Today’s Lecture

• Symbolic model checking with BDDs
  – Fairness
  – Counterexample/witness generation for general CTL
• Optimizations in Model checking
  – Abstraction (mostly next week)
  – Symmetry Reduction
  – Compositional Reasoning
• Simulation/Bisimulation
Fairness

• A computation path is defined as fair if a fairness constraint \( p \) is true infinitely often along that path
  – Fairness constraint is a state predicate
  – Generalized to set of fairness constraints \( \{p_1, p_2, \ldots, p_k\} \) by requiring each element of the subset to be true infinitely often

• Example: Every process in an asynchronous composition must be scheduled infinitely often

Why does Fairness matter?

• We need to model policies that enforce fairness in the model
  – Otherwise, we will get spurious counterexamples
  – Example: A scheduler might use round-robin scheduling amongst processes
    • Instead of verifying the system for a particular fixed fair scheduling strategy, we can verify it for all fair schedulers
Fairness in Symbolic Model Checking of CTL

• Suppose Fairness means that each element of \(\{p_1, p_2, \ldots, p_k\}\) must be true infinitely often

• Fair formulation of EG f is: The largest set of states \(Z\) such that
  – All of the states in \(Z\) satisfy \(f\)
  – For all fairness constraints \(p_i\), and all states \(s \in Z\), there is a path of length 1 or greater from \(s\) to a state in \(Z\) satisfying \(p_i\) such that all states along that path satisfy \(f\)

  \[\text{i.e., there is a next state of } \text{s satisfying } f \cup (Z \land p_i)\]

  – What’s the fixpoint formulation of EG f with fairness? For EGf: \(\forall Z. [f \land EX \ Z]\)
Fairness in Symbolic Model Checking of CTL

- Fair formulation of $\text{EG } f$ is: The largest set of states $Z$ such that
  - All of the states in $Z$ satisfy $f$
  - For all fairness constraints $p_i$, and all states $s \in Z$,
    - there is a path of length 1 or greater from $s$ to a state in $Z$ satisfying $p_i$ such that all states along that path satisfy $f$
    - i.e., there is a next state of $s$ satisfying $f \text{ U } (Z \land p_i)$
  - $\forall Z. f \land (\bigwedge_i \text{ EX } E[f \text{ U } (Z \land p_i)])$

Counterexample Generation under Fairness

- Algorithm needs to be adjusted accordingly
  - Need to find a cycle that visits each fairness constraint $p_i$ at least once
  - See Clarke et al. textbook for details
BDD-related Optimizations – Key Ideas

- Choose a good BDD variable ordering to start with
- Keep the support of computed BDDs as small as possible

What do we need to represent?

- Set of transitions: \( R(v, v') \)
- Sets of states: \( S_0(v) \), intermediate results of fixpoint computations
Representing $R(v, v')$

- How should the $v$ and $v'$ variables be ordered in the BDD relative to each other?
- Keep $v_i$ close to $v_i'$ (interleave)

Relational Product

- Recall that reachability analysis involved computing
  $$S_{i+1}(v) = S_i(v) \lor (\exists v \{ S_i(v) \land R(v,v') \}) [v/v']$$

- Relational Product operation is
  $$\exists v \{ S_i(v) \land R(v,v') \}$$

- This is done as one primitive BDD operation
  - Rather than an AND followed by EXISTS (why?)
Disjunctive Partitioning

• Suppose we have an asynchronous system composed of \( k \) processes
• Then, \( R(v, v') \) can be decomposed as
  \[ \bigvee_i R_i(v, v') \]
  – Plug into expression for relational product
  – Does \( \exists \) distribute over \( \bigvee \)? What use is that?

 Conjunctive Partitioning

• Suppose we have an synchronous system composed of \( k \) processes
• Then, \( R(v, v') \) can be decomposed as
  \[ \bigwedge_i R_i(v, v') \]
  – Can we do the same optimization as on the previous slide? If not, is a similar optimization possible?
Conjunctive Partitioning

• Suppose we have an synchronous system composed of \( k \) processes
• Then, \( R(v, v') \) can be decomposed as
  \[ \wedge_i R_i(v, v') \]
  – Can we do the same optimization as on the previous slide? If not, is a similar optimization possible?
  • We can choose an order in which to quantify out variables and push the quantifiers as far in as possible
  • What order do we pick?

Key Optimizations in (Symbolic) Model Checking

• Abstraction
  – Compute a smaller state graph by “merging states” s.t. if the property holds on the smaller system model, it holds on the larger one
• Symmetry Reduction
  – Group states into equivalence classes by exploiting symmetries in the model
• Compositional Reasoning
  – Compose proofs of correctness of modules to prove the overall system correct
Abstraction

• Reduce the size of the system model by throwing out information / grouping states
  – If this information is irrelevant to the property of interest (i.e., the property is true on the original model iff it is true on the abstract model) then it is a precise abstraction
  – If the property is true on the original model if it is true on the abstract model, it is a safe abstraction
Example

• Abstractions exhibit more behaviors
• Consider the foll 2 properties on the original model and abstraction:
  \[ G(\text{go } \rightarrow \text{X stop}) \quad \text{G F go} \]

A Simple Form of Abstraction

• Suppose the temporal logic property mentions only a subset of variable \( V' \) of the entire set \( V \)
• Can I use this information to construct a precise abstraction of the original model?
A Simple Form of Abstraction

• Suppose the temporal logic property mentions only a subset of variable \( V' \) of the entire set \( V \)

• Can I use this information to construct a precise abstraction of the original model?
  – YES. One such method is the “cone of influence” reduction.
    • Transitively propagate syntactic dependences on variables and “delete” all variables not in the transitive closure

Cone-of-Influence Reduction

• A staple part of all model checkers

• However: often most of the variables remain in the cone-of-influence
  – Need further abstraction
  – Will be covered in class next week