I/O-efficient Manipulation of Binary Decision Diagrams

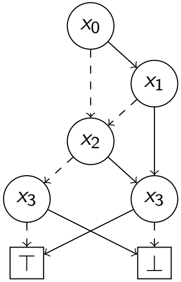
Steffan Christ Sølvsten

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

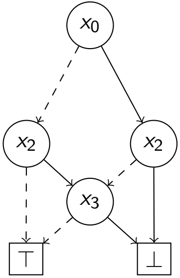
AARHUS
UNIVERSITY

Contents

Contents



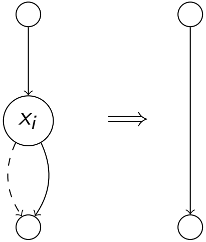
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$



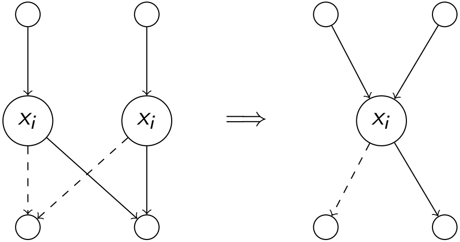
(b) $\neg(x_0 \ ? \ x_2 \wedge x_3 : x_2 \wedge x_3)$

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



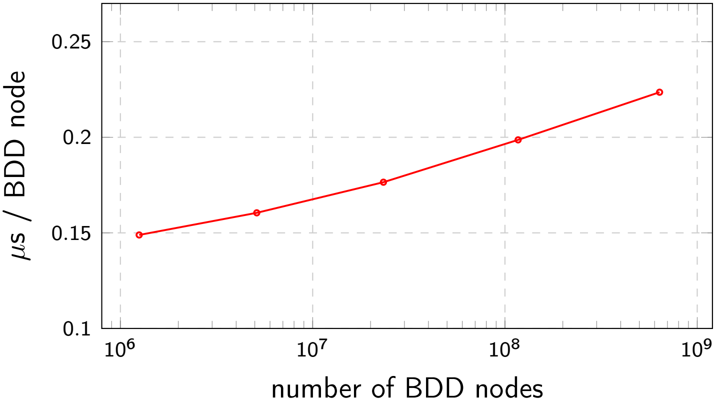
(2) Merge duplicate nodes

```
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)
```

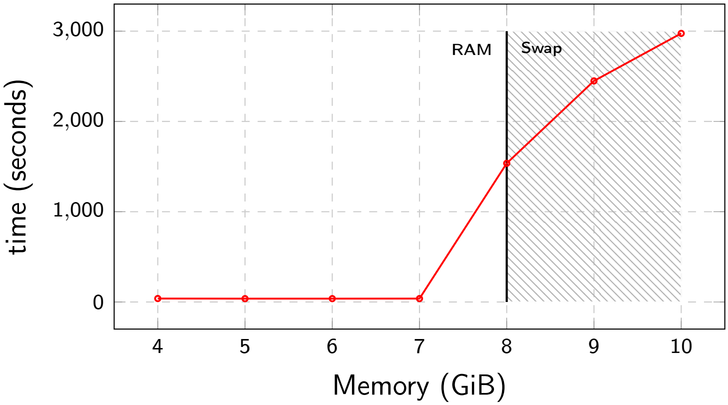
```
bdd_apply(f, g, ⊗):
  if f,g ∈ {⊥,⊤}
  then f ⊗ g
  else let i = top(f.(var), g.var)
           t = bdd_apply(f[x_i := ⊤], g[x_i := ⊤], ⊗)
           e = bdd_apply(f[x_i := ⊥], g[x_i := ⊥], ⊗)
       in make_node(i,t,e)
```

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*).

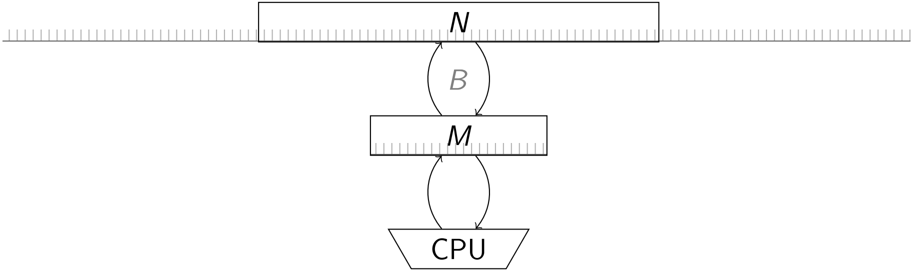


Running time of *BuDDy* for the *N*-Queens problem.



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

Contents



The I/O model by Aggarwal and Vitter '87

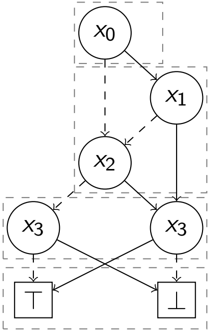
For any realistic values of N, M, and B we have that

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

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

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

CountPaths : *Example*

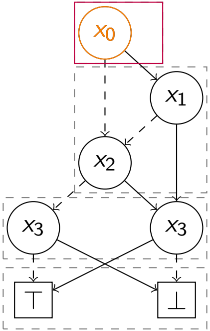


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4,\ B = 2$$

node I/Os	cache lookups
0	0

CountPaths : *Example*

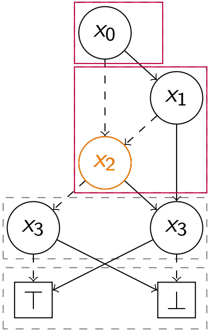


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4,\ B = 2$$

node I/Os	cache lookups
1	1

CountPaths : *Example*

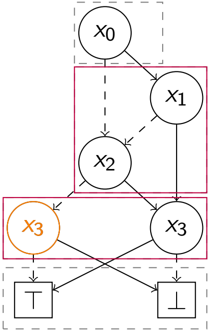


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4,\ B = 2$$

node I/Os	cache lookups
2	2

CountPaths : *Example*

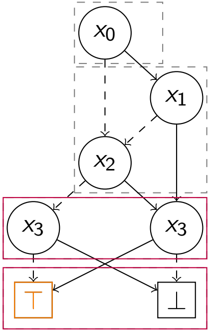


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4, \; B = 2$$

node I/Os	cache lookups
3	3

CountPaths : *Example*

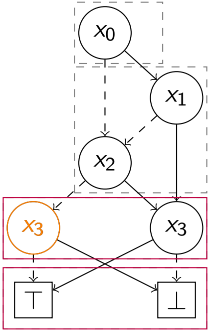


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4, \; B = 2$$

node I/Os	cache lookups
4	3

CountPaths : *Example*

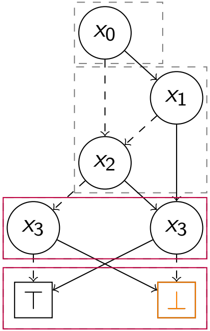


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4,\ B = 2$$

node I/Os	cache lookups
4	3

CountPaths : *Example*

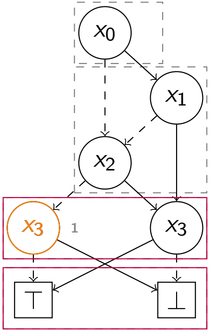


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4, \; B = 2$$

node I/Os	cache lookups
4	3

CountPaths : *Example*

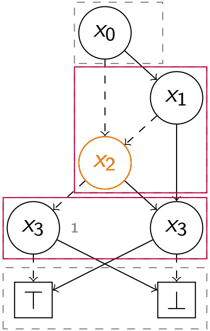


(a) $(x_0 \land x_1 \land x_3) \lor (x_2 \oplus x_3)$

$$M = 4, \; B = 2$$

node I/Os	cache lookups
4	3

CountPaths : *Example*

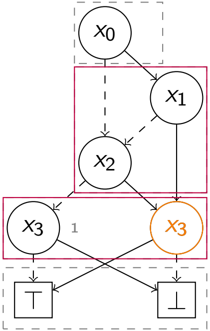


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4,\ B = 2$$

node I/Os	cache lookups
5	3

CountPaths : *Example*

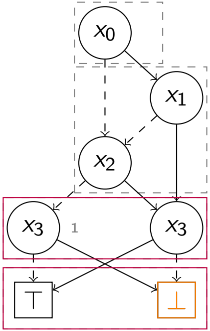


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4, \; B = 2$$

node I/Os	cache lookups
5	4

CountPaths : *Example*

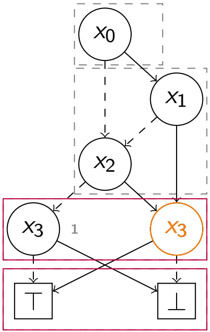


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4,\ B = 2$$

node I/Os	cache lookups
6	4

CountPaths : *Example*

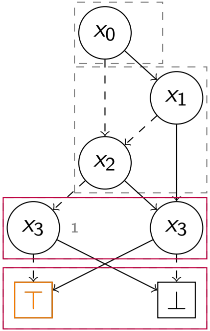


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4,\ B = 2$$

node I/Os	cache lookups
6	4

CountPaths : *Example*

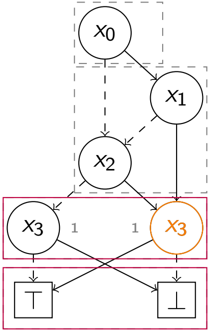


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4,\ B = 2$$

node I/Os	cache lookups
6	4

CountPaths : *Example*

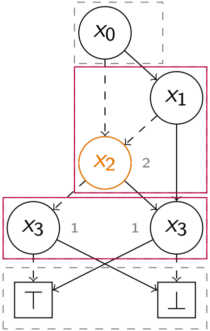


(a) $(x_0 \land x_1 \land x_3) \lor (x_2 \oplus x_3)$

$$M = 4, \; B = 2$$

node I/Os	cache lookups
6	4

CountPaths : *Example*

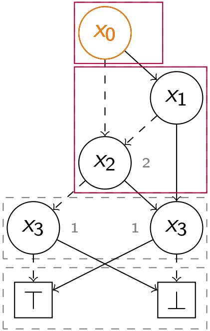


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4, \; B = 2$$

node I/Os	cache lookups
7	4

CountPaths : *Example*

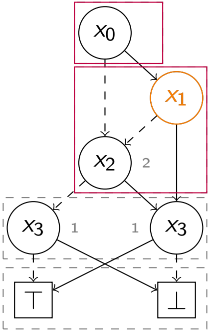


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4,\ B = 2$$

node I/Os	cache lookups
8	4

CountPaths : *Example*

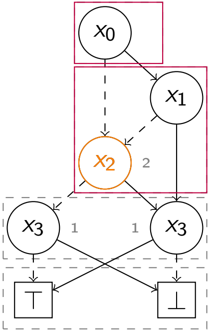


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4, \; B = 2$$

node I/Os	cache lookups
8	5

CountPaths : *Example*

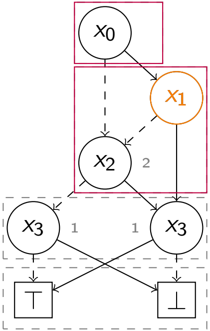


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4, \; B = 2$$

node I/Os	cache lookups
8	6

CountPaths : *Example*

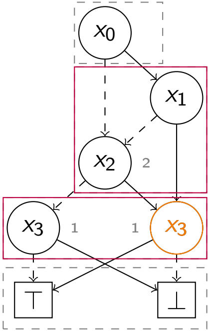


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4, \; B = 2$$

node I/Os	cache lookups
8	6

CountPaths : *Example*

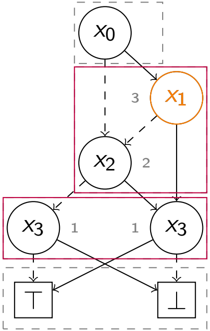


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4, \; B = 2$$

node I/Os	cache lookups
9	7

CountPaths : *Example*

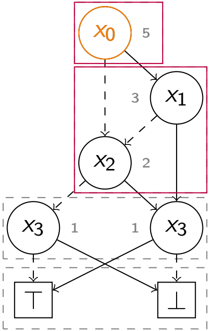


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4, \; B = 2$$

node I/Os	cache lookups
9	7

CountPaths : *Example*

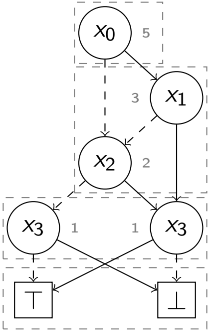


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

$$M = 4, \; B = 2$$

node I/Os	cache lookups
10	7

CountPaths : *Example*



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

$$M = 4,\ B = 2$$

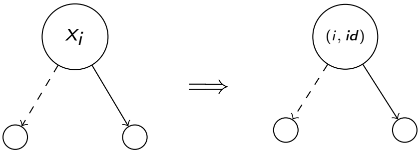
node I/Os	cache lookups
10	7

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`	$O(1)$

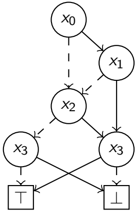
Algorithm	I/O-Complexity
`bdd_pathcount`	$O(N_f)$
`bdd_not`	$O(N_f)$
`bdd_restrict`	$O(N_f)$
`bdd_apply`	$O(N_f \cdot N_g)$
`bdd_equal`	$O(1)$

Contents

Contents

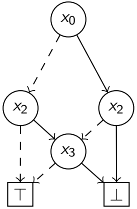


$$(i_1, id_1) < (i_2, id_2) \equiv i_1 < i_2 \vee (i_1 = i_2 \wedge id_i < id_j)$$



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

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

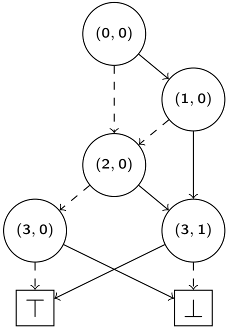


$$
\begin{array}{lcr}
[& ((0,0),(2,0),(2,1)) & , \\
 & ((2,0),\bot,(3,0)) & , \\
 & ((2,1),(3,0),\top) & , \\
 & ((3,0),\top,\bot) &]
\end{array}
$$

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

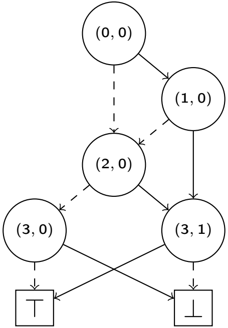
Node-based representation of prior shown BDDs

CountPaths : *Example*



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

CountPaths : *Example*



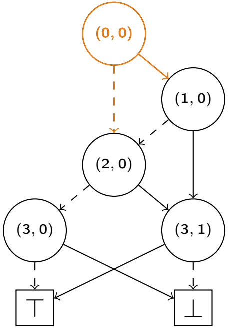
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Priority Queue: $Q_{\boldsymbol{count}}$:

[

]

CountPaths : *Example*



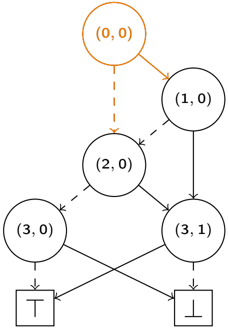
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Priority Queue: $Q_{\boldsymbol{count}}$:

[

]

CountPaths : *Example*

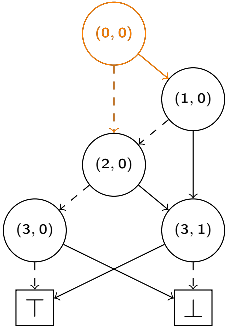


(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Priority Queue: $Q_{\boldsymbol{count}}$:

$$\begin{aligned}[\quad & ((0,0) \xrightarrow{\top} (1,0), \quad 1) \quad , \\ & ((0,0) \xrightarrow{\bot} (2,0), \quad 1) \quad , \\ & \quad]\end{aligned}$$

CountPaths : *Example*



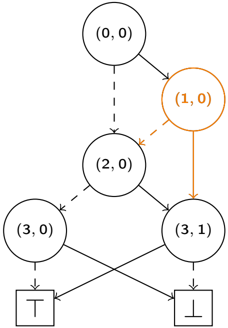
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(1, 0)$	0	0

Priority Queue: $Q_{\boldsymbol{count}}$:

$$[\quad ((0,0) \xrightarrow{\top} (1,0),\quad 1),$$
$$((0,0) \xrightarrow{\bot} (2,0),\quad 1),\quad]$$

CountPaths : *Example*



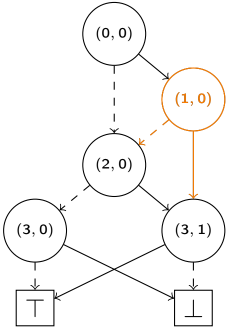
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(1, 0)$	0	0

Priority Queue: $Q_{\boldsymbol{count}}$:

$$[\quad ((0,0) \xrightarrow{\top} (1,0),\ 1), \quad ((0,0) \xrightarrow{\bot} (2,0),\ 1), \quad]$$

CountPaths : *Example*



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

Seek	Sum	Result
$(1, 0)$	1	0

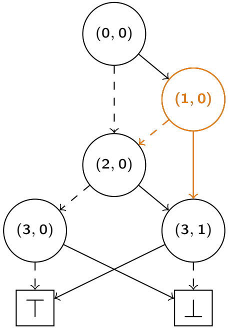
Priority Queue: Q_{count}:

[

$((0,0) \xrightarrow{\perp} (2,0), \quad 1)$,

]

CountPaths : *Example*



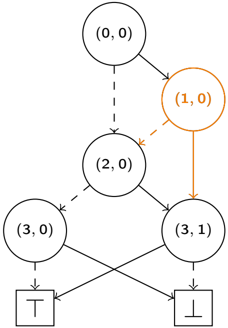
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
(1, 0)	1	0

Priority Queue: Q_{count}:

[
$((0,0) \xrightarrow{\perp} (2,0),\quad 1)$,
$((1,0) \xrightarrow{\perp} (2,0),\quad 1)$,
$((1,0) \xrightarrow{\top} (3,1),\quad 1)$,
]

CountPaths : *Example*



(a) $(x_0 \land x_1 \land x_3) \lor (x_2 \oplus x_3)$

Seek	Sum	Result
$(2, 0)$	0	0

Priority Queue: $Q_{\boldsymbol{count}}$:

[

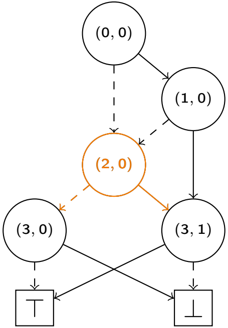
$((0,0) \xrightarrow{\perp} (2,0), \quad 1)$,

$((1,0) \xrightarrow{\perp} (2,0), \quad 1)$,

$((1,0) \xrightarrow{\top} (3,1), \quad 1)$,

]

CountPaths : *Example*



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

Seek	Sum	Result
$(2, 0)$	0	0

Priority Queue: $Q_{\boldsymbol{count}}$:

[

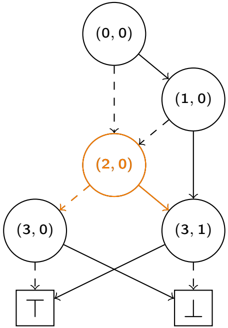
$((0,0) \xrightarrow{\perp} (2,0),\ \ 1)$,

$((1,0) \xrightarrow{\perp} (2,0),\ \ 1)$,

$((1,0) \xrightarrow{\top} (3,1),\ \ 1)$,

]

CountPaths : *Example*



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

Seek	Sum	Result
(2, 0)	1	0

Priority Queue: Q_{count}:

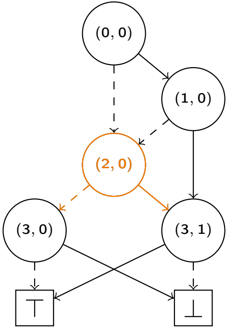
[

$((1,0) \xrightarrow{\perp} (2,0),\quad 1)$,

$((1,0) \xrightarrow{\top} (3,1),\quad 1)$,

]

CountPaths : *Example*



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

Seek	Sum	Result
(2, 0)	2	0

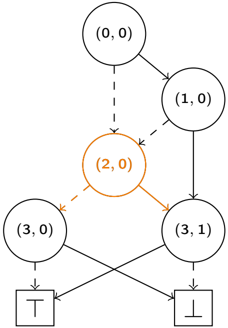
Priority Queue: Q_{count}:

[

$((1,0) \xrightarrow{\top} (3,1), \quad 1)$,

]

CountPaths : *Example*



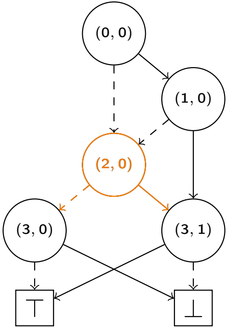
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(2, 0)$	2	0

Priority Queue: Q_{count}:

$$\left[\begin{array}{lll} ((2,0) \xrightarrow{\perp} (3,0), & 2) & , \\ ((1,0) \xrightarrow{\top} (3,1), & 1) & , \\ ((2,0) \xrightarrow{\top} (3,1), & 2) & \end{array}\right]$$

CountPaths : *Example*



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

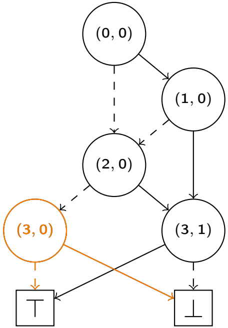
Seek	Sum	Result
(3, 0)	0	0

Priority Queue: Q_{count}:

[

$((2,0) \xrightarrow{\perp} (3,0),\quad 2)$,
$((1,0) \xrightarrow{\top} (3,1),\quad 1)$,
$((2,0) \xrightarrow{\top} (3,1),\quad 2)$]

CountPaths : *Example*



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

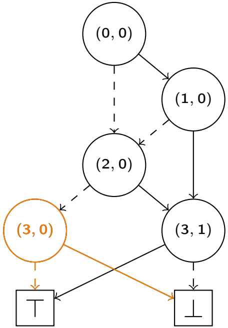
Seek	Sum	Result
$(3, 0)$	0	0

Priority Queue: $Q_{\boldsymbol{count}}$:

[

$((2,0) \xrightarrow{\bot} (3,0),\quad 2)$,
$((1,0) \xrightarrow{\top} (3,1),\quad 1)$,
$((2,0) \xrightarrow{\top} (3,1),\quad 2)$]

CountPaths : *Example*



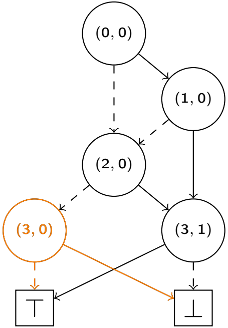
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
(3, 0)	2	0

Priority Queue: Q_{count}:

[
$((1,0) \xrightarrow{\top} (3,1), \quad 1)$,
$((2,0) \xrightarrow{\top} (3,1), \quad 2)$]

CountPaths : *Example*



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

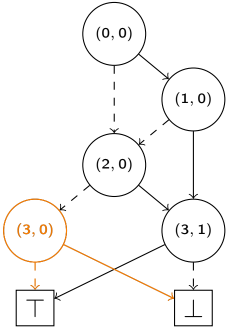
Seek	Sum	Result
(3, 0)	2	2

Priority Queue: $Q_{\boldsymbol{count}}$:

[

$((1,0) \xrightarrow{\top} (3,1),\quad 1)$,
$((2,0) \xrightarrow{\top} (3,1),\quad 2)$]

CountPaths : *Example*



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

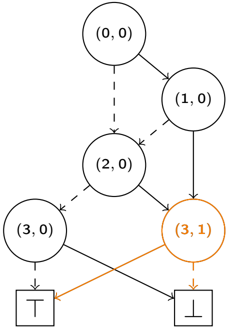
Seek	Sum	Result
$(3, 1)$	0	2

Priority Queue: Q_{count}:

[

$((1,0) \xrightarrow{\top} (3,1),\quad 1)$,
$((2,0) \xrightarrow{\top} (3,1),\quad 2)$]

CountPaths : *Example*



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

Seek	Sum	Result
$(3, 1)$	0	2

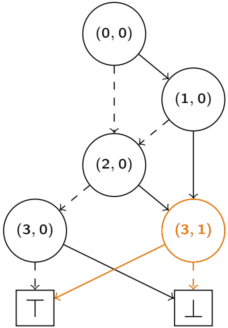
Priority Queue: Q_{count}:

[

$((1, 0) \xrightarrow{\top} (3, 1), \quad 1)$,

$((2, 0) \xrightarrow{\top} (3, 1), \quad 2)$]

CountPaths : *Example*



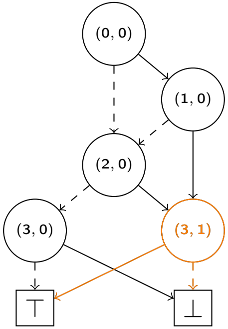
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$

Seek	Sum	Result
$(3, 1)$	1	2

Priority Queue: Q_{count}:

[$((2,0) \xrightarrow{\top} (3,1),\quad 2)$]

CountPaths : *Example*



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

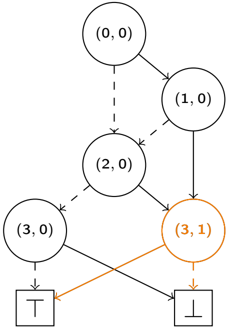
Seek	Sum	Result
$(3, 1)$	3	2

Priority Queue: Q_{count}:

[

]

CountPaths : *Example*



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

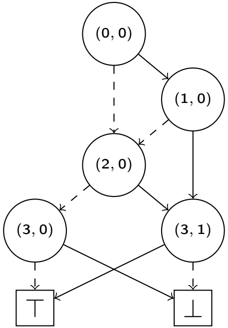
Seek	Sum	Result
$(3, 1)$	3	5

Priority Queue: Q_{count}:

[

]

CountPaths : *Example*



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

Result

5

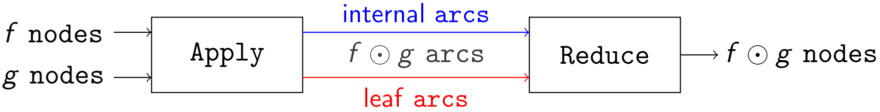
Priority Queue: $Q_{\boldsymbol{count}}$:

[

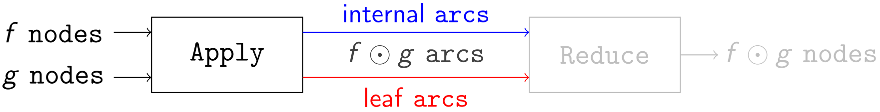
]

Contents

Apply

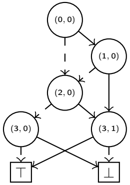

f nodes
g nodes
Apply
internal arcs
$f \odot g$ arcs
leaf arcs
Reduce
$f \odot g$ nodes

Apply

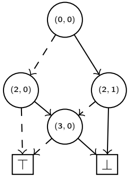


f nodes
g nodes
Apply
internal arcs
$f \odot g$ arcs
leaf arcs
Reduce
$f \odot g$ nodes

Apply : *Example*



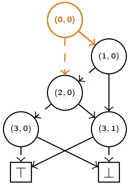
(a) $(x_{\mathbf{0}} \land x_{\mathbf{1}} \land x_{\mathbf{3}}) \lor (x_{\mathbf{2}} \oplus x_{\mathbf{3}})$



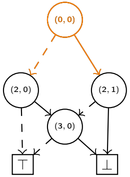
(b) $\neg(x_{\mathbf{0}}\ ?\ x_{\mathbf{2}} \lor x_{\mathbf{3}}\ :\ x_{\mathbf{2}} \land x_{\mathbf{3}})$

(c) $(a) \land (b)$

Apply : *Example*



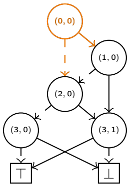
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$



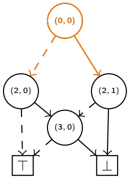
(b) $\neg(x_0 \ ? \ x_2 \vee x_3 \ : \ x_2 \wedge x_3)$

(c) $(a) \wedge (b)$

Apply : *Example*



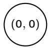
(a) $(x_0 \wedge x_1 \wedge x_3) \vee (x_2 \oplus x_3)$



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

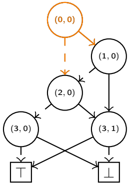
Priority Queue: $Q_{\boldsymbol{app}:\mathbf{1}}$:

$$[\quad (0,0) \xrightarrow{\top} ((1,0),(2,1)) \quad ,$$
$$(0,0) \xrightarrow{\bot} ((2,0),(2,0)) \quad ,$$
$$]$$

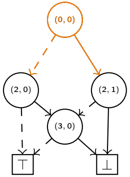


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:

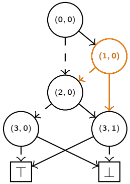
$\min((1, 0), (2, 1))$

Priority Queue: $Q_{app:1}$:

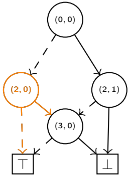
$$
\begin{array}{ll}
[\quad (0,0) \xrightarrow{\top} ((1,0),(2,1)) & , \\
\quad\;\; (0,0) \xrightarrow{\bot} ((2,0),(2,0)) & , \\
&]
\end{array}
$$

(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:

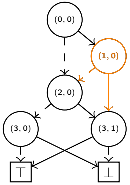
min((1, 0), (2, 1))

Priority Queue: $Q_{app:1}$:

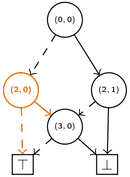
$$[\quad (0,0) \xrightarrow{\top} ((1,0),(2,1)) \quad , \quad (0,0) \xrightarrow{\bot} ((2,0),(2,0)) \quad , \quad]$$

(c) $(a) \wedge (b)$

Apply : *Example*



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

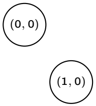


(b) $\neg(x_0 \ ? \ x_2 \vee x_3 \ : \ x_2 \wedge x_3)$

Seek:
$\min((1,0),(2,1))$

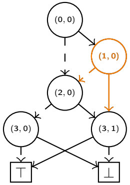
Priority Queue: $Q_{app:1}$:

$$[\ (0,0) \xrightarrow{\top} ((1,0),(2,1)),\ (0,0) \xrightarrow{\bot} ((2,0),(2,0)),\ (1,0) \xrightarrow{\bot} ((2,0),(2,1)),\ (1,0) \xrightarrow{\top} ((3,1),(2,1)),\ ]$$

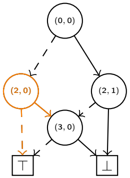


(c) $(a) \wedge (b)$

Apply : *Example*



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



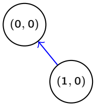
(b) $\neg(x_0\ ?\ x_2 \vee x_3\ :\ x_2 \wedge x_3)$

Seek:
$\min((1,0),(2,1))$

Priority Queue: $Q_{app:1}$:

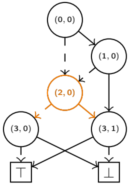
[
$(0,0) \xrightarrow{\perp} ((2,0),(2,0))$,
$(1,0) \xrightarrow{\perp} ((2,0),(2,1))$,
$(1,0) \xrightarrow{\top} ((3,1),(2,1))$,
]

Output:
$(0,0) \xrightarrow{\top} (1,0)$

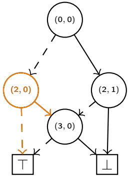


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
$\min((2,0),(2,0))$

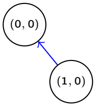
Priority Queue: $Q_{\boldsymbol{app}:\mathbf{1}}$:

[

$(0,0) \xrightarrow{\perp} ((2,0),(2,0))$,
$(1,0) \xrightarrow{\perp} ((2,0),(2,1))$,
$(1,0) \xrightarrow{\top} ((3,1),(2,1))$,

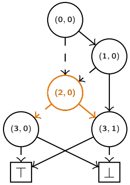
]

Output:

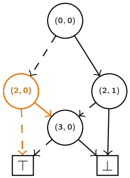


(c) $(a) \wedge (b)$

Apply : *Example*



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



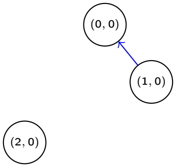
(b) $\neg(x_0 \ ? \ x_2 \vee x_3 \ : \ x_2 \wedge x_3)$

Seek:
$\min((2,0),(2,0))$

Priority Queue: $Q_{\boldsymbol{app}:\mathbf{1}}$:

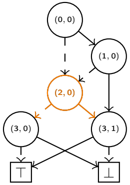
$$
\begin{aligned}
[\quad & \\
& (0,0) \xrightarrow{\perp} ((2,0),(2,0)) \quad , \\
& (1,0) \xrightarrow{\perp} ((2,0),(2,1)) \quad , \\
& (1,0) \xrightarrow{\top} ((3,1),(2,1)) \quad , \\
& (2,0) \xrightarrow{\top} ((3,1),(3,0)) \quad , \\
& (2,0) \xrightarrow{\perp} ((3,0),\top) \quad]
\end{aligned}
$$

Output:

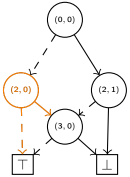


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:

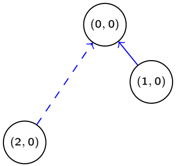
$\min((2,0),(2,0))$

Priority Queue: $Q_{app:1}$:

$$[\ (1,0) \xrightarrow{\perp} ((2,0),(2,1)),\quad (1,0) \xrightarrow{\top} ((3,1),(2,1)),\quad (2,0) \xrightarrow{\top} ((3,1),(3,0)),\quad (2,0) \xrightarrow{\perp} ((3,0),\top)\ ]$$

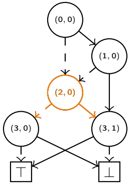
Output:

$(0,0) \xrightarrow{\perp} (2,0)$

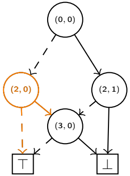


(c) $(a) \wedge (b)$

Apply : *Example*



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



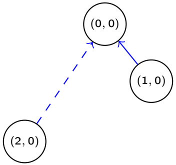
(b) $\neg(x_0 \; ? \; x_2 \vee x_3 \; : \; x_2 \wedge x_3)$

Seek:
$\min((2,0),(2,1))$

Priority Queue: $Q_{\boldsymbol{app}:\mathbf{1}}$:

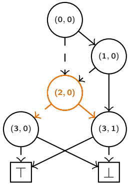
$$
\begin{aligned}
[\\
&(1,0) \xrightarrow{\perp} ((2,0),(2,1)) &,\\
&(1,0) \xrightarrow{\top} ((3,1),(2,1)) &,\\
&(2,0) \xrightarrow{\top} ((3,1),(3,0)) &,\\
&(2,0) \xrightarrow{\perp} ((3,0),\top) &]
\end{aligned}
$$

Output:

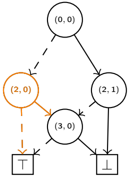


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
$\min((2,0),(2,1))$

Priority Queue: $Q_{app:1}$:

[

$(1,0) \xrightarrow{\top} ((3,1),(2,1))$,

$(2,0) \xrightarrow{\top} ((3,1),(3,0))$,

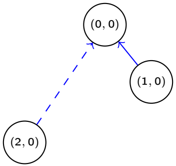
$(2,0) \xrightarrow{\perp} ((3,0),\top)$]

Priority Queue: $Q_{app:2}$:

[$(1,0) \xrightarrow{\perp} ((2,0),(2,1))$ $((3,0),(3,1))$,

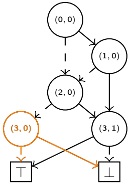
]

Output:

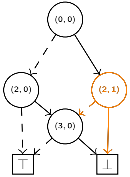


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
$\max((2,0),(2,1))$

Priority Queue: $Q_{\boldsymbol{app}:\mathbf{1}}$:

[

$(1,0) \xrightarrow{\top} ((3,1),(2,1))$,

$(2,0) \xrightarrow{\top} ((3,1),(3,0))$,

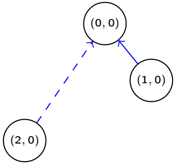
$(2,0) \xrightarrow{\bot} ((3,0),\top)$]

Priority Queue: $Q_{\boldsymbol{app}:\mathbf{2}}$:

[$(1,0) \xrightarrow{\bot} ((2,0),(2,1))$ $((3,0),(3,1))$,

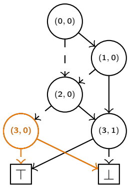
]

Output:

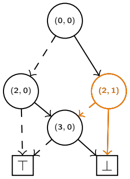


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
$\max((2,0),(2,1))$

Priority Queue: $Q_{app:1}$:

[

$(1,0) \xrightarrow{\top} ((3,1),(2,1))$,
$(2,1) \xrightarrow{\bot} ((3,0),(3,0))$,
$(2,0) \xrightarrow{\top} ((3,1),(3,0))$,

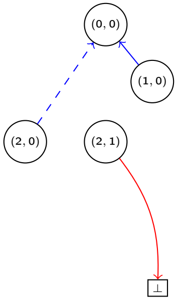
$(2,0) \xrightarrow{\bot} ((3,0),\top)$]

Priority Queue: $Q_{app:2}$:

[$(1,0) \xrightarrow{\bot} ((2,0),(2,1))$ $((3,0),(3,1))$,

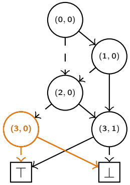
]

Output:
$(2,1) \xrightarrow{\top} \bot$

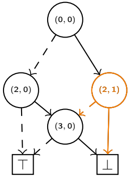


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
$\max((2,0),(2,1))$

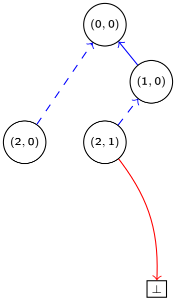
Priority Queue: $Q_{app:1}$:

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

Priority Queue: $Q_{app:2}$:

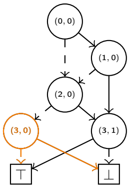
[ ]

Output:
$(1,0) \xrightarrow{\bot} (2,1)$

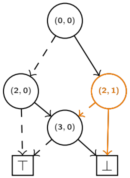


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
$\min((3,1),(2,1))$

Priority Queue: $Q_{app:1}$:

[

$(1,0) \xrightarrow{\top} ((3,1),(2,1))$,
$(2,1) \xrightarrow{\perp} ((3,0),(3,0))$,
$(2,0) \xrightarrow{\top} ((3,1),(3,0))$,

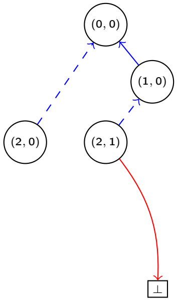
$(2,0) \xrightarrow{\perp} ((3,0),\top)$]

Priority Queue: $Q_{app:2}$:

[

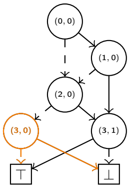
]

Output:

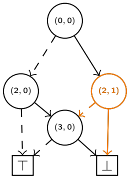


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:

$\min((3,1),(2,1))$

Priority Queue: $Q_{app:1}$:

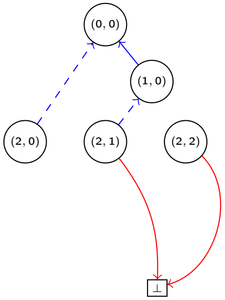
$$\begin{array}{ll} [& \\ (1,0) \xrightarrow{\top} ((3,1),(2,1)) & , \\ (2,1) \xrightarrow{\bot} ((3,0),(3,0)) & , \\ (2,0) \xrightarrow{\top} ((3,1),(3,0)) & , \\ (2,2) \xrightarrow{\bot} ((3,1),(3,0)) & , \\ (2,0) \xrightarrow{\bot} ((3,0),\top) &] \end{array}$$

Priority Queue: $Q_{app:2}$:

[ ]

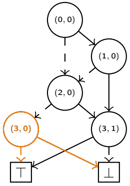
Output:

$(2,2) \xrightarrow{\top} \bot$

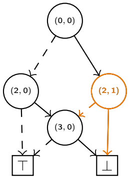


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
$\min((3,1),(2,1))$

Priority Queue: $Q_{app:1}$:

[

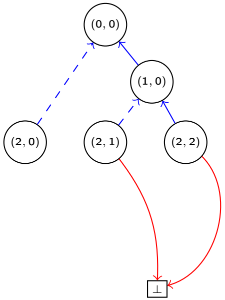
$(2,1) \xrightarrow{\perp} ((3,0),(3,0))$,
$(2,0) \xrightarrow{\top} ((3,1),(3,0))$,
$(2,2) \xrightarrow{\perp} ((3,1),(3,0))$,
$(2,0) \xrightarrow{\perp} ((3,0),\top)$]

Priority Queue: $Q_{app:2}$:

[

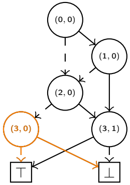
]

Output:
$(1,0) \xrightarrow{\top} (2,2)$

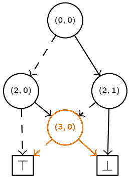


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
$\min((3,0),(3,0))$

Priority Queue: $Q_{\textit{app}:1}$:

[

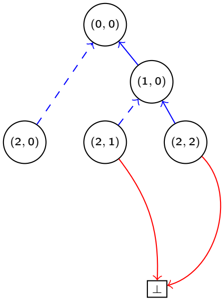
$(2,1) \xrightarrow{\bot} ((3,0),(3,0))$,
$(2,0) \xrightarrow{\top} ((3,1),(3,0))$,
$(2,2) \xrightarrow{\bot} ((3,1),(3,0))$,
$(2,0) \xrightarrow{\bot} ((3,0),\top)$]

Priority Queue: $Q_{\textit{app}:2}$:

[

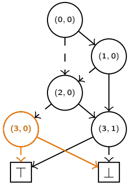
]

Output:

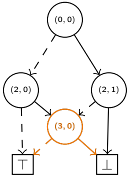


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
$\min((3,0),(3,0))$

Priority Queue: $Q_{app:1}$:

[

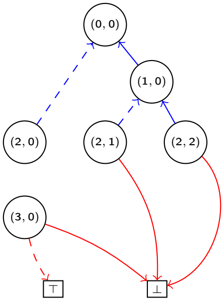
$(2,1) \xrightarrow{\perp} ((3,0),(3,0))$,
$(2,0) \xrightarrow{\top} ((3,1),(3,0))$,
$(2,2) \xrightarrow{\perp} ((3,1),(3,0))$,
$(2,0) \xrightarrow{\perp} ((3,0),\top)$]

Priority Queue: $Q_{app:2}$:

[

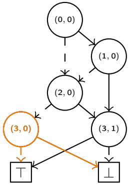
]

Output:
$(3,0) \xrightarrow{\perp} \top, (3,0) \xrightarrow{\top} \perp$

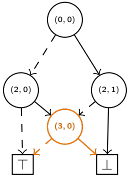


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
$\min((3,0),(3,0))$

Priority Queue: $Q_{app:1}$:

[

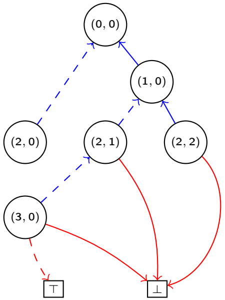
$(2,0) \xrightarrow{\top} ((3,1),(3,0))$,
$(2,2) \xrightarrow{\perp} ((3,1),(3,0))$,
$(2,0) \xrightarrow{\perp} ((3,0),\top)$]

Priority Queue: $Q_{app:2}$:

[

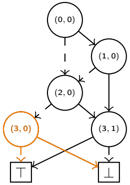
]

Output:
$(2,1) \xrightarrow{\perp} (3,0)$

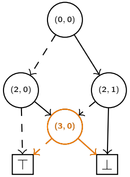


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
min((3, 1), (3, 0))

Priority Queue: $Q_{app:1}$:

[

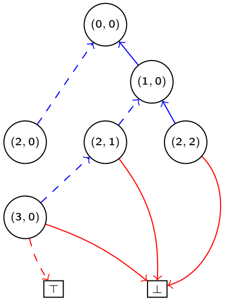
$(2,0) \xrightarrow{\top} ((3,1),(3,0))$,
$(2,2) \xrightarrow{\perp} ((3,1),(3,0))$,
$(2,0) \xrightarrow{\perp} ((3,0),\top)$]

Priority Queue: $Q_{app:2}$:

[

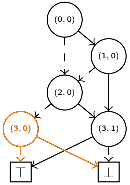
]

Output:

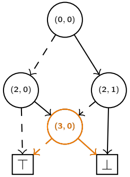


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
min((3, 1), (3, 0))

Priority Queue: $Q_{app:1}$:

[

$(2,0) \xrightarrow{\perp} ((3,0), \top)$]

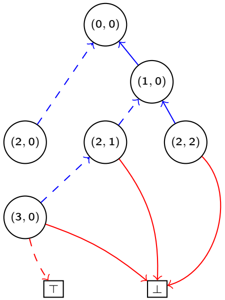
Priority Queue: $Q_{app:2}$:

[

$(2,0) \xrightarrow{\top} ((3,1),(3,0)) \quad (\top, \perp)$,

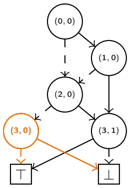
$(2,2) \xrightarrow{\perp} ((3,1),(3,0)) \quad (\top, \perp)$]

Output:

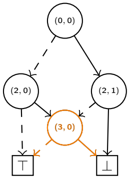


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:

min((3, 0), ⊤)

Priority Queue: $Q_{app:1}$:

[

$(2,0) \xrightarrow{\bot} ((3,0),\top)$]

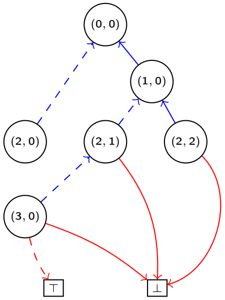
Priority Queue: $Q_{app:2}$:

[

$(2,0) \xrightarrow{\top} ((3,1),(3,0)) \quad (\top,\bot)$,

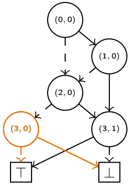
$(2,2) \xrightarrow{\bot} ((3,1),(3,0)) \quad (\top,\bot)$]

Output:

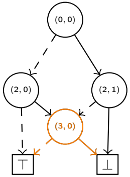


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:

$\min((3, 0), \top)$

Priority Queue: $Q_{app:1}$:

[

$(2, 0) \xrightarrow{\perp} ((3, 0), \top)$]

Priority Queue: $Q_{app:2}$:

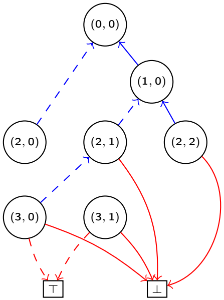
[

$(2, 0) \xrightarrow{\top} ((3, 1), (3, 0))$ $(\top, \perp)$,

$(2, 2) \xrightarrow{\perp} ((3, 1), (3, 0))$ $(\top, \perp)$]

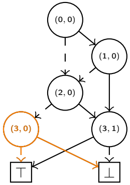
Output:

$(3, 1) \xrightarrow{\perp} \top, (3, 1) \xrightarrow{\top} \perp$

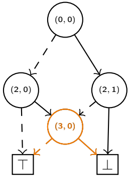


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
$\min((3,0), \top)$

Priority Queue: $Q_{app:1}$:

[

]

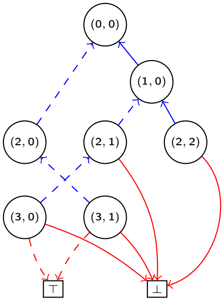
Priority Queue: $Q_{app:2}$:

[

$(2,0) \xrightarrow{\top} ((3,1),(3,0)) \quad (\top, \bot)$,

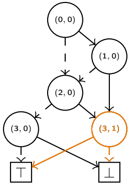
$(2,2) \xrightarrow{\bot} ((3,1),(3,0)) \quad (\top, \bot)$]

Output:
$(2,0) \xrightarrow{\bot} (3,1)$

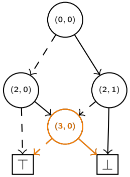


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
max((3, 1), (3, 0))

Priority Queue: $Q_{\boldsymbol{app}:\mathbf{1}}$:

[

]

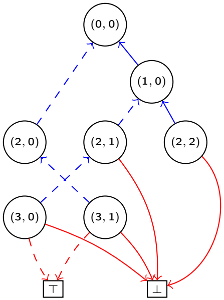
Priority Queue: $Q_{\boldsymbol{app}:\mathbf{2}}$:

[

$(2, 0) \xrightarrow{\top} ((3, 1), (3, 0)) \quad (\top, \bot)$,

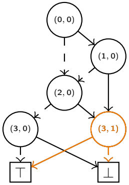
$(2, 2) \xrightarrow{\bot} ((3, 1), (3, 0)) \quad (\top, \bot)$]

Output:

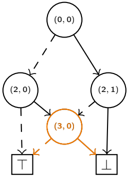


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:
$\max((3,1),(3,0))$

Priority Queue: $Q_{app:1}$:

[

]

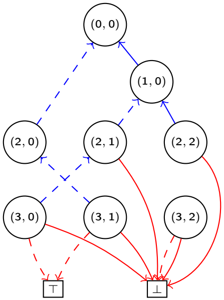
Priority Queue: $Q_{app:2}$:

[

$(2,0) \xrightarrow{\top} ((3,1),(3,0)) \quad (\top, \bot)$,

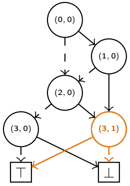
$(2,2) \xrightarrow{\bot} ((3,1),(3,0)) \quad (\top, \bot)$]

Output:
$(3,2) \xrightarrow{\bot} \bot, (3,2) \xrightarrow{\top} \bot$

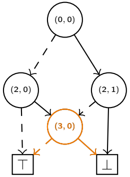


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Seek:

$\max((3,1),(3,0))$

Priority Queue: $Q_{app:1}$:

[

]

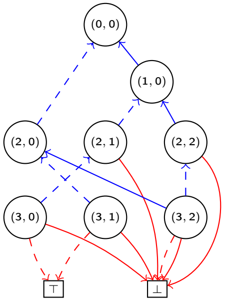
Priority Queue: $Q_{app:2}$:

[

]

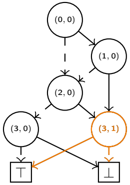
Output:

$(2,0) \xrightarrow{\top} (3,2), (2,2) \xrightarrow{\perp} (3,2)$

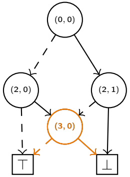


(c) $(a) \wedge (b)$

Apply : *Example*



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



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

Priority Queue: $Q_{app:1}$:

[

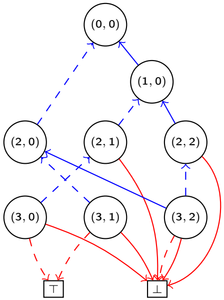
]

Priority Queue: $Q_{app:2}$:

[

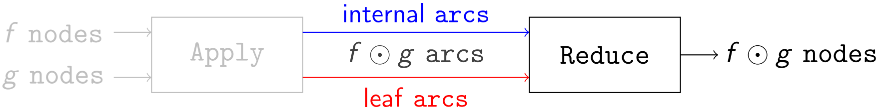
]

Output:



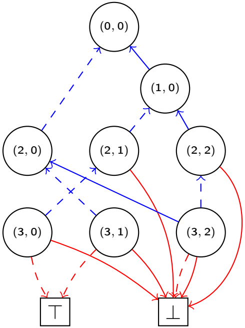
(c) $(a) \wedge (b)$

Apply



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

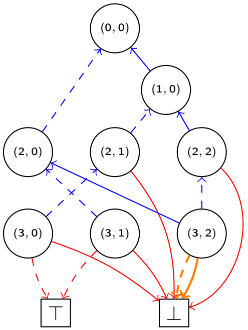
Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

(d) $(a) \wedge (b)$ **reduced**

Apply : *Example* *(Continued)*



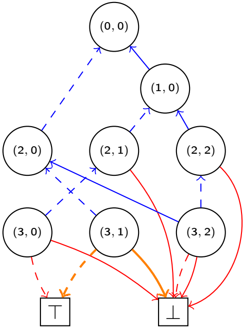
(c) $(a) \wedge (b)$

Level: 3

[$[(3,2) \mapsto \bot]$]

(d) $(a) \wedge (b)$ **reduced**

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

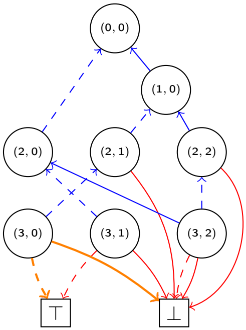
Level: 3

[$[(3,2) \mapsto \bot]$]

[$((3,1), \top, \bot)$,
]

(d) $(a) \wedge (b)$ **reduced**

Apply : *Example* *(Continued)*



(c) $(a) \land (b)$

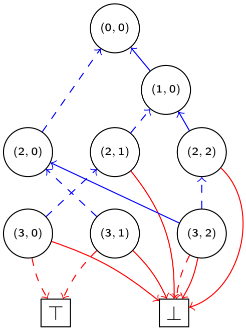
Level: 3

[$[(3,2) \mapsto \bot]$]

[$((3,1), \top, \bot)$,
$((3,0), \top, \bot)$]

(d) $(a) \land (b)$ **reduced**

Apply : *Example* *(Continued)*



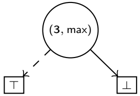
(c) $(a) \wedge (b)$

Level: 3

[$[(3, 2) \mapsto \bot]$]

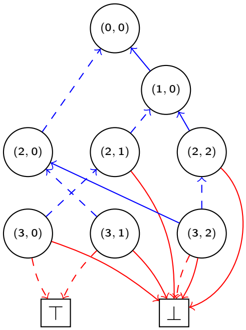
[$[(3, 1) \mapsto (3, \max)]$,
$((3, 0), \top, \bot)$]

Output:
$((3, \max), \top, \bot)$



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



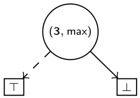
(c) $(a) \wedge (b)$

Level: 3

[$[(3,2) \mapsto \bot]$ **]**

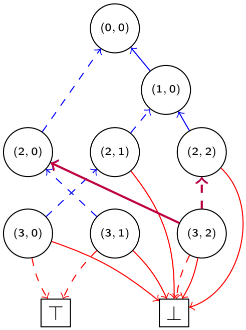
[$[(3,1) \mapsto (3, \text{max})]$,
$[(3,0) \mapsto (3, \text{max})]$ **]**

Output:



(d) $(a) \wedge (b)$ **reduced**

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\textit{red}}$:

[$(2,2) \xrightarrow{\perp} \perp$,

$(2,0) \xrightarrow{\top} \perp$,

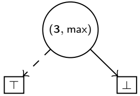
]

Level: 3

[ ]

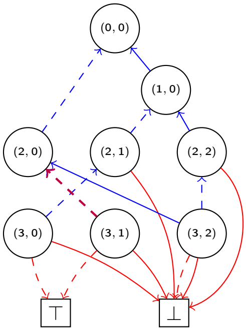
[$[(3,1) \mapsto (3, \text{max})]$,
$[(3,0) \mapsto (3, \text{max})]$]

Output:



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \land (b)$

Priority Queue: Q_{red}:

[$(2,2) \xrightarrow{\bot} \bot$,

$(2,0) \xrightarrow{\top} \bot$,

$(2,0) \xrightarrow{\bot} (3, \max)$,

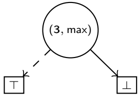
]

Level: 3

[ ]

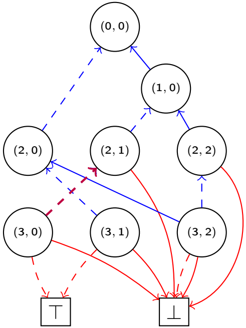
[,
$[(3,0) \mapsto (3, \max)]$]

Output:



(d) $(a) \land (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\boldsymbol{red}}$:

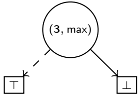
[
$(2,2) \xrightarrow{\perp} \perp$,
$(2,1) \xrightarrow{\perp} (3, \mathrm{max})$,
$(2,0) \xrightarrow{\top} \perp$,
$(2,0) \xrightarrow{\perp} (3, \mathrm{max})$,
]

Level: 3

[ ]

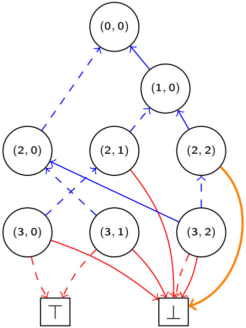
[,
]

Output:



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

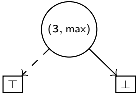
Priority Queue: $Q_{\boldsymbol{red}}$:

$$\left[\begin{array}{l} (2,1) \xrightarrow{\perp} (3, \mathrm{max}) \quad , \\ (2,0) \xrightarrow{\top} \perp \quad , \\ (2,0) \xrightarrow{\perp} (3, \mathrm{max}) \quad , \end{array}\right]$$

Level: 2

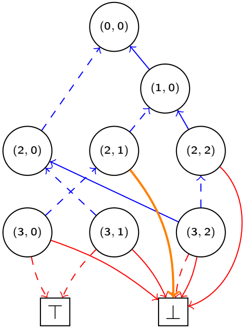
$$[\quad [(2,2) \mapsto \perp] \quad]$$

Output:



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\boldsymbol{red}}$:

[

$(2,0) \xrightarrow{\top} \bot$,

$(2,0) \xrightarrow{\bot} (3, \text{max})$,

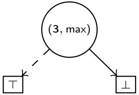
]

Level: 2

[$[(2,2) \mapsto \bot]$]

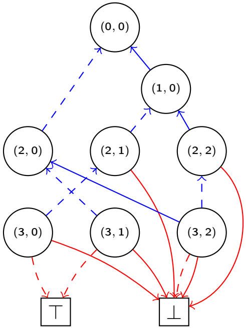
[$((2,1), (3, \text{max}), \bot)$,
]

Output:



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\boldsymbol{red}}$:

[

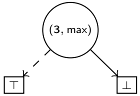
]

Level: 2

[$[(2, 2) \mapsto \bot]$]

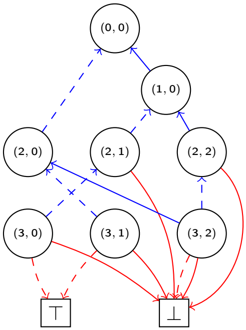
[$((2, 1), (3, \text{max}), \bot)$,
$((2, 0), (3, \text{max}), \bot)$]

Output:



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\textit{red}}$:

[

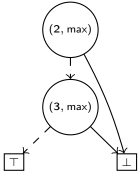
]

Level: 2

[$[(2,2) \mapsto \bot]$]

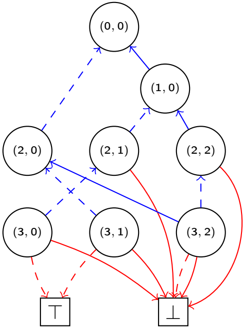
[$[(2,1) \mapsto (2, \text{max})]$,
$((2,0),(3,\text{max}),\bot)$]

Output:
$((2,\text{max}),(3,\text{max}),\bot)$



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\textit{red}}$:

[

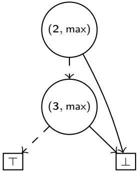
]

Level: 2

[$[(2,2) \mapsto \perp]$]

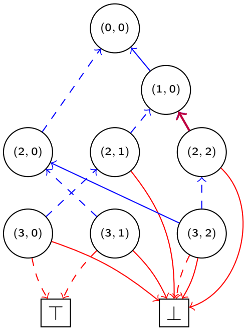
[$[(2,1) \mapsto (2,\max)]$,
$[(2,0) \mapsto (2,\max)]$]

Output:



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\boldsymbol{red}}$:

[

$(1,0) \xrightarrow{\top} \bot$,

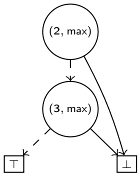
]

Level: 2

[ ]

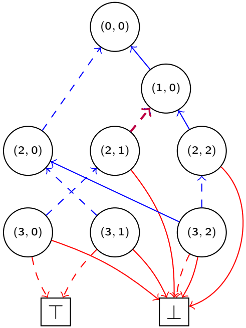
[$[(2,1) \mapsto (2,\mathrm{max})]$,
$[(2,0) \mapsto (2,\mathrm{max})]$]

Output:



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\boldsymbol{red}}$:

[

$(1,0) \xrightarrow{\top} \bot$,

$(1,0) \xrightarrow{\bot} (2, \text{max})$,

]

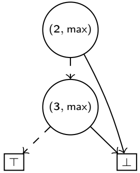
Level: 2

[ ]

[,

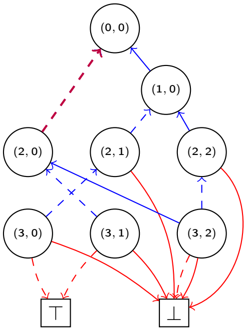
$[(2,0) \mapsto (2, \text{max})]$]

Output:



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\boldsymbol{red}}$:

[

$(1,0) \xrightarrow{\top} \bot$,

$(1,0) \xrightarrow{\bot} (2, \max)$,

$(0,0) \xrightarrow{\bot} (2, \max)$]

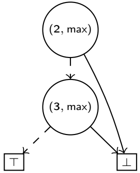
Level: 2

[ ]

[,

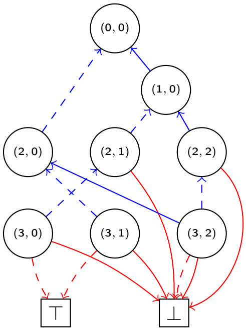
]

Output:



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: Q_{red}:

[

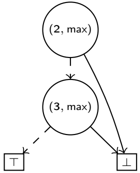
$(0,0) \xrightarrow{\perp} (2, \max)$]

Level: 1

[ ]

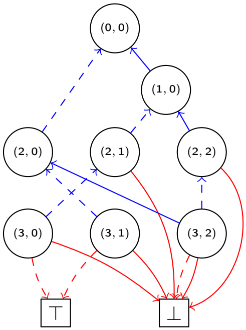
[$((1,0), (2, \max), \perp)$]

Output:



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\boldsymbol{red}}$:

[

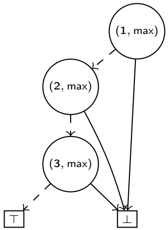
$(0,0) \xrightarrow{\perp} (2, \max)$]

Level: 1

[ ]

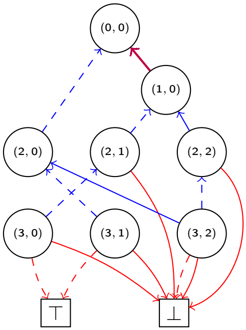
[$[(1,0) \mapsto (1, \max)]$]

Output:
$((1, \max), (2, \max), \perp)$



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\textit{red}}$:

[

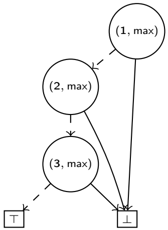
$(0,0) \xrightarrow{\top} (1, \max)$,
$(0,0) \xrightarrow{\perp} (2, \max)$]

Level: 1

[ ]

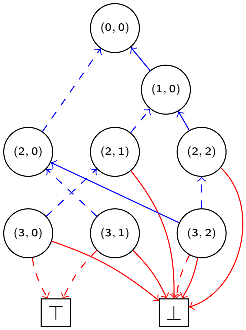
[ ]

Output:



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\boldsymbol{red}}$:

[

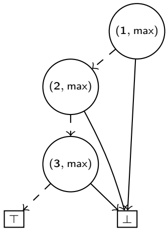
]

Level: 0

[ ]

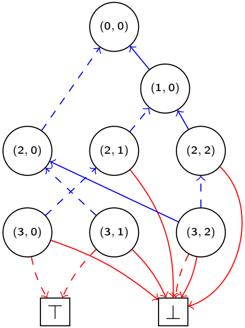
[((0, 0), (2, max), (1, max))]

Output:



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\boldsymbol{red}}$:

[

]

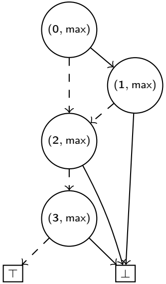
Level: 0

[ ]

[$[(0,0) \mapsto (0, \max)]$]

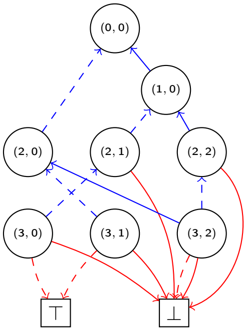
Output:

$((0, \max), (2, \max), (1, \max))$



(d) $(a) \wedge (b)$ reduced

Apply : *Example* *(Continued)*



(c) $(a) \wedge (b)$

Priority Queue: $Q_{\boldsymbol{red}}$:

[

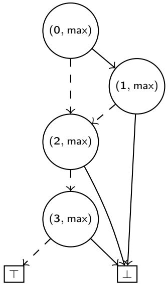
]

Level: 0

[ ]

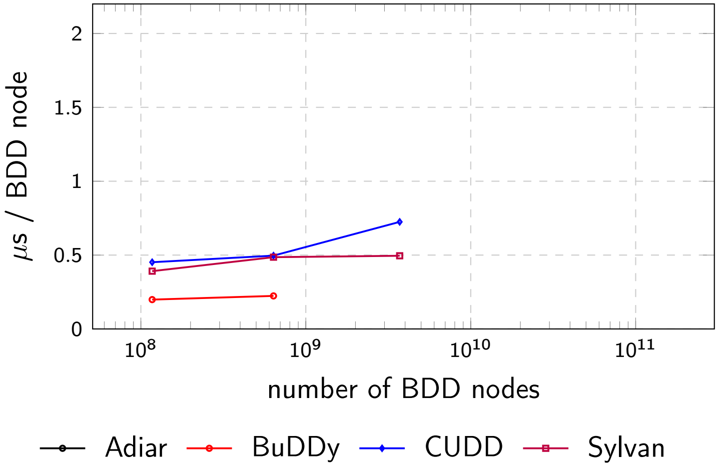
[ ]

Output:

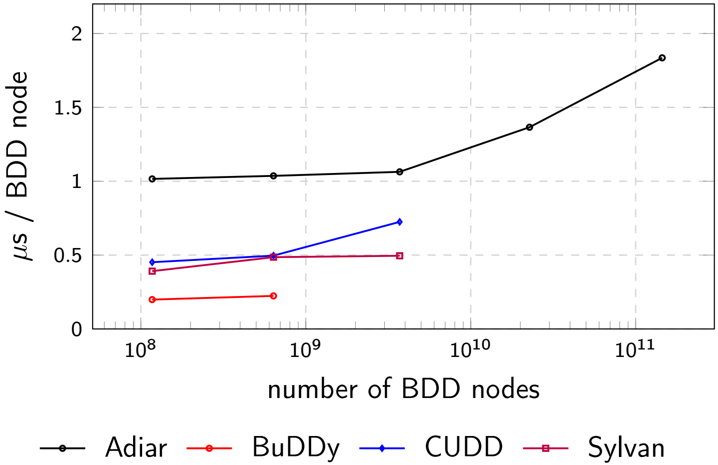


(d) $(a) \wedge (b)$ reduced

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`	$O(\text{sort}(N_f \cdot N_g))$



Running time for the *N-Queens* problems.



Running time for the *N-Queens* problems.

Contents

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`	?

Equality Checking

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

Equality Checking

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

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

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

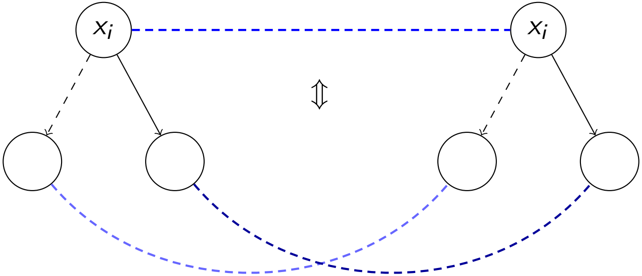
Theorem (Bryant '86)
Let π be a variable order and $f : \mathbb{B}^n \rightarrow \mathbb{B}$ then there exists a unique (up to isomorphism) Reduced Ordered Binary Decision Diagram representing f with ordering π.

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

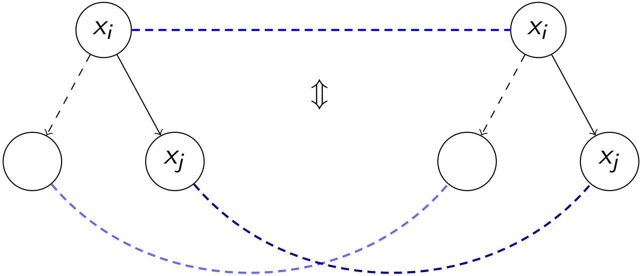
Equality Checking

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



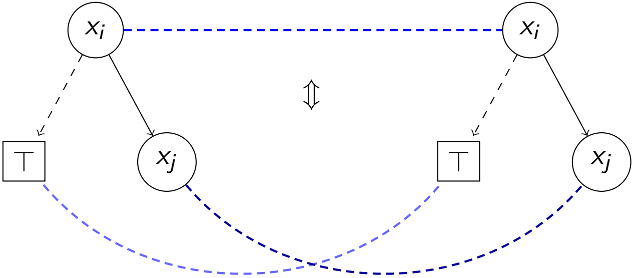
Equality Checking

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.



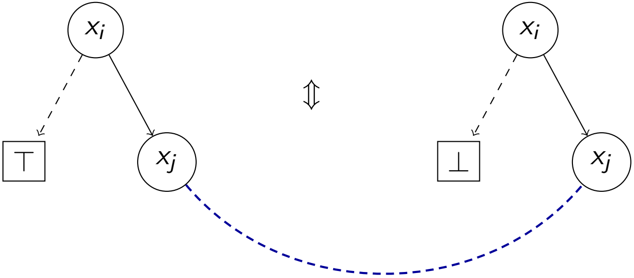
Equality Checking

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.



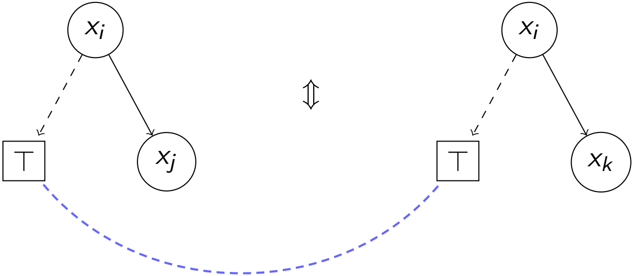
Equality Checking

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.



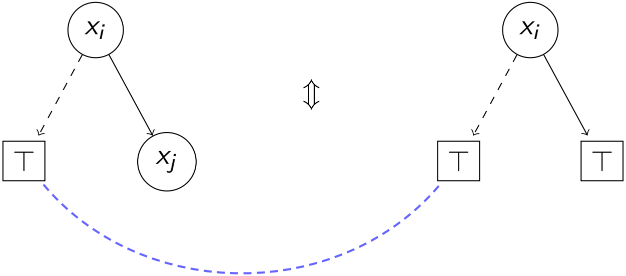
Equality Checking

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.

`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`)`

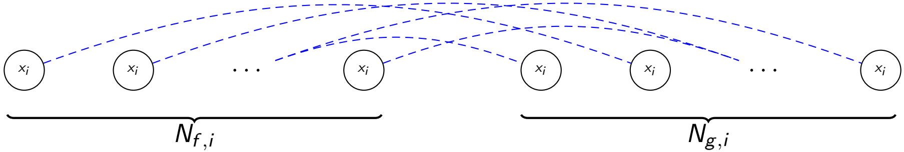
- Check whether root v_f of f and root v_g of g have a local violation.
- Check $low(v_f) \sim low(v_g)$ and $high(v_f) \sim high(v_g)$ "recursively".

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

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

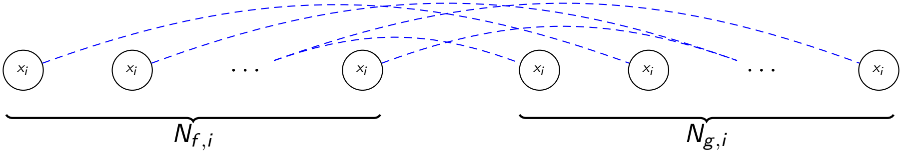
Equality Checking

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



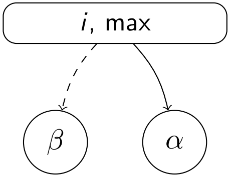
Return `false` if more than $N_{f,i} = N_{g,i}$ pairs of nodes are checked on level i.

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

Equality Checking

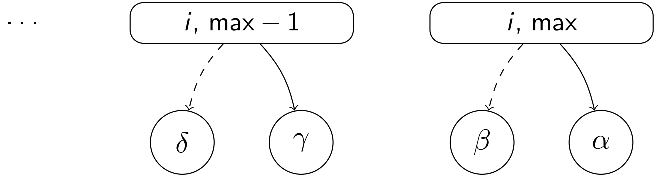
Observation
Each level output by the `Reduce` algorithm has the following properties:

Equality Checking



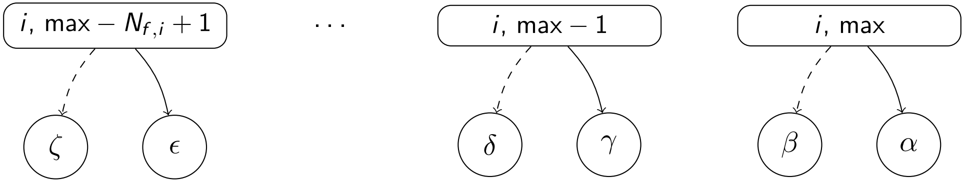
Observation
Each level output by the `Reduce` algorithm has the following properties:

Equality Checking



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

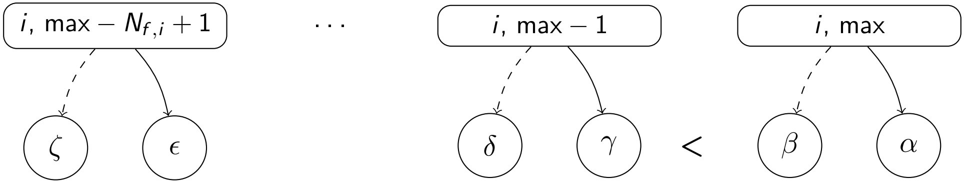
Equality Checking



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

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

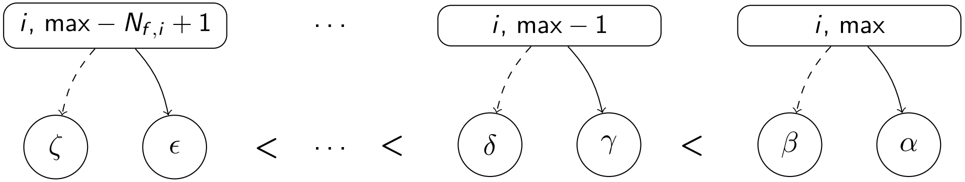
Equality Checking



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`*.*

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

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

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

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

Equality Checking

Algorithm	Time (s)
$f \leftrightarrow g \equiv \top$	0.38

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

Equality Checking

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

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

Equality Checking

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

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

Contents

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