Lecture 9: Symbolic Model Checking

- Basic CTL Fixpoint Theorems
- Ordered Binary Decision Diagrams (BDDs)
- Representing Transtion Systems with BDDs
- Symbolic Model Checking Algorithm

Breakthrough: Symbolic Model Checking

Method used by most "industrial strength" model checkers:

- uses boolean encoding for state machine and sets of states.
- can handle much larger designs – hundreds of state variables.
- BDDs traditionally used to represent boolean functions.

Symbolic Model Checking with BDDs

Ken McMillan implemented a version of the CTL model checking algorithm using Binary Decision Diagrams in 1987.

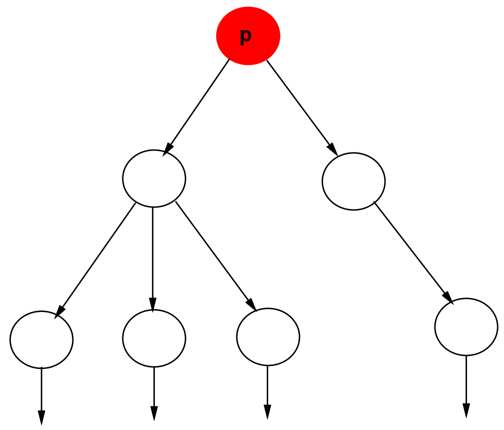
Carl Pixley at Motorola independently developed a similar algorithm, as did the French researchers, Coudert and Madre.

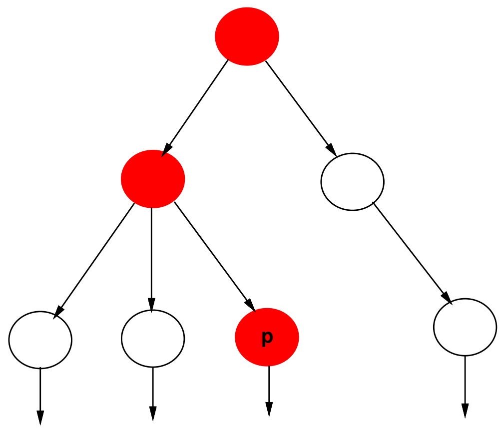
BDDs enabled handling much larger concurrent systems. (usually, an order of magnitude increase in hardware latches!)

- J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):pages 142–170, 1992.
- K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

Fixpoint Algorithms

$$\mathbf{EF}\, p = p \vee \mathbf{EX}\, \mathbf{EF}\, p$$





Fixpoint Algorithms (cont.)

Key properties of $\mathbf{EF}\,p$:

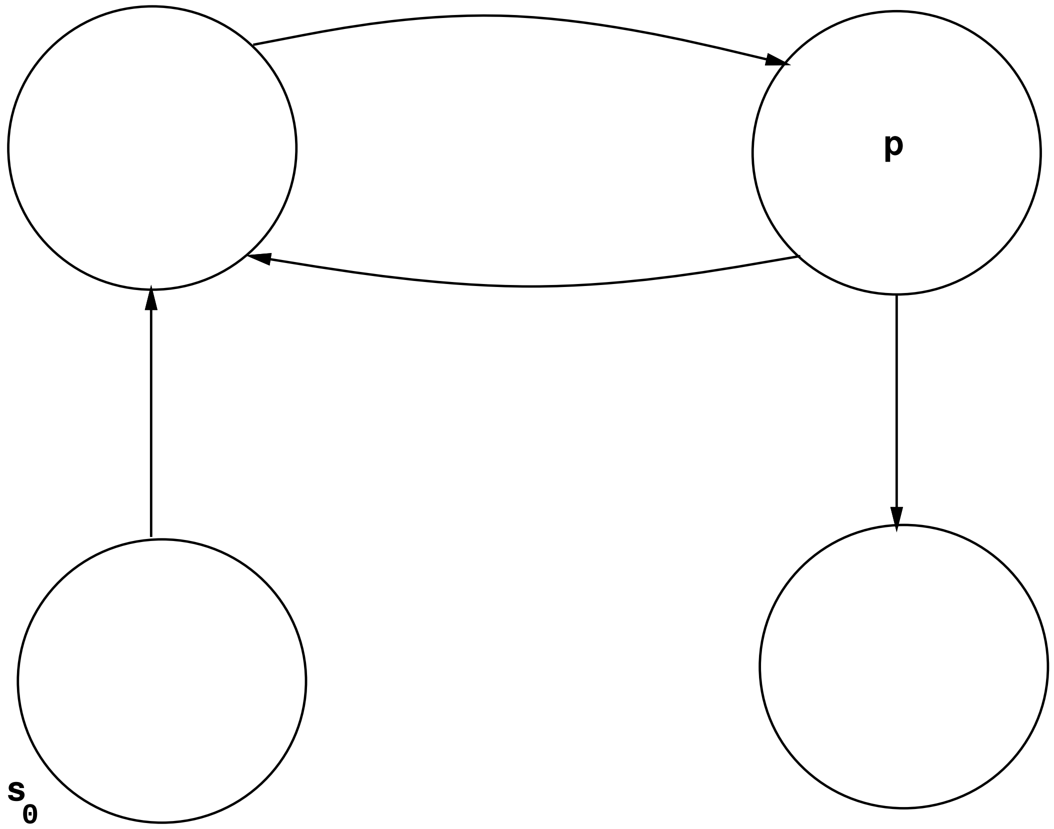
1. $\mathbf{EF}\,p = p \vee \mathbf{EX}\,\mathbf{EF}\,p$
2. $U = p \vee \mathbf{EX}\,U$ implies $\mathbf{EF}\,p \subseteq U$

We write $\mathbf{EF}\,p = \mathbf{Lfp}\; U.p \vee \mathbf{EX}\,U$.

How to compute $\mathbf{EF}\,p$:

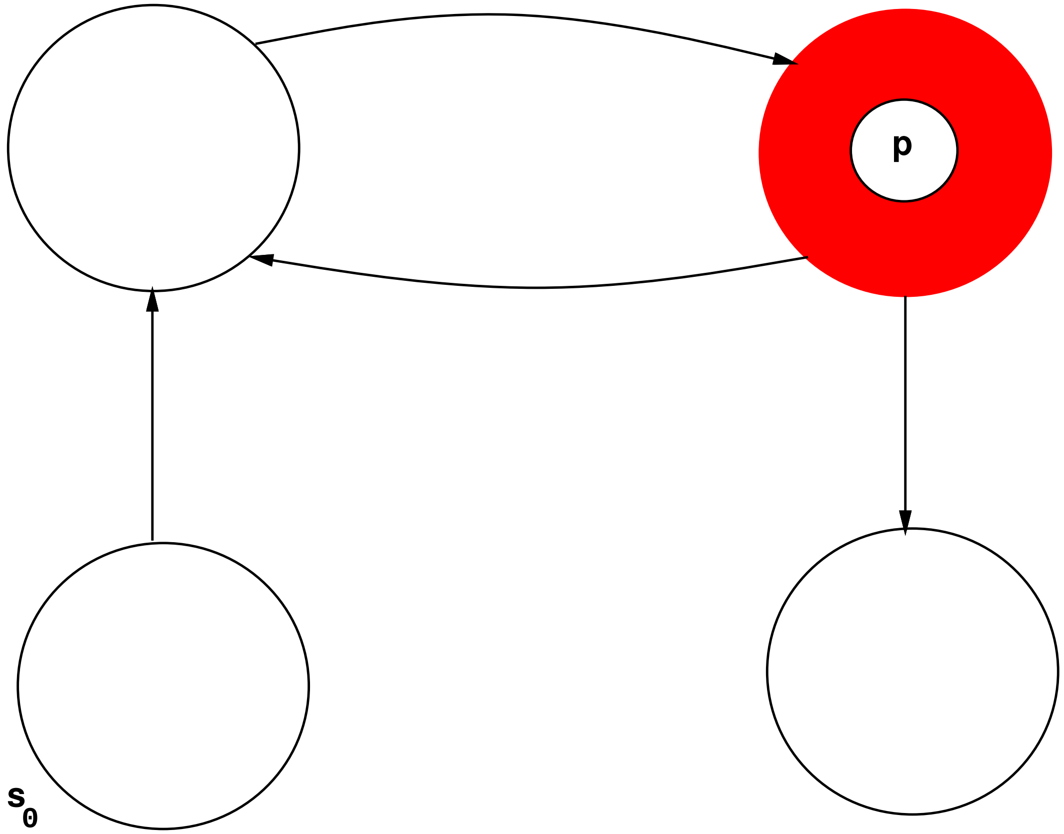
$$
\begin{array}{rcl}
U_0 & = & \mathbf{False} \\
U_1 & = & p \vee \mathbf{EX}\,U_0 \\
U_2 & = & p \vee \mathbf{EX}\,U_1 \\
U_3 & = & p \vee \mathbf{EX}\,U_2 \\
 & \vdots &
\end{array}
$$

$$M, s_0 \models \mathbf{EF}\, p\mathbf{?}$$



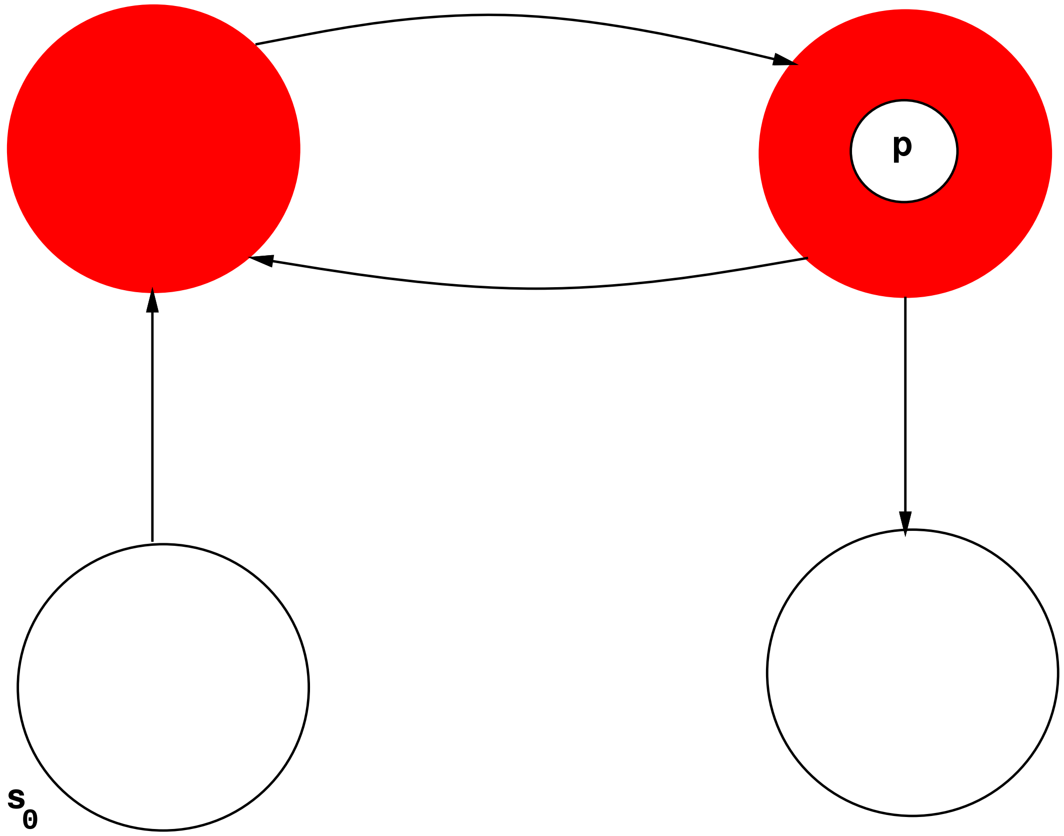
$U_0 = \emptyset$

$$M, s_0 \models \mathbf{EF}\, p?$$



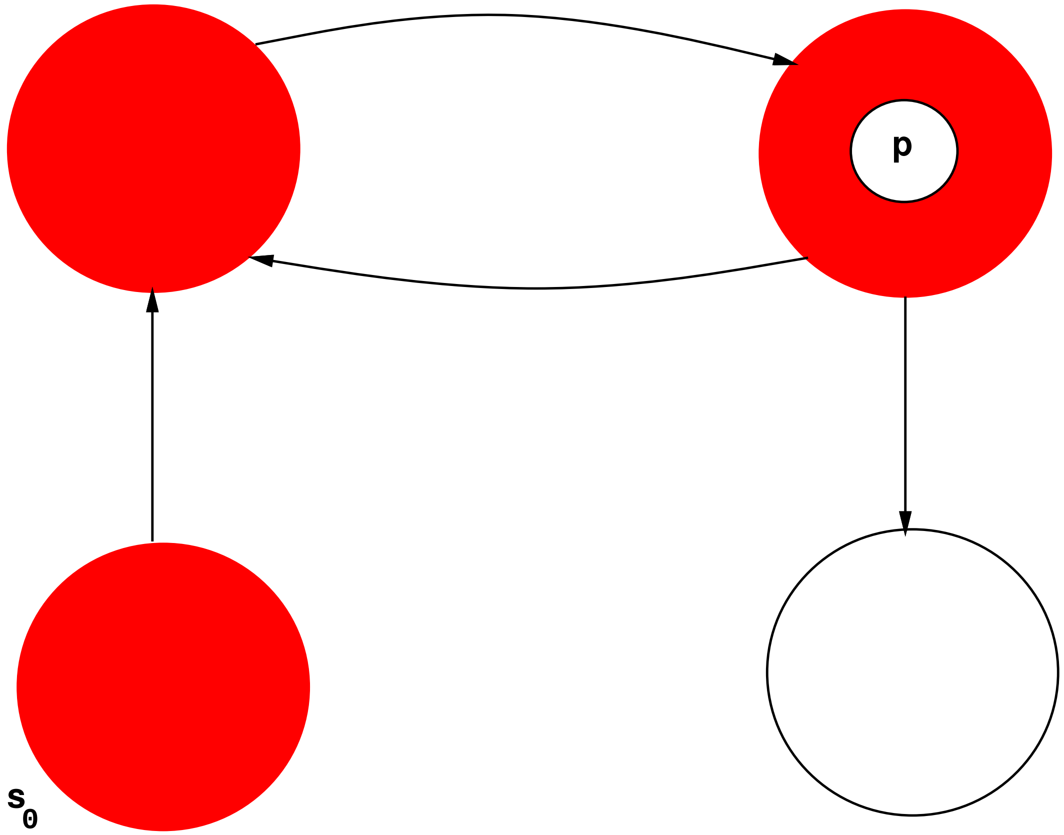
$U_1 = p \vee \mathbf{EX}\, U_0$

$$M, s_0 \models \mathbf{EF}\, p?$$



$U_2 = p \vee \mathbf{EX}\, U_1$

$M, s_0 \models \mathbf{EF}\, p$**?**



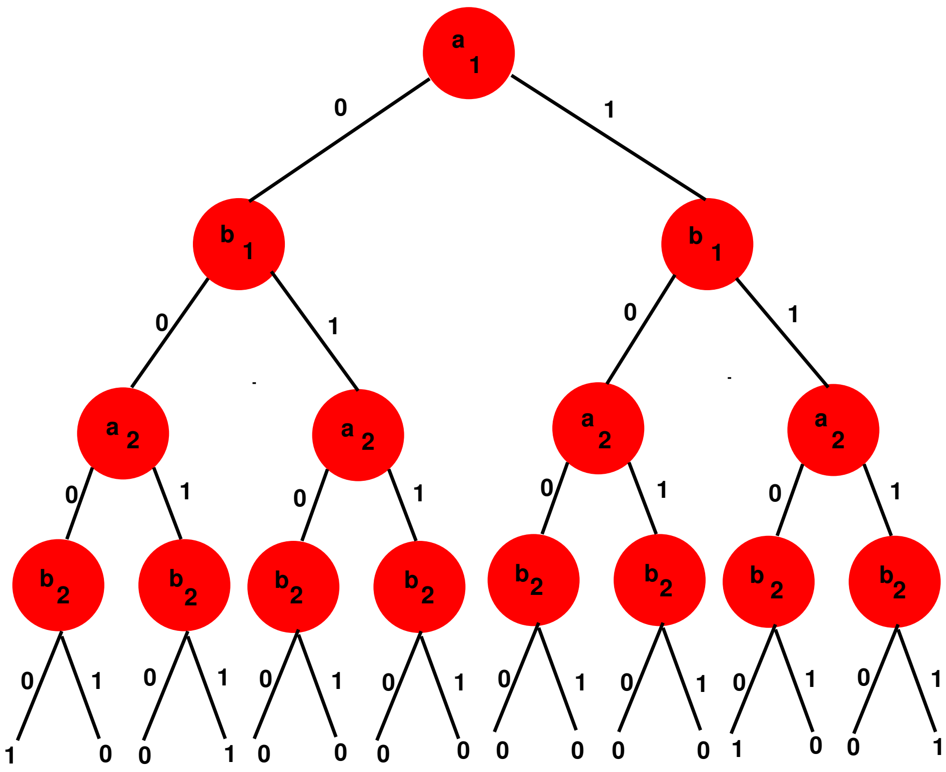
$U_3 = p \vee \mathbf{EX}\, U_2$

Ordered Binary Decision Trees and Diagrams

Ordered Binary Decision Tree for the two-bit comparator, given by the formula

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

is shown in the figure below:



From Binary Decision Trees to Diagrams

An Ordered Binary Decision Diagram (OBDD) is an ordered decision tree where

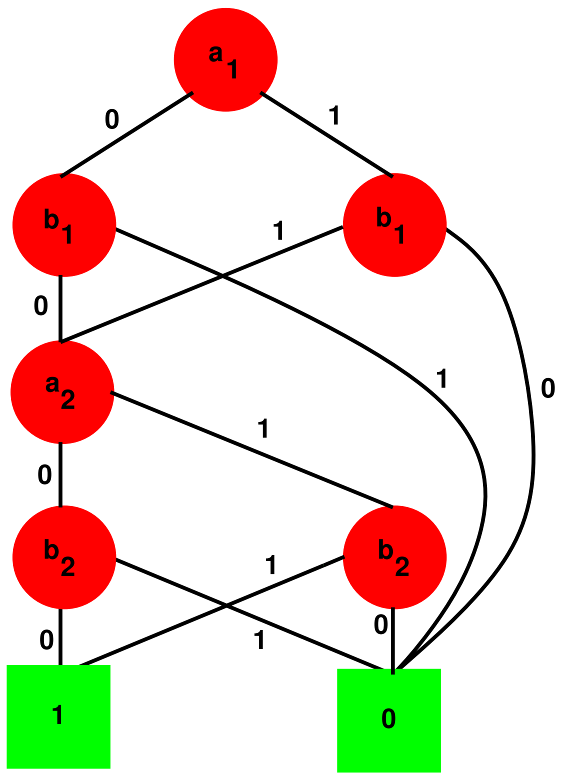
- All isomorphic subtrees are combined, and
- All nodes with isomorphic children are eliminated.

Given a parameter ordering, OBDD is unique up to isomorphism.

- R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

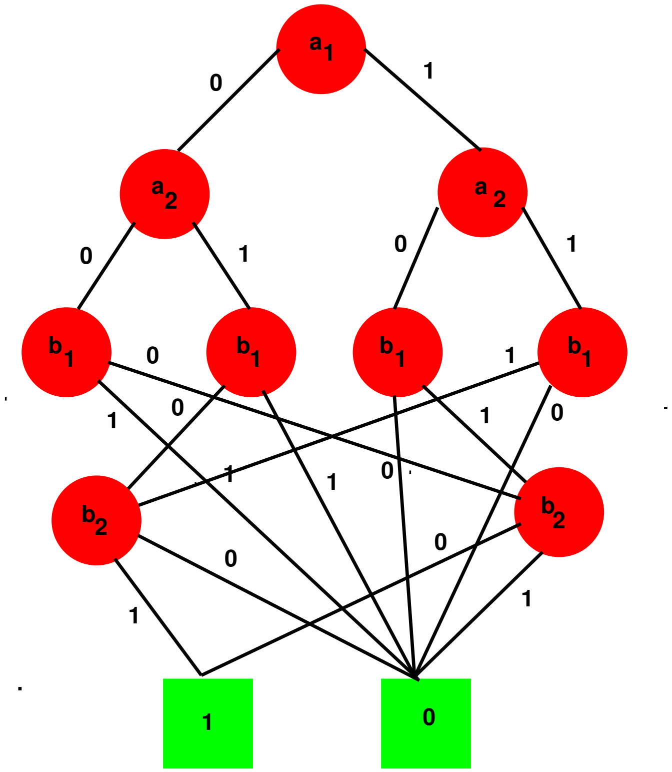
OBDD for Comparator Example

If we use the ordering $a_1 < b_1 < a_2 < b_2$ for the comparator function, we obtain the OBDD below:



Variable Ordering Problem

If we use the ordering $a_1 < a_2 < b_1 < b_2$ for the comparator function, we get the OBDD below:



Variable Ordering Problem (Cont.)

For an n-bit comparator:

- if we use the ordering $a_1 < b_1 < \ldots < a_n < b_n$, the number of vertices will be $3n + 2$.
- if we use the ordering $a_1 < \ldots < a_n < b_1 \ldots < b_n$, the number of vertices is $3 \cdot 2^n - 1$.

Moreover, there are boolean functions that have exponential size OBDDs for any variable ordering.

An example is the middle output (n^{th} output) of a combinational circuit to multiply two n bit integers.

Logical operations on OBDD's

- 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,

    $$f \cdot g = \bar{a} \cdot (f|_{\bar{a}} \cdot g|_{\bar{a}}) + a \cdot (f|_{a} \cdot g|_{a})$$

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

Logical operations (cont.)

- Boolean quantification: $\exists a : f(a, b, c, d)$
  - By definition,

$$\exists a : f = f|_{\bar{a}} \vee f|_{a}$$

  - $f(a, b, c, d)|_{\bar{a}}$: replace all a nodes by left sub-tree.
  - $f(a, b, c, d)|_{a}$: replace all a nodes by right sub-tree.

Using the above operations, we can build up OBDD's for complex boolean functions from simpler ones.

Symbolic Model Checking Algorithm

How to represent state-transition graphs with Ordered Binary Decision Diagrams:

Assume that system behavior is determined by n boolean state variables $v_1, v_2, \ldots, v_n$.

The Transition relation N will be given as a boolean formula in terms of the state variables:

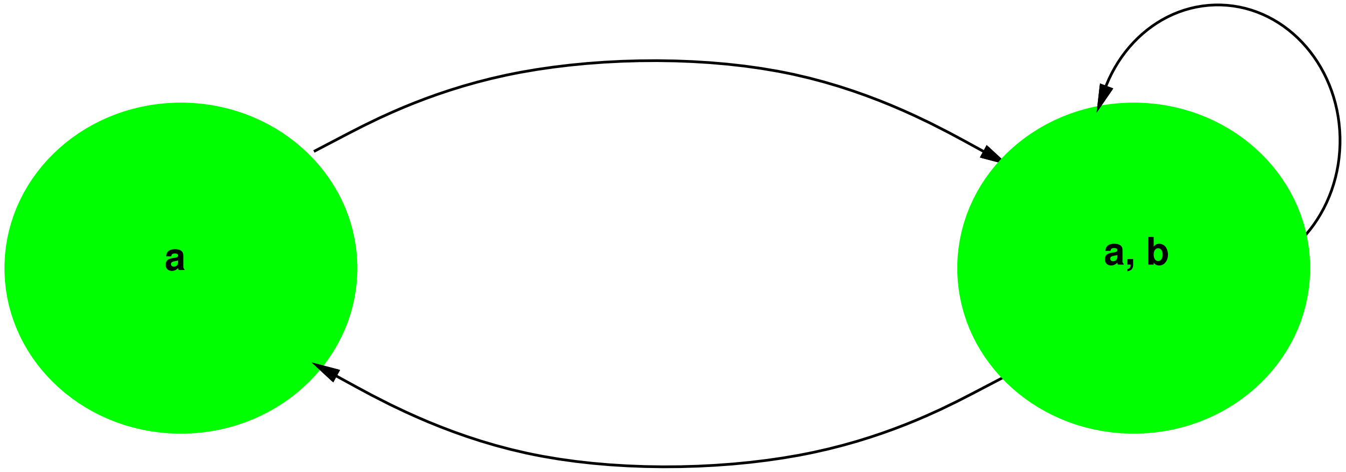
$$N(v_1, \ldots, v_n, v'_1, \ldots, v'_n)$$

where $v_1, \ldots v_n$ represents the current state and $v'_1, \ldots, v'_n$ represents the next state.

Now convert N to a OBDD!!

Symbolic Model Checking (cont.)

Representing transition relations symbolically:



Boolean formula for transition relation:

$$
\begin{array}{lccccccc}
 & (a & \wedge & \neg b & \wedge & a' & \wedge & b') \\
\vee & (a & \wedge & b & \wedge & a' & \wedge & b') \\
\vee & (a & \wedge & b & \wedge & a' & \wedge & \neg b')
\end{array}
$$

Now, represent as an OBDD!

Symbolic Model Checking (cont.)

Consider $f = \mathbf{EX}\, p$.

Now, introduce state variables and transition relation:

$$f(\bar{v}) = \exists \bar{v}'[N(\bar{v}, \bar{v}') \wedge p(\bar{v}')]$$

Compute OBDD for relational product on right side of formula.

Symbolic Model Checking (cont.)

How to evaluate fixpoint formulas using OBDDs:

$$\mathbf{EF}\, p = \mathbf{Lfp}\, U.\, p \vee \mathbf{EX}\, U$$

Introduce state variables:

$$\mathbf{EF}\, p = \mathbf{Lfp}\, U.\, p(\bar{v}) \vee \exists \bar{v}' \, [N(\bar{v}, \bar{v}') \wedge U(\bar{v}')]$$

Now, compute the sequence

$$U_0(\bar{v}), U_1(\bar{v}), U_2(\bar{v}), \ldots$$

until convergence.

Convergence can be detected since the sets of states $U_i(\bar{v})$ are represented as OBDDs.