Model Checking

Demonstration Of
SPIN

By
Mitra Purandare

Motivation

- More and more complex systems
- Increased dependability: everything important depends on computers
- Increased functionality: security, mobility
- Testing is becoming humanly un-manageable!

Testing

- Dynamic Execution/Simulation of System
- Generating test-cases: Limited by tester's ability to devise test-cases
- To Prove: Absence of a certain bug?
- To Prove: Presence of a certain property?
- Is CSARDAS 100% correct?
- Testing: Not formal/Mathematical!

Implications

- More efficient methods for test and verification needed.
- Formal Verification is the most promising approach.
- Experts in these new methods lacking!!
- Job: A great motivation to study formal methods!

Formal Verification

- Problem: Does an implementation satisfy a property?
- Two Basic categories
  - Theorem provers: infinite state systems, time-consuming, not really automated
  - Model Checkers: exhaustive state space exploration, finite state systems, automated

Model Checking

- Introduced by Clarke and Emerson, Quielle and Sifakis in 1981
- Given a property (P) and a system (M), does M \models P?
- Yes, P holds in M
- No, generate a trace which shows the property violation
**Example: Model and Property**

- **Model:** Kripke Structure, Finite State Machine, Automaton
- **Property:** CTL/LTL
  - Safety Property
  - Liveness Property

```
Model: Kripke Structure, Finite State Machine, Automaton
Property: CTL/LTL
```

\[
\begin{align*}
\text{Result} & : \text{CTL/LTL} \\
G \neg p & : a, b, c \\
\neg G \neg p & : a, b, a, b, a, ... \\
F q & : a, b, a, b \\
q U p & : \\
\end{align*}
\]

**Tools**

- SPIN (Bell Labs)
- SMV, NuSMV (CMU)
- Mocha (Penn)
- JPF (Java Path Finder, NASA)
- Bandera (KSU)
- BLAST (Berkeley)
- MAGIC (CMU)
- FormalCheck (Cadence)
- RuleBase (IBM, Haifa)
- SLAM, Zing (Microsoft Research)
- FormalPro (Mentor Graphics)

**SPIN (Simple Promela Interpreter)**

- Developed by G.Z. Holzman@Bell Labs
- Promela (PROtocol MEta LAnguage)
- Publicly available since 1991
- Prestigious ACM System Software Award for 2001
- Most efficient and scalable
- Still active research -> good support

**SPIN**

- Explicit state LTL model checker
- On-the-fly reachability
- Partial order reduction to reduce state space
- Targets software verification
- Scales well for large problem sizes

**The Cabbage-Goat-Wolf problem!**

- Ferryman with C, G, W and a boat on one side of a river
- Only ferryman can row the boat
- Ferryman can take only one item at a time
- Not goat and wolf together without ferryman
- Not goat and cabbage together without ferryman
- GOAL: Ferryman wants to take all 3 items to the other side!
**Property**

- Goal: \( \text{wolf\_location} = \text{destination} \land \text{goat\_location} = \text{destination} \land \text{ferryman\_location} = \text{destination} \)
- Restriction 1: \( \text{wolf\_location} = \text{goat\_location} \land \text{ferryman\_location} \neq \text{wolf\_location} \)
- Restriction 2: \( \text{goat\_location} = \text{cabbage\_location} \land \text{ferryman\_location} \neq \text{cabbage\_location} \)
- \!(\text{Restriction 1} \lor \text{Restriction 2}) \lor \text{Goal}

**State Transition Diagram**

- 4 variables, ferryman, cabbage, goat, wolf respectively
- 1: on this bank, 0: other bank i.e. destination

**SPIN References**

- Model Checking: Clarke, Grumberg and Peled
- Symbolic Model Checking: Kenneth L. McMillan
- OR Come To the H-Floor! :)

**Challenges**

- Coverage
- Reliability
- Repair
- Scalability
- Infinite State System
- Specification
- InterOperability

**Future**

- Bounded Model Checking
- SAT Solvers
- Abstraction and refinement
- Hybrid Systems