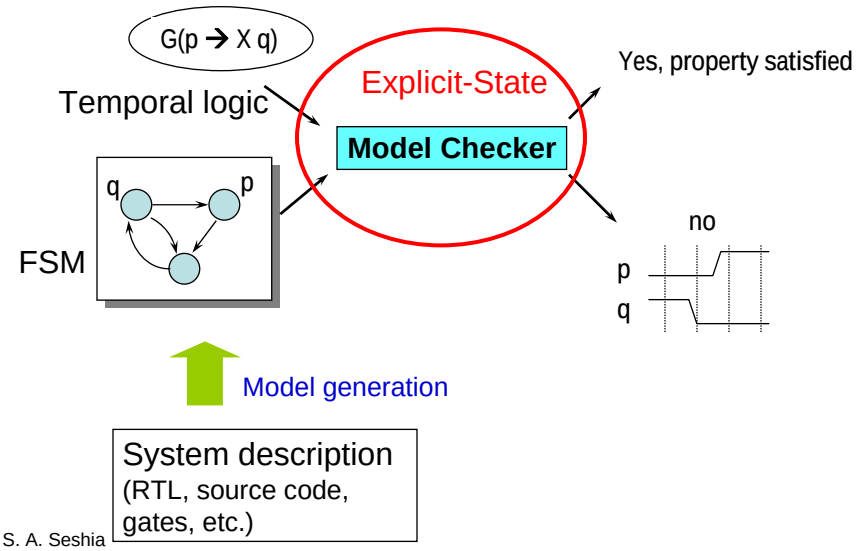
EECS 219C: Computer-Aided Verification

Properties as Automata and Explicit-State Model Checking

Sanjit A. Seshia
EECS, UC Berkeley

Finite-State Model Checking



Explicit-State Model Checking

- Model checking exhaustively enumerates the states of the system
- State space can be viewed as a graph
- Explicit-state model checking
  - Explicitly enumerates each state and traverses each edge of the graph
- We will focus on explicit-state techniques as used in SPIN [G. Holzmann, won ACM Software Systems Award]

Issues with Explicit-State MC

- The graph is usually HUGE (> 10^6 nodes)
  - So can't compute it a-priori
- But we are given an initial state (s_0) and a way of going from state to state (transition relation R)
  - In particular, we'll assume that R is specified as a "set of actions", each having a "enabling condition" and a "set of assignments" that cause a state change

Model Checking G p

- Consider the simplest property G p
  - p is a system invariant to be satisfied by all states
- Given the state graph, how can we check this?

Model Checking G p

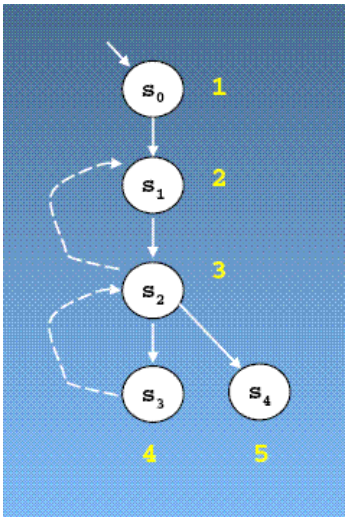
- Consider the simplest property G p
  - p is a system invariant to be satisfied by all states
- Given the state graph, how can we check this?
  - Graph traversal: DFS or BFS

Depth-First Search (DFS)

Maintain 2 data structures:

1. Set of visited states
2. Stack with current path from the initial state

Potential problems?

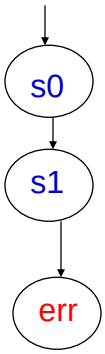


Generating counterexamples

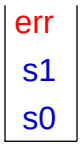
If the DFS algorithm finds an “error” state (in which p is not satisfied), how can we generate a counterexample trace from the initial state to that state?

Generating counterexamples

If the DFS algorithm finds an “error” state (in which p is not satisfied), how can we generate a counterexample trace from the initial state to that state?



Stack:



Will this be the shortest counterexample?

DFS without State Set

- Only keep track of current stack
- No set of states to maintain
  - Each time you visit a state, check whether it’s on the stack
    - If so, don’t explore its edges
    - If not, do.
- Q1: Will this terminate?
- Q2: If yes: on state graph with n states, how long will it take?

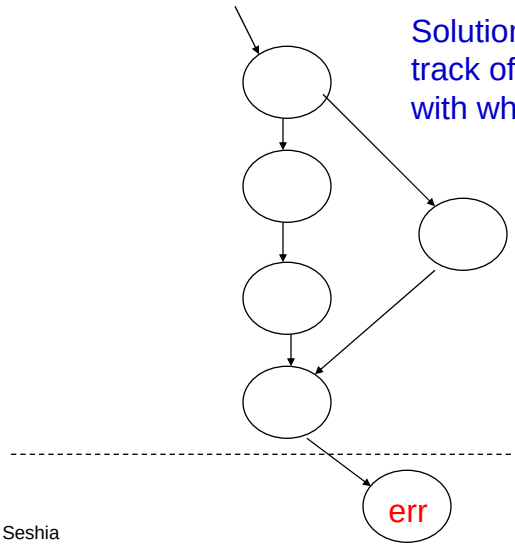
Bounded Model Checking with DFS

- Same as the original DFS, except that you only allow your stack to grow up to B elements deep
  - Keep track of set of all visited states and explore a state only if it is not in this set
- If this returns “no error within B steps from initial state”, can you trust it?

Bounded Model Checking with DFS

- Same as the original DFS, except that you only allow your stack to grow up to B elements deep
  - Keep track of set of all visited states and explore a state only if it is not in this set
- If this returns “no error within B steps from initial state”, can you trust it?
  - NO! Example on next slide

Example



Breadth-First Search

- Visit states in order of distance from initial state
- Uses queue, No stack: how to generate counterexamples?
- Are the generated counterexamples the shortest?

Comparing DFS and BFS for Gp

- Pros of BFS over DFS
  - Shortest counterexample generated
- Cons of BFS
  - Need to store back-pointers to predecessor with each state in the state space representation (increased memory requirement)
  - Does not efficiently extend to liveness properties
    - Need to do cycle detection

What about non-Gp safety properties?

- Recall: safety properties → finite counterexample trace
- So we can construct a monitor automaton with an "error" state that must be avoided
  - Construct product of that automaton with original system
  - Error state of product has "error" in the component corresponding to the monitor