An Introduction to Model Checking

Scott A. Smolka
Stony Brook University

Jan. 20, 2011
CMACS Ugrad Workshop
Outline

• What is CMACS?

• What is Model Checking?

• Model Checking Example

• Applying MC to Heart Cells
• 5-Year, $10M NSF Expedition in Computing Award
• 7 academic institutions + NASA JPL
• 18 Principal Investigators, with Ed Clarke of CMU the Lead Investigator
• Seek to apply next-generation Model Checking and Abstract Interpretation techniques to Biological and Embedded Systems
• Challenge problems in Pancreatic Cancer, Atrial Fibrillation, and Automotive & Aerospace systems
Does system model $M$ satisfy system property $\varphi$?

- $M$ given as a state machine.
- $\varphi$ usually specified in temporal logic.
Clarke Emerson Sifakis Receive 2007 Turing Award

... they developed this fully automated approach [Model Checking] that is now the most widely used verification method in the hardware and software industries.
Model Checking

Answer: Yes if model sat property
Counterexample otherwise
What is Model Checking?

[Clarke & Emerson 1981]:
“Model checking is an automated technique that, given a finite-state model of a system and a logical property, systematically checks whether this property holds for (a given initial state in) that model.”

Model checking tools automatically verify whether $M \models \varphi$ holds, where $M$ is a (finite-state) model of a system and property $\varphi$ is stated in some formal notation.

Problem: state space explosion! Although finite-state, the model of a system typically grows exponentially.
Common Design Flaws in Concurrent Systems

- Deadlock
- Livelock, starvation
- Underspecification
  - unexpected reception of messages
- Overspecification
  - Dead code
- Violations of constraints
  - Buffer overruns
  - Array bounds violations
- Assumptions about speed
  - Logical correctness vs. real-time performance
System Development

System Engineering → Analysis

Analysis → Design

Design → Code

Code → Testing

Testing → Maintenance

"Classic" Model Checking

"Modern" Model Checking

Classic “waterfall model” [Pressman 1996]
The SMV Model Checker

- Developed at CMU in the 1990s
- System model given as an FSA
- System properties given as \textit{CTL} formulas
- SMV program has 3 parts:
  - (finite) variable \textit{declarations}
  - (nondeterministic) variable \textit{assignments}
  - property \textit{specification}
A Simple Two-Tank Pumping System

Source Tank

Sink Tank
Pump System Specification

• Initially, both tanks are empty.
• **Pump** switched on as soon as water level in **tank A** reaches **ok**, provided **tank B** not full.
• **Pump** remains on as long as **tank A** not empty and **tank B** not full.
• **Pump** switched off as soon as **tank A** empty or **tank B** full.
• System should not attempt to switch the **pump** off (on) if it's already off (on).
Pumping System Specification (Part I)

MODULE main

VAR
level_a : {empty, ok, full}; -- source tank
level_b : {empty, ok, full}; -- sink tank
pump : {on, off};
Pumping System Specification (Part II)

ASSIGN
  next(level_a) := case
    level_a = empty : {empty, ok};
    level_a = ok & pump = off : {ok, full};
    level_a = ok & pump = on : {ok, empty, full};
    level_a = full & pump = off : full;
    level_a = full & pump = on : {ok, full};
    1 : {ok, empty, full};
  esac;
Pumping System Specification (Part III)

next(level_b) := case
    level_b = empty & pump = off : empty;
    level_b = empty & pump = on : {empty, ok};
    level_b = ok & pump = off : {ok, empty};
    level_b = ok & pump = on : {ok, empty, full};
    level_b = full & pump = off : {ok, full};
    level_b = full & pump = on : {ok, full};
    1 : {ok, empty, full};
esac;
next(pump) := case
    pump = off & (level_a = ok | level_a = full) &
    (level_b = empty | level_b = ok) : on;
    pump = on & (level_a = empty | level_b = full) : off;
    1 : pump; -- keep pump status as it is
esac;

INIT
    (pump = off)
Pumping System Specification (Part V)

SPEC

-- pump is always off if source tank is empty or sink tank is full
AG AF (pump = off -> (level_a = empty | level_b = full))

-- always possible to reach a state when the source tank is ok or full
AG EF (level_b = ok | level_b = full)
Model Executions

- Initially, system could be in any of nine states where no restrictions on water level in A or B but the pump is off.
- Denote a state by an ordered tuple \(<A,B,P>\) where A and B are current water level in tank A and B, and P is current pump status.
- Assume initial state to be \(<\text{empty, empty, off}>\).
- Next state could be \(<\text{empty, empty, off}>\) or \(<\text{ok, empty, on}>\).
- From \(<\text{ok, empty, on}>\), next state could be \(<\text{ok, empty, on}>\), \(<\text{ok, ok, on}>\), \(<\text{full, empty, on}>\), \(<\text{full, ok, on}>\), \(<\text{empty, empty, off}>\), or \(<\text{empty, ok, off}>\).
- For each of these states, we could calculate next possible states.
Initial Portion of Execution Tree
The temporal logic CTL allows us to specify properties of paths (and states along paths) of an execution tree. It is an extension of Boolean propositional logic.

**CTL Operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>EX $\phi$</td>
<td>true in current state if formula $\phi$ is true in at least one of the next states</td>
</tr>
<tr>
<td>EF $\phi$</td>
<td>true in current state if there exists some state in every path beginning in current state that satisfies the formula $\phi$</td>
</tr>
<tr>
<td>EG $\phi$</td>
<td>true in current state if every state in every path beginning in current state satisfies the formula $\phi$</td>
</tr>
<tr>
<td>AX $\phi$</td>
<td>true in current state if formula $\phi$ is true in every one of the next states</td>
</tr>
<tr>
<td>AF $\phi$</td>
<td>true in current state if there exists some state in every path beginning in current state that satisfies the formula $\phi$</td>
</tr>
<tr>
<td>AG $\phi$</td>
<td>true in current state if every state in every path beginning in current state satisfies the formula $\phi$</td>
</tr>
</tbody>
</table>

**E** (for some path) and **A** (for all paths) are *path quantifiers* for paths beginning from a state. **F** (for some state) and **G** (for all states) are *state quantifiers* for states along a path.
Intuition for CTL Operators

(a) EF q
(b) AF q
(c) EG q
(d) AG q
Simple CTL Properties of Pump System

• **AF (pump = on).** For every path beginning at initial state, there's state in that path at which pump is on.

• **False**, since there's a path from initial state in which the pump always remains off (e.g., if tank A forever remains empty).

• SMV generates following **counterexample**. (Loop indicates infinite sequence of states beginning at initial state such that tank B is full in every state of path and hence pump is off.)
SMV Counterexample

-- specification AF pump = on is false
-- as demonstrated by following execution sequence
-- loop starts here
  state 1.1:
  level_a = full
  level_b = full
  pump = off
  state 1.2:
Another Simple CTL Property

• Dual property $AF \ (pump = \text{off})$. For every path beginning at initial state, there's a state in that path at which the pump is off.

• Trivially true, since in the initial state itself (which is included in all paths) $pump = \text{off}$ is true.
What does MC have to do with Bio?

• And what does it have to do with heart cells, and atrial fibrillation, and … ?

• Can view Flavio’s minimal model as a special kind of state machine and try to apply MC to that!
Hybrid Automaton Model: Cardiac Cell
Hybrid Automaton Model: Cardiac Cell

- Early Repolarization
- Plateau
- Upstroke
- Stimulated
- Final Repolarization
- Resting

- $v \geq V_E$
- $v \leq V_P$
- $v \leq V_F$
- $v \leq V_R$

- $v \geq V_U$
- $s_{on}$
- $s_{off} \land v < V_U$
- $s_{off}$
- $v < V_U$
Hybrid Automaton Model: Cardiac Cell

HA Network (Spatial) Simulation

- Fibrillation/Defibrillation protocol
- 400 x 400 HA cell array
(Finite) Mode Abstraction

- Preserves spatial properties \(4^{160,000}\) images
CMACS Wants You!

- NSF REUs
- Summer Internships
- RAs in CMACS graduate programs
I'll climb up this strand of DNA to see where life takes me.
Emergent Behavior in Cardiac Tissue

ECG

Surface

Simulation

Experiment