Model Checking IV
Symbolic Model Checking

Edmund M. Clarke, Jr.
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

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!!

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:

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

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.

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

Given OBDDs for f and R, the OBDD for

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

is computed as described in the first lecture.

Symbolic Model Checking (Cont.)

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

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

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

This sequence converges to $\mathbf{E}[f \ \mathbf{U}\ g]$ 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 $\mathbf{E}[f \ \mathbf{U}\ g]$ 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**:

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

Given the OBDD for f, the function Gfp is used to compute the OBDD for **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 **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

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

Hence,

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

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

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

ω-automata

There are many types of ω-automata. However, we will only consider deterministic Büchi automata.

A finite *Büchi automaton* is a 5-tuple

$$M = \langle K, p_0, \Sigma, \Delta, A \rangle,$$

where

- K is a finite set of *states*
- $p_0 \in K$ is the *initial state*
- Σ is a finite *alphabet*
- $\Delta \subseteq K \times \Sigma \times K$ is the *transition relation*
- $A \subseteq K$ is the *acceptance set*.

M is *deterministic* if for all $p, q_1, q_2 \in K$ and $\sigma \in \Sigma$, if $\langle p, \sigma, q_1 \rangle, \langle p, \sigma, q_2 \rangle \in \Delta$ then $q_1 = q_2$.

Language Acceptance

An infinite sequence of states $p_0 p_1 p_2 \ldots \in K^\omega$ is a *path* in M if there exists an infinite sequence $a_0 a_1 a_2 \ldots \in \Sigma^\omega$ such that $\forall i \geq 0 : \langle s_i, a_i, s_{i+1} \rangle \in \Delta$.

Let $p = p_0 p_1 p_2 \ldots \in K^\omega$ be a path in M. The *infinitary set* of p is the set of states that occur infintely often on p.

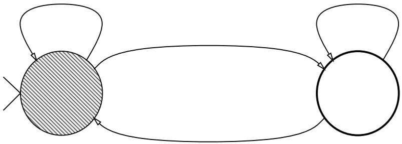
A sequence $a_0 a_1 a_2 \ldots \in \Sigma^\omega$ is *accepted* by M if there is a corresponding path $p = p_0 p_1 p_2 \ldots \in K^\omega$ such that the infinitary set of p contains at least one element of A.

The set of sequences accepted by an automaton M is called the *language of M* and is denoted $\mathcal{L}(M)$.

Büchi Automata Examples

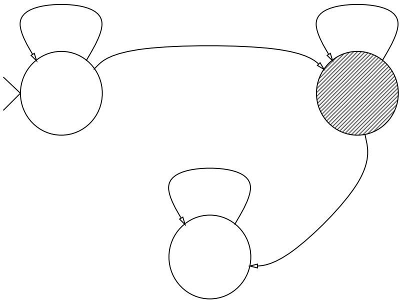
The alphabet for these examples is the set $\Sigma = \{p, q, r\}$. States in the acceptance set are shaded.

This automaton accepts infinite length strings with the property that every occurrence of p is eventually followed by an occurrence of q.



Büchi Automata Examples (cont.)

This automaton accepts infinite length strings with the property that p occurs almost always in the string.



Product Construction

Let M and M' be two Büchi automata over the same alphabet Σ.

Consider the Kripke structure

$$\mathcal{K}(M, M') = (\mathrm{AP}, K \times K', \langle p_0, p_0' \rangle, L, R),$$

where

- $\mathrm{AP} = \{q, q'\}$ is the set of atomic propositions
- $\langle s, s' \rangle \models q$ *iff* $s \in A$
- $\langle s, s' \rangle \models q'$ *iff* $s' \in A'$
- $\langle s, s' \rangle R \langle r, r' \rangle$ *iff* $\exists a \in \Sigma : \langle s, a, r \rangle \in \Delta$ and $\langle s', a, r \rangle \in \Delta'$.

Checking Containment

It is possible to show that, if M' is deterministic,

$$\mathcal{L}(M) \subseteq \mathcal{L}(M') \Leftrightarrow K(M, M') \models \mathbf{A}[\mathbf{GF}\, q \Rightarrow \mathbf{GF}\, q']$$

The above formula is in CTL* but not in CTL. However, it belongs to a class of formulas which can be checked in polynomial time.

In fact, $\mathbf{A}[\mathbf{GF}\, q \Rightarrow \mathbf{GF}\, q']$ is equivalent to $\mathbf{AG}\,\mathbf{AF}\, q'$ under the fairness constraint "infinitely often q".

Checking this formula with the given fairness constraint can be handled by the technique described previously.