Today’s Lecture

- Optimizations in Model checking
  - Symmetry Reduction
- Simulation/Bisimulation
Simulation and Bisimulation

Simulation --- Intuition

- Two finite state machines (Kripke structures) $M$ and $M'$
- $M'$ simulates $M$ if
  - $M'$ can start in a similarly labeled state as $M$
  - For every step that $M$ takes from $s$ to $t$, $M'$ can mimic it by stepping to a state with similar label as $t$
Simulation

- \( M = (S, S_0, R, L) \) and \( M' = (S', S_0', R', L') \)
- A relation \( H \subseteq S \times S' \) is a simulation relation between \( M \) and \( M' \) means that:
  - For all \( (s, s') \), if \( H(s, s') \) then:
    - \( L'(s') = L(s) \cap AP' \)
    - For every state \( t \) s.t. \( R(s, t) \) there is a state \( t' \) such that \( R'(s', t') \) and \( H(t, t') \)

- \( M' \) simulates \( M \) if
  - there exists a simulation relation \( H \) between them, and
  - For each \( s_0 \in S_0 \), there exists \( s_0' \in S_0' \) s.t. \( H(s_0, s_0') \)

Example

Atomic propositions: go and stop

Which machine simulates which?
**Bisimulation**

- M and M’ are bisimulation equivalent (bisimilar) if
  - M can match each step of M’ and vice-versa
  - Note: this is NOT the same as “M simulates M’ and M’ simulates M”

- A relation $H \subseteq S \times S'$ is a bisimulation relation between M and M’ means that:
  - For all $(s, s')$, if $H(s, s')$ then:
    - $L'(s') = L(s) \cap AP'$
    - For every state $t$ s.t. $R(s, t)$ there is a state $t'$ such that $R'(s', t')$ and $H(t, t')$
    - For every state $t'$ s.t. $R'(s', t')$ there is a state $t$ such that $R(s, t)$ and $H(t, t')$

---

**(Bi)Simulation and (A)CTL***

- If M’ simulates M, then any ACTL* property satisfied by M’ is satisfied by M

- If M’ and M are bisimilar, any CTL* property satisfied by one is also satisfied by the other
Symmetry Reduction

Symmetry

• Many systems have inherent symmetry
  – Overall system might be composed of $k$ identical modules
  – E.g., a multi-processor system with $k$ processors
  – E.g., a multi-threaded program with $k$ threads executing the same code with same inputs
  – Anything with replicated structure

• Question: How can we detect and exploit the symmetry in the underlying state space for model checking?
Symmetry in Behavior

• Given a system with two identical modules
  – Run: $s_0, s_1, s_2, \ldots$
  – Trace: $L(s_0), L(s_1), L(s_2), \ldots$

  – Each $s_i = (s_{i1}, s_{i2}, \text{rest})$ comprises \textit{values to variables} of both modules 1 and 2
  – If we can interchange these without changing the set of traces of the overall system, then there is symmetry in the system behavior

Exploiting Symmetry

• If a state space is symmetric, we can group states into equivalence classes
  – Just as in abstraction

• Resulting state graph/space is called \textit{“quotient”} graph/space
  – Model check this quotient graph
**Quotient (first attempt)**

\[ M = (S, S_0, R, L) \]

Let \( \equiv \) be an equivalence relation on \( S \)

Assume: \( s \equiv t \iff L(s) = L(t) \)

& \( s \in S_0 \iff t \in S_0 \)

Quotient: \( M' = (S', S'_0, R', L') \)

- \( S' = S/\equiv \) , \( S'_0 = S_0/\equiv \) (states are equivalence classes with respect to \( \equiv \))
- \( R'([s], [t]) \) whenever \( R(s, t) \)
- \( L'([s]) = L(s) \)

---

**Is that definition enough?**

Suppose we want to check an invariant:

Does \( M \) satisfy \( \varphi \) ?

Instead if we check:

Does quotient \( M' \) satisfy \( \varphi \) ?

If \( M' \) is constructed using the definition of \( \equiv \) on the previous slide, will the above check generate spurious counterexamples?
Stable Equivalences

Equivalence $\equiv$ is called stable if:

$$R(x, y) \Rightarrow$$

for every $s$ in $[x]$ there exists some $t$ in $[y]$ such that $R(s, t)$

Claim: Suppose $\equiv$ is stable, then:

$M$ satisfies $\varphi$ iff $M'$ satisfies $\varphi$

(Proof idea: show $M$ and $M'$ are bisimilar)

Detecting Symmetry

• Given symmetry expressed as an equivalence relation between states, we know how to exploit it

• How do we detect/compute this equivalence relation?
  – Need to characterize it more formally
Symmetry as Permutation

- Symmetry in the state space can be viewed as “equivalence under permutation”
- Permute the set of states so that the set of traces remains the same
  - A subset of states that remains the same under permutation forms the needed equivalence class
- A representation of all possible such permutations represents symmetry in the system

Automorphisms

A permutation function
\[ f : S \rightarrow S \]
is an automorphism if:
\[ R(s, t) \Leftrightarrow R(f(s), f(t)) \]

What is an example automorphism for this state space?
Automorphisms

\[f: \begin{align*}
    f(0,0) &= 1,1 & f(1,1) &= 0,0 \\
    f(0,1) &= 0,1 & f(1,0) &= 1,0
\end{align*}\]

\[g: \begin{align*}
    g(0,0) &= 0,0 & g(1,1) &= 1,1 \\
    g(0,1) &= 1,0 & g(1,0) &= 0,1
\end{align*}\]

\[A = \{ f, g, f \circ g, \text{id} \}\]

The set of all automorphisms forms a group!

Equivalence using Automorphisms

Let \( s \cong t \) if there is some automorphism \( f \) such that
\[f(s) = t \quad \text{(and } L(s) = L(t) \land s \in S_0 \iff t \in S_0)\]

The equivalence classes of an automorphism (sets mapped to themselves) are called \textit{orbits}

Claim 1: \( \cong \) is an equivalence
Claim 2: \( \cong \) is stable \quad (why? - HW)
Orbits

\[(0,0), (1,1)\]
\[(0,1), (1,0)\]

Symmetry reduction

Map each state to its representative in the orbit
How Symmetry Reduction works in practice

- A permutation (automorphism) group is *manually* constructed
  - Syntactically specify which modules are identical
- Orbit relation (equivalence relation) automatically generated from this
  - Using fixpoint computation (MC, Sec. 14.3)
- An (lexicographically smallest) element of each equivalence class is picked as its representative
- $S_0'$ and $R'$ generated from orbit relation
- Model checking explores only representative states

Symmetry reduction

- Implemented in many model checkers
  - E.g., SMV, Mur$\varphi$ (finite-state systems), Brutus (security protocols)