Temporal Skeletons for Verifying Time

Gustaf Naeser, Kristina Lundqvist and Lars Asplund
Mälardalen University
And
MIT

SigAda 2005
2005-11-15

Outline

- Formal Methods
- Hardware Monitoring

Application: Robocup – Soccer playing robots

Assuring that a system works

- There are several strategies for ensuring that a developed system meets the safety demands that
  - Testing
  - Inspection
  - Reviewing
  - Simulation
  - Formal verification

Formal verification

- Formal methods = mathematically based
  - The word mathematics often scares designers and developers
  - Likely to require domain experts
  - Complement to testing

- Two kinds
  - Theorem proving
  - Model checking

The SafetyChip framework parts

The Framework
Modelling
Can be visualised over the original source code

Verification
The model and the RTK are used to verify the behaviour of the system

The SafetyChip framework
- Desire to use FM in development of embedded high-integrity real-time systems
  - Must know properties of the software
  - Must know properties of the hardware
  - Reduce investment
    - Between single computer and triple redundant
  - Reuse investment

Ada Ravenscar source code and ASIS
- Source code parsed using the Ada Semantic Interface Specification
  - ASIS provides a parse tree
  - Additional information must be added (scopes are not available in ASIS)
  - ASIS can effectively be used to decide static properties of a program
  - Dynamic properties cannot be decided

Intermediate code
- Intermediate code is used in order to reuse of the framework itself, i.e., to allow several front ends and back ends
  - Frontends: Input languages
    - Ada, VHDL
  - Backends: Verification tools
    - UPPAAL
- Automated transformation to and from the intermediate format!

The Ada code for Task2
```ada
task Task2;
  task body Task2 is
    T_2 : Task_ID := 2;
    begin
      loop
        FO.Open(T_2);
        if (Global = 30) then
          Global := 0;
          Work(0, 0, 3, T_2);
          FO.Call(T_2);
        end if;
      end loop;
      end Task2;
```
**Temporal skeletons**

- Blocks of instructions that only consume time (full lined boxes)
- Blocks with instructions that “actively” change temporal behaviour (dashed boxes)

**UPPAAL automata**

- UPPAAL used for verification
- Skeletons are automatically translated to the tool’s TA
- TA optimisation to reduce state space

**VHDL Translator**

- Enable formal verification of legacy components

**RoboCup**

- 1997 Deep Blue beat Kasparov in chess
- In 50 years robots are going to beat the humans in Soccer
- Different Leagues
Mini League

- A video camera above the field
- Radio communication 433MHz
- Two motors
- Rotating disc as kicker
- Field the size of table tennis table

Middle Sized League

- Field size 5m*8m
- Robot size < 50 (45) height < 80 cm weight < 80 kg!!
- No global sensors
- Communication OK
- Server
- Colour scheme robots, field, goals and ball

Sony Legged Robots

- Radio communication 433MHz
- Two motors
- Rotating disc as kicker
- Field the size of table tennis table

Swedish Wheels

- Field size 5m*8m
- Robot size < 50 (45) height < 80 cm weight < 80 kg!!
- No global sensors
- Communication OK
- Server
- Colour scheme robots, field, goals and ball
**Embedded System**

- Software on the robots are Ada-95
  - Ravenscar profile
  - FPGA for all IO; VHDL
- Software in the coach computer is full Ada
- Communication between server and robots are by using RadioLAN
- Communication inside the robot is by using an Optical CAN-bus

---

**Aros Robot**

- Two disks
- Omnidirectional wheels
- Power-pole
- Kicker

---

**Team Aros**

---

**Translation**

---

**Rotation**
SafetyChip
A Time Monitoring and Policing Device

Gustaf Naeser, Lars Asplund and Johan Furunäs
Mälardalen University

Outline
- Formal Methods
- **Hardware Monitoring**

Application: Robocup – Soccer playing robots

Hardware Monitoring and Policing
- How can the operation of a system during run time be
  - Observed
  - Modified
- Aids development
- The largest part of a system’s life is after deployment
Kinds of monitoring

**Intrusive**
- Changes the behaviour of the monitored system
  - Code is added to generate observable events
  - Monitor can be off target
  - Instructions are kept in the final system to keep the behaviour

**Non-intrusive**
- Does not change the systems behaviour
  - Harder to implement since it requires access to tap points “inside” the system

The SafetyChip framework parts

- Application
- FM
- RTK
- HW

The Framework

- Component based
- Allows timing analysis

System on Chip

A Ravenscar RTK
- Component based
- Allows timing analysis

Monitoring and Policing

Reuses the model from the verification

Non-Intrusive Fault-Tolerance

- RavenHART
- JMDA RAM
- FPGas
- Monitoring Chip
- System Master

IP CORE

DATABUS
Developments in hardware

- Field Programmable Gate Arrays (FPGA) is a programmable hardware combining properties from both hardware and software
  - Faster than SW, slower than HW (ASIC)
  - Easier to change than HW, harder than SW

RavenHaRT

- Hardware implemented RTK
  - Based on the Ravenscar profile of Ada95
  - Implemented using the Xilinx Virtex II Pro
  - Deterministic run-time behavior
  - Non-Deadlocking Inter-task communication

RTK

- Ravenscar profile supported
- Supports a more functionality at low cost
  - e.g., dynamic priorities
  - Multiple processors
- Component design to allow
  - Easy change
  - HW/SW locality in final system
- Implemented using generic templates
  - Allows the kernel to be tailored to each application

RTK components

- Ready Queue
- Delay Queue
- Protected Objects Queue
- Interrupt Queue (uses PO)
- Hardware
  - Processors with null tasks
  - clock
- Application
  - Tasks and protected objects

The Ada Code for task T1

UPPAAL of Task T1
**Timing of Task T1**

- create
- runnable
- run
- change
- noop
- call

---

**Task T3**

---

** Behaviour for Task T3**

<table>
<thead>
<tr>
<th>Address</th>
<th>Work</th>
<th>Code</th>
<th>Node</th>
<th>REU</th>
<th>MeU</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00</td>
<td>02</td>
<td>03</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00</td>
<td>04</td>
<td>05</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00</td>
<td>06</td>
<td>07</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00</td>
<td>08</td>
<td>09</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00</td>
<td>10</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00</td>
<td>12</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00</td>
<td>14</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

---

**RoboCup**

- Continued

---

**Team Aros**

---
No Synchronization

Searching

Synchronized - contact

IR/Ultra between 4 robots

Ultra sonic ‘radar’

Ir – Collision Avoidance
Gyro

Rotation

X1 = Omega * R
X2 = -Omega * R
Y1 = 0
Y2 = 0
X1-X2 = 2 * Omega * R
X1+X2 = 0
Y1+Y2 = 0
X1-X2 styr motorn (kompensation)

X-translaction

X1 = a (v eller s)
X2 = a
Y1 = 0
Y2 = 0
X1-X2 = 0
X1+X2 = 2 * a
Y1+Y2 = 0

Y-translaction

Y1 = 0
X2 = 0
Y1 = a
Y2 = a
X1-X2 = 0
X1+X2 = 0
Y1+Y2 = 2 * a

Murphy

Arm 2*(2+1) = 6
Back = 3
Leg 2*(3+1) = 8
Feet 2*2 = 4
Total = 21
New Name = Anders??

Soccer playing robot Anders

The End