Symbolic Model Checking

Sérgio Campos, Edmund Clarke

Breakthrough!

Ken McMillan implemented a version of the CTL model checking algorithm using OBDDs in the fall of 1987.

Subsequently, we were able to handle much larger concurrent systems!!

- 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.
- J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan, and D. L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 13(4):401–424, 1994.

Representing Transition Relations

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.

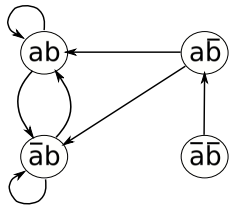
Now convert N to a OBDD!!

Representing Transition Relations

For a program with variables a and b:

- Create BDD variables a, b, a' and b'
- Represent each transition as a formula between these variables:

- The transition relation is the disjunction of all transitions:



$$N = (b \wedge b') \vee (a \wedge b') \vee (\neg a \wedge \neg b \wedge a' \wedge \neg b')$$

Symbolic Model Checking

Check takes a CTL formula as its argument and returns the OBDD for the set of states that satisfy the formula:

If f is an atomic proposition v_i, then $Check(f)$ is simply the OBDD for v_i.

Formulas of the form $f \vee g$ and $\neg f$ are handled using the standard OBDD algorithms for these connectives.

EX f, **E**$[f$ **U** $g]$, and **EG** f are handled by auxiliary procedures:

$$\begin{aligned} Check(\mathbf{EX}\, f) &= CheckEX(Check(f)) \\ Check(\mathbf{E}[f\ \mathbf{U}\ g]) &= CheckEU(Check(f), Check(g)) \\ Check(\mathbf{EG}\, f) &= CheckEG(Check(f)) \end{aligned}$$

AX f, **A**$[f$ **U** $g]$ and **AG** f are rewritten in terms of above operators.

Symbolic Model Checking (Cont.)

CheckEX is simple since **EX** f is true in a state if it has a successor in which f is true.

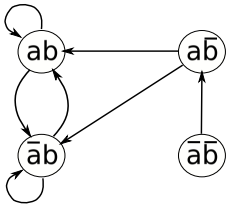
$$CheckEX(f(\bar{v})) = \exists \bar{v}' \left[f(\bar{v}') \wedge N(\bar{v}, \bar{v}') \right].$$

Given OBDDs for f and R, the OBDD for

$$\exists \bar{v}' \left[f(\bar{v}') \wedge N(\bar{v}, \bar{v}') \right].$$

is computed as described in the previously.

EX f Example



$$N = (b \wedge b') \vee (a \wedge b') \vee (\neg a \wedge \neg b \wedge a' \wedge \neg b')$$

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

	$\mathbf{EX}(a \wedge \neg b)$	$\mathbf{EX}(b)$
$f(\bar{v}')$	$a' \wedge \neg b'$	b'
$f \wedge N$	$\neg a \wedge \neg b \wedge a' \wedge \neg b'$	$(b \wedge b') \vee (a \wedge b')$
$\exists \bar{v}'[f \wedge N]$	$\neg a \wedge \neg b$	$a \vee b$

Symbolic Model Checking (Cont.)

$CheckEU(f(\bar{v}), g(\bar{v}))$ is given by

$$\mathbf{lfp}\, Z(\bar{v}) \left[g(\bar{v}) \vee \left(f(\bar{v}) \wedge CheckEX(Z(\bar{v})) \right) \right].$$

The function Lfp is used to compute the sequence of approximations $Z_0, Z_1, \ldots$.

This sequence converges to **E**$[f$ **U** $g]$ in a finite number of steps.

The OBDD for Z_{i+1} is computed from the OBDDs for f, g, and Z_i.

Since OBDDs are a canonical form for boolean functions, convergence is easy to detect.

When $Z_i = Z_{i+1}$, Lfp terminates. The state set for **E**$[f$ **U** $g]$ is given by the OBDD for Z_i.

Symbolic Model Checking (Cont.)

CheckEG is similar. In this case, the procedure is based on the greatest fixpoint characterization for the CTL operator **EG**:

$$CheckEG(f(\bar{v})) = \mathbf{gfp}\, Z(\bar{v})\left[f(\bar{v}) \wedge CheckEX(Z(\bar{v}))\right]$$

Given the OBDD for f, the function Gfp is used to compute the OBDD for $\mathbf{EG}\, f$.

CTL with Fairness Constraints

A *fairness constraint* can be an arbitrary formula of CTL.

Let $H = \{h_1, \ldots, h_n\}$ be a set of such fairness constraints.

A path p is *fair* with respect to H if each $h_i \in H$ holds *infinitely often* on p.

The path quantifiers in CTL formulas are restricted to fair paths.

EG with Fairness Constraints

Consider the formula $\mathbf{EG}\, f$ with the set of fairness constraints H.

This formula will be true at a state s if there is a path p starting at s such that

- f holds globally on p, and
- each formula in H holds infinitely often on p.

The operator **EG** (Cont.)

Let S be the largest set of states with the following two properties:

1. all of the states in S satisfy f, and
2. for all fairness constraints $h_k \in H$ and all states $s \in S$
   - there is a non-empty sequence of states from s to a state in S satisfying h_k, and
   - all states in the sequence satisfy the formula f.

It can be shown that each state in S is the beginning of a path on which f is always true.

Furthermore, every formula in H holds infinitely often on this path.

The operator **EG** (Cont.)

It follows that $\mathbf{EG}\, f$ can be expressed as a greatest fixed point of a predicate transformer:

$$\mathbf{EG}\, f = \mathbf{gfp}\, S \left[f \wedge \bigwedge_{k=1}^{n} \mathbf{EX}(\mathbf{E}[f\ \mathbf{U}\ S \wedge h_k]) \right]$$

This formula can be used to compute the set of states that satisfy $\mathbf{EG}\, f$.

Other Operators

Checking the formulas $\mathbf{EX}\, f$ and $\mathbf{E}[f\ \mathbf{U}\ g]$ under fairness constraints is simpler.

The set of all states which are the start of some fair computation is

$$fair = \mathbf{EG}\, true.$$

Hence,

$$\mathbf{EX}(f) = \mathbf{EX}(f \wedge fair),$$
$$\mathbf{E}[f\ \mathbf{U}\ g] = \mathbf{E}[f\ \mathbf{U}\ g \wedge fair]$$

Remaining CTL operators can be expressed in terms of **EX**, **EG**, and **EU**. For example,

$$\mathbf{A}[f\ \mathbf{U}\ g] \equiv \neg\, \mathbf{E}[\neg g\ \mathbf{U}\ \neg f \wedge \neg g] \wedge \neg\, \mathbf{EG}\, \neg g$$