SysML Models
Verification Relying on
Dependency Graphs

Ludovic APVRILLE
Pierre de SAQUI-SANNE
Oana HOTESCU
Alessandro TEMPIA CALVINO

Modelsward’2022
Context: Verifying Models

Model verification

- Syntax checking
- Simulation
  - Random or user-guided
- Formal verification
  - Model transformed into a formal description
  - Use of a model-checker

Solutions to combinatory explosion

- Higher abstraction level
- Improving proofs
Automated Simplification of Models

Only the relevant part of a model is used to prove a property

→ Use of dependency graphs
# Application to SysML Models and CTL Properties

**SysML Model**

- Structure: block definition diagrams and internal block diagrams
- Behavior: state machine diagrams (1 state machine per block)
- "SysML design" = Structure + Behavior

**Properties**

- Properties in CTL format
- \( p = A|E \ <\ > |[] \ expr \) where \( expr \) is as follows:
  - Block.state
  - A boolean expression built upon blocks’ attributes
# Dependencies in SysML Models

<table>
<thead>
<tr>
<th>Model elements</th>
<th>Dependency graph</th>
<th>Diagrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>One vertex per state</td>
<td>State machines</td>
</tr>
<tr>
<td>Transition</td>
<td>One edge per transition</td>
<td>State machines</td>
</tr>
<tr>
<td>Message sending / receiving (asynchronous)</td>
<td>One vertex per message sending/receiving. One edge between senders to all possible receivers.</td>
<td>State machines and internal block diagrams</td>
</tr>
<tr>
<td>Message sending / receiving (synchronous)</td>
<td>Like asynchronous, but double edge between potential couples sending / receiving actions</td>
<td>State machines and internal block diagrams</td>
</tr>
</tbody>
</table>
Implementation in TTool

Free and open-source toolkit supporting several SysML extensions (AVATAR, DIPLODOCUS, SysML-Sec).

1. Support of SysML design and CTL properties
2. Generation of dependency graphs from SysML designs
3. Generation of a reduced SysML Model from a dependency graph and one property
4. Internal model-checking from a (reduced) SysML design + one property [modelsward’2021]
5. Back-tracing to models

Case Study: An AFDX Network

AFDX (Avionics Full DupleX switched Ethernet)

- De-facto embedded avionics network, (e.g. Airbus A380)
- Properties to be verified:
  - Message arrival time < deadline
  - Total message ordering
  - No message loss due to buffer overflow or failure
- AFDX configuration (this paper)

ES: End System
SW: Switch
VL: Virtual Link
SysML Model of the AFDX Network

User
ES1
- flow1 : Flow;
- flow2 : Flow;
- cnt1 = 0 : int;
- cnt2 = 0 : int;
- nbrOfMessages1 = 0 : int;
- nbrOfMessages2 = 0 : int;
~ out flowToSend1(Flow flow1)
~ out flowToSend2(Flow flow2)

Mechanism
TrafficRegulator11
- flow1 : Flow;
- msg1 : Message;
- maxPeriod = 128 : int;
- currentTime = 0 : int;
- lastTime = 0 : int;
- cnt1 = 0 : int;
~ out regulatedMessage1(Message msg1)
~ in flow1_in(Flow flow1)

Mechanism
TrafficRegulator12
- flow2 : Flow;
- msg2 : Message;
- maxPeriod = 128 : int;
- currentTime = 0 : int;
- lastTime = 0 : int;
- cnt2 = 0 : int;
~ out regulatedMessage2(Message msg2)
~ in flow2_in(Flow flow2)

Mechanism
Scheduler1
- dataRate = 1 : int;
- msg1 : Message;
- msg2 : Message;
- totalWaitingTime = 0 : int;
- transmissionDelay = 1 : int;

State machine diagram of
TrafficRegulator11

IDLE
COUNTING_TIME
flow1_in(flow1)
regulatedMessage1(msg1)
RegulatedF1
END
[ else]
[ waitingTime>0 && ((lastTime==0) || (waitingTime==countTime)) ]
[ waitingTime==0 ]
after(1)
countTime=0
currentTime=currentTime+1

lastTime=currentTime
waitingTime=0
countTime=0
cnt1=cnt1+1

msg1.flowID=flow1.id
msg1.bag=flow1.period
msg1.trafficPriority=flow1.trafficPriority
msg1.length=flow1.msgLength

[ waitingTime==0 ]
[ waitingTime==countTime ]
[ waitingTime>0 ]
[ countTime!=waitingTime ]
after(1)
countTime=countTime+1
currentTime=currentTime+1

cnt1<flow1.nbrOfMessages]
Properties of the AFDX Network

Properties cover important network mechanisms (regulation, scheduling, filtering, demultiplexing)

### Reachability
1. $E \leftrightarrow \text{TrafficRegulator.RegulatedF3}$
2. $E \leftrightarrow \text{SWScheduler.ScheduledMsg1}$
3. $E \leftrightarrow \text{Filtering.FilteredF3}$
4. $E \leftrightarrow \text{Demultiplexer.ReceivedF2}$

### Liveness
1. $A[]\text{TrafficRegulator.RegulatedF3}$
2. $A[]\text{SWScheduler.ScheduledMsg1}$
3. $A[]\text{Filtering.FilteredF3}$
4. $A[]\text{Demultiplexer.ReceivedF2}$
### Experiment

- Dependency graph size: 500 states, 600 transitions
- Reachability graph size: 165k states, 316k transitions

<table>
<thead>
<tr>
<th>Property</th>
<th>Proof duration (ms)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No DG</td>
<td>With DG</td>
<td></td>
<td>No DG</td>
<td>With DG</td>
<td></td>
<td>Gain</td>
</tr>
<tr>
<td></td>
<td>$E \leftrightarrow A[]$</td>
<td>Total</td>
<td>$E \leftrightarrow A[]$</td>
<td>DG processing</td>
<td>Total</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>50</td>
<td>56</td>
<td>6</td>
<td>8</td>
<td>15</td>
<td>29</td>
</tr>
<tr>
<td>2</td>
<td>104</td>
<td>495</td>
<td>599</td>
<td>18</td>
<td>36</td>
<td>15</td>
<td>69</td>
</tr>
<tr>
<td>3</td>
<td>98</td>
<td>737</td>
<td>835</td>
<td>19</td>
<td>51</td>
<td>15</td>
<td>85</td>
</tr>
<tr>
<td>4</td>
<td>114</td>
<td>1734</td>
<td>1848</td>
<td>34</td>
<td>86</td>
<td>15</td>
<td>135</td>
</tr>
</tbody>
</table>

Tests were run on an Intel core i9 computer, with 8 cores at 2.3GHz, 32 Go RAM, running MacOS and Java 8, with TTool build version 13854 (August 2021).
Application to Model Refinement

Avoiding redoing proofs not impacted by a model refinement

Updated Design

DG' == DG?

no

Yes:

Reuse previous proof result

Design property

prover

DG

Reduced Design

True / false

prover

DG'

Updated Design property

prover

DG

Reduced Design

True / false

prover

DG'
Conclusion

Improving model-checking

- Model adaptation for each property
- Use of dependency graphs
- Important gain demonstrated on one case study

Future work

- Other case studies
- Model refinement: defining bisimulation relation
- Extending dependencies to SysML allocations
To Go Further...

TTool

An open source toolkit provided by

ttool.telecom-paris.fr