BDD MODEL CHECKING

BINARY DECISION DIAGRAMS

Loïc Massin

University of Oslo
INF5140 / Spring 2017

Basic Model Checking Problem

System describe by states.

Basic approach : represent each state individually.

➔Problem, size of the state space increases exponentially.

➔State Space Explosion.

- Need too much memory;
- Need too much time.

ONE SOLUTION

Symbolic model checking :

- Idea: represent set of states by Boolean formula over Boolean variables.

  $$f : Bool^n \rightarrow Bool$$

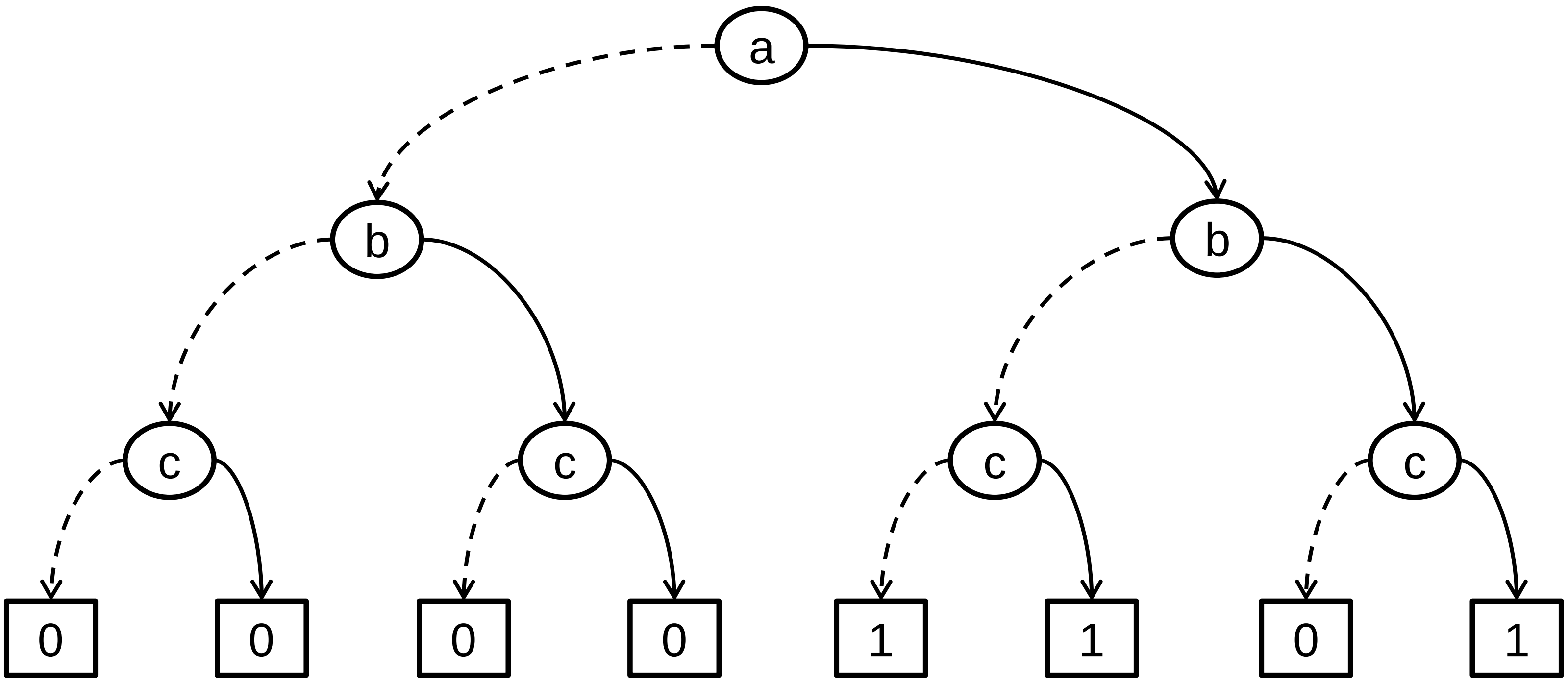
- Need efficient representation and manipulation for state sets and transition relation.

➔ Use Binary Decision Diagrams

Binary Decision Trees

- **Directed acyclic graphs.**
- **One or two Terminal nodes / Leaves:** labelled with 0 or 1;
- **Set of variables nodes u of out-degree two:**
  - Non-Terminal nodes: each are labelled with a variable var(u);
  - Branches / Children: low(v) / high(v), correspond to assignment of 0 or 1 for the variable in the node

EXAMPLE OF BDT

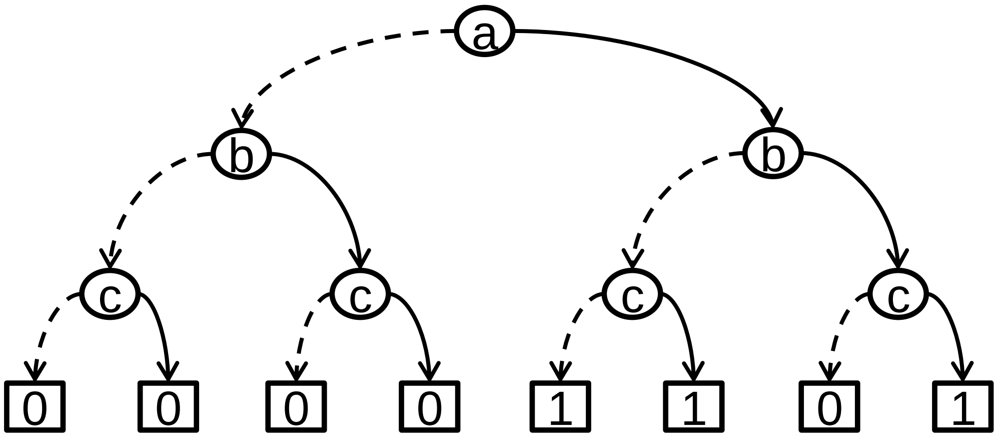


$$a \wedge (\neg b \vee c)$$

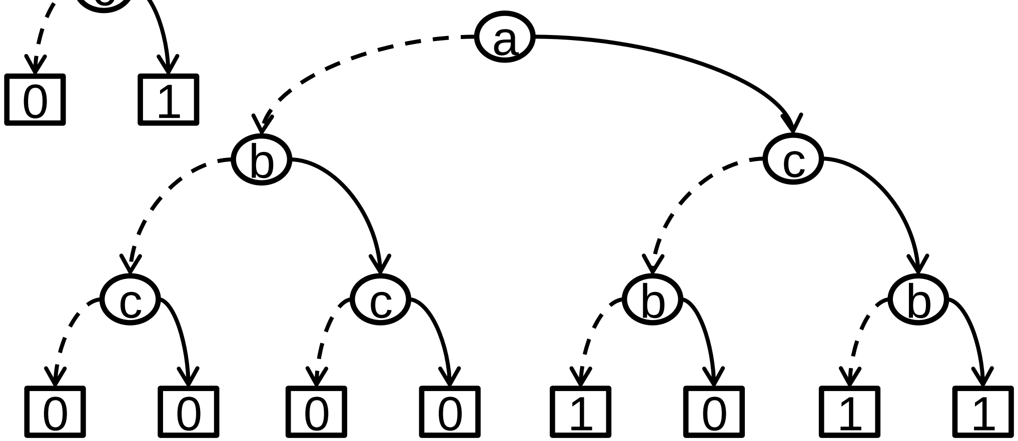
Dashed lines denote low-branches, solid lines high-branches

PROBLEMS

- Still exponential;
- Several BDT can verify the same formula.



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



BASICS BDD PROPERTIES

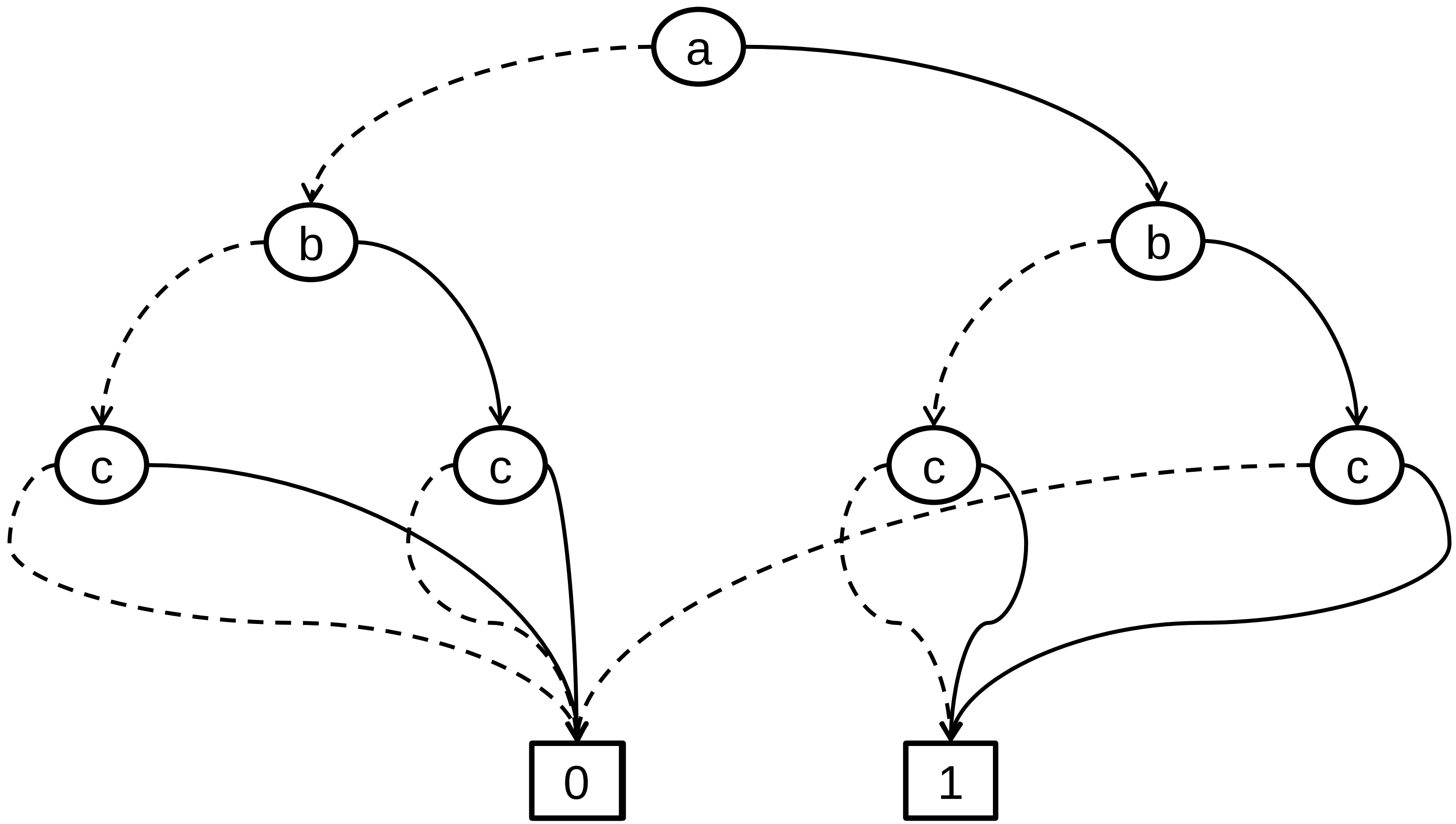
To move from BDT to BDD:

- Merge terminal nodes;

Ordered BDD (OBDD):

- Define a variable ordering: on all paths from root to leafs, variables appear in same order, without repetitions (there exists a global ordering of variables).

EXAMPLE OF OBDD



a ∧ (¬b ∨ c) with ordering a < b < c

Reduced Ordered BDD (1)

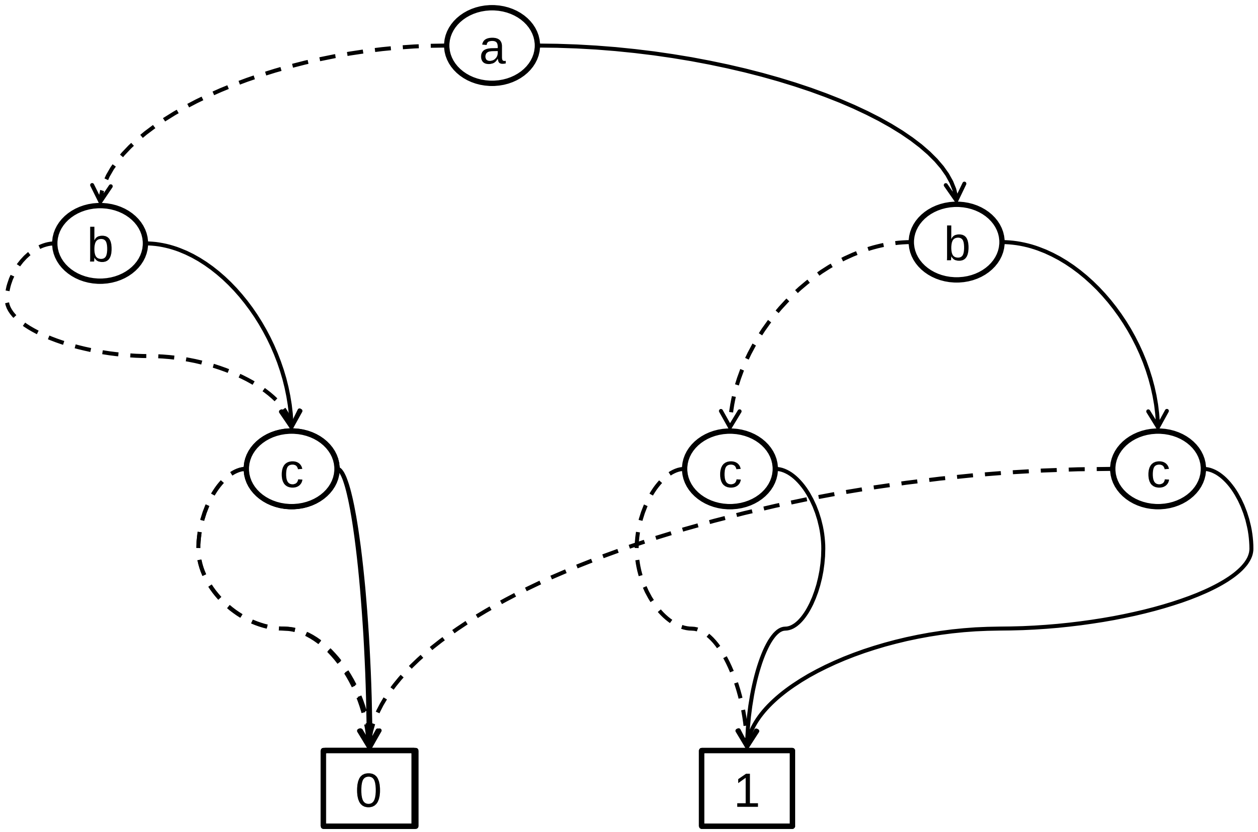
Uniqueness: no two distinct nodes v and w have the same variable name and low- and high- children.

- ➔ Merge isomorphic subgraphs;

Non-redundant tests: No variable node v has identical low- and high- children.

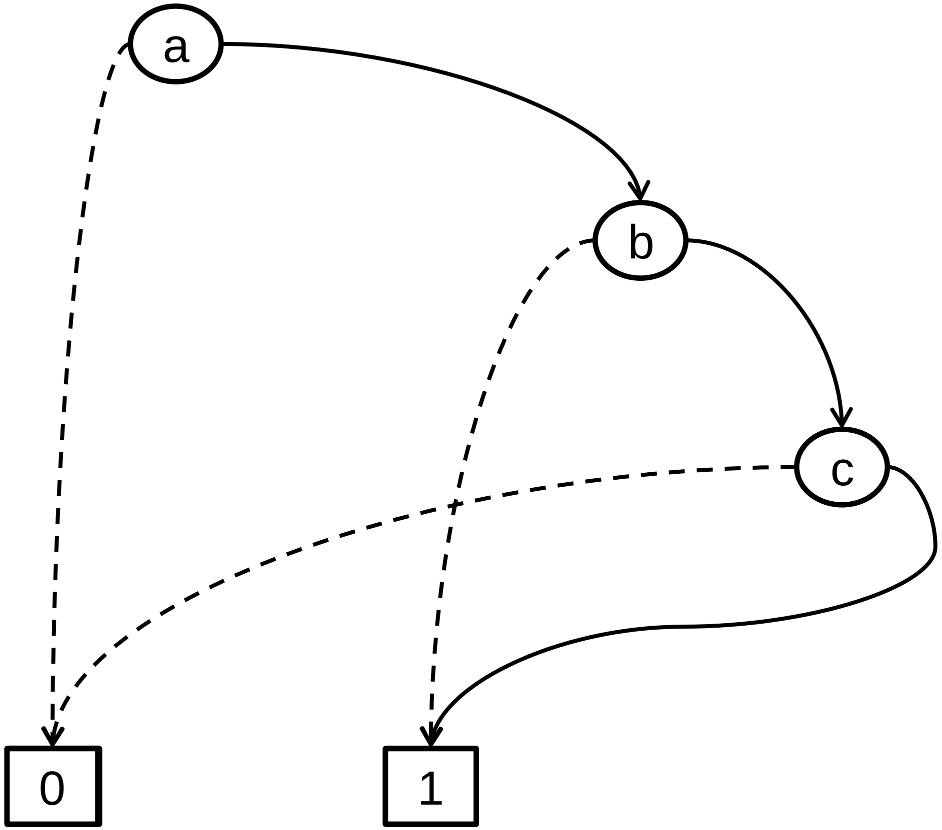
- ➔ Remove redundancy.

MERGE ISOMORPHIC SUBGRAPHS



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

REMOVE REDUNDANCY



a ∧ (¬b ∨ c)

ROBDD (2)

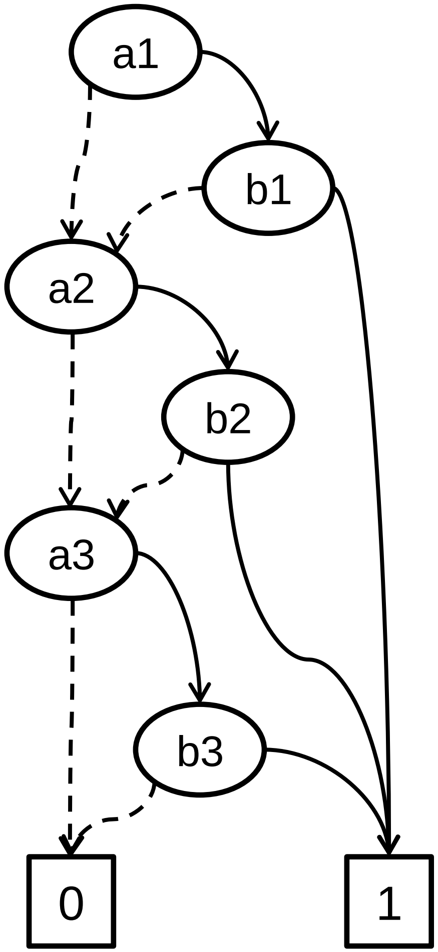
Canonical (unique) representation of a Boolean formula for a particular variable order:

For every function f : $\text{Bool}^n \rightarrow \text{Bool}$ *and variable ordering* $x_1 < x_2 < \cdot\cdot\cdot < x_n$, *there exists exactly one ROBDD representing this function.*

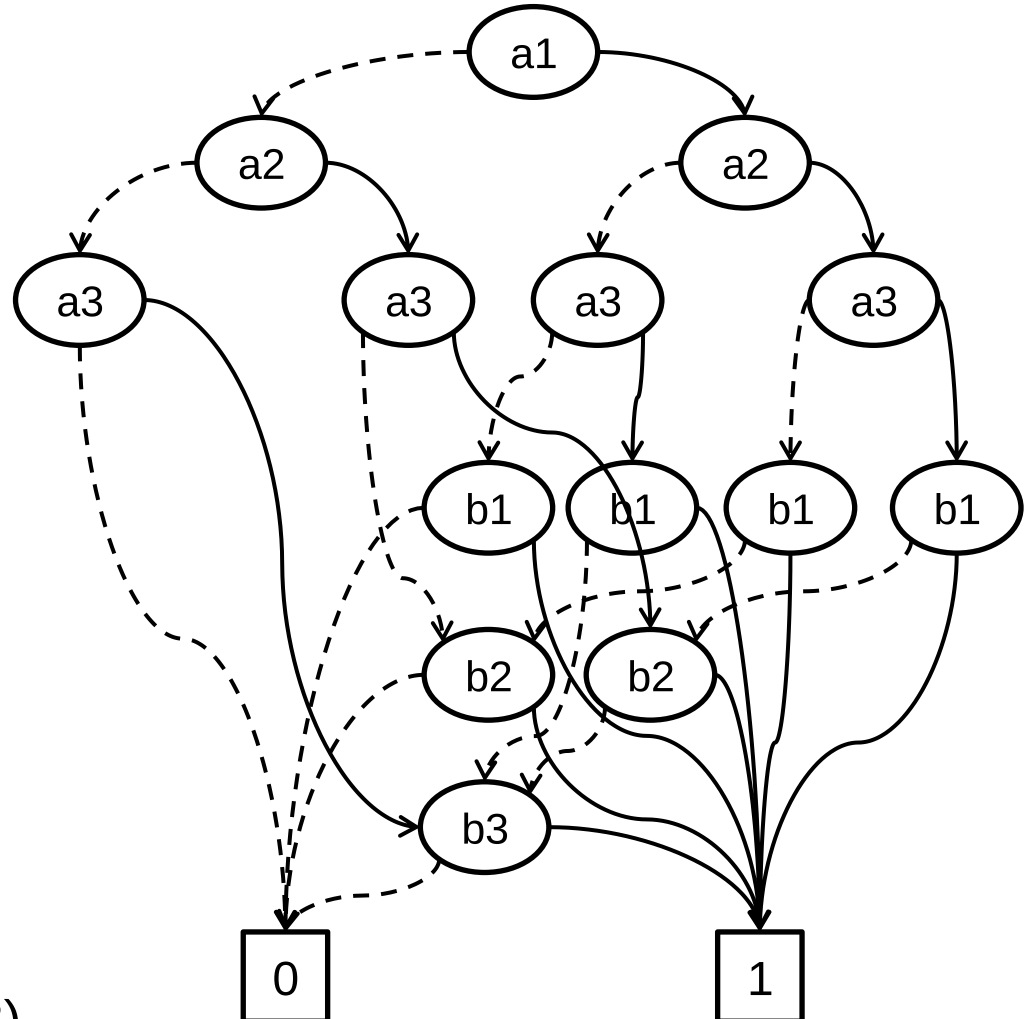
Equivalence checking in linear time, and satisfiability checking in constant time.

Most of time, we will refer to ROBDD simply as BDDs.

SENSITIVITY TO VARIABLE ORDERING (1)



(a1 ∧ b1) ∨ (a2 ∧ b2) ∨ (a3 ∧ b3)



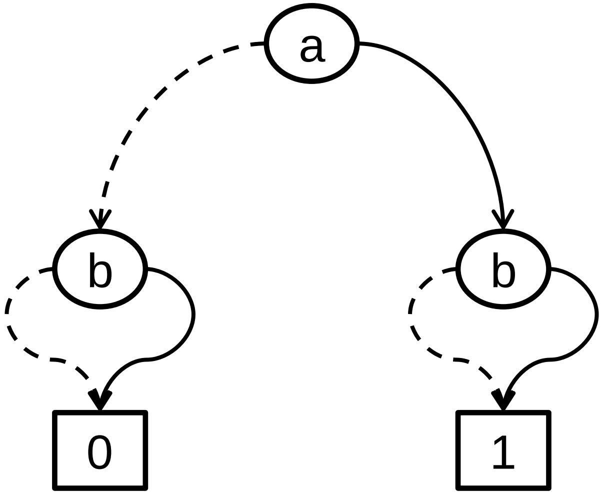
Sensitivity to Variable Ordering (2)

- Two different variable ordering lead to tow different ROBDD.
- Crucial importance in practice, determine the efficiency of ROBDD-based model checking.
- Finding the best variable ordering is **NP-hard**. It exists several heuristics to approach the problem.

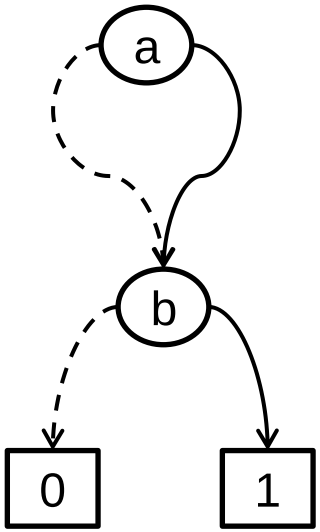
The algorithm APPLY (1)

- If Bϕ and Bψ are two OBDDs, the call **apply(op, Bϕ, Bψ)** computes the OBBD of the formula ϕ op ψ.
- Operates recursively on the structure of the two OBDDs:
  - We start at the root and follow parallel paths on the two OBDDs to the leaves;
  - Once we arrive at the leaves, we apply the given boolean operation to the boolean constants 0 and 1 to form the result for that particular path.

THE ALGORITHM APPLY (2)

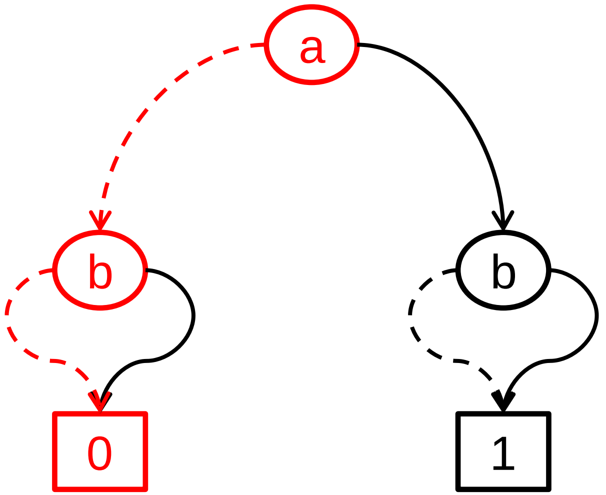


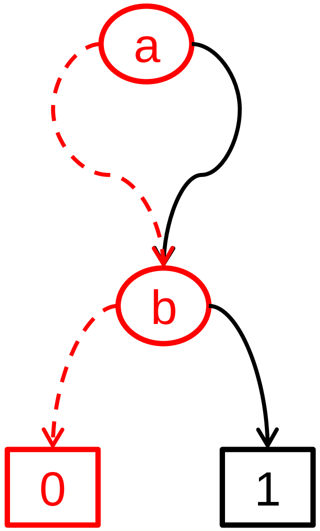
f(a,b) = a
with a>b



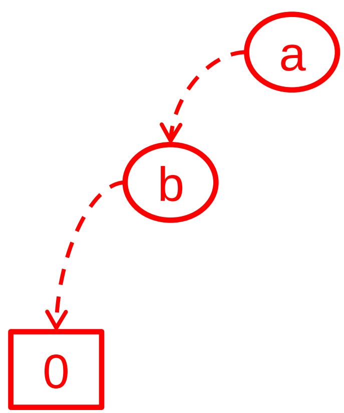
f(a,b) = b
with a>b

THE ALGORITHM APPLY (2)

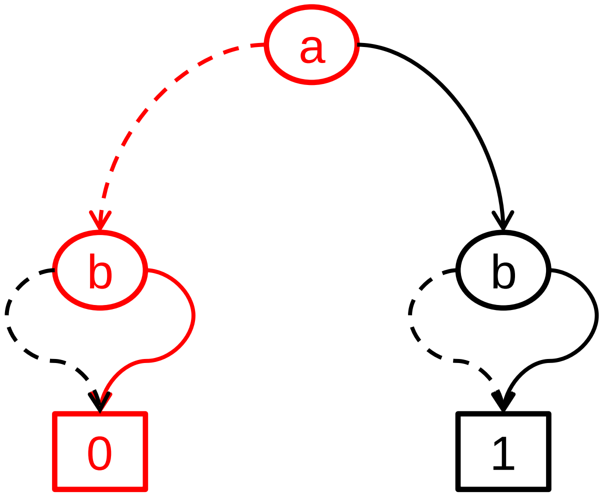


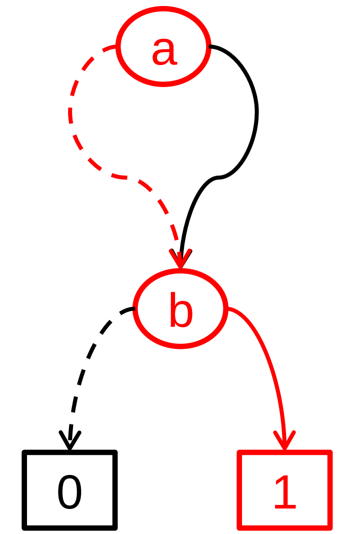


$f(a,b) = a \vee b$
with $a>b$

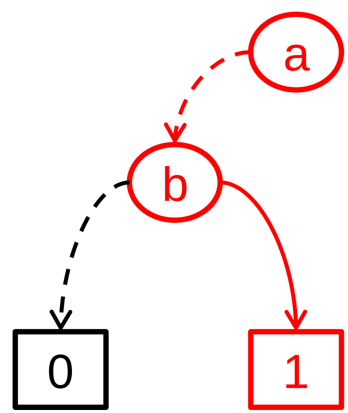


The algorithm APPLY (2)

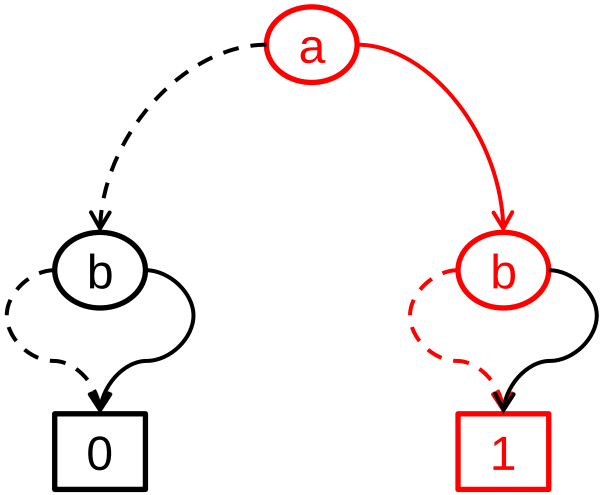


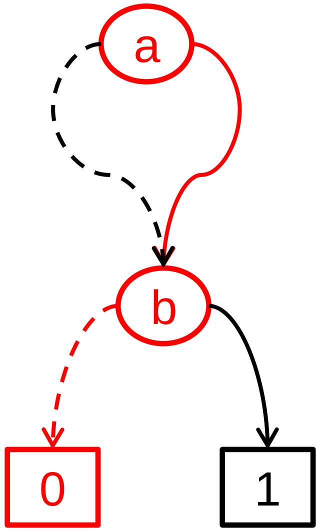


f(a,b) = a ∨ b
with a>b

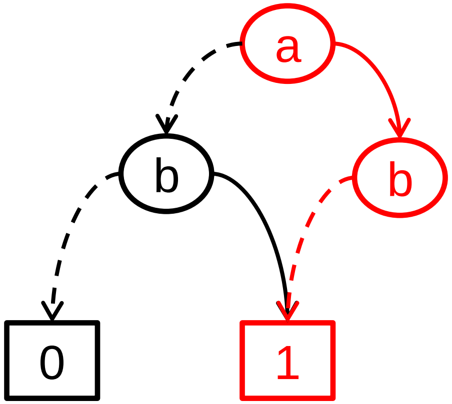


THE ALGORITHM APPLY (2)

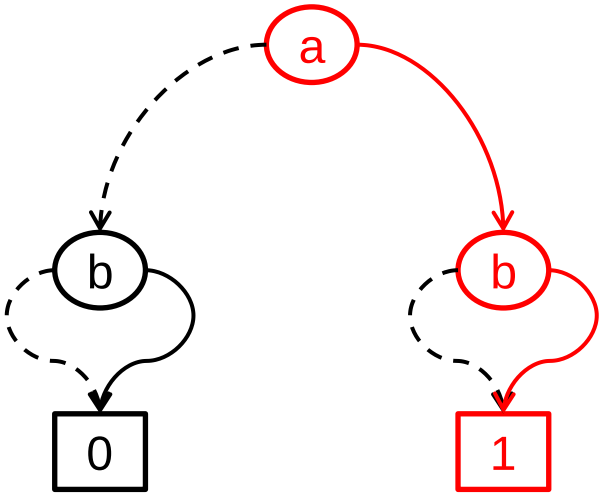


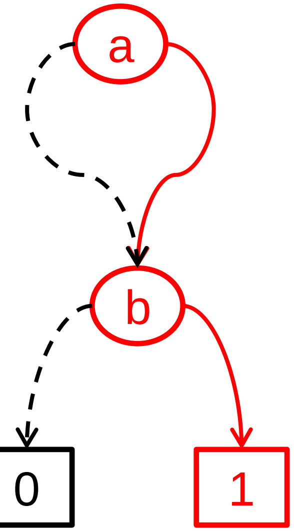


f(a,b) = a v b
with a>b



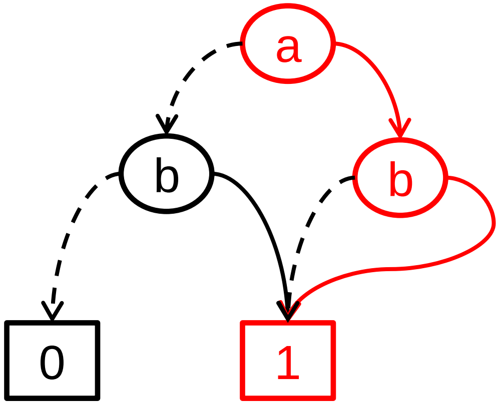
THE ALGORITHM APPLY (2)





$f(a,b) = a \vee b$

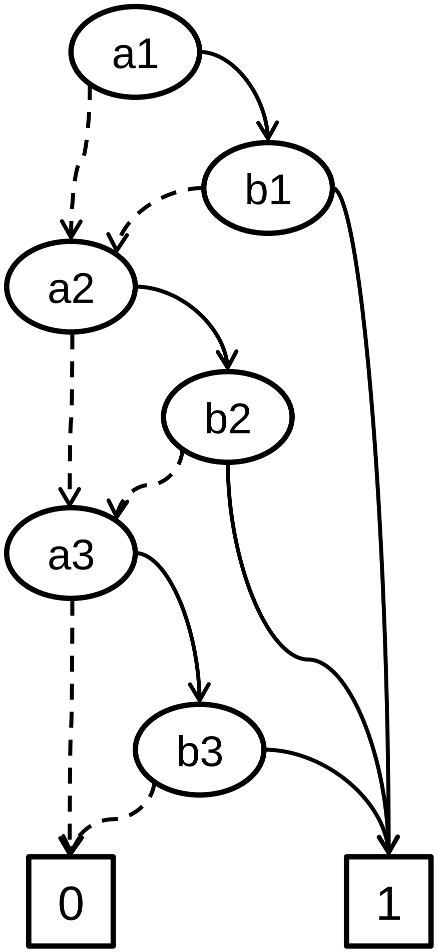
with $a>b$



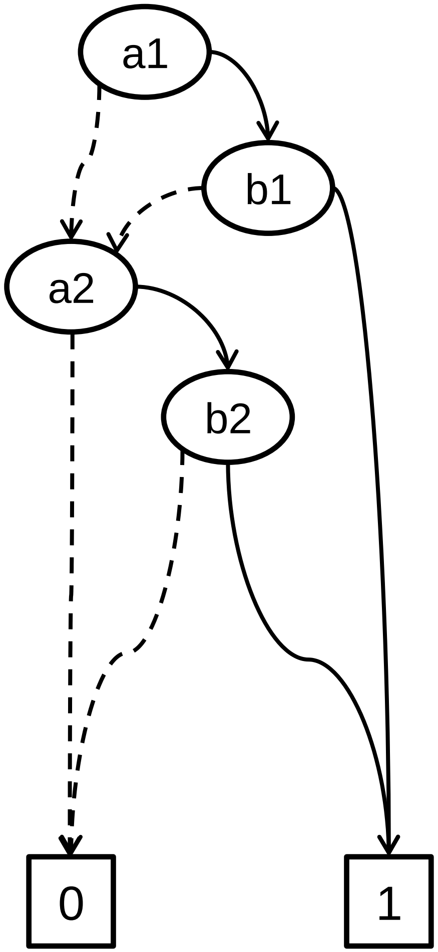
THE ALGORITHM RESTRICT (1)

- If Bϕ is a OBDD, the call **restrict(0, x, Bϕ)** (respectively restrict(1, x, Bϕ)) the OBDD for $\phi[0/x]$ (respectively $\phi[1/x]$).
- **restrict(0, x, Bϕ)**
  - For each node v labeled with x:
    - ➔ Incoming edges are redirected to low(v);
    - ➔ Node v is removed.
- **restrict(1, x, Bϕ)**
  - As above but redirected to high(v).

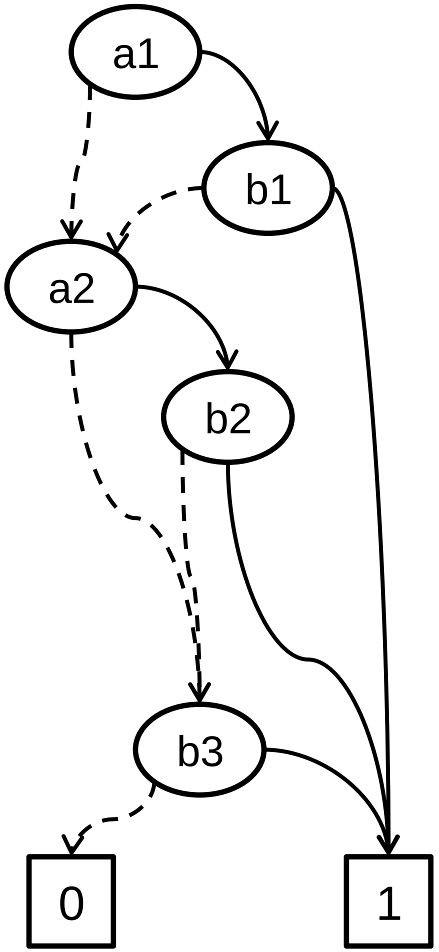
THE ALGORITHM RESTRICT (1)



Bϕ



restrict(0, a3, Bϕ)



restrict(1, a3, Bϕ)

References

- Henrik Reif Andersen, An Introduction to Binary Decision Diagrams. *The IT University of Copenhagen, Fall 1999*
- Alessandro Artale, Formal Methods Lecture VI, Binary Decision Diagrams. http://www.inf.unibz.it/~artale/FM/slide7.pdf *(visited on 05.17.2017)*
- A. Pnueli, Symbolic Model Checking. http://www.cs.nyu.edu/courses/spring07/G22.3033-002/lecture6_h4.pdf *(visited on 05.17.2017)*