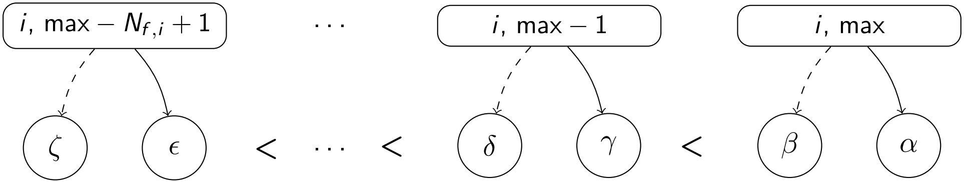
Equality Checking



Observation
Each level output by the `Reduce` algorithm has the following properties:

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

Equality Checking



Observation
Each level output by the `Reduce` algorithm has the following properties:

- Nodes on level i have their identifiers *consecutively* numbered.
- Nodes on level i are output sorted by their children.

Equality Checking

Theorem
If G_f and G_g are outputs of `Reduce`*.*

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

Proof.
$\Leftarrow$: Must describe the exact same graph.

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

Equality Checking

Theorem
If G_f and G_g are outputs of `Reduce`*.*

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

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.*

Contents

What are Binary Decision Diagrams?		
Why do they break?		
How can we fix it?	*Depth-First*	*Time-Forwarded*
CountPaths	$O(N_f)$	$O(\text{sort}(N_f))$
Apply	$O(N_f \cdot N_g)$	$O(\text{sort}(N_f))$
Equality Checking	$O(1)$	$2N/B$