State Space Exploration of Coloured Petri Nets and the ASAP Model Checking Platform

*Petri Nets 2010 Tutorial*

Lars M. Kristensen  
Department of Computer Engineering  
Bergen University College, Norway  
[lmkr@hib.no](mailto:lmkr@hib.no)

Michael Westergaard  
Computer Science Department,  
University of Aarhus, Denmark  
[mw@cs.au.dk](mailto:mw@cs.au.dk)

Tutorial Outline (1)

- Part 1: Introduction
  - State space methods for Coloured Petri Nets (CPNs) and the research area.
  - Example of a practical application: verification of an edge router discovery protocol with Ericsson.
  - Overview of the ASAP model checking platform.

- Part 2: User perspective
  - Managing verification projects.
  - Creating and executing verification jobs: The JoSEL language.
  - Safety properties and LTL model checking.
Tutorial Outline (2)

- **Part 3: Advanced state space methods**
  - Compact in-memory storage: the comback method
  - State deletion: the sweep-line method.
  - State space partitioning for external memory and distributed model checking.

- **Part 4: Research perspective**
  - Extending ASAP with new state space methods.
  - Benchmarking and profiling.
  - Status and outlook.
State Space Exploration

- One of the main approaches to **model-based verification** of finite-state concurrent systems:

<table>
<thead>
<tr>
<th>System</th>
<th>Model</th>
<th>State space</th>
</tr>
</thead>
</table>

Reachability graph analysis
Occurrence graph analysis
State space exploration
Model checking

**Guarantees complete coverage of executions:**
Error detection + Verification
Explicit State Space Exploration

1: NODES ← \{M_0\}
2: UNPROCESSED ← \{M_0\}
3: ARCS ← ∅
4: while UNPROCESSED ≠ ∅ do
5:     Select a marking M in UNPROCESSED
6:     UNPROCESSED ← UNPROCESSED − \{M\}
7:     for all binding elements (t, b) such that \( M \xrightarrow{(t,b)} M' \) do
8:         Calculate \( M' \) such that \( M \xrightarrow{(t,b)} M' \)
9:         ARCS ← ARCS ∪ \{(M, (t, b), M')\}
10:        if \( M' \not\in \text{NODES} \) then
11:            NODES ← NODES ∪ \{M'\}
12:            UNPROCESSED ← UNPROCESSED ∪ \{M'\}
13:         end if
14:     end for
15: end while
State Space Exploration Methods

- **Advantages:**
  - Highly automatic support by computer tools (construction and analysis algorithms).
  - Much of the underlying mathematics can be hidden.
  - Rich set of behavioural properties can be analysed.
  - Counter examples and diagnostics information.
  - Even partial state spaces provide a systematic and effective error-detection technique.

- **Disadvantages:**
  - Verification relative to specific system configuration.
  - Inherent state explosion problem.
State Space Methods Zoo

- Mostly modelling language independent.
- Typically exploits specific system characteristics.

![Diagram showing state space methods]

- Directed model checking
- Büchi automata
- Stubborn sets
- Comback method
- Modular state spaces
- Symmetry method
- State caching
- Equivalence method
- LTL model checking
- Language equivalence
- Hash compaction
- Bit-state hashing
- Ample sets
- Distributed model checking
- Symbolic model checking
- CTL model checking

Behavioural properties

space

time
Coloured Petri Nets (CPNs)

- Combination of Petri Nets and Standard ML.

- Construction, simulation, and basic state space exploration is supported by CPN Tools.

Petri Nets:
- concurrency
- control structures
- synchronisation
- communication
- resource sharing

Standard ML:
- data manipulation
- compact modelling

CPN / ASAP Tutorial - 9
CPNs and State Space Methods

- **A main guidelines has been to support the full CPN modelling language:**
  - The rich data types yields state vectors of typically 100-1000 bytes.
  - The complex inscriptions make it infeasible to exploit structural properties.
  - Unfolding to low-level Petri Nets is not an viable option.
  - Calculation of enabling binding element (events) is expensive.

- **Advantages of the CPN modelling language:**
  - The possibility of compact modelling yields smaller state spaces (model level reduction).
  - The hierarchical structure facilitates sharing of sub-states.
  - Petri net locality can be exploited to reduce time spent on calculating enabled binding elements (events).
Practical Applications

- State space methods for CPNs have been widely used for verification purposes:
  - Danfoss Flowmeter systems.
  - Bang & Olufsen Beolink System.
  - Scheduling at Australien Defence Force.
  - Ericsson Edge Router Discovery Protocol.
  - Several Internet protocols (e.g., WAP, IOTP, TCP, DCCP, SIP, DYMO).
  - ...

- For further examples:
An Example Application

- Design of an Edge Router Discovery Protocol (ERDP) for mobile ad-hoc networks.

- A CPN model was constructed constituting a formal specification of the ERDP protocol.

- State space exploration was applied to conduct a formal verification of key properties of ERDP.

- Modelling and verification helped in identifying several omissions and errors in the design.
Edge Router Discovery Protocol

- Protocol for **prefix configuration** executed between **edge routers and gateways**:

  - Internet
  - Edge router
  - Gateway
  - Mobile ad-hoc network
Configuration of a gateway

Currently no assigned prefixes

Unicast router solicitation (RS)

Update prefixes

Gateway

GW Buffer

ER Buffer

Edge Router

Unsolicited RA

Unsolicited RA

Unsolicited RA

RS[]

RS[]

RS[]

Solicited RA [P1]

Solicited RA [P1]

Assignment:
P1

Periodical multi-cast of unsolicited router advertisement (RA)

Prefixes have a limited lifetime – must be refreshed – otherwise they will expire

New prefix

Unicast solicited router advertisement (RA)
The Modelling Phase

- CPN modelling applied for specification of the protocol software design:
  - First a conventional natural language specification was developed by the protocol software engineers.
  - Next a CPN model reflecting the specification was developed.

- The ERDP protocol and the CPN model was then developed in an iterative process:
  - CPN model discussed and reviewed in each iteration.
  - CPN model used as a basis for discussion of protocol design.
  - Interactive simulation used for detailed investigations of the protocol software.
Module Hierarchy

Abstract view

Gateway
- ProcessUnsolicitedRA
- ReceiveSolicitedRA
- SendRouterSolicitation
- GWDiscardPrefixes

EdgeRouter
- SendUnsolicitedRA
- ERDiscardPrefixes
- ProcessRS

GW_ER_Link
- NoUnusedPrefix
- AssignNewPrefix

Gateway

Edger Router

Wireless communication link between gateway and edge router
ERDP Top-level Module

Three substitution transitions

Four packet buffers
Results from Modelling

- Several software design issues and errors were identified in the modelling phase:

<table>
<thead>
<tr>
<th>Category</th>
<th>Review 1</th>
<th>Review 2</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incompleteness and ambiguity in the ERDP specification</td>
<td>3</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>Errors in the protocol</td>
<td>2</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>Simplifications of the protocol</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Additions</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Total</td>
<td><strong>11</strong></td>
<td><strong>13</strong></td>
<td><strong>24</strong></td>
</tr>
</tbody>
</table>

- Approximately 70 person-hours were used on CPN modelling and reviews.
State Space Exploration

- State space exploration was pursued after the three iterations of modelling.

- The first step was to obtain a finite state space:
  - The CPN model above can have an arbitrary number of tokens on the packet buffers.
  - An upper integer bound of 1 was imposed on each of the packet buffers (GWIn, GWOut, ERIn, EROut).
  - This also prevents overtaking among the packets transmitted across the wireless link.
  - The number of tokens simultaneously on the four packet buffers was limited to 2.
Verification of ERDP

- Key property of the ERDP protocol:

  From any state with a non-configured prefix P it is possible to reach a state where P is consistently configured.

- Investigated using state space exploration starting from the simplest possible configuration.
One prefix, no loss, no expiration

- State space: 46 nodes and 65 arcs.
- A single dead marking.

- Visual inspection showed that the dead marking is inconsistently configured.
  - The edge router has assigned a prefix to the gateway.
  - BUT, the gateway is not configured with the prefix.

- The error-trace was visualised by means of a message sequence chart.
Error trace MSC

- The edge router sends two unsolicited RAs.
- The first one gets through and we obtain a consistent configuration with prefix P1.
- When the second reaches the edge router there are no unassigned prefixes available.
- A Solicited RA with the an empty list of prefixes is sent.
- The gateway updates its prefixes to be the empty list.
One prefix, no loss, no expiration (rev)

- The protocol was modified such that the edge router always replies with the list of all currently assigned prefixes.

- State space: 34 nodes and 49 arcs.
- No dead markings and 11 home markings (constituting a single terminal SCC).

- Inspection shows that all home markings are consistently configured with the prefix.
  - It is always possible to reach a consistently configured state for the prefix.
  - When such a state has been reached, the protocol entities will remain consistently configured.
Results from Verification

- The verification was conducted in three steps where assumptions were gradually removed.

- **Step 1 [no packet loss and no expire of prefixes]:**
  - Synchronisation error between edge router and gateway.
  - The error was corrected and the key property was verified.

- **Step 2 [packet loss on wireless link added]:**
  - Synchronisation error when certain unsolicited RAs was lost.
  - Livelock error in processing of router advertisement in gateway.
  - The errors were corrected and the key property was verified.

- **Step 3 [expire of prefixes added]:**
  - Property verified: Consistent configuration always possible.
### State Space Statistics

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>No loss/No expire</th>
<th>Loss/No expire</th>
<th>Loss/Expire</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>34</td>
<td>49</td>
<td>68</td>
<td>160</td>
</tr>
<tr>
<td>2</td>
<td>72</td>
<td>121</td>
<td>172</td>
<td>425</td>
</tr>
<tr>
<td>3</td>
<td>110</td>
<td>193</td>
<td>337</td>
<td>851</td>
</tr>
<tr>
<td>4</td>
<td>148</td>
<td>265</td>
<td>582</td>
<td>1,489</td>
</tr>
<tr>
<td>5</td>
<td>186</td>
<td>337</td>
<td>926</td>
<td>2,390</td>
</tr>
<tr>
<td>6</td>
<td>224</td>
<td>409</td>
<td>1,388</td>
<td>3,605</td>
</tr>
<tr>
<td>7</td>
<td>262</td>
<td>481</td>
<td>1,987</td>
<td>5,185</td>
</tr>
<tr>
<td>8</td>
<td>300</td>
<td>553</td>
<td>2,742</td>
<td>7,181</td>
</tr>
<tr>
<td>9</td>
<td>338</td>
<td>625</td>
<td>3,672</td>
<td>9,644</td>
</tr>
<tr>
<td>10</td>
<td>376</td>
<td>697</td>
<td>4,796</td>
<td>12,625</td>
</tr>
</tbody>
</table>

- When a state space has been generated, the verification of the key properties was be done in a few seconds.
Conclusions

- **Start state space exploration from the simplest possible configurations:**
  - Errors often manifest themselves in the simplest configurations and strongest assumptions.
  - The assumptions are then gradually lifted and larger configurations considered.

- **For the ERDP protocol we did not encounter state explosion.**

- **The key properties could be verified for the number of prefixes that are envisioned to appear in practice.**
Observations

- State space methods can now be used to validate industrial-sized practical systems.
- No state space method will work well on all systems.
- **Active research area:** tomorrow will bring even better state space methods and techniques.

**Implications for computer tools:**
- A computer tool must support a wide range of state space methods.
- A computer tools must provide a platform for continuously extending the supported methods.
Brief History of State Space Tool Support for Coloured Petri Nets

- **1G - Occurrence Graph Analyzer (OGA) [1992 - 1994]:**
  - Stand-alone tool based on loading SML images from Design/CPN.
  - Standard ML interface and implementation.
  - Support for first state space exploration and simple visualisation.

- **2G - Design/CPN Occurrence Graph Tool [1994 - 2003]:**
  - Integration of OGA into Design/CPN.
  - Direct support in the graphical user interface.
  - Prototype implementations of equivalence and symmetry method,
  - Time condensed state space, and the sweep method.

- **2.5G - CPN Tools [since 2003]:**
  - Port of Design/CPN Occurrence Graph Tool to CPN Tools.
  - Based on the faster simulation engine used in CPN Tools.
The ASAP Platform

- A new model checking computer tool supporting a large collection of state space methods.

- **Key features:**
  - Graphical specification language for verification jobs.
  - Uniform access to a wide range of state space methods.
  - Allows users to work at different abstraction levels.
  - Support a coherent integration of new state space methods.
  - Verification projects for managing methods, queries, and models.

- **Available via:** [www.cs.au.dk/~ascoveco/](http://www.cs.au.dk/~ascoveco/)
ASAP Status

- ASAP has been developed in the context of the ASCoVeCo Research project.
- Implementation started 08/2007.
- Current status:
  - Comback, sweepline, hash compaction, bit-state hashing, full state space exploration, state caching.
  - Safety and LTL (on-the-fly and offline analysis).
  - Reporting facilities, including visualisation of error traces and state spaces.
  - Plug-in architecture for adding new methods.