Real-Time Systems

Lecture 21: Wrapup & Questions

2018-02-06

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Content

- Lecture 20 Continued:
  - Formal Methods in the Development Process
    - Verification
    - Model Decomposition, Resource Consumption
  - Conclusion

- Lecture 21: Code Generation

- Looking Back (and Forward: Exam)

- Advertisements
The Story So Far...

Project, Situation, Requirements

The Project: Wireless Fire Alarm System

- Develop new communication protocol for wireless fire alarm systems (WFAS).
- Main functionality:
  - self-monitoring and (display non-operational sensors at central unit)
  - alarm notification (display fire indications (smoke, heat, etc.) at central unit)
- Timing constraints are regulated by European Norm EN 54, Part 25.
- Goal: satisfy EN 54-25 — and have a good, robust, efficient overall product.

Formal Methods for SME

- Two broad directions:
  - Option 1: teach DC (usually not economic).
  - Option 2: serve as translator / mediator.

Requirements Validation Cont’d

- Option 1: use SMEs, quality not measurements.
- Option 2: use frameworks / evaluations.

Self-Monitoring Sensor
The Project: Wireless Fire Alarm System

- Develop new communication protocol for wireless fire alarm systems (WFAS).
- Main functionality:
  - self-monitoring and display non-operational sensors at central unit.
  - Alarm notification and display fire indications (smoke, heat, etc.) at central unit.
- Timing constraints are regulated by European Norm EN 54, Part 25.
- Goal: satisfy EN 54-25 — and have a good, robust, efficient overall product.

Formal Methods for SME

Design SME

Glossary

Detailed Models

Formal Specifications

Formalized Requirements

Verification Results

Finished Product

Requirements Validation Cont’d

Two broad directions:
- Option 1: teach DC (usually not economic).
- Option 2: serve as translator / mediator.

Self-Monitoring: Sensor

Self-Monitoring: Model Architecture

Content

- Lecture 20 Continued:
  - Formal Methods in the Development Process
    - Verification
      - Model Decomposition, Resource Consumption
    - Conclusion
  - Lecture 21: Code Generation
  - Looking Back (and Forward: Exam)
  - Advertisements
Formal Verification
**Queries:**

- $E<> \text{switcher.DETECTION}$
  
  **sanity-check:** "it is possible to detect one missing sensor"
  
  (check with sensor switcher and with channel blocker)

- $A[] \text{not deadlock}$
  
  **sanity-check:** no deadlock

- $A[] (\text{switcher.DETECTION imply switcher.timer <= 300*Second})$  
  
  **requirement:** "detection takes at most 300 s"
  
  (check with sensor switcher and with channel blocker)

- $A[] !\text{center.ERROR}$
  
  **requirement:** "no spurious errors"
  
  (check without sensor switcher, with channel blocker)
## Verification Results: Self-Monitoring

### Sensors as slaves, N = 126.

<table>
<thead>
<tr>
<th>Query</th>
<th>States explored</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection possible</td>
<td>26,445,788</td>
</tr>
<tr>
<td>$E&lt;&gt;$ switcher.DETECTION</td>
<td></td>
</tr>
<tr>
<td>No message collision</td>
<td>68,022,052</td>
</tr>
<tr>
<td>$A[]$ not deadlock</td>
<td></td>
</tr>
<tr>
<td>Detect$_T$</td>
<td>190,582,600</td>
</tr>
<tr>
<td>$A[]$ (switcher.DETECTION imply switcher.timer &lt;= 300*Second)</td>
<td></td>
</tr>
<tr>
<td>NoSpur$_T$</td>
<td>640,943</td>
</tr>
<tr>
<td>$A[]$ !center.ERROR</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query</th>
<th>States explored</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection possible</td>
<td>10,205.13</td>
</tr>
<tr>
<td>$E&lt;&gt;$ switcher.DETECTION</td>
<td></td>
</tr>
<tr>
<td>No message collision</td>
<td>12,895.17</td>
</tr>
<tr>
<td>$A[]$ not deadlock</td>
<td></td>
</tr>
<tr>
<td>Detect$_T$</td>
<td>36,070.78</td>
</tr>
<tr>
<td>$A[]$ (switcher.DETECTION imply switcher.timer &lt;= 300*Second)</td>
<td></td>
</tr>
<tr>
<td>NoSpur$_T$</td>
<td>97.44</td>
</tr>
<tr>
<td>$A[]$ !center.ERROR</td>
<td></td>
</tr>
</tbody>
</table>

### Repeaters as slaves, N = 10.

<table>
<thead>
<tr>
<th>Query</th>
<th>States explored</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection possible</td>
<td>1,250,596</td>
</tr>
<tr>
<td>$E&lt;&gt;$ switcher.DETECTION</td>
<td></td>
</tr>
<tr>
<td>No message collision</td>
<td>9,600,062</td>
</tr>
<tr>
<td>$A[]$ not deadlock</td>
<td></td>
</tr>
<tr>
<td>Detect$_T$</td>
<td>6,009,120</td>
</tr>
<tr>
<td>$A[]$ (switcher.DETECTION imply switcher.timer &lt;= 300*Second)</td>
<td></td>
</tr>
<tr>
<td>NoSpur$_T$</td>
<td>3.94</td>
</tr>
<tr>
<td>$A[]$ !center.ERROR</td>
<td></td>
</tr>
</tbody>
</table>

(Opteron 6174 2.2Ghz, 64GB, UPPAAL 4.1.3 (64-bit), options -s -t0 -u)
### Models and Corresponding Sizes

<table>
<thead>
<tr>
<th>Model</th>
<th>Templates</th>
<th>Instances</th>
<th>Total Locations</th>
<th>Clocks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Self-Monitoring:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensors as slaves</td>
<td>9</td>
<td>137</td>
<td>1040</td>
<td>6</td>
</tr>
<tr>
<td>Repeaters as slaves</td>
<td>9</td>
<td>21</td>
<td>82</td>
<td>6</td>
</tr>
<tr>
<td><strong>Alarm:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>One alarm</td>
<td>6</td>
<td>16</td>
<td>101</td>
<td>16</td>
</tr>
<tr>
<td>Two alarms in 2 seconds</td>
<td>5</td>
<td>16</td>
<td>108</td>
<td>12</td>
</tr>
<tr>
<td>Ten simultaneous alarms</td>
<td>6</td>
<td>25</td>
<td>200</td>
<td>15</td>
</tr>
</tbody>
</table>

### From DC Formulae to Queries: Alarm

- **Queries:**
  - \( A[] \neg \text{Center.ALARMED} \text{ imply time} < 10\text{\*Second} \)
    
    **requirement:** \text{“exactly one alarm displayed within 10 s”}
  - \( A[] \left( \neg \text{Sensor0.DONE} \lor \neg \text{Sensor1.DONE} \right) \text{ imply time} \leq 10\text{\*Second} \)
    
    **requirement:** \text{“exactly two (simultaneous) alarms displayed within 10 s”}
  - \( A[] \left( \neg \text{Sensor0.DONE} \lor \neg \text{Sensor1.DONE} \lor \ldots \lor \neg \text{Sensor9.DONE} \right) \text{ imply time} \leq 100\text{\*Second} \)
    
    **requirement:** \text{“exactly ten (simultaneous) alarms displayed within 100 s”}
### Verification Results: Alarm

#### Time $T = T_1$ (palm tree, full collision)

<table>
<thead>
<tr>
<th>Query</th>
<th>ids</th>
<th>$T$ (seconds)</th>
<th>$MB$ (MB)</th>
<th>States expl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm1</td>
<td></td>
<td>3.6 ± 1</td>
<td>43.1 ± 1</td>
<td>59k ± 15k</td>
</tr>
<tr>
<td>A[] !Center.ALARMED imply time &lt; 10*Second</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alarm2</td>
<td>sequential</td>
<td>4.7</td>
<td>67.1</td>
<td>110, 207</td>
</tr>
<tr>
<td>A[] (Sensor0.DONE</td>
<td></td>
<td>Sensor1.DONE) imply time &lt;= 10*Second</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alarm10</td>
<td>sequential</td>
<td>44.6 ± 11</td>
<td>311.4 ± 102</td>
<td>641k ± 159k</td>
</tr>
<tr>
<td>A[] (!Sensor0.DONE</td>
<td></td>
<td>!Sensor1.DONE</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Alarm10</td>
<td>optimized</td>
<td>41.8 ± 10</td>
<td>306.6 ± 80</td>
<td>600k ± 140k</td>
</tr>
</tbody>
</table>

#### Time $T = T_2$ (palm tree, limited collision)

<table>
<thead>
<tr>
<th>Query</th>
<th>ids</th>
<th>$T$ (seconds)</th>
<th>$MB$ (MB)</th>
<th>States expl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm1</td>
<td></td>
<td>1.4 ± 1</td>
<td>38.3 ± 1</td>
<td>36k ± 14k</td>
</tr>
<tr>
<td>A[] !Center.ALARMED imply time &lt; 10*Second</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alarm2</td>
<td>sequential</td>
<td>0.5</td>
<td>24.1</td>
<td>19, 528</td>
</tr>
<tr>
<td>A[] (Sensor0.DONE</td>
<td></td>
<td>Sensor1.DONE) imply time &lt;= 10*Second</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alarm10</td>
<td>sequential</td>
<td>17.3 ± 6</td>
<td>179.1 ± 61</td>
<td>419k ± 124k</td>
</tr>
<tr>
<td>A[] (!Sensor0.DONE</td>
<td></td>
<td>!Sensor1.DONE</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Alarm10</td>
<td>optimized</td>
<td>17.1 ± 6</td>
<td>182.2 ± 64</td>
<td>412k ± 124k</td>
</tr>
</tbody>
</table>

### Testing the Real System

<table>
<thead>
<tr>
<th>Test Scenario</th>
<th>First Alarm</th>
<th>All 10 Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model sequential</td>
<td>3.26s</td>
<td>29.03s</td>
</tr>
<tr>
<td>Model optimized</td>
<td>2.14s</td>
<td>27.08s</td>
</tr>
<tr>
<td>Model test scenario</td>
<td>3.31s</td>
<td>29.81s</td>
</tr>
<tr>
<td>Measured Avg.</td>
<td>2.79s ± 0.53s</td>
<td>29.65s ± 3.26s</td>
</tr>
</tbody>
</table>
Conclusion
### Conclusion

- Verifying "a whole system design" (i.e., every bit and detail of: car, plane, even WFAS) can be very expensive, gaining confidence into "the core design ideas" (or crucial aspects of the design) can be much more feasible.

- **One approach:**
  - fix a **budget** (time, effort, ...),
  - identify and **formalise** core requirements (balance priority and budget),
  - **validate** using positive / negative examples,
  - **model** as far as possible, on an appropriate level of abstraction (balance level of detail and budget),
  - **validate** using simulation of example runs,
  - **verify** as far as possible (if infeasible: limit considered scenarios, at least simulate).

- **Other way round:** fix the **goal** of the formal analysis.
In my opinion,

- **Everybody in this room** (or on the “broadcast receiver” at home)

- has been exposed to all the **knowledge** and **experience**

- that it takes to **do the WFAS project**.

**What’s your opinion?**
Motivation (F.A., W., et al., FAOC, 2016)

Question: Can't we generate the code automatically?
**Code Generation from TA in the Literature**


**The Rendezvous Transition Rule may Block Senders**

**Example:** (sender blocked in some configurations)

| l₀ | a! | l₁ |
| m₀ | m₁ | m₂ |

**Example:** (sender never blocked)

| l₀ | x > 1 | a! | l₁ |
| m₀ | m₁ | m₂ |

**Another Example:** (one of the senders blocked)

| l₀ | a! | l₁ |
| m₀ | m₁ | n₀ | a! | n₁ |
Recall: Operational Semantics of Networks of TA

Operational semantics:
labelled transition relations \( \Rightarrow \subseteq \text{Conf}(N) \times \text{Conf}(N) \), \( \text{Conf}(N) = \{ \langle \ell, \nu \rangle \mid \nu \models I(\ell) \} \).

- **(delay transition)** \( \langle \ell, \nu \rangle \xrightarrow{t} \langle \ell, \nu + t \rangle, t \in \mathbb{R}_0^+, \) if and only if \( \forall t' \in [0, t] \bullet \nu + t' \models I(\ell) \).

- **(local action transition)** \( \langle \ell, \nu \rangle \xrightarrow{e} \langle \ell', \nu' \rangle, \) if and only if there is an edge \( e = (\ell, \tau, \varphi, \ell', \nu') \in A_i \) such that \( \langle \ell, \nu \rangle \vdash e \text{ and } \langle \ell', \nu' \rangle = \langle \ell, \nu \rangle [e] \).

- **(rendezvous transition)** \( \langle \ell, \nu \rangle \xrightarrow{e_0} \langle \ell_0, \nu' \rangle, \) if and only if there is an edge \( e_0 = (\ell, a!, \varphi, \ell_0, \nu') \) in \( A_i \) such that \( \langle \ell, \nu \rangle \vdash e_0 \), and there is an edge \( e_1 = (\ell, a?, \varphi, \ell_1, \nu') \) in \( A_j \), \( i \neq j \), such that \( \langle \ell, \nu \rangle \vdash e_1 \), and \( \langle \ell, \nu \rangle = \langle \ell_0, \nu' \rangle [e_0; e_1] \).

Characterising “Dependency on Global Scheduler”

**Lemma.** A closed component network \( N_{\text{loc}} = \{ A_1, \ldots, A_n \} \) does not depend on a global scheduler if and only if
- in each reachable configuration,
  - if there is a sending edge locally enabled, then
    - there is at least one locally enabled receiver in a different automaton,
    - and no other sending edge in a different automaton.
    i.e.
    \[
    \forall e \in \text{Conf}(N_{\text{loc}})_{\text{reach}} \forall 1 \leq i \leq n \forall a \in A \forall e \in E(A_i)_{|a} \bullet c \vdash_{\text{loc}} e \\
    \Rightarrow (e \vdash e \wedge 1 \leq j \leq n \forall b \in A \forall e' \in E(A_j)_{|b} \bullet c \vdash_{\text{loc}} e' \Rightarrow j = i). \]
Wrapup
Content

Introduction
• Observables and Evolutions
• Duration Calculus (DC)
• Semantical Correctness Proofs
• DC Decidability
• DC Implementables
• PLC-Automata

\[
\text{obs}: \text{Time} \rightarrow \mathcal{P}(\text{obs})
\]

• Timed Automata (TA), Uppaal
• Networks of Timed Automata
• Region/Zone-Abstraction
• TA model-checking
• Extended Timed Automata
• Undecidability Results

Automatic Verification...
...whether a TA satisfies a DC formula, observer-based

Recent Results:
• Timed Sequence Diagrams, or Quasi-equal Clocks,
or Automatic Code Generation, or ...

Looking Back

• Lect. 1: real-time system (vs. hybrid), state variables
• Lect. 2: evolutions, timing diagrams, classes of timed properties
• Lect. 3: DC symbols, state assertions, terms (syntax / semantics)
• Lect. 4: DC formulae, abbreviations, satisfiable / realisable / valid (from it)
• Lect. 5: semantics-based correctness proof, real-world obstacles
• Lect. 6: DC calculus: decidability of RDC / discrete time
• Lect. 7: undecidability of RDC / continuous time
• Lect. 8: DC Implementables, standard forms, control automata
• Lect. 9: PLC characteristics, programming model
• Lect. 10: PLC automata, DC semantics
• Lect. 11: timed automata (syntax / semantics); tr. seq. / comp. path / run
• Lect. 12: parallel composition of TA (syntactical / semantical); Uppaal
• Lect. 13: TA location reachability, time-abstract system, regions
• Lect. 14: zones, zone-based reachability, Difference-Bounds-Matrices
• Lect. 15: Extended Timed Automata (variables, urgent/committed)
• Lect. 16: query language, evolutions vs. transition sequences
• Lect. 17: testability, observer construction, untestable DC formulae
• Lect. 18: undecidability results for Timed Büchi Automata
• Lect. 19: quasi-equal clocks, bisimulation
• Lect. 20: formal methods for RTS in practice
Advertisements (Again)
Advertisements

- BSc. / MSc. projects (6 – 16 ECTS)
- BSc. / MSc. thesis
  - modelling real-time systems
  - extend timed automata tools
  - work on timed automata theory
  - ⟨ your (real-time) topic here ⟩
- Student assistant jobs
  - programming
  - modelling
- Tutor jobs
  - e.g., Software Engineering in Summer 2018

→ contact me

Advertise

* ADVERTISEMENTS *
References

I have improved my capabilities in scientific problem solving.
(Ich habe meine Fähigkeiten im wissenschaftlichen Problemlösen verbessert.)

- (1) task (in own words), (2) solution (in full sentences), (3) correctness argument.
  That’s already “half of the story”. ;-)}