Lecture 15: Implementing a Symbolic Model Checker

- Representing Transition Relations
- Implementing Basic CTL Operators
- Fairness Constraints
- Buchi Automata
- Omega Regular Languages
- Checking Language Containment
How To Build a CTL Model Checker

The following papers describe how to build a Symbolic Model Checker including fairness constraints:

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

\textit{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 $\text{Check}(f)$ is simply the OBDD for $v_i$.

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

\textbf{EX} $f$, \textbf{E}[f U g], and \textbf{EG} $f$ are handled by auxiliary procedures:

\begin{align*}
    \text{Check(} \textbf{EX} f \text{)} & = \text{CheckEX(} \text{Check}(f) \text{)} \\
    \text{Check(} \textbf{E}[f \text{ U } g] \text{)} & = \text{CheckEU(} \text{Check}(f), \text{Check}(g) \text{)} \\
    \text{Check(} \textbf{EG} f \text{)} & = \text{CheckEG(} \text{Check}(f) \text{)}
\end{align*}

\textbf{AX} $f$, \textbf{A}[f U g] and \textbf{AG} $f$ are rewritten in terms of above operators.
Symbolic Model Checking (Cont.)

*CheckEX* is simple since $\textbf{EX} f$ is true in a state if it has a successor in which $f$ is true.

\[
\text{CheckEX}(f(\overline{v})) = \exists \overline{v}' \ [f(\overline{v}') \land R(\overline{v}, \overline{v}')] .
\]

Given OBDDs for $f$ and $R$, the OBDD for

\[
\exists \overline{v}' \ [f(\overline{v}') \land R(\overline{v}, \overline{v}')] .
\]

is computed as described in the first lecture.
Symbolic Model Checking (Cont.)

\[ \text{CheckEU}(f(\bar{v}), g(\bar{v})) \text{ is given by} \]
\[ \text{lfp } Z(\bar{v}) \left[ g(\bar{v}) \lor \left( f(\bar{v}) \land \text{CheckEX}(Z(\bar{v})) \right) \right]. \]

The function \text{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 \(Z_i = Z_{i+1}\), \text{lfp} terminates. The state set for \(E[f \ U g]\) is given by the OBDD for \(Z_i\).
Symbolic Model Checking (Cont.)

Check$_{EG}$ is similar. In this case, the procedure is based on the greatest fixpoint characterization for the CTL operator $\text{EG}$:

\[
\text{Check}_{EG}(f(\bar{v})) = \text{gfp} \ Z(\bar{v}) \ [f(\bar{v}) \land \text{Check}_{EX}(Z(\bar{v}))]
\]

Given the OBDD for $f$, the function Gfp is used to compute the OBDD for $\text{EG} \ f$. 
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.
Consider the formula $\text{EG } f$ 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 $p$. 
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 $\text{EG}$ (Cont.)

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

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

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

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

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

$$fair = \text{EG } \text{true}.$$ 

Hence,

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

Remaining CTL operators can be expressed in terms of $\text{EX}$, $\text{EG}$, and $\text{EU}$. For example,

$$\text{A}[f \ U \ g] \equiv \neg \text{E}[
eg g \ U \neg f \land \neg g] \land \neg \text{EG } \neg g.$$
There are many types of $\omega$-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
- $\Sigma$ 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 \( p_0p_1p_2 \ldots \in K^\omega \) is a path in \( M \) if there exists an infinite sequence \( a_0a_1a_2 \ldots \in \Sigma^\omega \) such that \( \forall i \geq 0 : \langle s_i, a_i, s_{i+1} \rangle \in \Delta \).

Let \( p = p_0p_1p_2 \ldots \in K^\omega \) be a path in \( M \). The infinitary set of \( p \) is the set of states that occur infinitely often on \( p \).

A sequence \( a_0a_1a_2 \ldots \in \Sigma^\omega \) is accepted by \( M \) if there is a corresponding path \( p = p_0p_1p_2 \ldots \in K^\omega \) 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 \( L(M) \).
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 $q$.

- 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 $\Sigma$.

Consider the Kripke structure

$$K(M, M') = (AP, K \times K', \langle p_0, p'_0 \rangle, L, R),$$

where

- $AP = \{q, q'\}$ 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,

$$\mathcal{L}(M) \subseteq \mathcal{L}(M') \iff K(M, M') = A[\varphi q \Rightarrow \varphi q']$$

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, $A[\varphi q \Rightarrow \varphi q']$ is equivalent to $AG AF q'$ under the fairness constraint “infinitely often $q$”.

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