Verification

Lecture 12

Bernd Finkbeiner
Peter Faymonville
Michael Gerke

UNIVERSITÄT
DES
SAARLANDES

REVIEW: Boolean functions

- Boolean functions $f : \mathbb{B}^n \to \mathbb{B}$ for $n \geq 0$ where $\mathbb{B} = \{ 0, 1 \}$
  - examples: $f(x_1, x_2) = x_1 \wedge (x_2 \vee \neg x_1)$, and $f(x_1, x_2) = x_1 \leftrightarrow x_2$
- Finite sets are boolean functions
  - let $|S| = N$ and $2^{n-1} < N \leq 2^n$
  - encode any element $s \in S$ as boolean vector of length n: $[\![\ ]\!] : S \to \mathbb{B}^n$
  - $T \subseteq S$ is represented by f_T such that:

$$f_T([\![s]\!]) = 1 \quad \text{iff} \quad s \in T$$

  - this is the characteristic function of T
- Relations are boolean functions
  - $\mathcal{R} \subseteq S \times S$ is represented by $f_{\mathcal{R}}$ such that:

$$f_R([\![s]\!], [\![t]\!]) = 1 \quad \text{iff} \quad (s, t) \in \mathcal{R}$$

REVIEW: Representing boolean functions

representation	compact?	sat	$\wedge$	$\vee$	$\neg$
propositional formula	often	hard	easy	easy	easy
DNF	sometimes	easy	hard	easy	hard
CNF	sometimes	hard	easy	hard	hard
(ordered) truth table	never	hard	hard	hard	hard
reduced ordered binary decision diagram	often	easy	medium	medium	easy

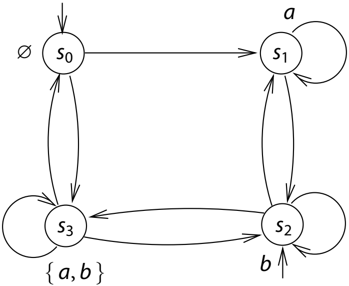
Explicitly representing transition systems

$TS = (S, Act, \rightarrow, I, AP, L)$ with $|S| = N$, $|Act| = M$ and $|AP| = K$:

- Identify the N states by numbers
- Represent the set of initial states I as boolean vector $\underline{i}$
  - $\underline{i}(s_j) = 1$ if and only if state $s_j \in I$
- Represent $\xrightarrow{\alpha}$ by M boolean matrices $\mathbf{T}_\alpha$ of size $N \times N$
  - $\mathbf{T}_\alpha(s_i, s_j) = 1$ if and only if $s_i \xrightarrow{\alpha} s_j$
- Represent L by an $N \times K$-boolean matrix $\mathbf{L}$
  - $\mathbf{L}(s_i, a_j) = 1$ if and only if $a_j \in L(s_i)$

$\Rightarrow$ Use sparse matrix representations for $\mathbf{T}$ and $\mathbf{L}$

Example (no actions)



$$\underline{i} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \text{and} \quad \mathbf{T} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{L} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

for simplicity, actions are omitted here

Transition systems as boolean functions

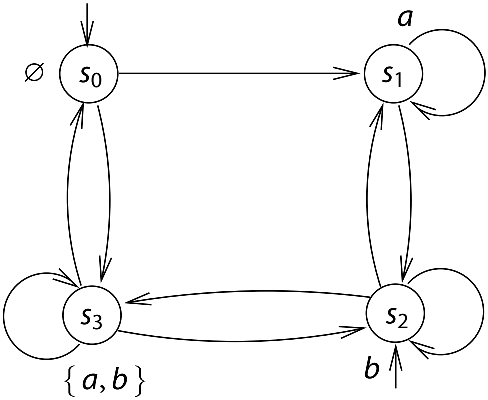
- Assume each state is uniquely labeled
  - $L(s) = L(s')$ implies $s = s'$
  - no restriction: if needed extend *AP* and label states uniquely
- Assume a fixed total order on propositions: $a_1 < a_2 < \ldots < a_K$
- Represent a state by a boolean function
  - over the boolean variables x_1 through x_K such that

$$[\![s]\!] = x_1^* \wedge x_2^* \wedge \ldots \wedge x_K^*$$

  - where the literal x_i^* equals x_i if $a_i \in L(s)$, and $\neg x_i$ otherwise

  $\Rightarrow$ no need to explicitly represent function L
- Represent I and $\rightarrow$ by their characteristic (boolean) functions
  - e.g., $f_{\rightarrow}([\![s]\!], [\![\alpha]\!], [\![t]\!]) = 1$ if and only if $s \xrightarrow{\alpha} t$

An example (no actions)



state	bit-vector	boolean function
s_0	$\langle 0, 0 \rangle$	$\neg x_1 \wedge \neg x_2$
s_1	$\langle 0, 1 \rangle$	$\neg x_1 \wedge x_2$
s_2	$\langle 1, 0 \rangle$	$x_1 \wedge \neg x_2$
s_3	$\langle 1, 1 \rangle$	$x_1 \wedge x_2$

- States:
- Initial states:

$$f_I(x_1, x_2) = (\neg x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_2)$$

Example (continued)

- Transition relation:

$f_\rightarrow$	$\langle 0,0\rangle$	$\langle 0,1\rangle$	$\langle 1,0\rangle$	$\langle 1,1\rangle$
$\langle 0,0\rangle$	0	1	0	1
$\langle 0,1\rangle$	0	1	1	0
$\langle 1,0\rangle$	0	1	1	1
$\langle 1,1\rangle$	1	0	1	1

- Alternatively: $f_\rightarrow(\underbrace{x_1, x_2}_{s}, \underbrace{x_1', x_2'}_{s'}) = 1$ if and only if $s \rightarrow s'$

$$
\begin{array}{rl}
f_\rightarrow(x_1, x_2, x_1', x_2') = & (\neg x_1 \wedge \neg x_2 \wedge \neg x_1' \wedge x_2') \\
\vee & (\neg x_1 \wedge \neg x_2 \wedge x_1' \wedge x_2') \\
\vee & (\neg x_1 \wedge x_2 \wedge x_1' \wedge \neg x_2') \\
\vee & \ldots \\
\vee & (x_1 \wedge x_2 \wedge x_1' \wedge x_2')
\end{array}
$$

Binary decision trees

- Let X be a set of boolean variables and $<$ a total order on X
- Binary decision tree (BDT) is a complete binary tree over $\langle X, < \rangle$
  - each leaf v is labeled with a boolean value $val(v) \in \mathbb{B}$
  - non-leaf v is labeled by a boolean variable $Var(v) \in X$
  - such that for each non-leaf v and vertex w:

$$w \in \{\, left(v), right(v) \,\} \;\Rightarrow\; (Var(v) < Var(w) \;\vee\; w \text{ is a leaf})$$

$\Rightarrow$ On each path from root to leaf, variables occur in the same order

Shannon expansion

- Each boolean function $f : \mathbb{B}^n \longrightarrow \mathbb{B}$ can be written as:

$$f(x_1, \ldots, x_n) = (x_i \ \wedge \ f[x_i := 1]) \ \vee \ (\neg x_i \ \wedge \ f[x_i := 0])$$

  - where $f[x_i := 1]$ stands for $f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$
  - and $f[x_i := 0]$ for $f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$
- The boolean function $f_B(v)$ represented by vertex v in BDT B is:
  - for v a leaf: $f_B(v) = val(v)$
  - otherwise:

$$f_B(v) = (Var(v) \ \wedge \ f_B(right(v))) \ \vee \ (\neg Var(v) \ \wedge \ f_B(left(v)))$$

- $f_B = f_B(v)$ where v is the root of B

Considerations on BDTs

- BDTs are not compact
  - a BDT for boolean function $f : \mathbb{B}^b \to \mathbb{B}$ has 2^n leafs
  - $\Rightarrow$ they are as space inefficient as truth tables!
- $\Rightarrow$ BDTs contain quite some redundancy
  - all leafs with value one (zero) could be collapsed into a single leaf
  - a similar scheme could be adopted for isomorphic subtrees
- The size of a BDT does not change if the variable order changes

Ordered Binary Decision Diagram

share equivalent expressions [Akers 76, Lee 59]

- Binary decision diagram (OBDD) is a directed graph over $\langle X, < \rangle$ with:
  - each leaf v is labeled with a boolean value $val(v) \in \{ 0, 1 \}$
  - non-leaf v is labeled by a boolean variable $Var(v) \in X$
  - such that for each non-leaf v and vertex w:

$$w \in \{ left(v), right(v) \} \;\Rightarrow\; (Var(v) < Var(w) \;\vee\; w \text{ is a leaf})$$

$\Rightarrow$ An OBDD is acyclic

  - f_B for OBDD B is obtained as for BDTs

Isomorphism

- B and B′ over $\langle X, < \rangle$ are isomorphic iff their roots are isomorphic
- Vertices v in B and w in B′ are isomorphic, denoted $v \cong w$, iff

  there exists a bijection H between the vertices of B and B′ such that:

  1. if v is a leaf, then $H(v) = w$ is a leaf with $val(v) = val(H(v))$
  2. if v is a non-leaf, then $H(v) = w$ is a non-leaf such that

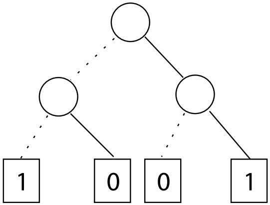
  $$Var(v) = Var(w) \wedge H(left(v)) = left(H(v)) \wedge H(right(v)) = right(H(v))$$

- Testing B $\cong$ B′ can be done in linear time
  - due to the labels (0 and 1) of the edges.

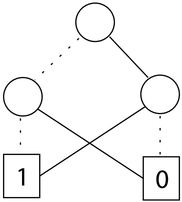
Reducing OBDDs

- Generate an OBDD (or BDT) for a boolean expression, then reduce
  - by means of a recursive descent over the OBDD
- Elimination of duplicate leafs
  - for a duplicate 0-leaf (or 1-leaf), redirect all incoming edges to just one of them
- Elimination of "don't care" (non-leaf) vertices
  - if $\mathit{left}(v) = \mathit{right}(v) = w$, eliminate v and redirect all its incoming edges to w
- Elimination of isomorphic subtrees
  - if $v \neq w$ are roots of isomorphic subtrees, remove w
  - and redirect all incoming edges to w to v

How to reduce an OBDD?



becomes

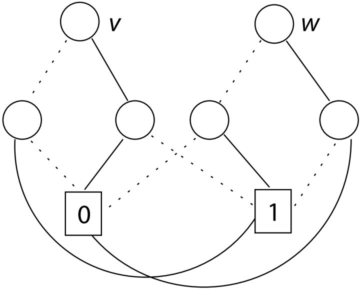


eliminating identical leafs

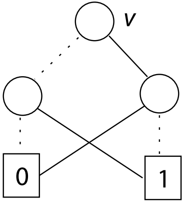
How to reduce an OBDD?

eliminating “don’t care” vertices

How to reduce a BDD?



becomes



eliminating isomorphic subtrees

Reduced OBDDs

OBDD B over $\langle X, < \rangle$ is called reduced iff:

1. for each leaf v, w: $(val(v) = val(w)) \Rightarrow v = w$

   $\Rightarrow$ identical terminal vertices are forbidden

2. for each non-leaf v: $left(v) \neq right(v)$

   $\Rightarrow$ non-leafs may not have identical children

3. for each non-leaf v, w:

$$(Var(v) = Var(w) \wedge right(v) \cong right(w) \wedge left(v) \cong left(w)) \Rightarrow v = w$$

$\Rightarrow$ vertices may not have isomorphic sub-dags

this is what is mostly called BDD; in fact it is an ROBDD!

Dynamic generation of ROBDDs

Main idea:

- Construct directly an ROBDD from a boolean expression
- Create vertices in depth-first search order
- On-the-fly reduction by applying hashing
  - on encountering a new vertex *v*, check whether:
  - an equivalent vertex *w* has been created (same label and children)
  - $left(v) = right(v)$, i.e., vertex *v* is a "don't care" vertex

ROBDDs are canonical

[Fortune, Hopcroft & Schmidt, 1978]

For ROBDDs B and B′ over $\langle X, < \rangle$ we have:

$(f_B = f_{B'})$ implies B and B′ are isomorphic

$\Rightarrow$ for a fixed variable ordering, any boolean function
can be uniquely represented by an ROBDD (up to isomorphism)

The importance of canonicity

- Absence of redundant vertices
  - if f_{B} does not depend on x_i, ROBDD B does not contain an x_i vertex
- Test for equivalence: $f(x_1, \ldots, x_n) \equiv g(x_1, \ldots, x_n)$?
  - generate ROBDDs B_f and B_g, and check isomorphism
- Test for validity: $f(x_1, \ldots, x_n) = 1$?
  - generate ROBDD B_f and check whether it only consists of a 1-leaf
- Test for implication: $f(x_1, \ldots, x_n) \rightarrow g(x_1, \ldots, x_n)$?
  - generate ROBDD $\mathrm{B}_f \wedge \neg \mathrm{B}_g$ and check if it just consist of a 0-leaf
- Test for satisfiability
  - f is satisfiable if and only if B_f is not just the 0-leaf

Variable ordering

- Different ROBDDs are obtained for different variable orderings
- The size of the ROBDD depends on the variable ordering
- Some boolean functions have linear and exponential ROBDDs
- Some boolean functions only have polynomial ROBDDs
- Some boolean functions only have exponential ROBDDs

The even parity function

$f(x_1, \ldots, x_n) = 1$ iff the number of variables x_i with value 1 is even

truth table or propositional formula for f has exponential size

but an ROBDD of linear size is possible

Symmetric functions

$$f[x_1 := b_1, \ldots, x_n := b_n] = f[x_1 := b_{i_1}, \ldots, x_{i_n} := b_{i_n}]$$

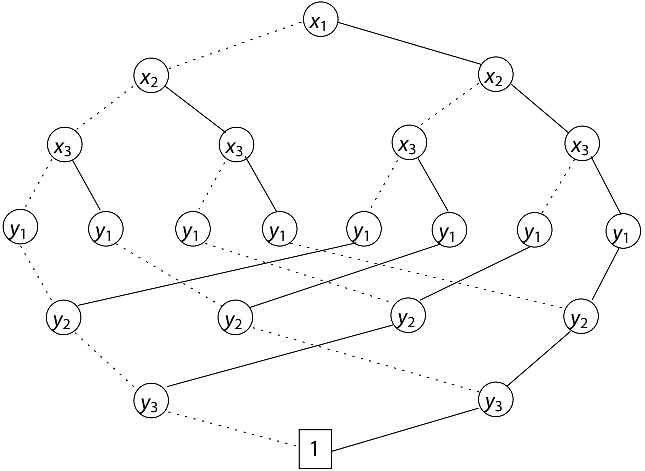
for each permutation $(i_1, \ldots, i_n)$ of $(1, \ldots, n)$

$\Rightarrow$ The value of f depends only on the number of ones!

Examples: $f(\ldots) = x_1 \oplus \ldots \oplus x_n$,
$f(\ldots) = 1$ iff $\geq k$ variables x_i are true

symmetric boolean functions have ROBDDs of size in $\mathcal{O}(n^2)$

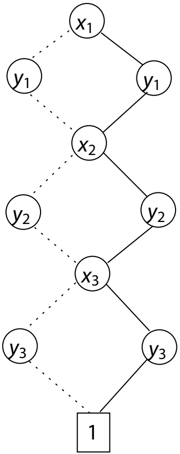
The function stable with exponential ROBDD



The ROBDD of $f(\overline{x}, \overline{y}) = (x_1 \leftrightarrow y_1) \wedge \ldots \wedge (x_n \leftrightarrow y_n)$

has $3 \cdot 2^n - 1$ vertices under ordering $x_1 < \ldots < x_n < y_1 < \ldots < y_n$

The function stable with linear ROBDD



The ROBDD of $f(\overline{x}, \overline{y}) = (x_1 \leftrightarrow y_1) \wedge \ldots \wedge (x_n \leftrightarrow y_n)$

has $3 \cdot n + 2$ vertices under ordering $x_1 < y_1 < \ldots < x_n < y_n$

The multiplication function

- Consider two n-bit integers
  - let $b_{n-1}b_{n-2}\ldots b_0$ and $c_{n-1}c_{n-2}\ldots c_0$
  - where b_{n-1} is the most significant bit, and b_0 the least significant bit
- Multiplication yields a $2n$-bit integer
  - the ROBDD $\mathrm{B}_{f_{n-1}}$ has at least 1.09^n vertices
  - where f_{n-1} denotes the the $(n-1)$-st output bit of the multiplication

Optimal variable ordering

- The size of ROBDDs is dependent on the variable ordering
- Is it possible to determine $<$ such that the ROBDD has minimal size?
  - the optimal variable ordering problem for ROBDDs is NP-complete
  - polynomial reduction from the 3SAT problem

(Bollig & Wegener, 1996)

- There are many boolean functions with large ROBDDs
  - for almost all boolean functions the minimal size is in $\Omega(\frac{2^n}{n})$
- How to deal with this problem in practice?
  - guess a variable ordering in advance
  - rearrange the variable ordering during the manipulations of ROBDDs

Sifting algorithm

(Rudell, 1993)

Dynamic variable ordering using variable swapping:

1. Select a variable x_i
2. By successive swapping of x_i, determine $|B|$ at any position for x_i
3. Shift x_i to its optimal position
4. Go back to the first step until no improvement is made

- Characteristics:
  - a variable may change position several times during a single sifting iteration
  - often yields a local optimum, but works well in practice

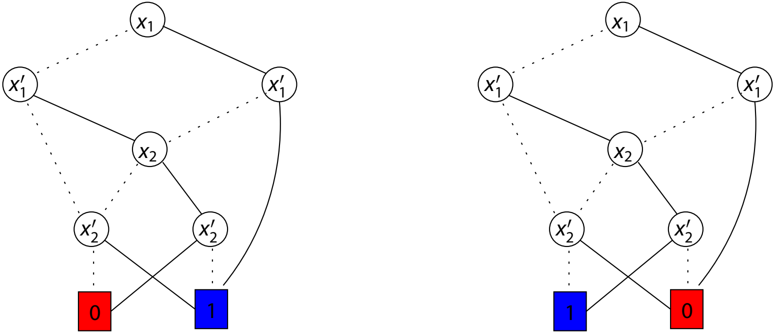
Interleaved variable ordering

- Which variable ordering to use for transition relations?
- The interleaved variable ordering:
  - for encodings $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ of state s and t respectively:

$$x_1 < y_1 < x_2 < y_2 < \ldots < x_n < y_n$$

- This variable ordering yields compact ROBDDs for binary relations

Negation



negation amounts to interchange the 0- and 1-leaf

Apply

- Shannon expansion for binary operations:

$$\begin{aligned} f \ op \ g \quad = & \quad (x_1 \wedge (f[x_1 := 1] \ op \ g[x_1 := 1])) \\ & \vee \ (\neg x_1 \wedge (f[x_1 := 0] \ op \ g[x_1 := 0])) \end{aligned}$$

- A top-down evaluation scheme using Shannon's expansion:
  - let v be the variable highest in the ordering occurring in B_f orB_g
  - split the problem into subproblems for $v := 0$ and $v := 1$, and solve recursively
  - at the leaves, apply the boolean operator *op* directly
  - reduce afterwards to turn the resulting OBDD into an ROBDD
- Efficiency gain is obtained by dynamic programming
  - the time complexity of constructing the ROBDD of $B_{f \ op \ g}$ is in $\mathcal{O}(|B_f| \cdot |B_g|)$

Algorithm Apply(op, B_f, B_g)

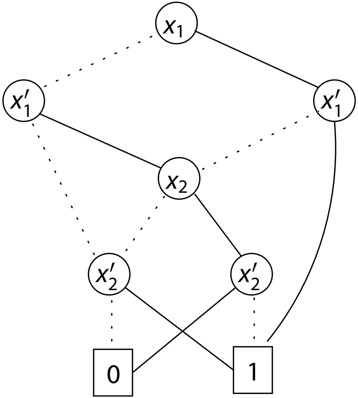
B.root := Apply(op, B_f.root, B_g.root);

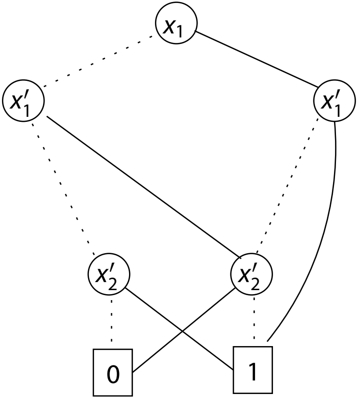
if $G(v_1, v_2) \neq$ empty **then return** $G(v_1, v_2)$ **fi**; {lookup in hashtable}
if (v_1 and v_2 are terminals) **then** res := $val(v_1)\ op\ val(v_2)$ **fi**;
else if (v_1 is terminal and v_2 is nonterminal)
then res :=
$MakeNode(Var(v_2), \text{Apply}(op, v_1, left(v_2)), \text{Apply}(op, v_1, right(v_2)))$;
else if (v_1 is nonterminal and v_2 is terminal)
then res :=
$MakeNode(Var(v_1), \text{Apply}(op, left(v_1), v_2), \text{Apply}(op, right(v_1), v_2))$;
else if ($Var(v_1) = Var(v_2)$)
then res :=
$MakeNode(Var(v_1), \text{Apply}(op, left(v_1), left(v_2)), \text{Apply}(op, right(v_1), right(v_2)))$;
else if ($Var(v_1) < Var(v_2)$)
then res :=
$MakeNode(Var(v_1), \text{Apply}(op, left(v_1), v_2), \text{Apply}(op, right(v_1), v_2))$;
else {$Var(v_1) > Var(v_2)$}
res := $MakeNode(Var(v_2), \text{Apply}(op, v_1, left(v_2)), \text{Apply}(op, v_1, right(v_2)))$;
$G(v_1, v_2)$:= res; {memoize result}
return res

Algorithm Restrict(B, x, b)

- For each vertex v labeled with variable x:
  - if $b = 1$ then redirect incoming edges to $right(v)$
  - if $b = 0$ then redirect incoming edges to $left(v)$
  - remove vertex v, and all vertices only reachable through v
  - (if necessary) reduce (only above v)

Restrict





performing Restrict$(B, x_2, 1)$: replace x_2 by constant 1

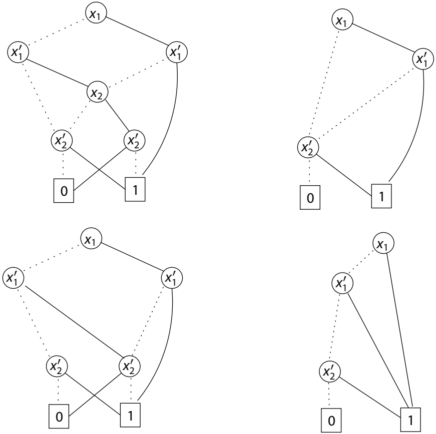
Abstract

- Existential quantification over x_i:

$$\exists x_i.\, f(x_1, \ldots, x_n) = f[x_i := 1] \ \vee \ f[x_i := 0]$$

- Naive realization: $\text{Apply}(\vee, \text{Restrict}(B_f, x_i, 1), \text{Restrict}(B_f, x_i, 0))$
- Efficiency gain:
  - observe that $\text{Restrict}(B_f, x_i, 1)$ and $\text{Restrict}(B_f, x_i, 0)$ are equal up to x_i
  - ... the resulting ROBDD also has the same structure up to x_i
  - replace each node labeled with x_i by the result of applying $\vee$ on its children
- This can easily be generalized to $\exists x_1. \ldots \exists x_k. f(x_1, \ldots x_n)$

Example



ROBBDs B_f (left up), $B_{f[x_2:=0]}$ (right up), $B_{f[x_2:=1]}$ (left down), and $B_{\exists x_2.\, f}$ (right down)

Operations on ROBDDs

Algorithm	Output	Time complexity	Space complexity
Reduce	B′ (reduced) with $f_{\mathrm{B}} = f_{\mathrm{B}'}$	$\mathcal{O}(\lvert B_f \rvert \cdot \log \lvert \mathrm{B}_f \rvert)$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$
Not	$\mathrm{B}_{\neg f}$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$
Apply	$\mathrm{B}_{f\ op\ g}$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert \cdot \lvert \mathrm{B}_g \rvert)$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert \cdot \lvert \mathrm{B}_g \rvert)$
Restrict	$\mathrm{B}_{f[x:=b]}$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$
Rename	$\mathrm{B}_{f[x:=y]}$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert)$
Abstract	$\mathrm{B}_{\exists x.\, f}$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert^2)$	$\mathcal{O}(\lvert \mathrm{B}_f \rvert^2)$

operations are only efficient if f and g have compact ROBDD representations