Announcements

• Meet with me in early March to discuss your paper presentation
• Slots assigned in the order in which you will present (will be sent by e-mail)
• Default meeting time is my Mon/Wed office hour
Today’s Lecture

• Symbolic model checking with BDDs
  – Checking CTL properties: quick recap
  – Fairness
  – Counterexample/witness generation for general CTL
  – Optimizations
• Abstraction

Least and Greatest Fixpoints

• Let
  – $S = \{s_0, s_1\}$
  – $\tau(Z) = Z \cup \{s_0\}$, $Z \subseteq S$
• What’s the least fixpoint of $\tau$? The greatest fixpoint? Are they the same?

• Notation: “fixpoint” and “fixed point” sometimes used interchangeably
Model Checking CTL Properties

• We define a general recursive procedure called “Check” to do the fixpoint computations

• Definition of Check:
  – Input: A CTL property \( \Pi \) (and implicitly, \( R \))
  – Output: A Boolean formula \( B \) representing the set of states satisfying \( \Pi \)

• If \( S_0(v) \rightarrow B(v) \), then \( \Pi \) is true

The “Check” procedure

Cases:
• If \( \Pi \) is a Boolean formula, then \( \text{Check}(\Pi) = \Pi \)
• Else:
  – \( \Pi = \text{EX} \psi \), then \( \text{Check}(\Pi) = \text{CheckEX}(\text{Check}(\psi)) \)
  – \( \Pi = \text{E} (\psi_1 \ U \psi_2) \), then
    \[ \text{Check}(\Pi) = \text{CheckEU}(\text{Check}(\psi_1), \text{Check}(\psi_2)) \]
  – \( \Pi = \text{E} \ G \psi \), then \( \text{Check}(\Pi) = \text{CheckEG}(\text{Check}(\psi)) \)

• Note: What are the arguments to CheckEX, CheckEU, CheckEG? CTL properties or Boolean formulas?
CheckEU

• CheckEU(p, q) returns a set of states, each of which is such that
  – Either q is true in that state
  – Or p is true in that state and you can get from it to a state in which p U q is true

• Let $Z_0$ be our initial approximation to the answer to CheckEU(p, q)

• $Z_k(v) = \{ q(v) + [ p(v) . \exists v' \{ R(v, v') . Z_{k-1}(v') \} ] \}$

• What’s $Z_0$? Why will this terminate?

Counterexample/Witness Generation for CTL

• Counterexample = run showing how the property is violated
  – Formulas with universal path quantifier A
• Witness = run showing how the property is satisfied
  – Formulas with existential path quantifier E
  – Can also view as counterexample for the negated property
    • E.g. E G p and A F ¬ p
Witness Generation for EG p

• Fixpoint formulation for E G p:
  – $\nu Z . p \land EX Z$
  – $\tau(Z) = p \land EX Z$

• Fixpoint computation yields sequence $Z_0, Z_1, \ldots, Z_k$
  – $Z_0 = True$ (universal set)
  – $Z_1 = \tau(True) = ?$
  – each $Z_i$ is a BDD representing a set of states
  – How would you describe an element of $Z_i$?

• We need to generate the counterexample from $S_0, R, Z_0, Z_1, \ldots, Z_k$

Witness Generation for EG p

• Fixpoint computation yields sequence $Z_0, Z_1, \ldots, Z_k$
  – A state in $Z_i$ ($i > 0$) satisfies $p$ and there is a path of length $i-1$ from that state comprising states satisfying $p$
  – How would you describe an element of $Z_k$?
    • Remember: it’s the fixpoint
Witness Generation for EG p

- Fixpoint computation yields sequence $Z_0, Z_1, \ldots, Z_k$
  - A state in $Z_i$ satisfies $p$ and there is a path of length $i-1$ from that state comprising states satisfying $p$
  - How would you describe an element of $Z_k$?
    - State in $Z_k$ has path from it of length $k-1$ or more (including a cycle) with all states satisfying $p$
    - If $S_0$ is contained in $Z_k$, any initial state has such a path

Witness Generation for EG p

- Let $s_0$ be an initial state with a desired witness path
  - We need to reproduce one such witness
  - How can we do this?
Witness Generation for EG p

- Let $s_0$ be an initial state with a desired witness path
  - We need to reproduce one such witness
  - How can we do this?
    - Main insight: desired successor of $s_0$ also satisfies EG $p$, and so on
    - Look for a cycle in such a computed chain
      - Why should there be a cycle?

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 \( \text{EG} \ f \) 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 \( f \)

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 \( f \cup (Z \land p_i) \)
  – What’s the fixpoint formulation of \( \text{EG} \ f \) with fairness?
Fairness in Symbolic Model Checking of CTL

• 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 \( f \)
    • i.e., there is a next state of \( s \) satisfying \( f \cup (Z \land p_i) \)

\[
\forall Z. f \land (\land_i \text{EX} E[f \cup (Z \land p_i)])
\]

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 $\lor$? What use is that?

Conjunctive Partitioning

- Suppose we have a 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
  \[ \bigwedge_i R_i(v, v') \]
  - 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?

**Abstraction**

- Reduce the size of the system model by throwing out information
  - 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
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

Next class

• More on abstraction
• Symbolic model checking without BDDs