Model Checking Distributed Software

Sagar Chaki
September 19, 2014

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Model Checking and Me

1997 : Ed visits IIT Kharagpur
  • Just finished 2nd year undergrad
  • Couldn’t understand most of the talks

1998-99: Final year UG
  • Developed symbolic model checker, read papers, found a typo

1999-2004: CMU PhD
  • Some of the best years of my life
  • Did not coin the term “CEGAR”
  • Ed, Martha, Pankaj, Somesh, Orna, Helmut, Daniel, Joel, …

2004-Present: SEI
  • Verifying Cyber Physical Systems
  • Meetings & Lunch in my office responding to email
Motivation

Distributed algorithms have always been important

• File Systems, Resource Allocation, Internet, …

Increasingly becoming safety-critical

• Robotic, transportation, energy, medical

Prove correctness of distributed algorithm implementations

• Pseudo-code is verified manually (semantic gap)
• Implementations are heavily tested (low coverage)
Approach: Verification + Code Generation

Program in Domain Specific Language

- Distributed Application
- Safety Specification

Debug Application, Refine Specification

Verification

Failure → Success

Code Generation

Binary

- Run on Physical Device
- Run within simulator

The Verifying Compiler: A Grand Challenge for computing research

Tony Hoare
Verification

Program in Domain Specific Language

Distributed Application

Sequentialization (assuming synchronous communication)

Single-Threaded C Program

Software Model Checking (CBMC, BLAST etc.)

Safety Specification

Failure

Success

Model Checking

Automatic verification technique for finite state concurrent systems.

- Developed independently by Clarke and Emerson and by Queille and Sifakis in early 1980’s.
- ACM Turing Award 2007

Specifications are written in propositional temporal logic. (Pnueli 77)

- Computation Tree Logic (CTL), Linear Temporal Logic (LTL), …

Verification procedure is an intelligent exhaustive search of the state space of the design
Code Generation

Program in Domain Specific Language

- Distributed Application
- Safety Specification

Add synchronizer protocol

C++/MADARA Program

Compile (g++, clang, MSVC, etc.)

Binary

MADARA Middleware

A database of facts: $DB = Var \leftrightarrow Value$

Node $i$ has a local copy: $DB_i$

- update $DB_i$ arbitrarily
- publish new variable mappings
  - Immediate or delayed
  - Multiple variable mappings transmitted atomically

Implicit “receive” thread on each node

- Receives and processes variable updates from other nodes
- Updates ordered via Lamport clocks

Portable to different OSes (Windows, Linux, Android etc.) and networking technology (TCP/IP, UDP, DDS etc.)
Case Study: Synchronous Collision Avoidance
Example: Synchronous Collision Avoidance
Example: Synchronous Collision Avoidance

(0,3) (3,3)

(0,0) (3,0)

Reserve

Reserve

Reserve

X

Y
Example: Synchronous Collision Avoidance

Reservation Contention Resolved based on Node ID. No collision possible if no over-booking.
Collision Avoidance Protocol

- **REQUEST**
  - If time to move to next coordinate
  - If no other node is locking the next coordinate

- **NEXT**
  - Reached the next coordinate
  - Moving to the next coordinate

- **MOVE**
  - If no other node “with higher id” is trying to lock the next coordinate

- **WAITING**
  - If no other node is trying to lock the next coordinate
Synchronous Collision Avoidance Code

MOC_SYNC;

CONST X = 4; CONST Y = 4;
CONST NEXT = 0;
CONST REQUEST = 1;
CONST WAITING = 2;
CONST MOVE = 3;

EXTERN int
MOVE_TO (unsigned char x, unsigned char y);

NODE uav (id) { ... }

void INIT () { ... }

void SAFETY { ... }

INIT {

FORALL_NODE(id)
state.id = NEXT;
//assign x.id and y.id non-deterministically
//assume they are within the correct range
//assign lock[x.id][y.id][id] appropriately

FORALL_DISTINCT_NODE_PAIR (id1,id2)
ASSUME(x.id1 != x.id2 || y.id1 != y.id2);

FORALL_DISTINCT_NODE_PAIR (id1,id2)
ASSERT(x.id1 != x.id2 || y.id1 != y.id2);

SAFETY {

FORALL_DISTINCT_NODE_PAIR (id1,id2)
ASSUME(x.id1 != x.id2 || y.id1 != y.id2);

} }

global bool lock [X][Y][#N];
local int state,x,y,xp,yp,xf,yf;

void NEXT_XY () { ... }
void ROUND () {
   if(state == NEXT) { ...
      state = REQUEST;
   } else if(state == REQUEST) { ...
      state = WAITING;
   } else if(state == WAITING) { ...
      state = MOVE;
   } else if(state == MOVE) { ...
      state = NEXT;
   }
}

NODE uav (id) {
   ... }
Synchronous Collision Avoidance Code

```c
if(state == NEXT) {
    // compute next point on route
    if(x == xf && y == yf) return;
    NEXT_XY();
    state = REQUEST;
} else if(state == REQUEST) {
    // request the lock but only if it is free
    if(EXISTS_OTHER(idp, lock[xp][yp][idp] != 0)) return;
    lock[xp][yp][id] = 1;
    state = WAITING;
} else if(state == WAITING) {
    // grab the lock if we are the highest
    // id node to request or hold the lock
    if(EXISTS_HIGHER(idp, lock[xp][yp][idp] != 0)) return;
    state = MOVE;
}
else if(state == MOVE) {
    // now we have the lock on (xp, yp)
    if(MOVE_TO()) return;
    lock[x][y][id] = 0;
    x = xp; y = yp;
    state = NEXT;
}
```
Tool Usage

Project webpage (http://mcda.googlecode.com)
  • Tutorial (https://code.google.com/p/mcda/wiki/Tutorial)

Verification
  • daslc --nodes 3 --seq --rounds 3 --seq-dbl --out tutorial-02.c tutorial-02.dasl
  • cbmc tutorial-02.c (takes about 10s to verify)

Code generation & simulation
  • daslc --nodes 3 --madara --vrep --out tutorial-02.cpp tutorial-02.dasl
  • g++ ...
  • mcda-vrep.sh 3 outdir ./tutorial-02 ...
Demonstration: Synchronous Collision Avoidance
## Contact Information Slide Format

<table>
<thead>
<tr>
<th><strong>Sagar Chaki</strong></th>
<th><strong>U.S. Mail</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Principal Researcher</td>
<td>Software Engineering Institute</td>
</tr>
<tr>
<td>SSD/CSC</td>
<td>Customer Relations</td>
</tr>
<tr>
<td>Telephone: +1 412-268-1436</td>
<td>4500 Fifth Avenue</td>
</tr>
<tr>
<td>Email: <a href="mailto:chaki@sei.cmu.edu">chaki@sei.cmu.edu</a></td>
<td>Pittsburgh, PA 15213-2612</td>
</tr>
<tr>
<td></td>
<td>USA</td>
</tr>
</tbody>
</table>

### Web

- [www.sei.cmu.edu](http://www.sei.cmu.edu)
- [www.sei.cmu.edu/contact.cfm](http://www.sei.cmu.edu/contact.cfm)

### Customer Relations

- Email: [info@sei.cmu.edu](mailto:info@sei.cmu.edu)
- Telephone: +1 412-268-5800
- SEI Phone: +1 412-268-5800
- SEI Fax: +1 412-268-6257