Hardware Verification 2IMF20

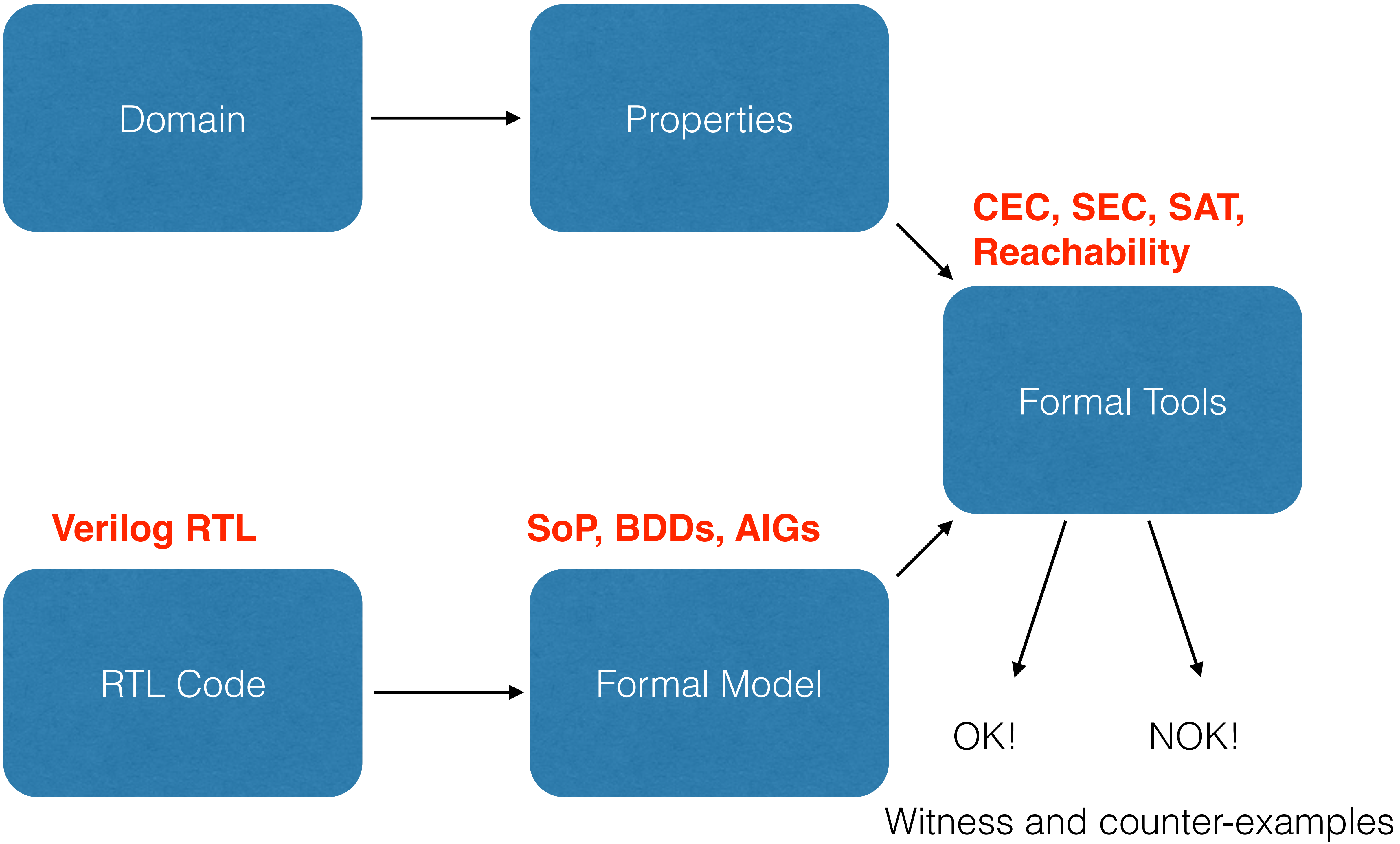
Julien Schmaltz

Lecture 03 (continued): Symbolic reachability with BDDs

Technische Universiteit Eindhoven
University of Technology

Where innovation starts

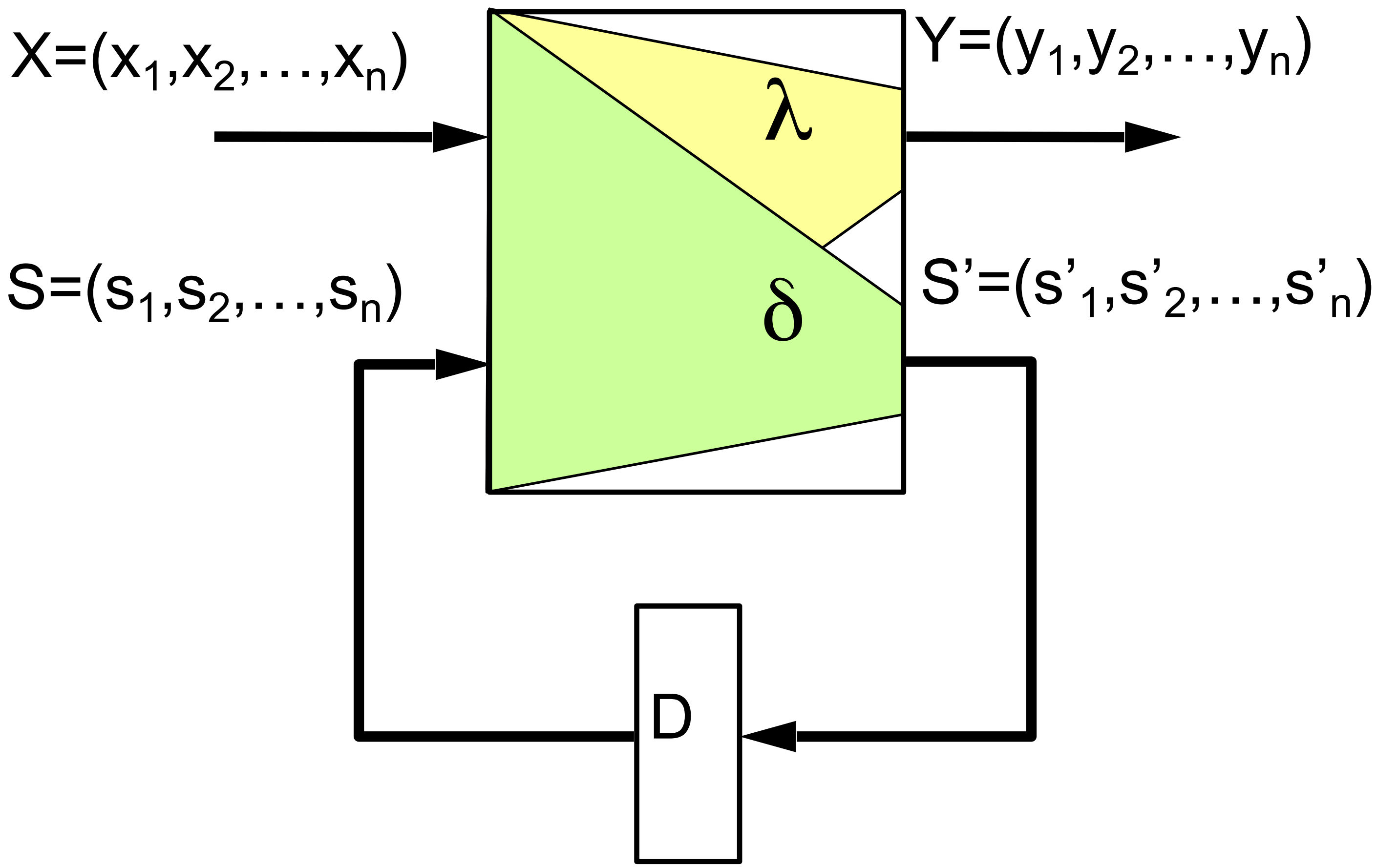
Course content - Covered so far



Previously …

- Sequential equivalence checking
- Reachability
  - forward/backward

Basic Model Finite State Machines



$M(X,Y,S,S_0,\delta,\lambda)$:

X: Inputs

Y: Outputs

S: Current State

S_0: Initial State(s)

δ: $X \times S \rightarrow S$ (next state function)

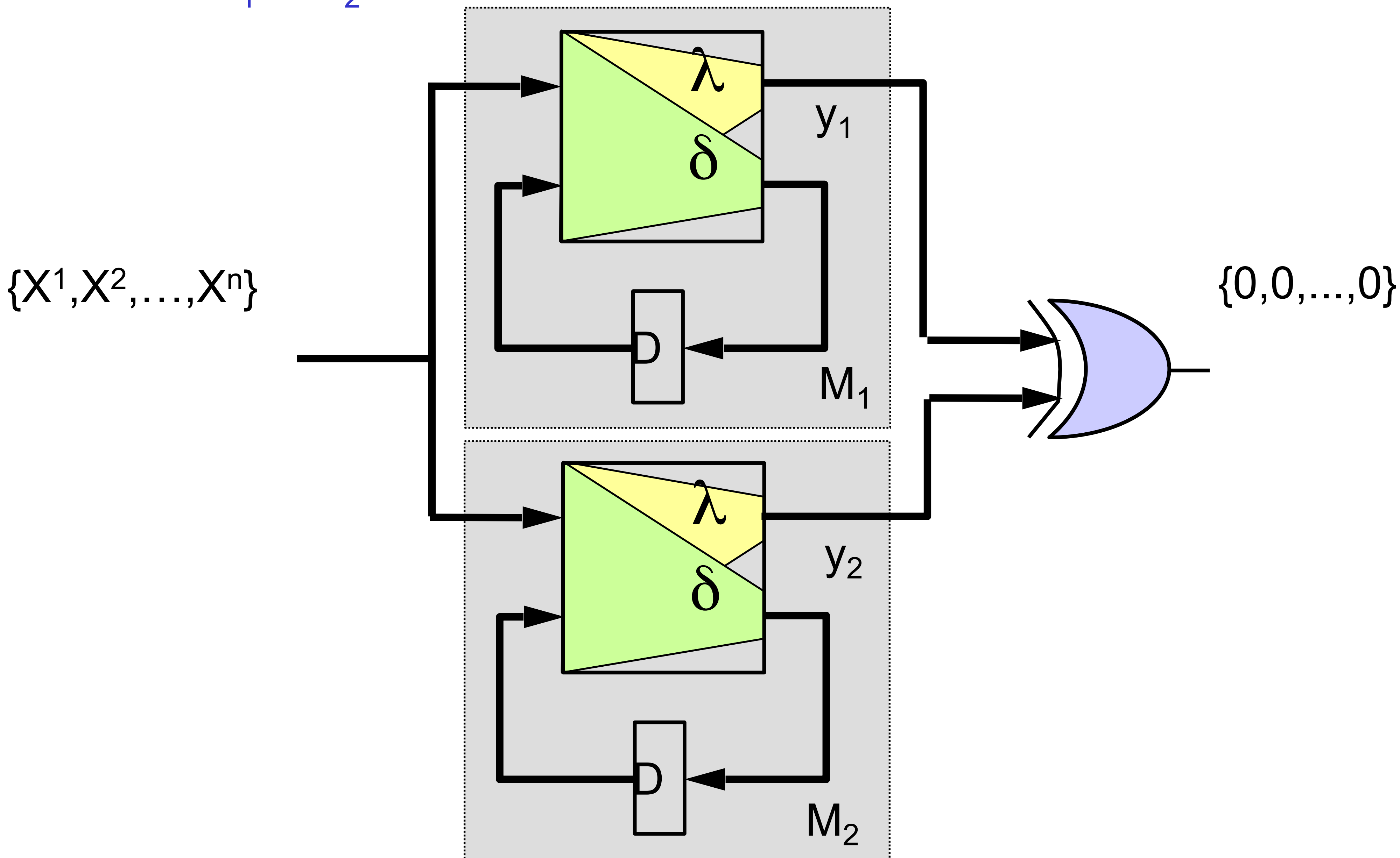
λ: $X \times S \rightarrow Y$ (output function)

Delay element(s):

- Clocked: synchronous
  - single-phase clock, multiple-phase clocks
- Unclocked: asynchronous

Finite State Machines Equivalence

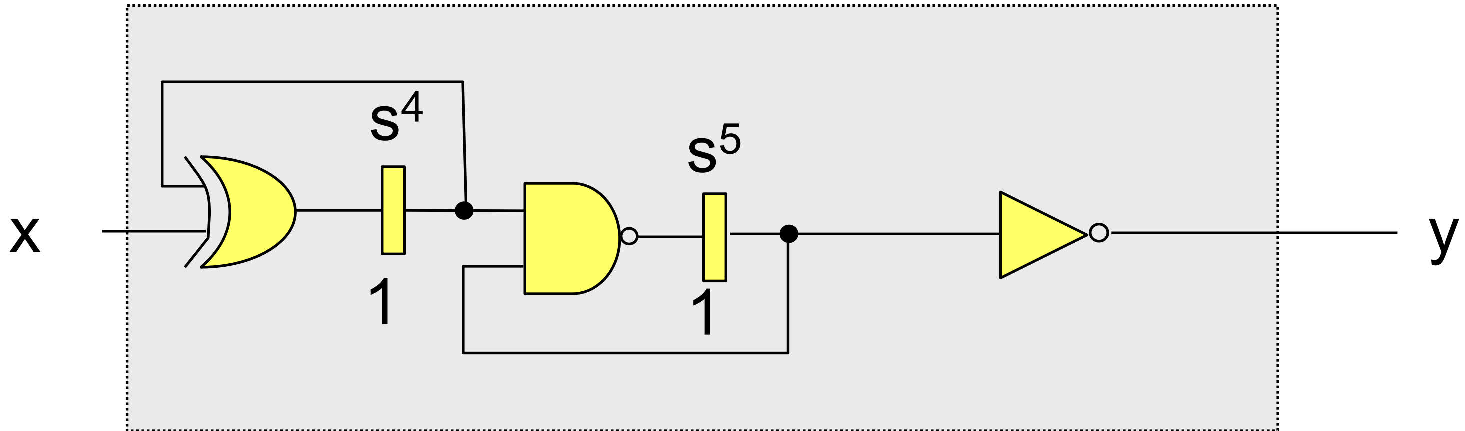
Build Product Machine $M_1 \times M_2$:



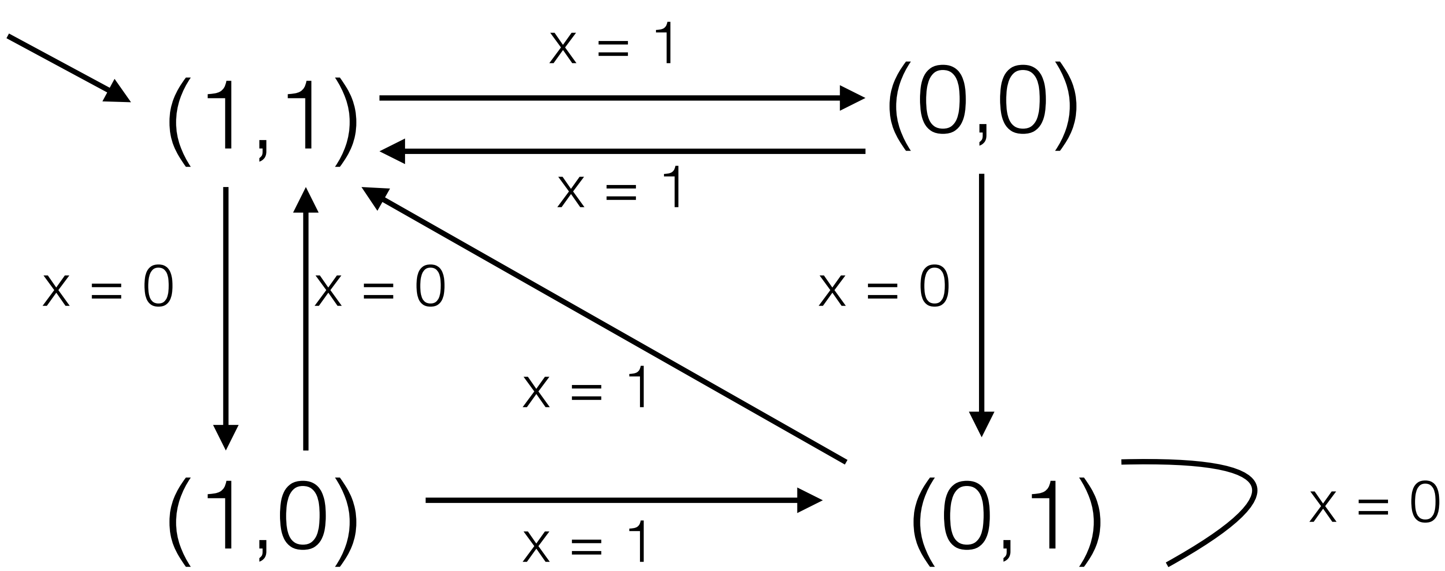
Definition:

M_1 and M_2 are functionally equivalent iff the product machine $M_1 \times M_2$ produces a constant 0 for all valid input sequences $\{X_1, \ldots, X_n\}$.

Bwd image - Example.

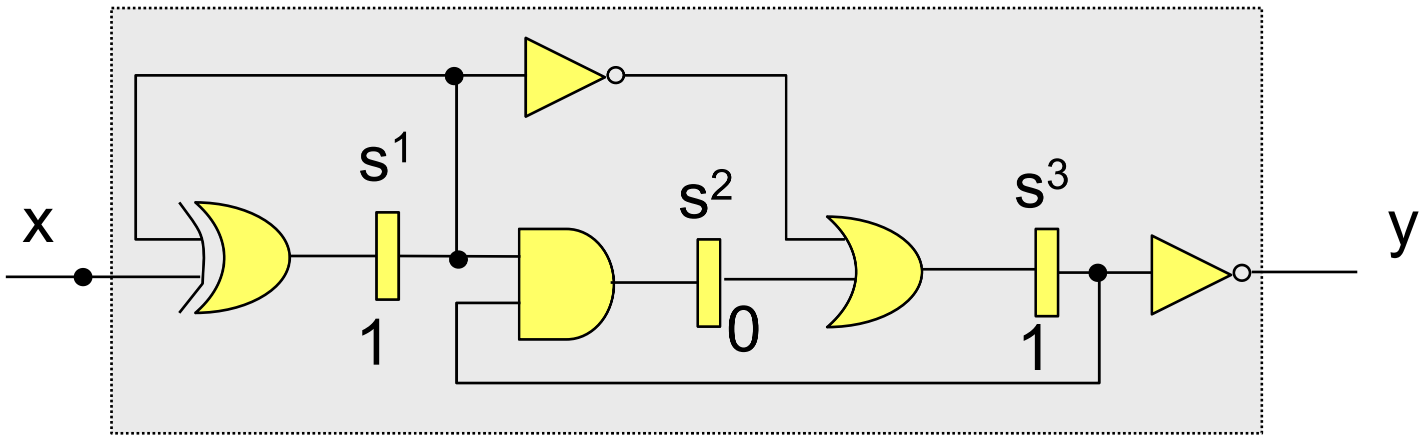


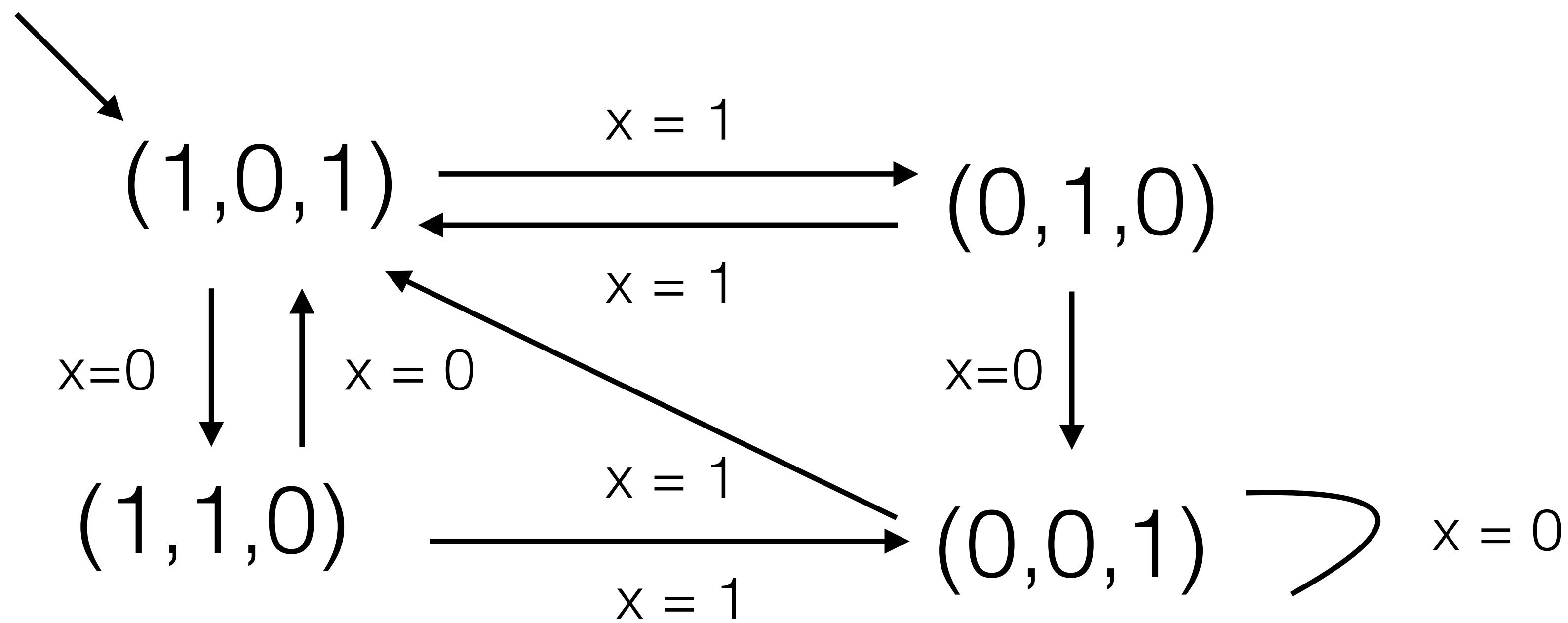
sequential circuit



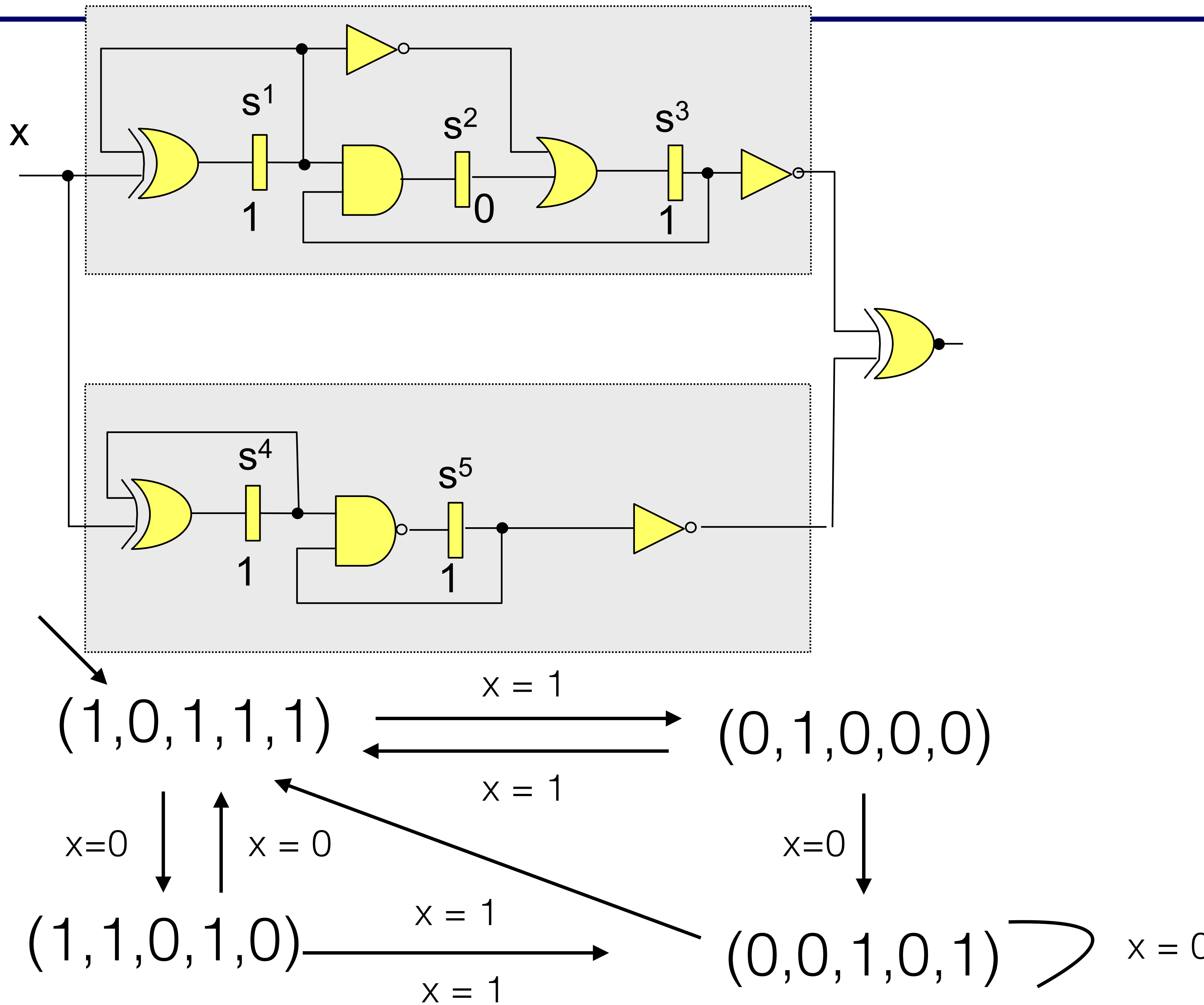
T = { (11,10),(11,00),(00,01),(00,11),
(10,11),(10,01),(01,01),(01,11)}

Another sequential circuit





Product machine states



Implementation of SEC

- ABC tool
  - pdr engine (Property Directed Reachability a.k.a. IC3)
  - dprove

- Short demo

Symbolic reachability

Let’s ook at BDDs once more

(RO)BDD's (Reduced Ordered) Binary Decision Diagrams

[Bryant 1986]

- Canonical form representation for Boolean functions
- Substantially more compact than CNF or DNF
- Efficient manipulation of BDD's

Shannon and Binary Decision Trees

- Shannon expansion for Boolean function f

$$f = (\neg a \wedge f|_{a=0}) \vee (a \wedge f|_{a=1})$$

- Using this expansion and a variable ordering, one can build a binary decision tree
- Binary Decision Trees are not very compact (same size as truth tables)

Bryant’s rules for ROBDD

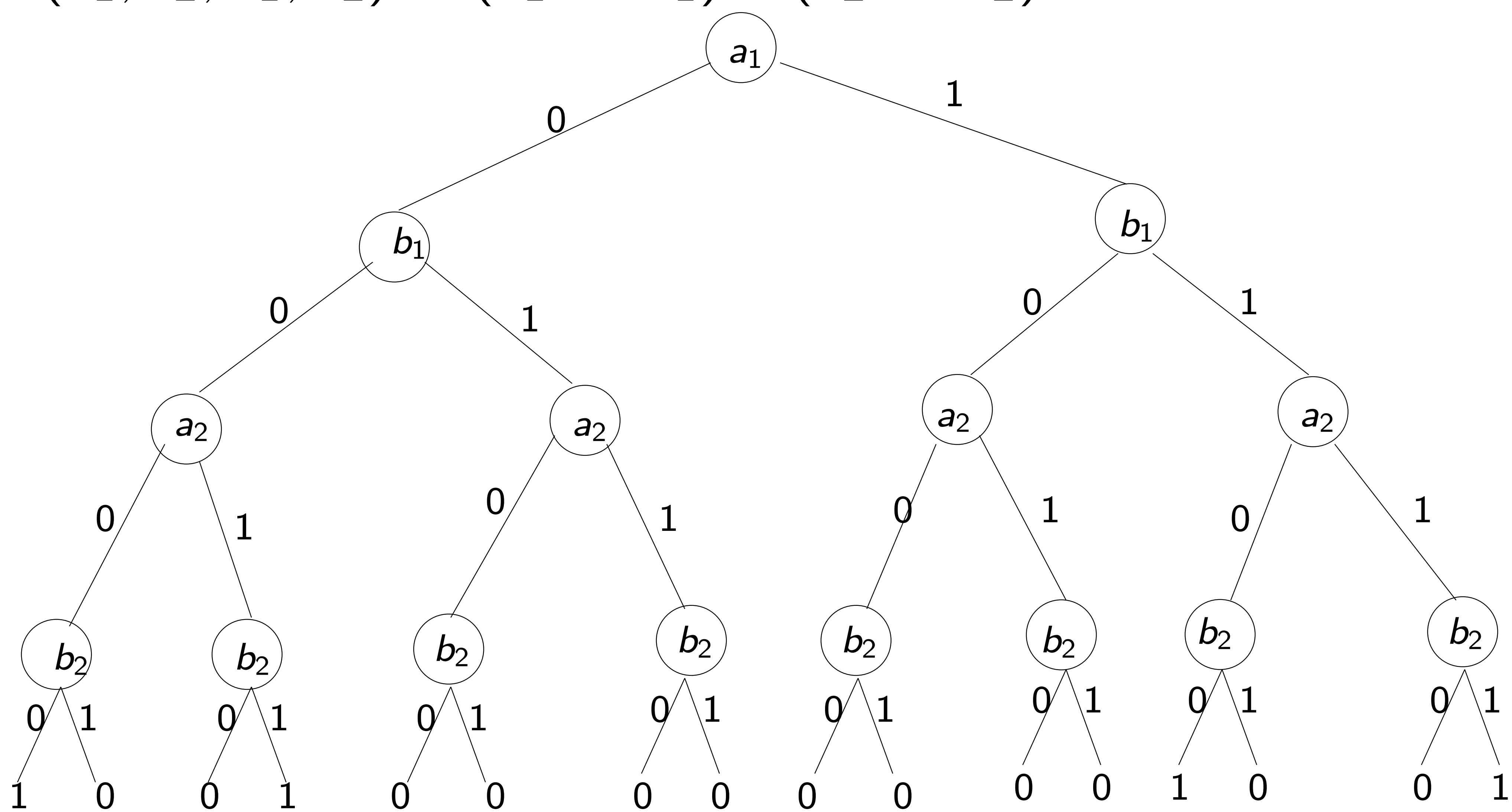
» (1) Remove duplicate terminals

» (2) Remove duplicate non-terminals

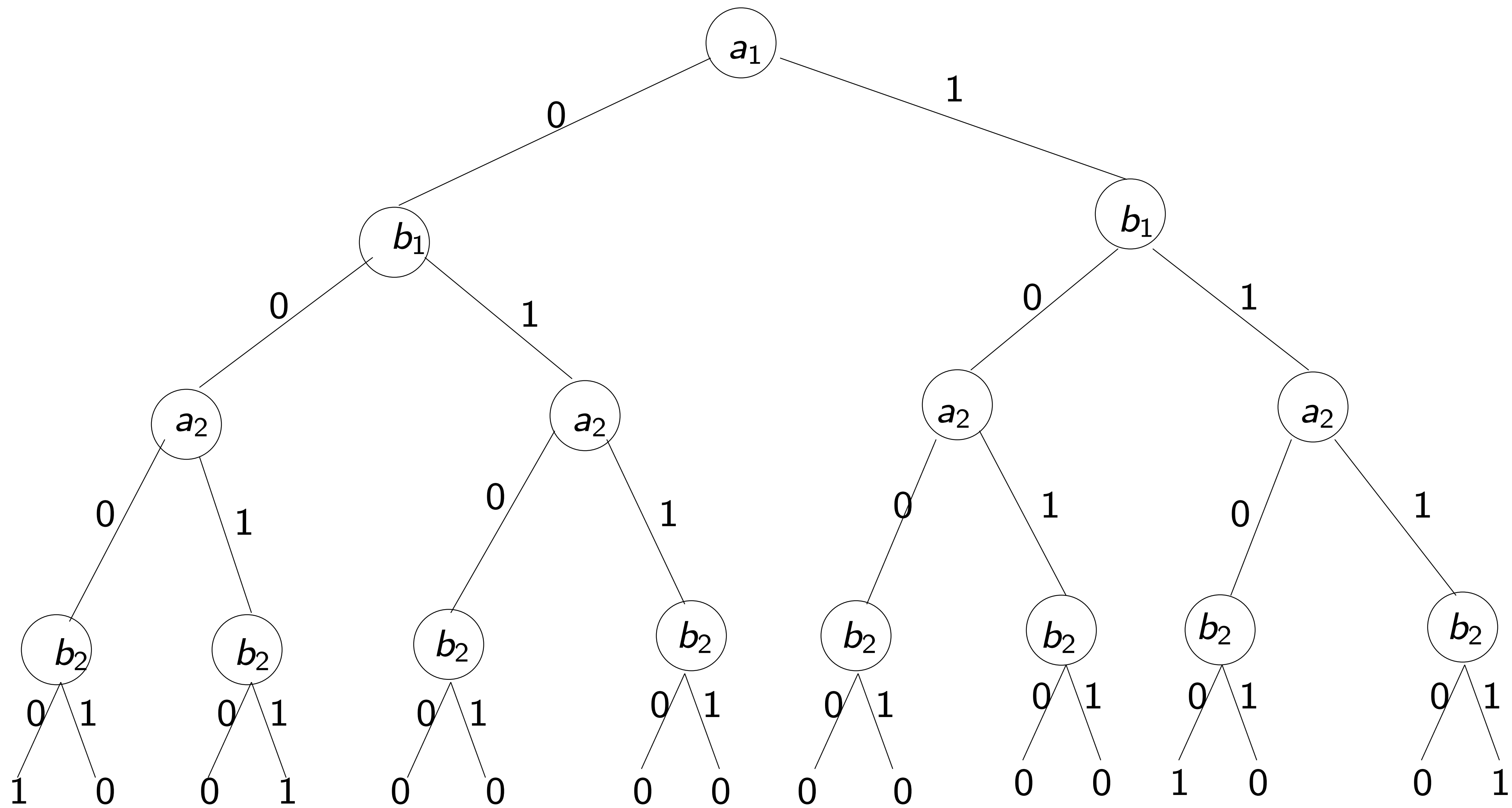
» (3) Remove redundant tests

Binary Decision Tree for a 2-bit comparator

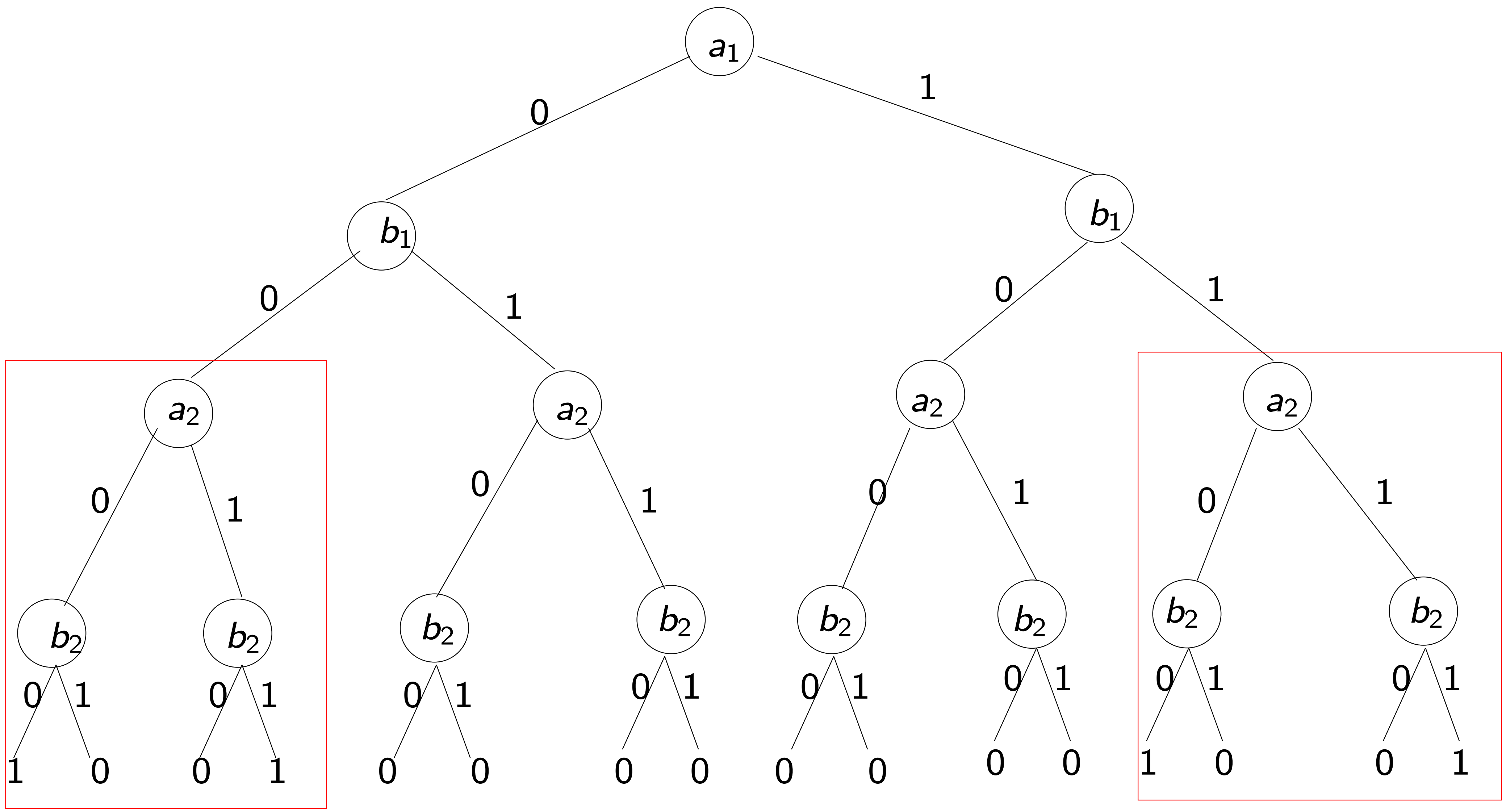
$$f(a_1, a_2, b_1, b_2) = (a_1 \Leftrightarrow b_1) \wedge (a_2 \Leftrightarrow b_2)$$



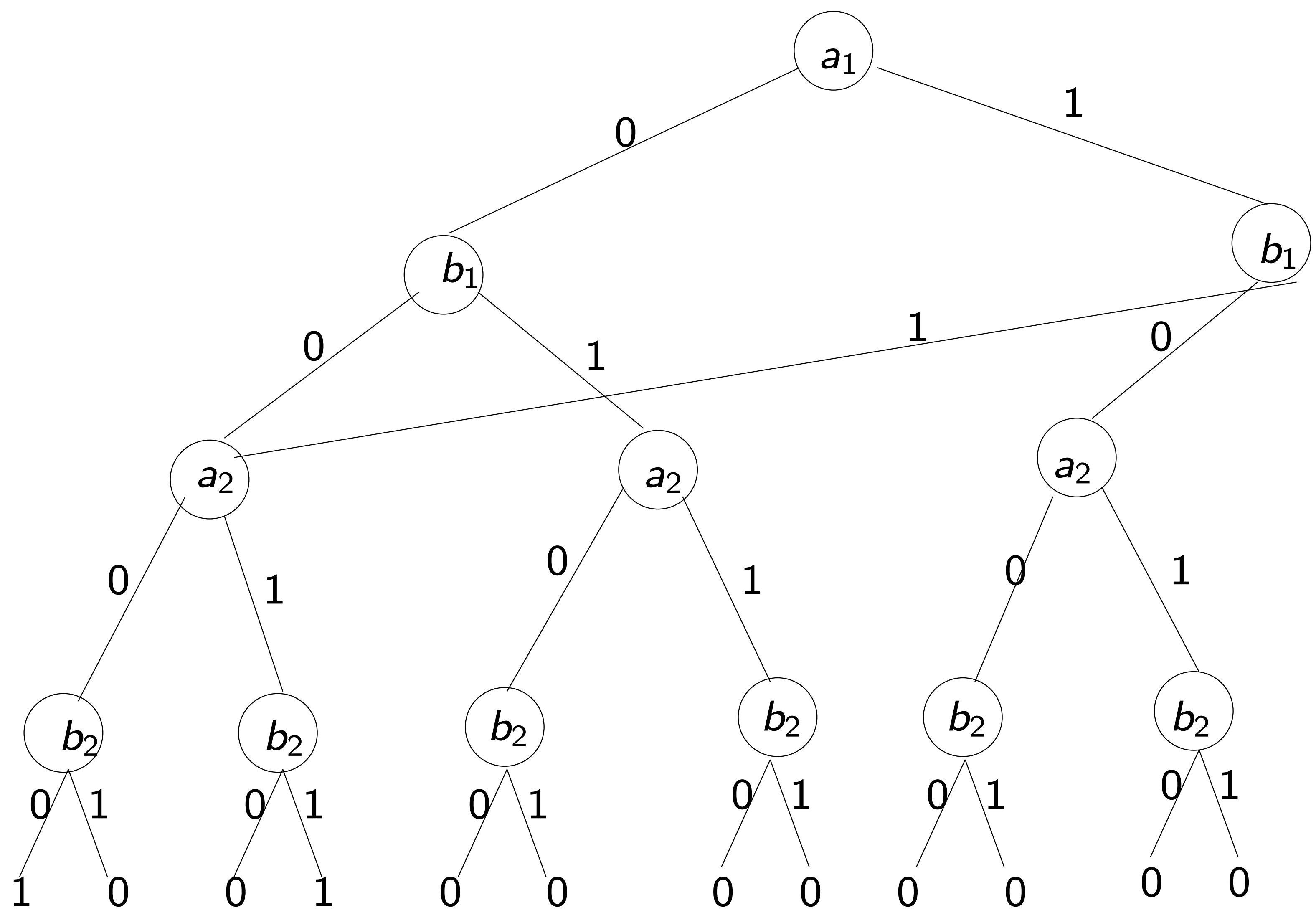
ROBDD for 2-bit comparator with ordering $a_1 < b_1 < a_2 < b_2$



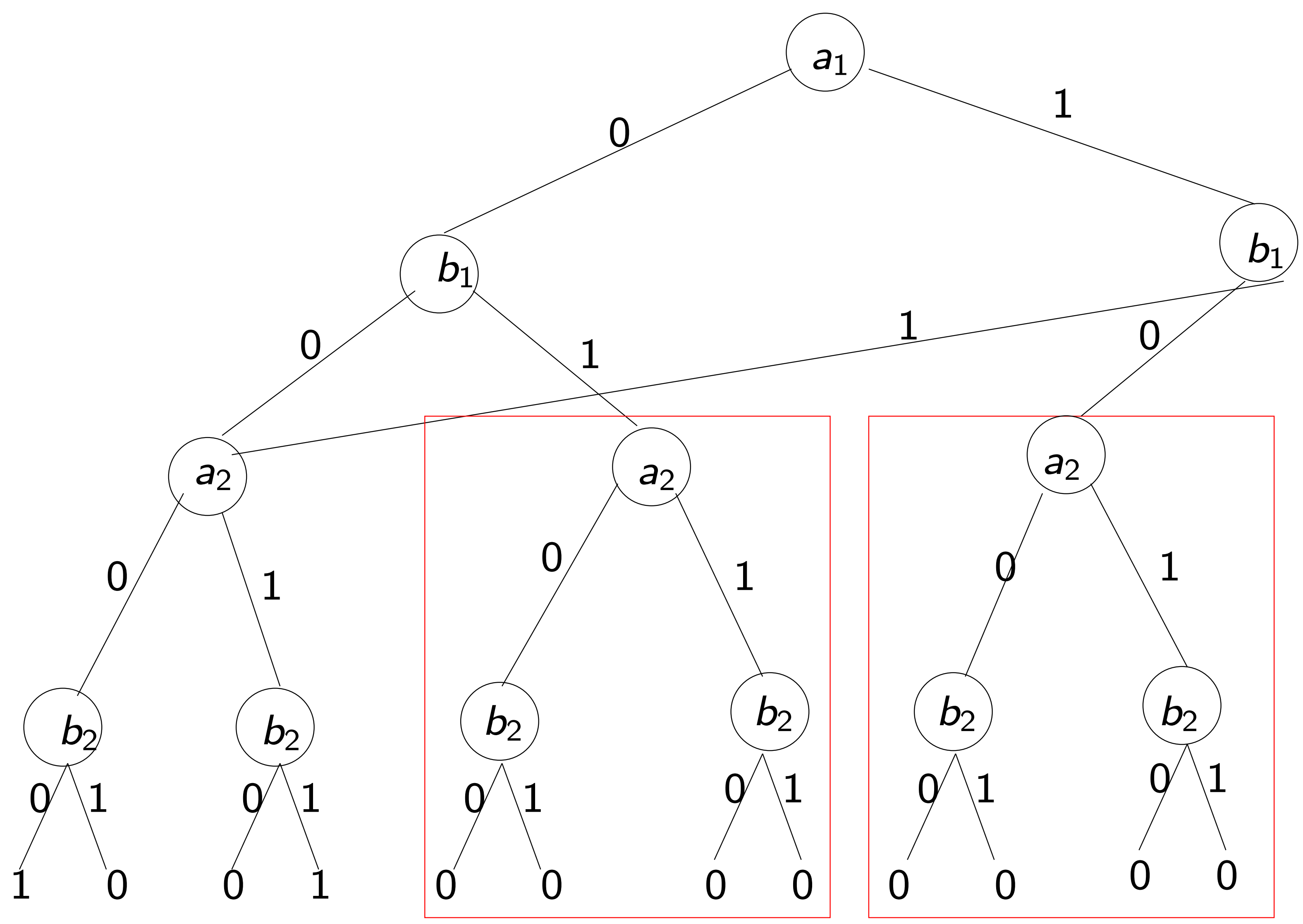
ROBDD for 2-bit comparator with ordering $a_1 < b_1 < a_2 < b_2$



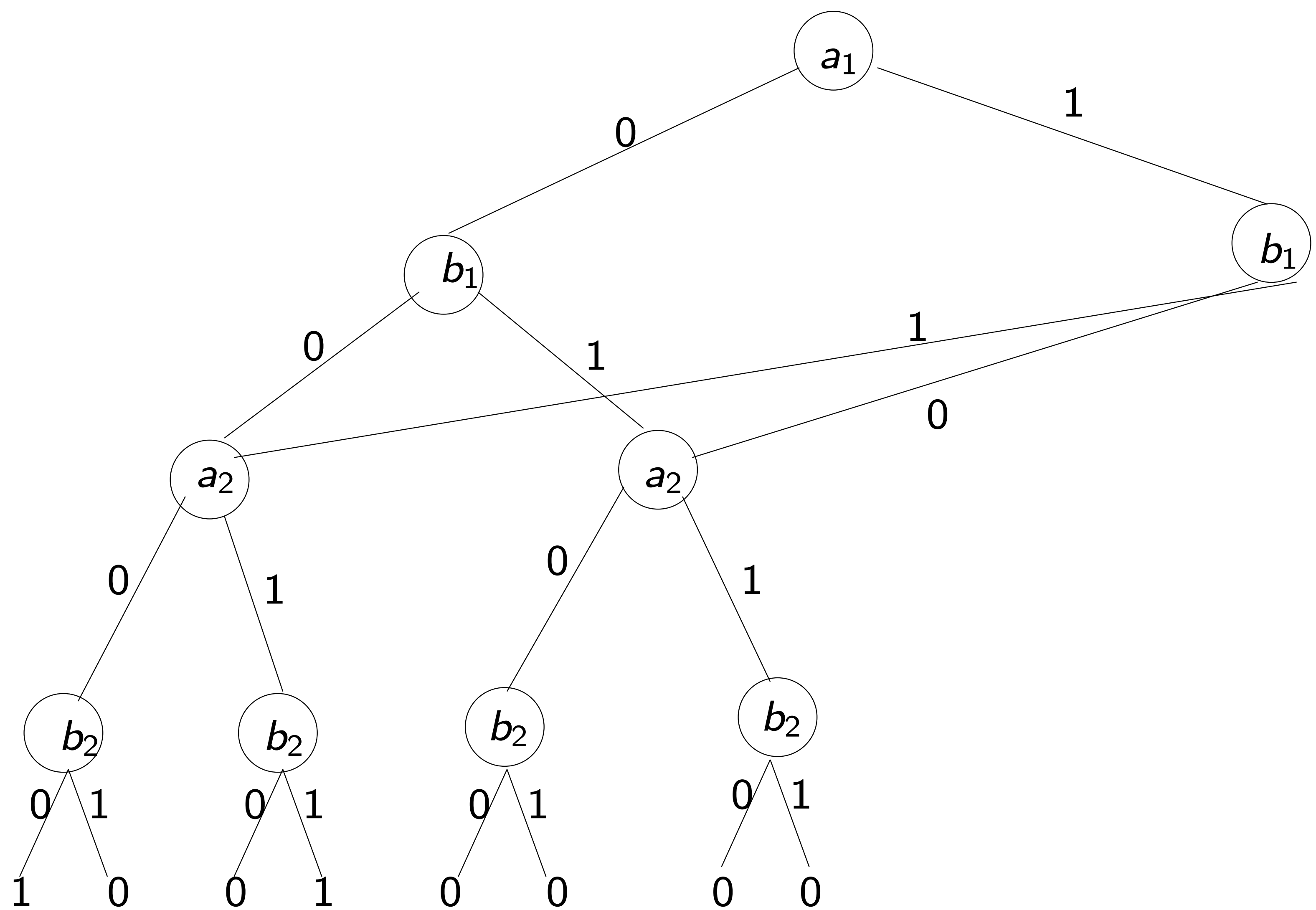
ROBDD for 2-bit comparator with ordering $a_1 < b_1 < a_2 < b_2$



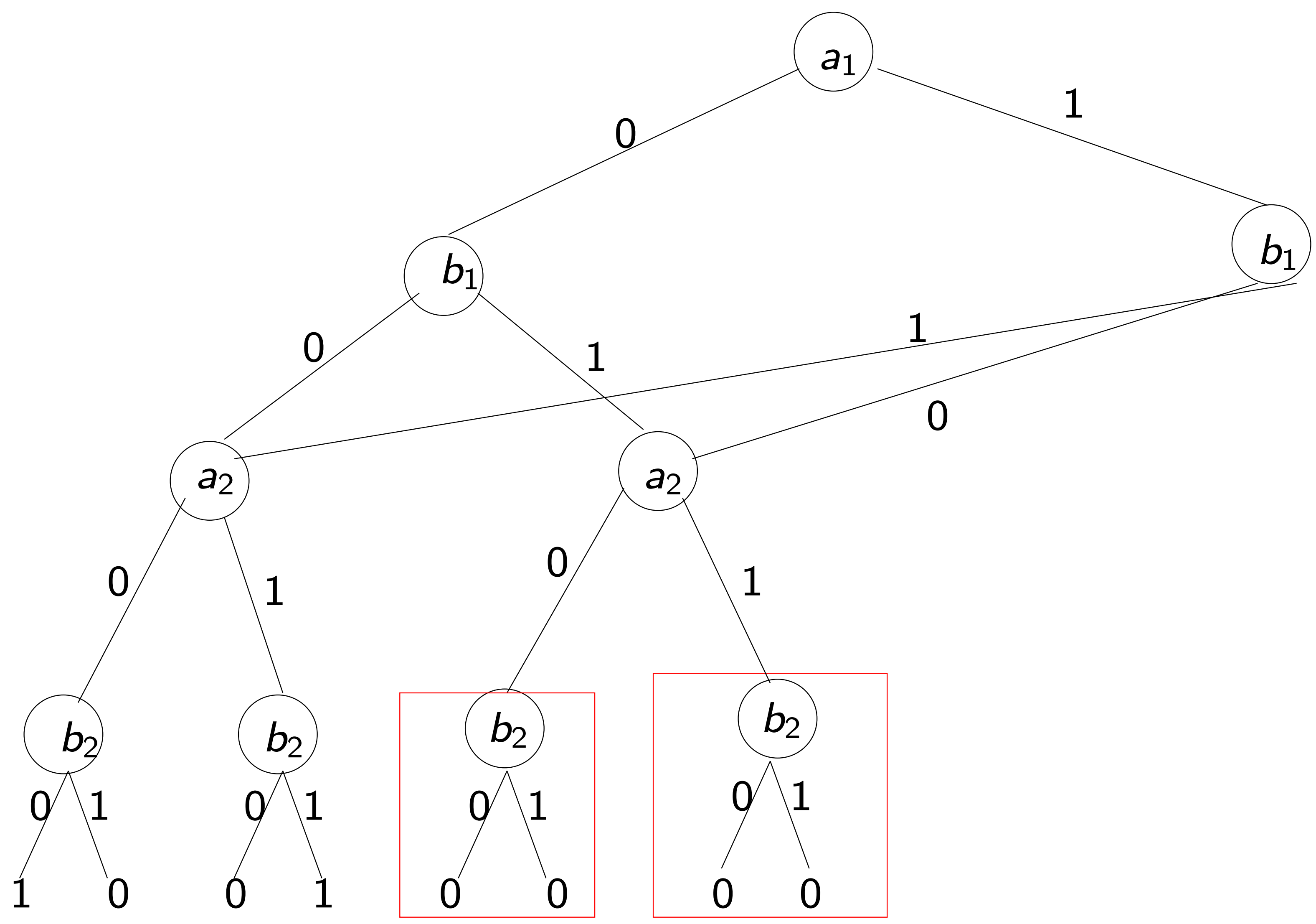
ROBDD for 2-bit comparator with ordering $a_1 < b_1 < a_2 < b_2$



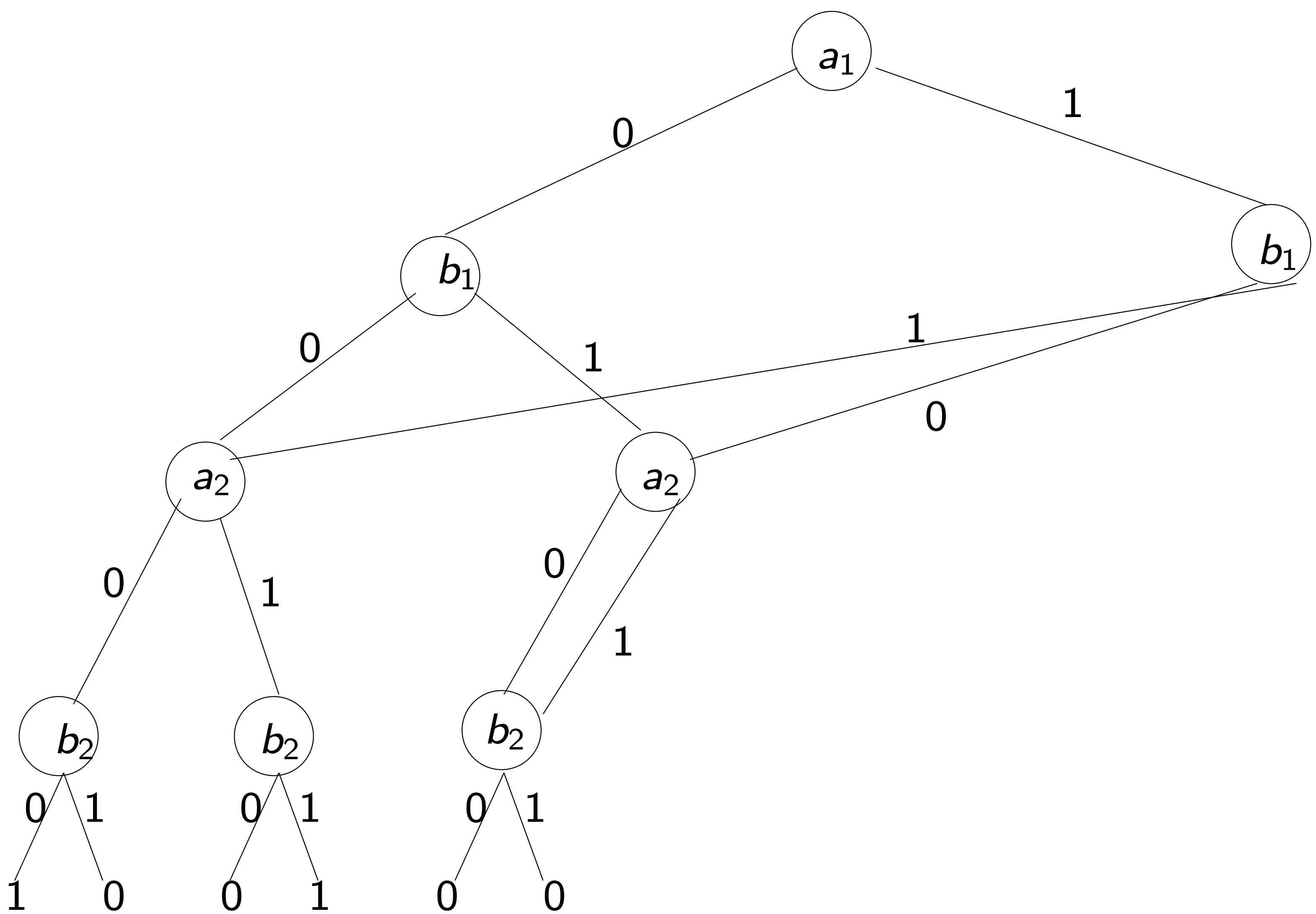
ROBDD for 2-bit comparator with ordering $a_1 < b_1 < a_2 < b_2$



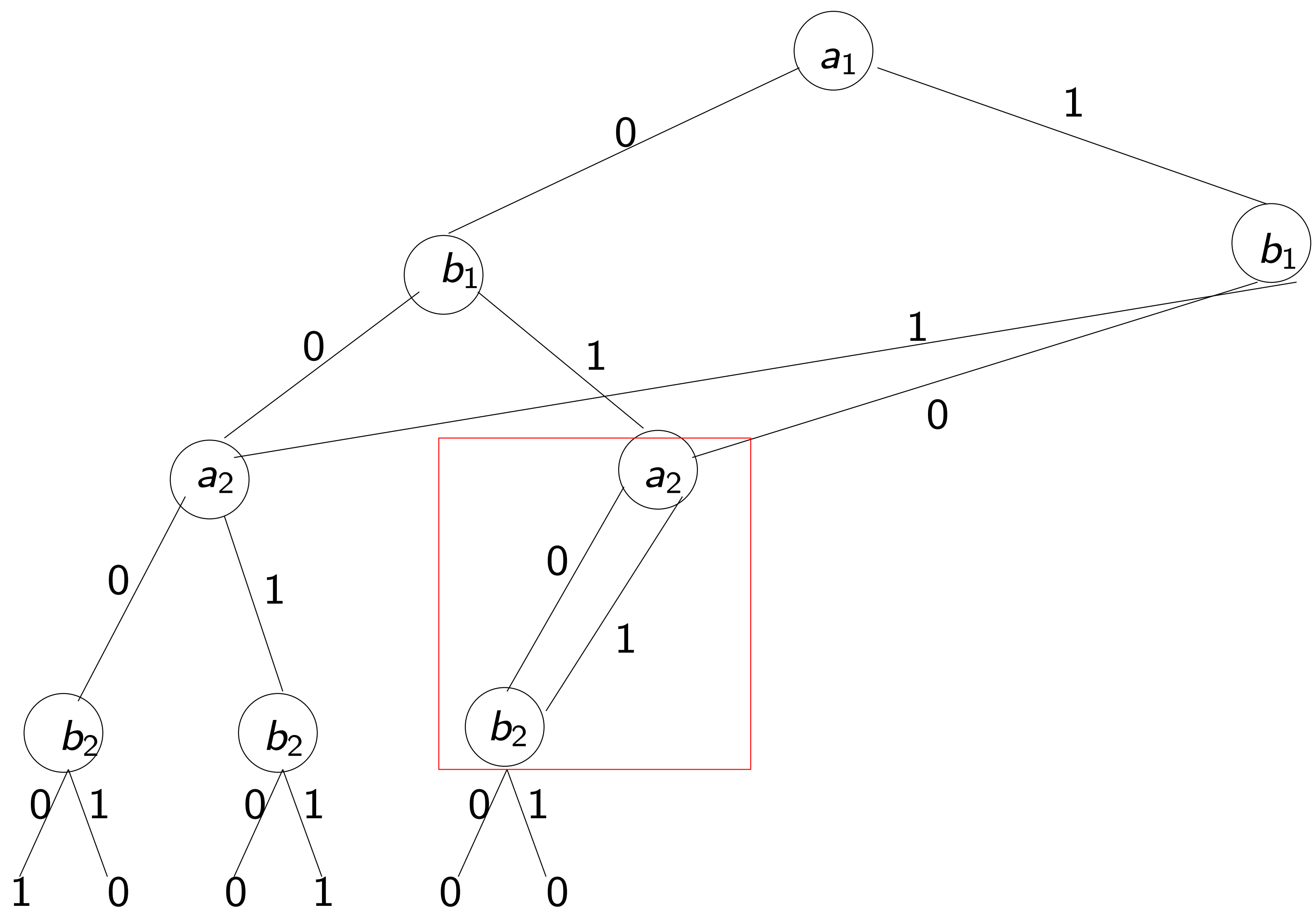
ROBDD for 2-bit comparator with ordering $a_1 < b_1 < a_2 < b_2$



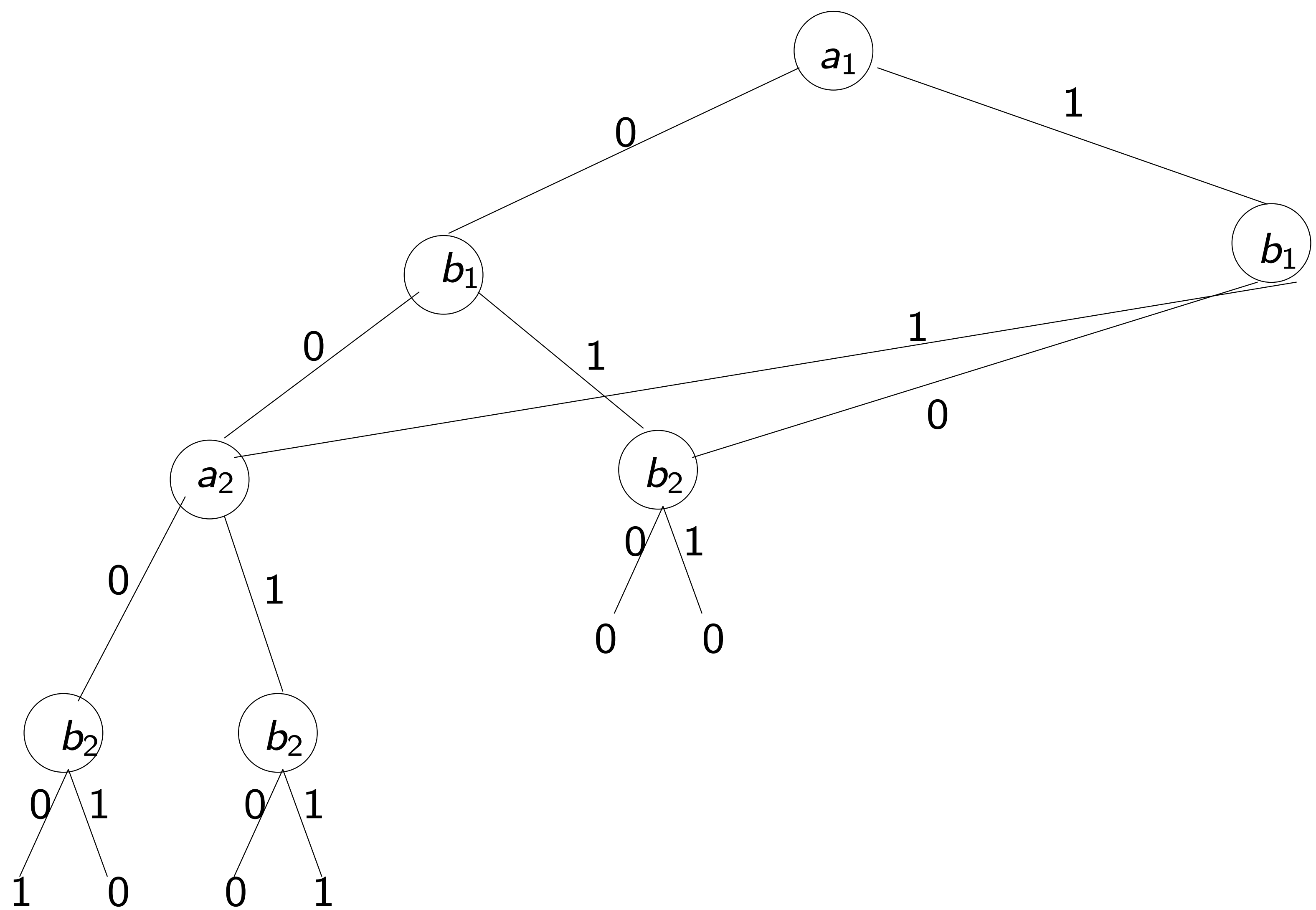
ROBDD for 2-bit comparator with ordering $a_1 < b_1 < a_2 < b_2$



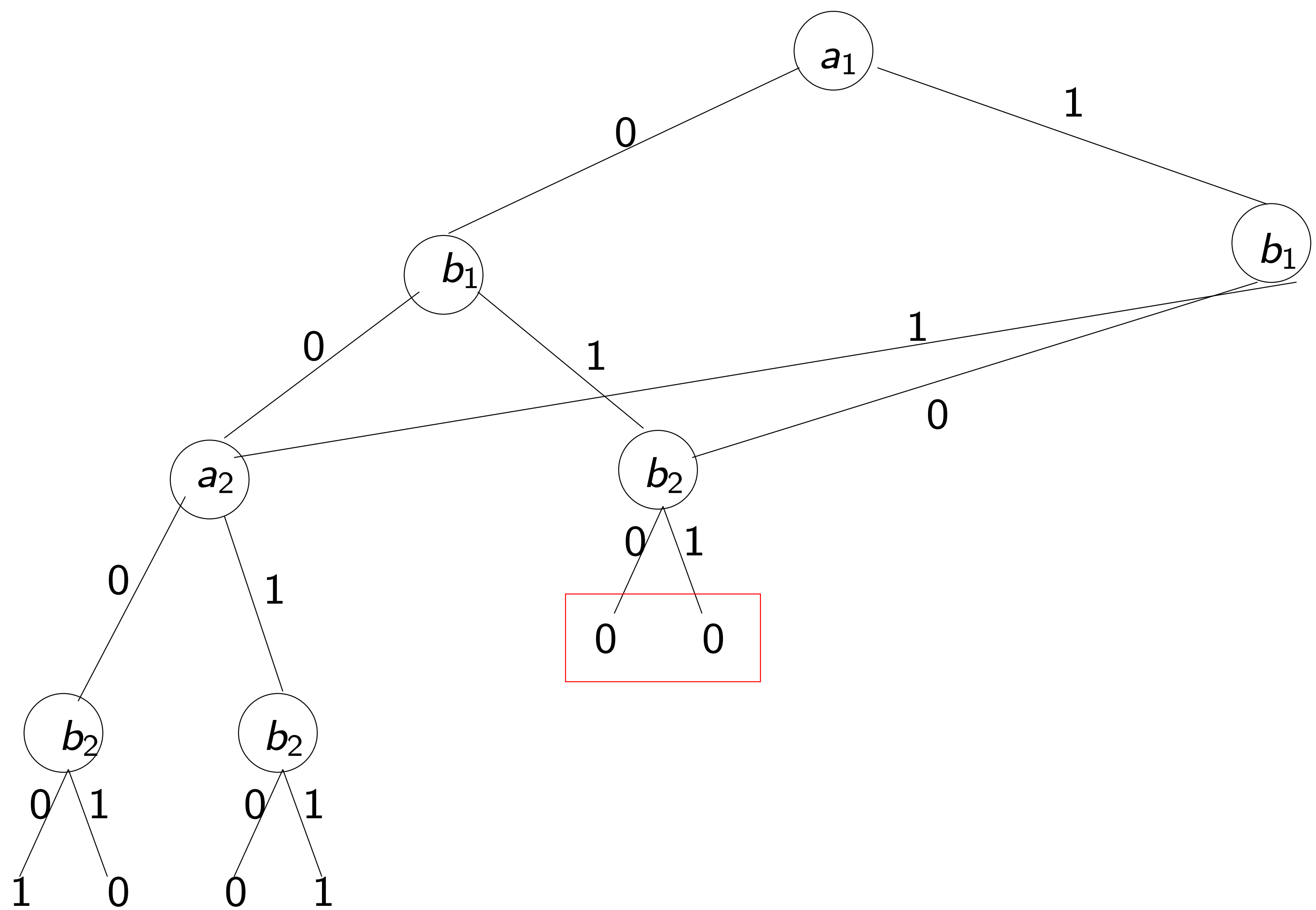
ROBDD for 2-bit comparator with ordering $a_1 < b_1 < a_2 < b_2$



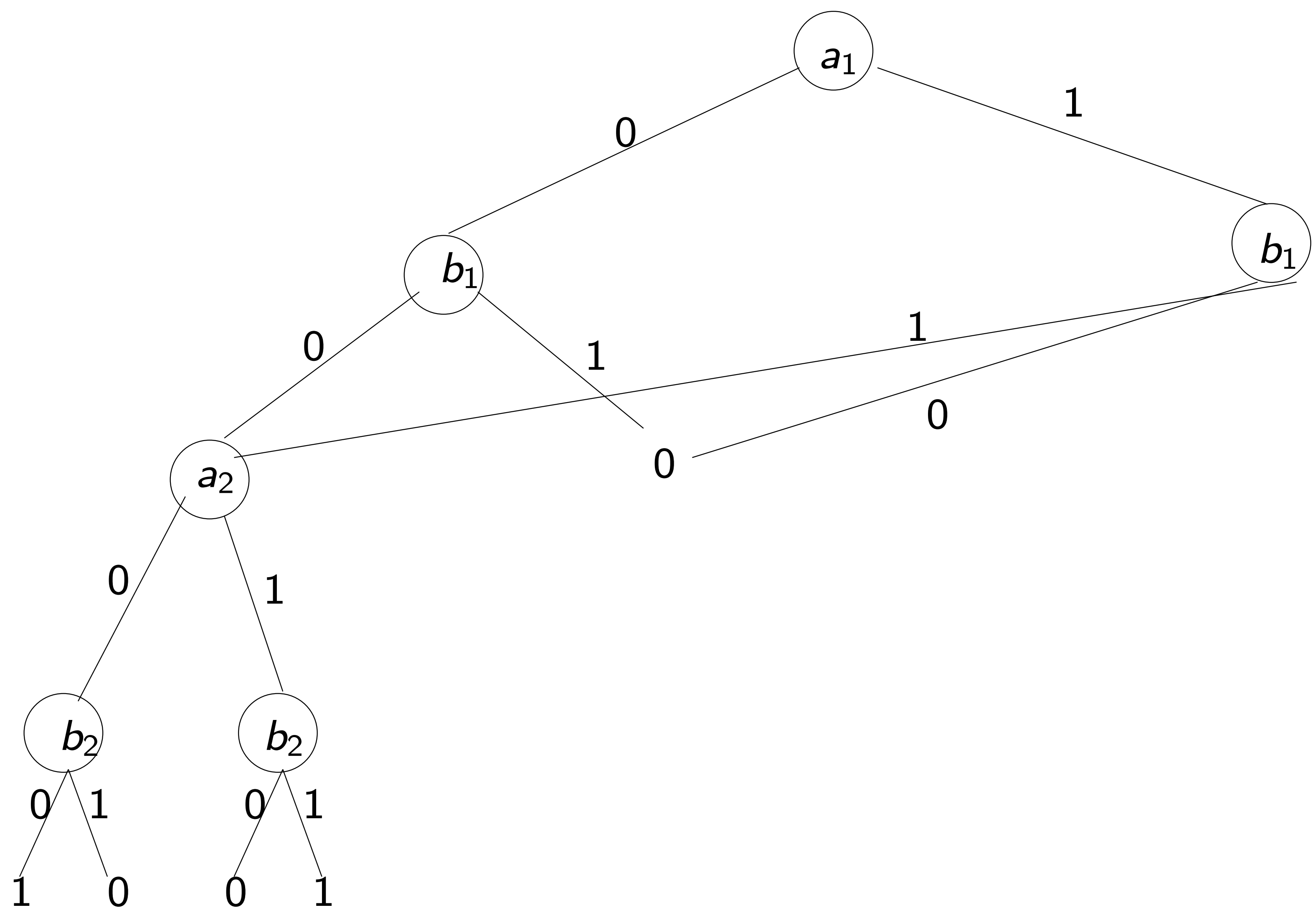
ROBDD for 2-bit comparator with ordering $a_1 < b_1 < a_2 < b_2$



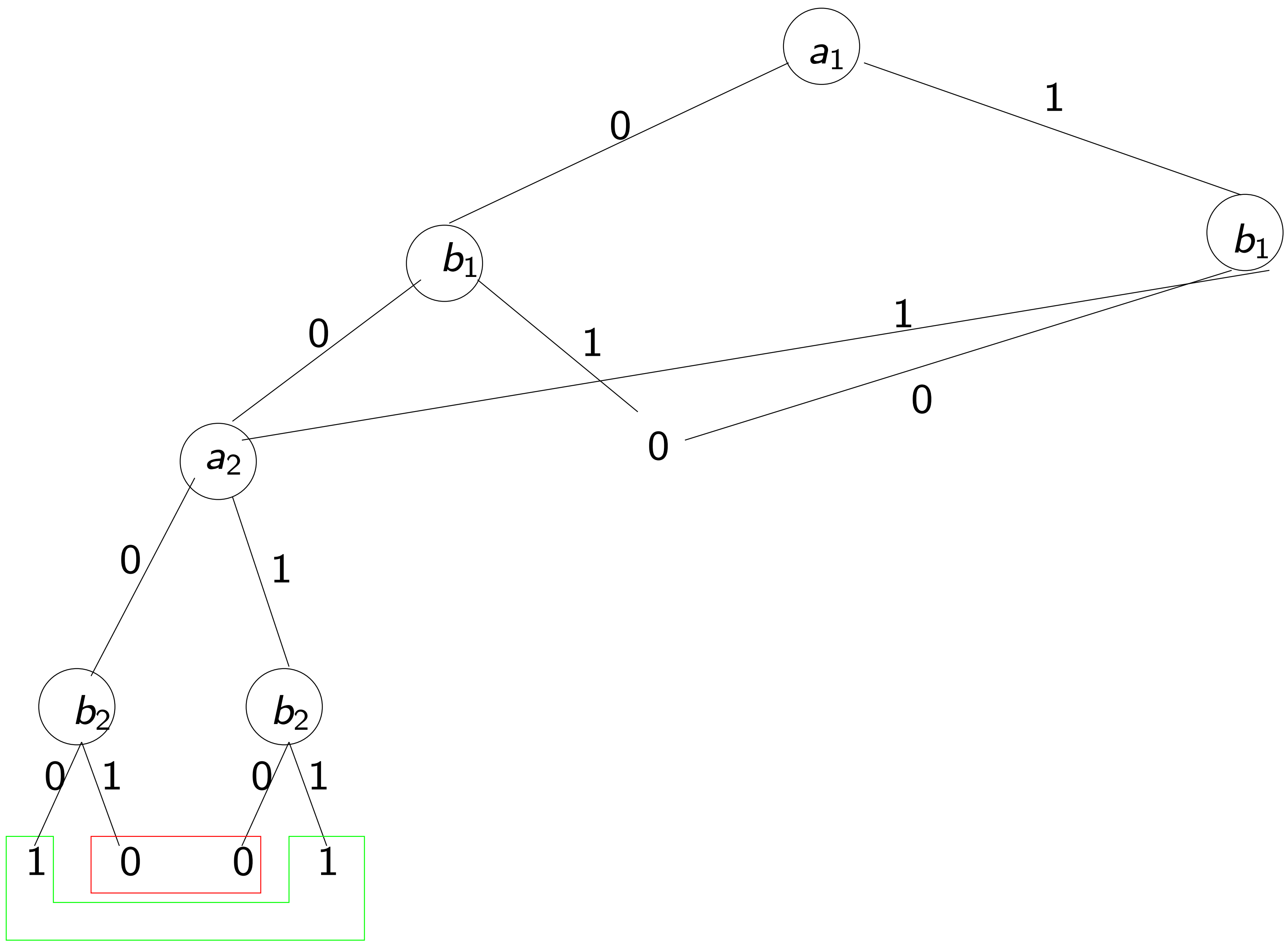
ROBDD for 2-bit comparator with ordering $a_1 < b_1 < a_2 < b_2$



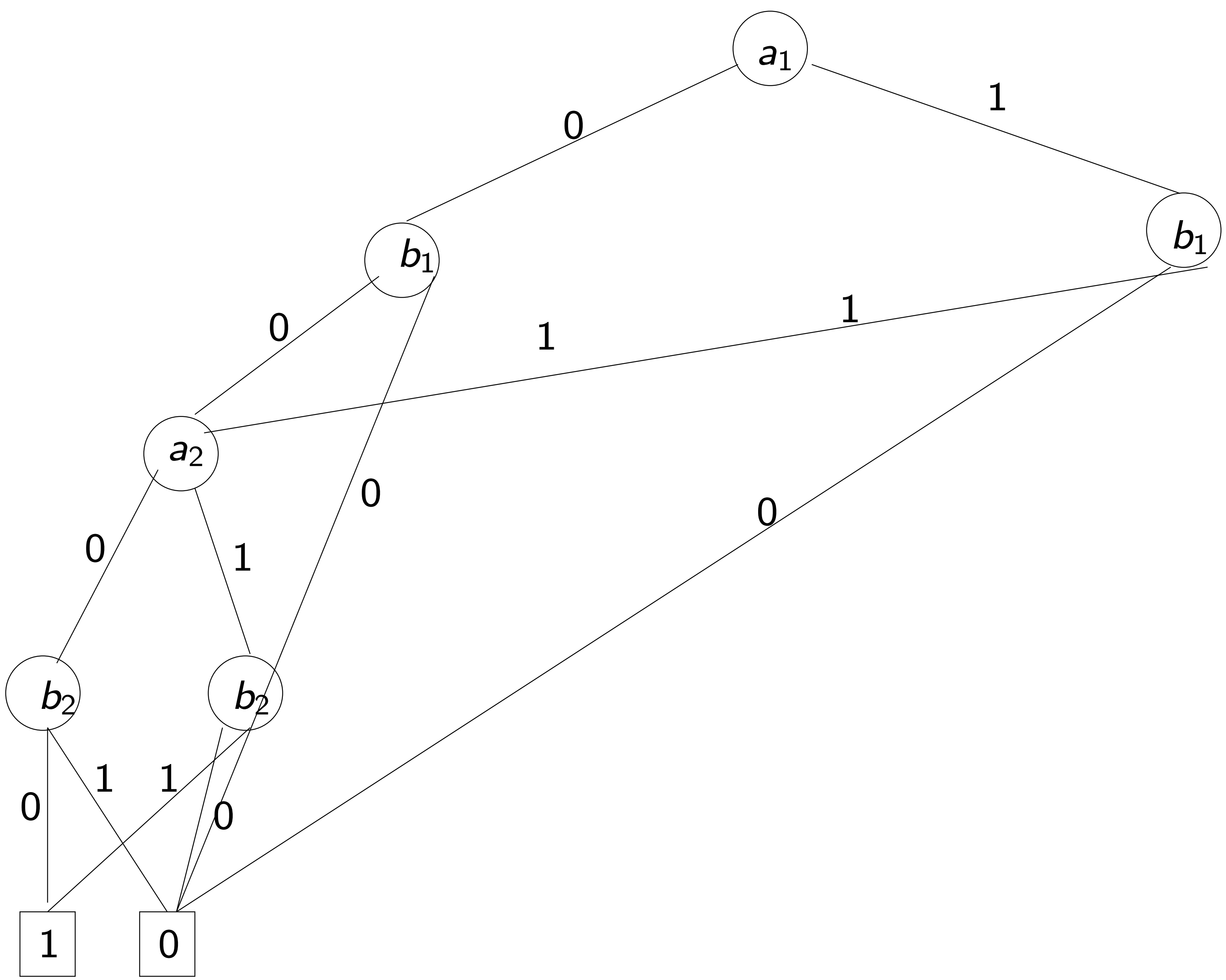
ROBDD for 2-bit comparator with ordering $a_1 < b_1 < a_2 < b_2$



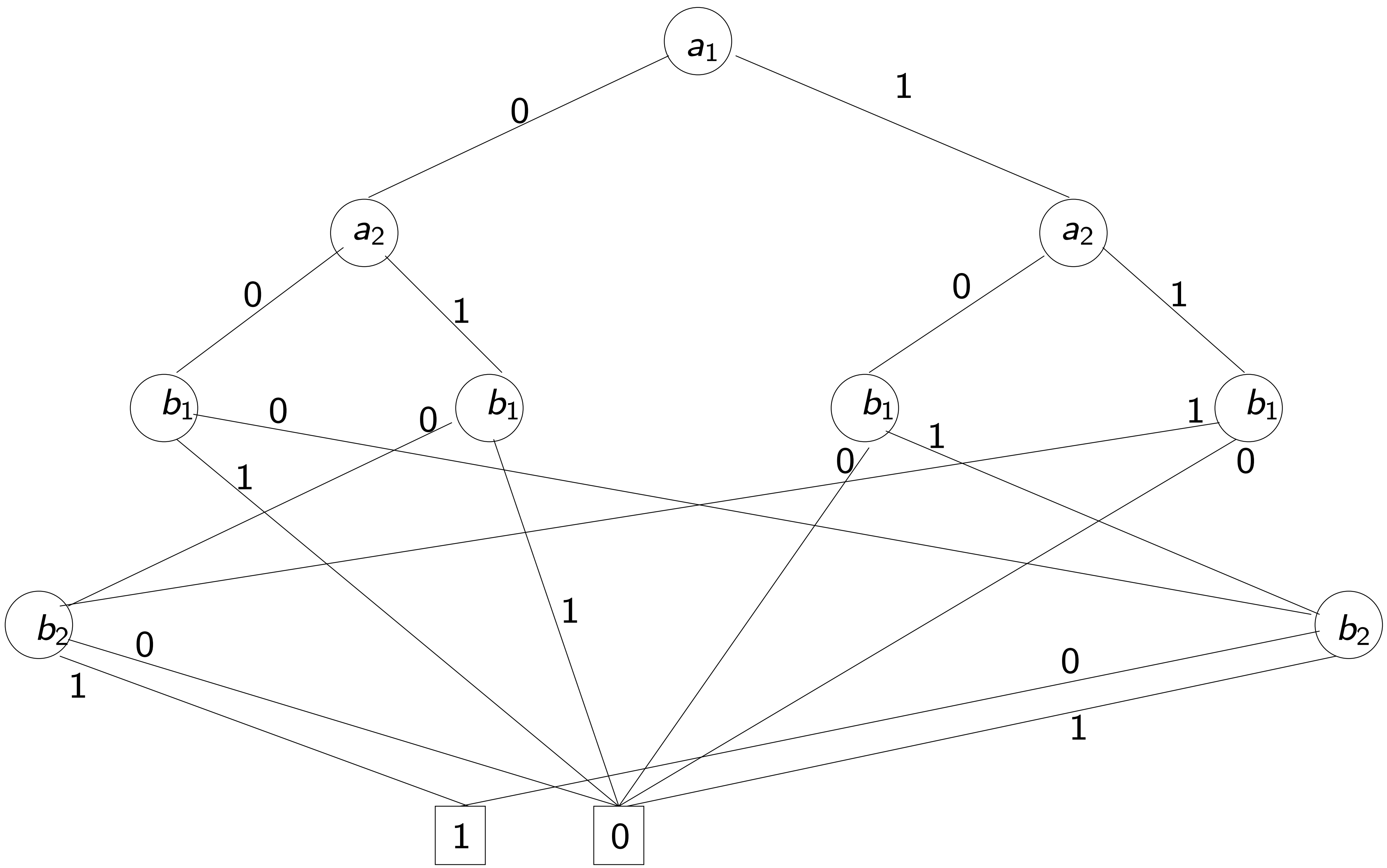
ROBDD for 2-bit comparator with ordering $a_1 < b_1 < a_2 < b_2$



ROBDD for 2-bit comparator with ordering $a_1 < b_1 < a_2 < b_2$



ROBDD for 2-bit comparator with ordering $a_1 < a_2 < b_1 < b_2$



Logical operations on ROBDD's (1)

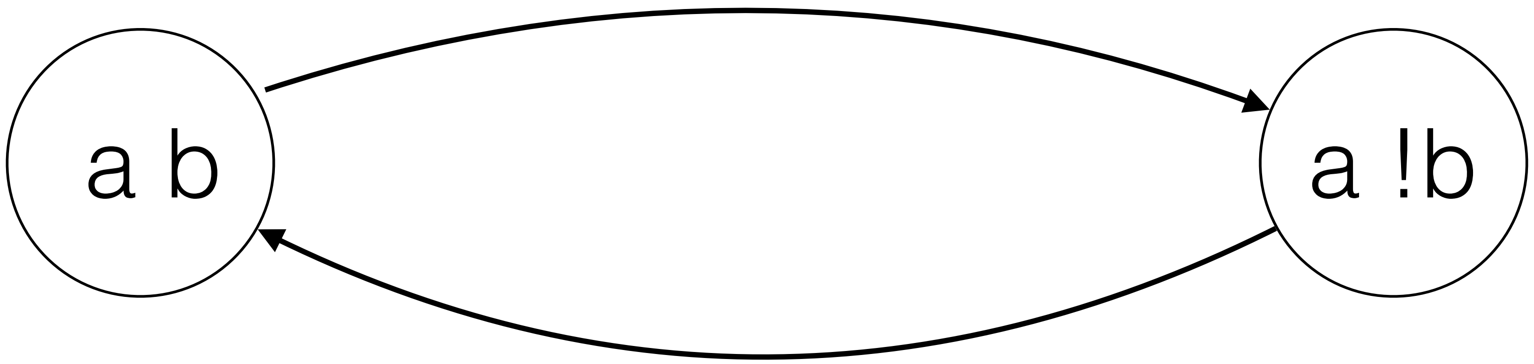
- Logical negation $\neg f(a, b, c, d)$

  Replace each leaf by its negation
- Logical conjunction $f(a, b, c, d) \wedge g(a, b, c, d)$
  - Use Shannon's expansion as follows

    $$f \wedge g = \neg a \wedge (f|_{\neg a} \wedge g|_{\neg a}) \vee a \wedge (f\ |_{a} \wedge g\ |_{a})$$

    to break the problem into two sub-problems. Solve sub-problems recursively.
  - Always combine isomorphic subtrees and eliminate redundant nodes
  - Hash tables stores previously computed sub-problems
  - Number of sub-problems bounded by $|f| \cdot |g|$

Simple example

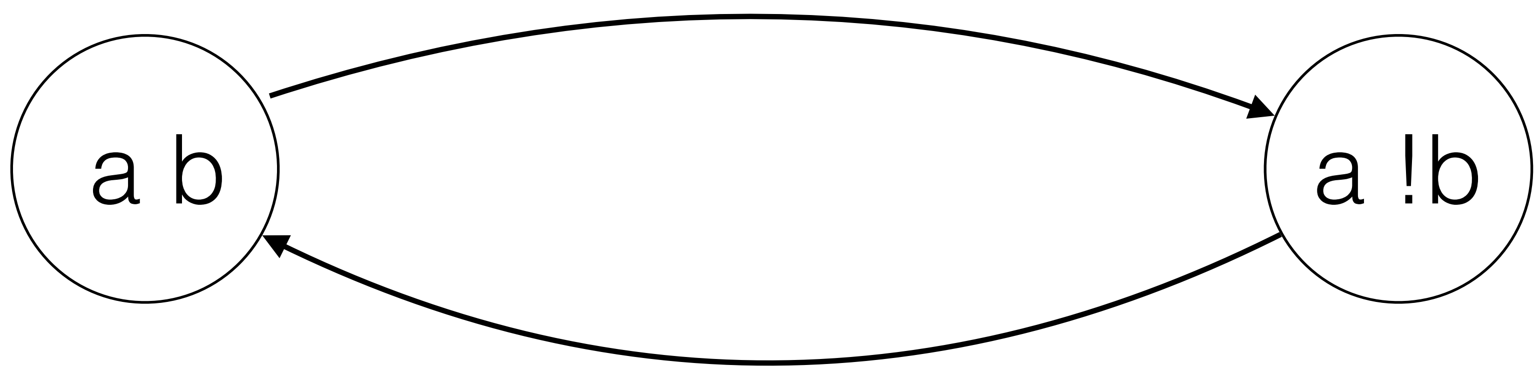


Transition relation as characteristic function

T(a,b,a'',b') = (a & !b & a' & b') | (a & b & a' & !b')

Represent as a ROBDD!

Do it on the black board ?



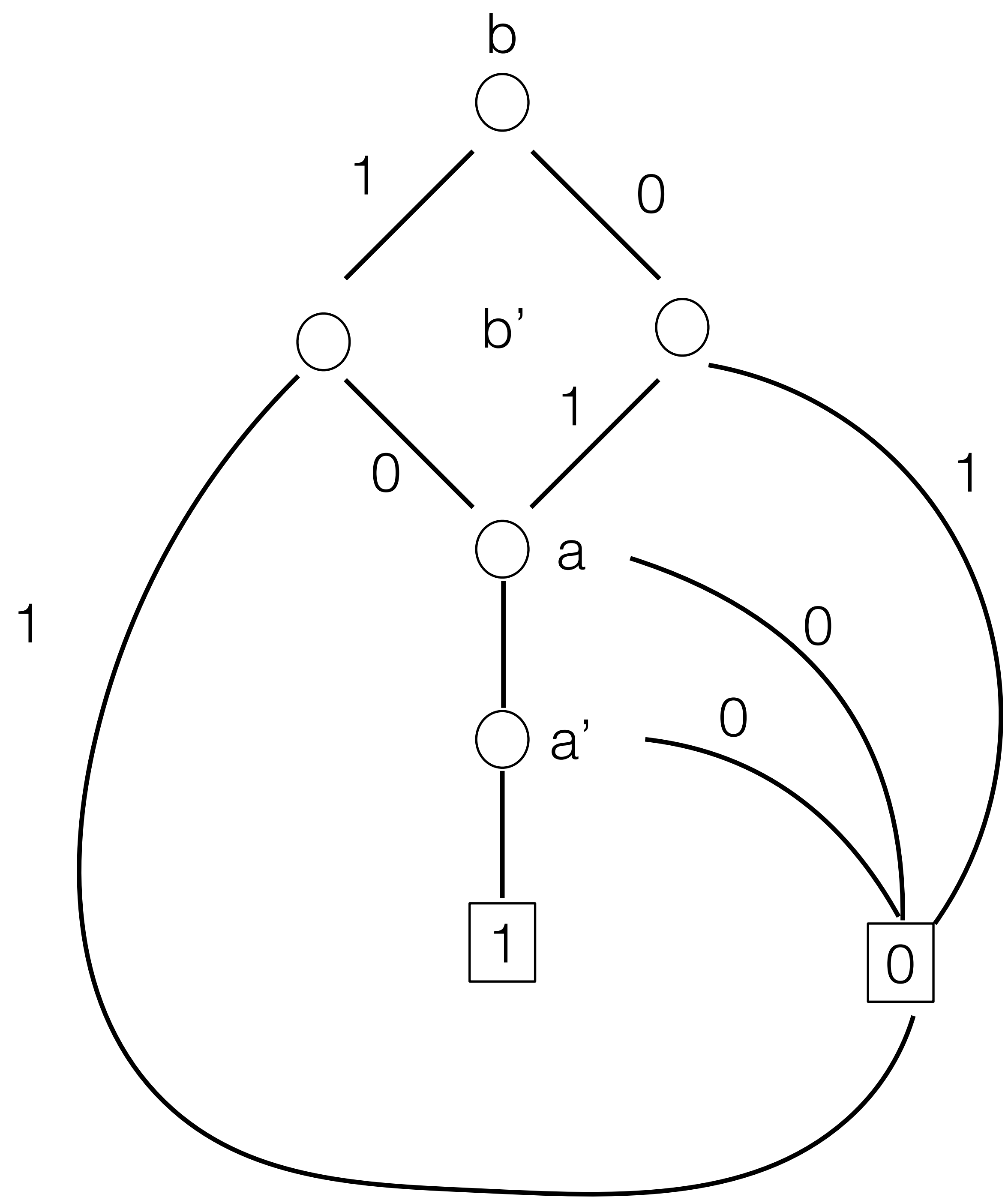
Transition relation as characteristic function

T(a,b,a’’,b’) = (a & !b & a’ & b’) | (a & b & a’ & !b’)

Represent as a ROBDD!

ROBDD for the example

Ordering = b b’ a a’



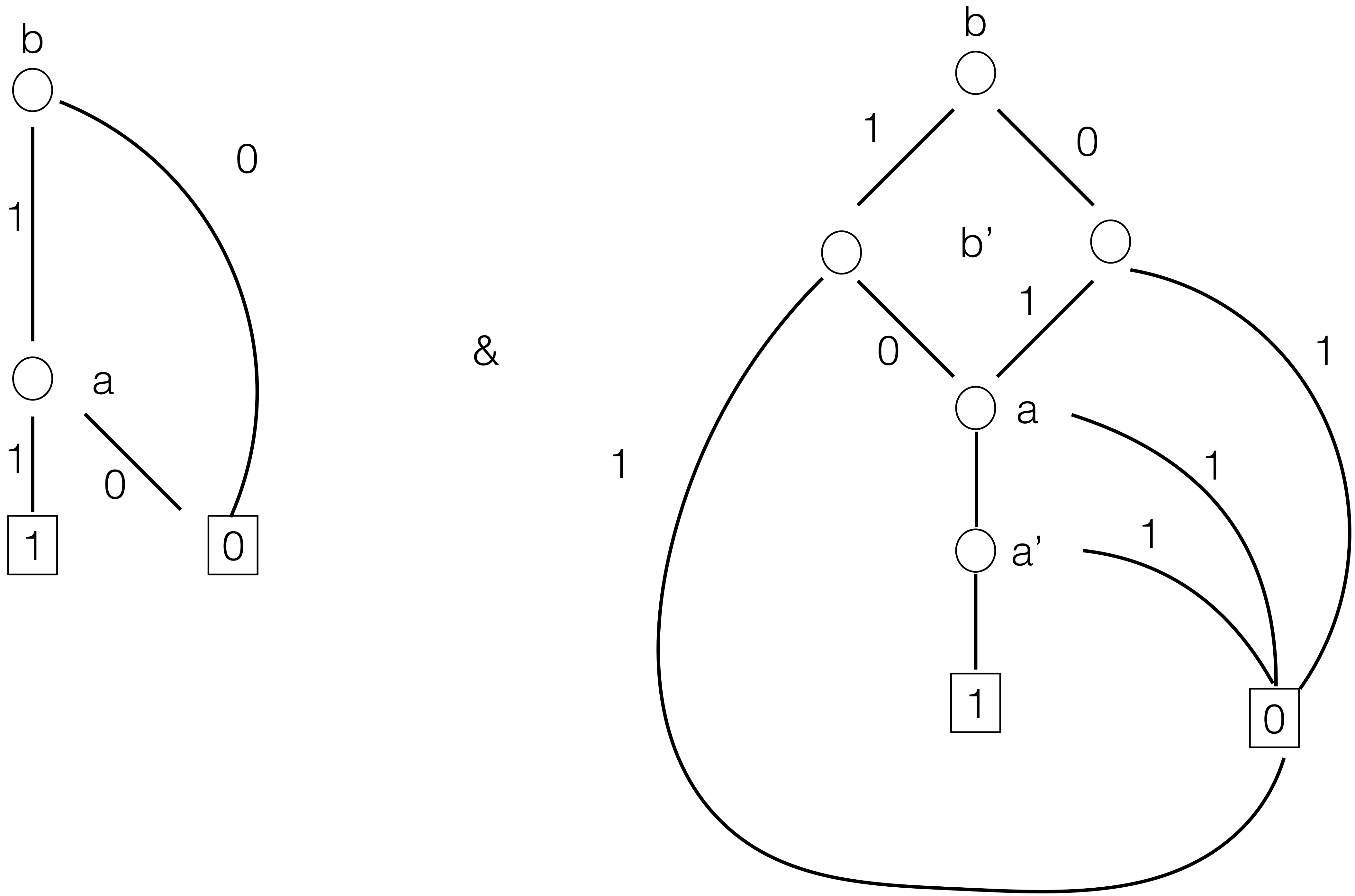
Forward image as existential quantifier

$$Fwd(P,T) = \{s' | \exists s. s \in P \wedge (s', s) \in T\}$$

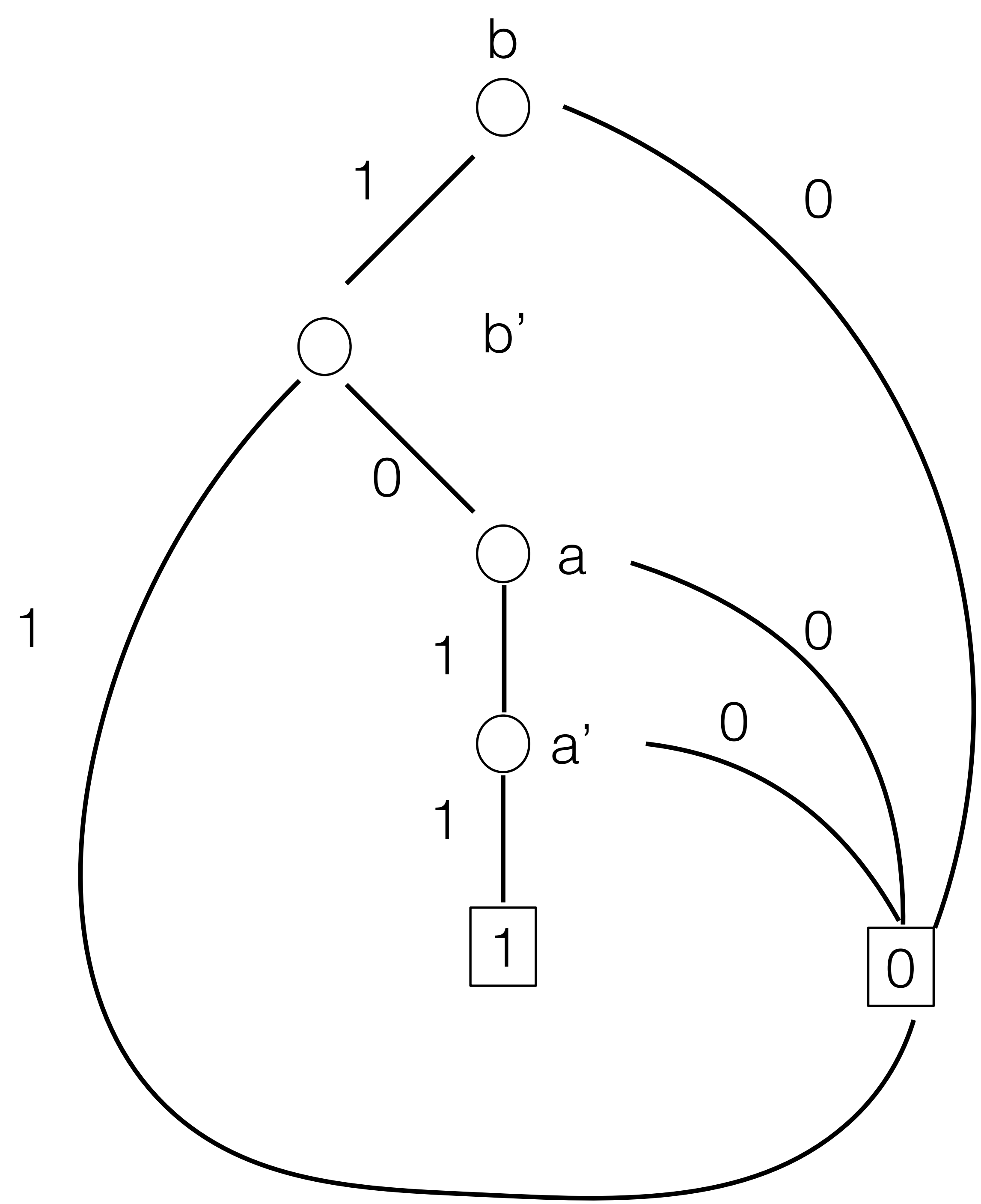
Operation on ROBDD:

- By definition: Exists a: f = f | !a or f | a
- Replace all a-nodes by negative sub-tree
- Replace all a-nodes by positive sub-tree

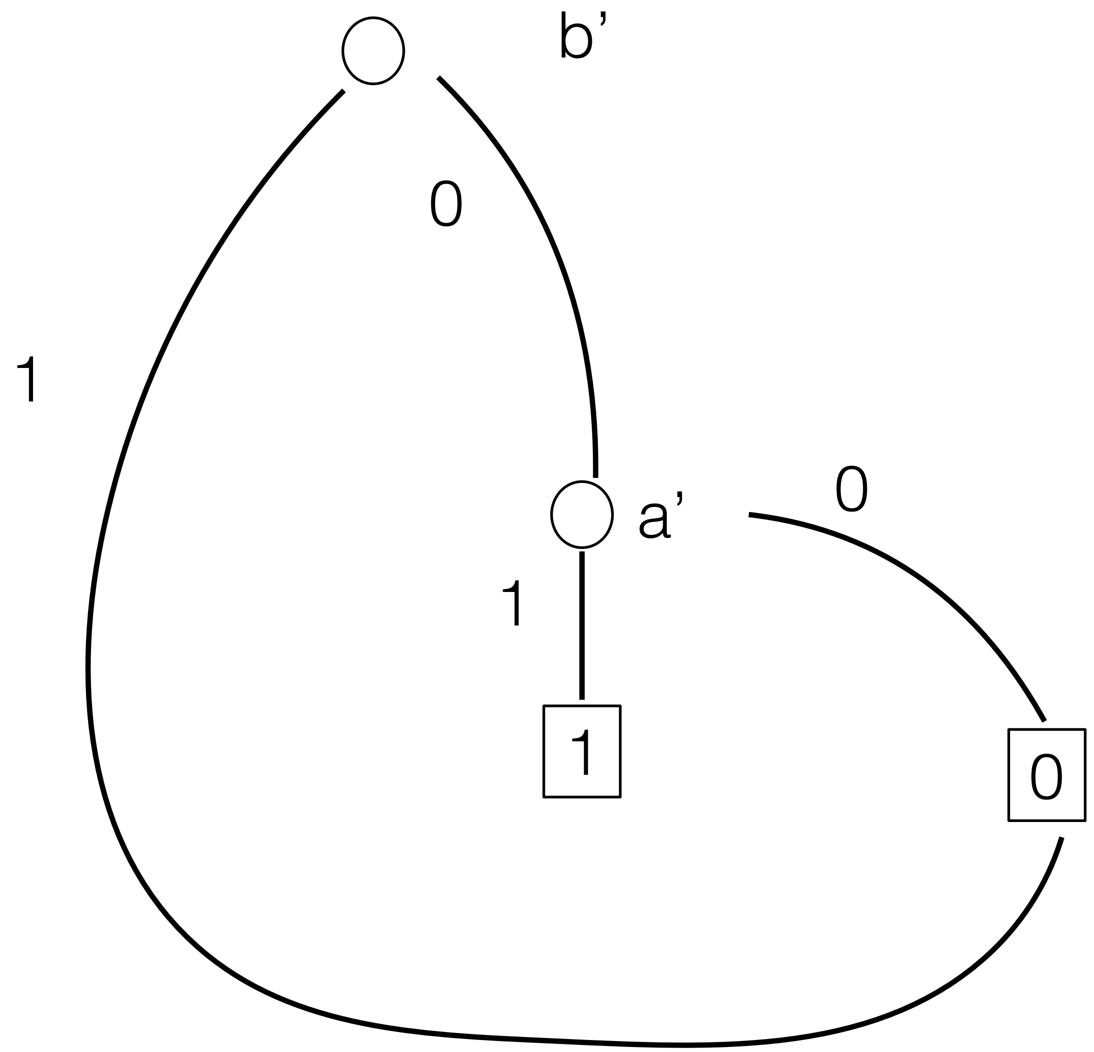
States in current & in transition relation



States in current & in transition relation

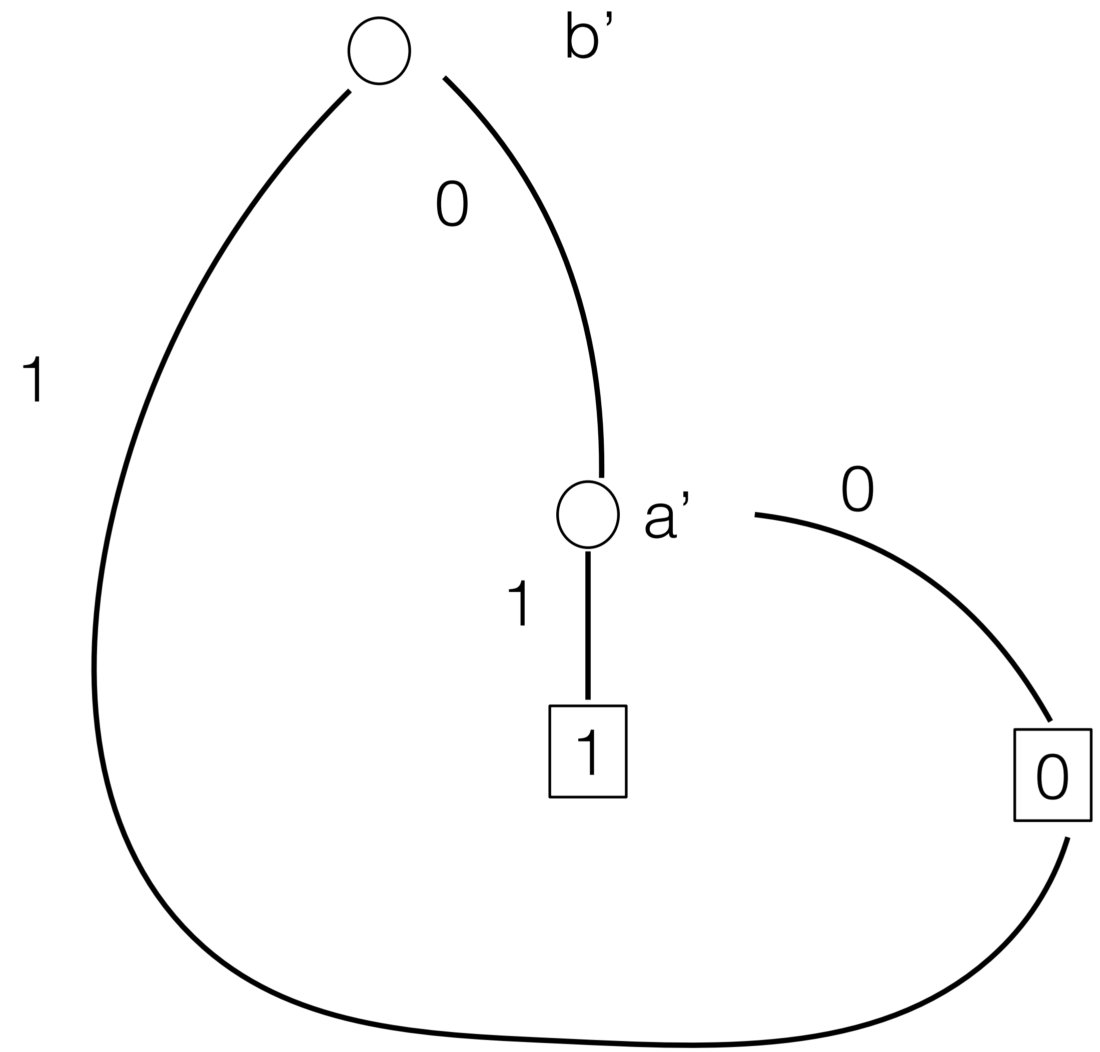


Existential quantifier on a and b



Exercise compute one more

As an exercise, compute the set of states reached from this state.



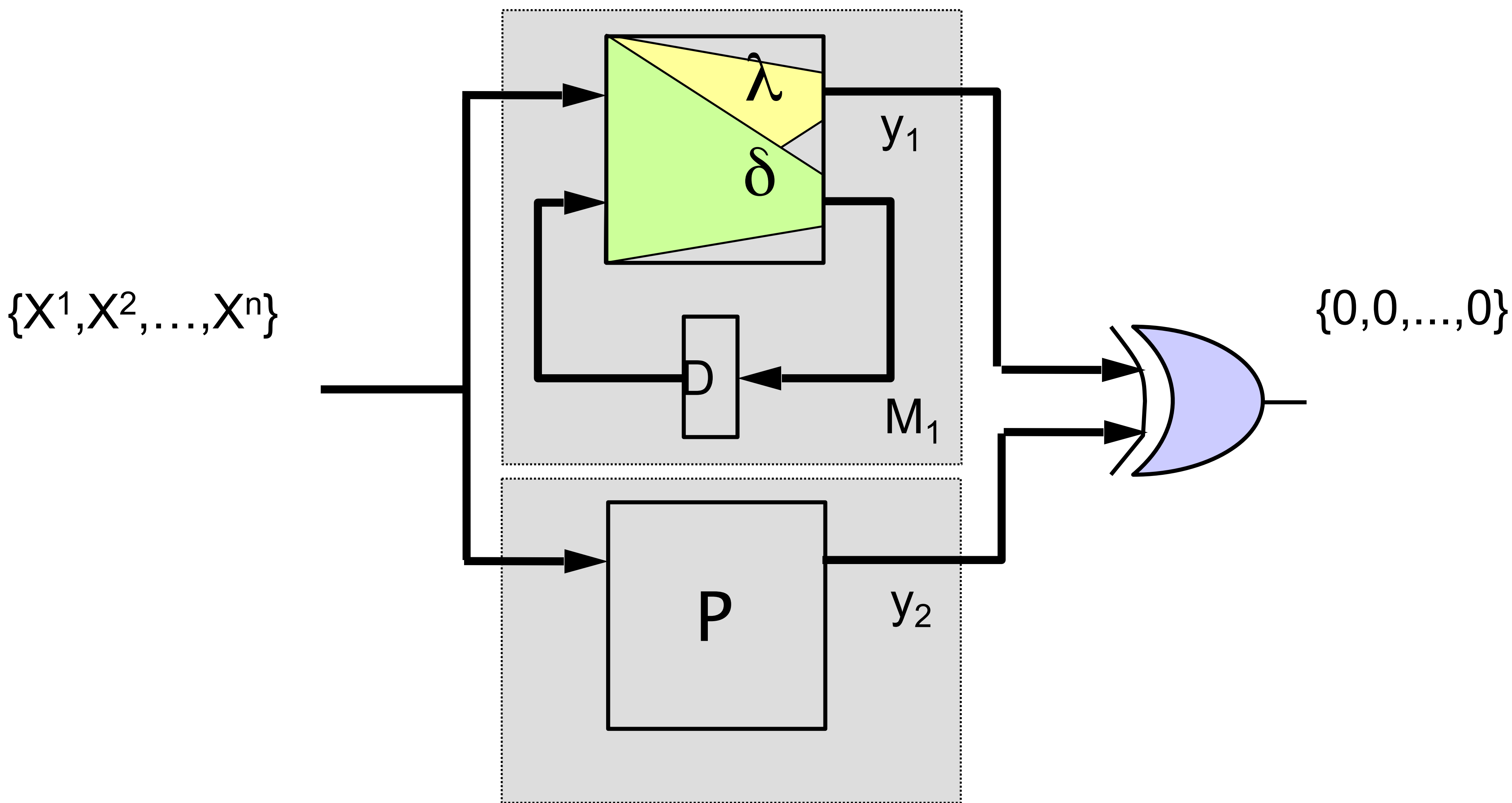
Summary

We looked at combinational and sequential equivalence.

We looked at reachability analysis: forward, backward, symbolic.

We looked at different representations for Boolean functions: AIGs, BDDs.

Generalising equivalence to properties



Intuition: In all (reachable) states, Machine M1 satisfies property P.

Another good reference on BDD

http://www.cs.utexas.edu/~isil/cs389L/bdd.pdf