Analysis of system behaviour using the mCRL2 toolset

Aad Mathijssen

Design and Analysis of Systems group
Laboratory for Quality Software (LaQuSo)

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven

NL-TWINS Meeting
Logica, Eindhoven

23th January 2009

Analysis techniques

Development of *distributed systems* is inherently complex:

- Needed: assessment and improvement of quality
- Means: analysis techniques

Analysis techniques used in distributed system development:

- Structure: what *things* are in the system?
- Behaviour: what *happens* in the system?

The two techniques complement each other because they focus on *different aspects* of the system.

Analysis of system behaviour

What is analysis of system behaviour about?

- Modelling: create an *abstract* model of the *behaviour* of the system
  - *gain insight* in the behaviour
  - *reduce complexity* to allow for validation and verification
- Validation: are we building the right product?
  - *test requirements* on the model for a number of paths and configurations
- Verification: are we building the product right?
  - *verify requirements* on the model for all possible paths and configurations

mCRL2 toolset

For analysing the behaviour of distributed systems in *industry*, tool support is essential.

The mCRL2 toolset:

- *Supports* many aspects of analysis of system behaviour (modelling, validation, verification)
- Can be *used* to:
  - *detect errors* in the design or implementation of software
  - *prevent errors* already in the design of software

Goals of the mCRL2 toolset:

- Generic basis for the analysis of system behaviour
- Research and development of verification techniques
- Industrial application of verification techniques

mCRL2 toolset: overview

Overview of the mCRL2 toolset:

- 20 years of history:
  - Late 1980s: Common Representation Language (CRL)
  - From 1990: μCRL
  - During 1990s: μCRL toolset
  - From 2004: mCRL2 and mCRL2 toolset
- Collection of tools
- External languages and tools are supported:
  μCRL, CADP, χ, PNML, TorX, LySa, SystemC, LTSmin
- Multi-platform: Windows, Mac and UNIX variants
- Free software licence: Boost licence
- Release policy: fixed release cycle (January and July)

mCRL2 toolset: modelling

Ingredients for modelling:

- *Actions* (push_button, place_order, call_f)
- *Non-deterministic choice*
  (*either* push_button *or* place_order)
- *Sequence* (*first* push_button, *then* place_order)
- *Processes* (Client, WebShop)
- *Parallelism* (Client *in parallel with* WebShop)
- *Synchronous communication*
  (push_button *communicates with* place_order)
- *Data types*
  (push_button(on), Client(1), call_f$(\{x|prime(x)\})$)

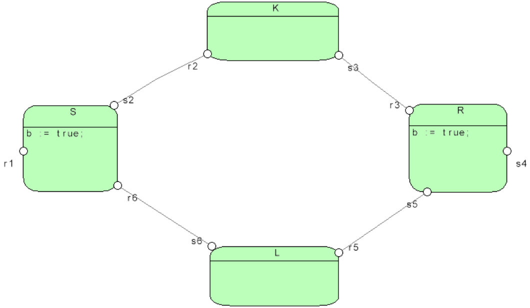
mCRL2 toolset: modelling (2)

The toolset supports two kinds of modelling:

- *Textual*:

$$\textbf{init} \quad \nabla_{\{\mathsf{r1,s4,c2,c3,c5,c6,i}\}}(\Gamma_{\{\mathsf{r2|s2\rightarrow c2,r3|s3\rightarrow c3,r5|s5\rightarrow c5,r6|s6\rightarrow c6}\}}(\mathsf{S}(\mathit{true}) \parallel \mathsf{K} \parallel \mathsf{L} \parallel \mathsf{R}(\mathit{true})));$$

- *Graphical*:



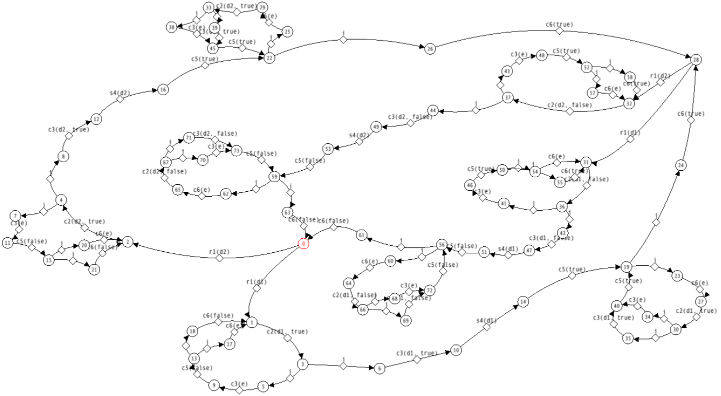
mCRL2 toolset: validation

Validation of models supported by the toolset:

- Manual or semi-automated simulation
- Automated testing using the TorX test tool
- Different types of visualisation

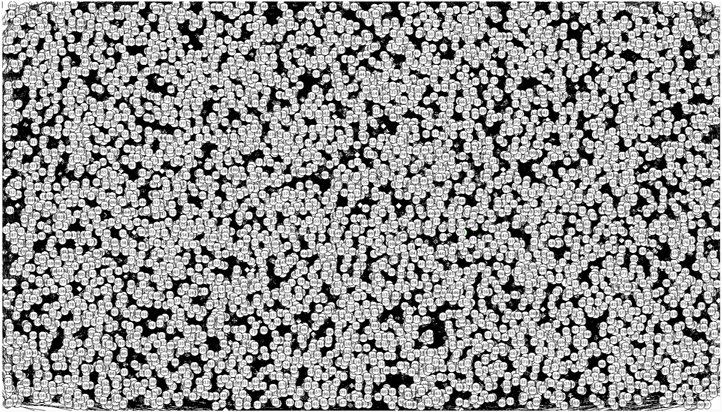
mCRL2 toolset: visualisation

Visualisation as a directed graph using *automatic positioning*:



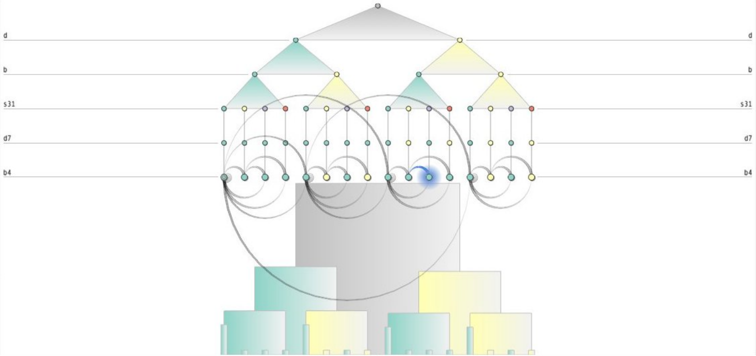
mCRL2 toolset: visualisation

Visualisation as a directed graph is limited to *small models*:



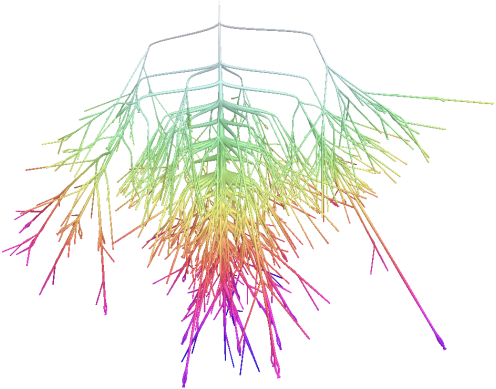
mCRL2 toolset: visualisation

Visualisation as a graph of clusters of states:



mCRL2 toolset: visualisation

Visualisation as a 3D tree of clusters of states:

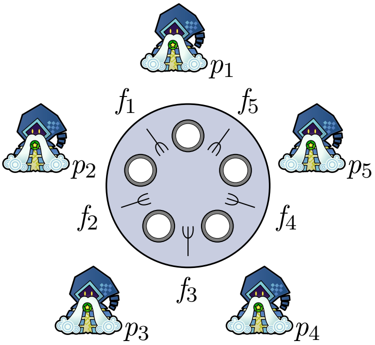


mCRL2 toolset: verification

Toolset support for automated verification
of requirements on the complete model:

- Occurrences of *deadlocks*
- Occurrences of specific *actions*
- *Equivalence* of models
- *Formula checking*:
  - express requirements as *temporal logic formulas*
  - check these formulas on the model

Example: dining philosophers



Abstractly represents various concurrency issues such as *deadlock* and *starvation*.

Example: dining philosophers

Modelling the behaviour of the philosophers:

$$\begin{aligned}
&\textbf{proc}\ \ \mathsf{Phil}(p : \mathit{PhilId}) = \\
&\qquad (\mathsf{get}(p, \mathit{lf}(p)) \cdot \mathsf{get}(p, \mathit{rf}(p)) + \mathsf{get}(p, \mathit{rf}(p)) \cdot \mathsf{get}(p, \mathit{lf}(p))) \\
&\qquad \cdot\, \mathsf{eat}(p) \\
&\qquad \cdot\, (\mathsf{put}(p, \mathit{lf}(p)) \cdot \mathsf{put}(p, \mathit{rf}(p)) + \mathsf{put}(p, \mathit{rf}(p)) \cdot \mathsf{put}(p, \mathit{lf}(p))) \\
&\qquad \cdot\, \mathsf{Phil}(p);
\end{aligned}$$

Modelling the behaviour of the forks:

$$\begin{aligned}
&\textbf{proc}\ \ \mathsf{Fork}(f : \mathit{ForkId}) = \\
&\qquad \textstyle\sum_{p:Phil} \mathsf{up}(p, f) \cdot \mathsf{down}(p, f) \cdot \mathsf{Fork}(f);
\end{aligned}$$

Example: dining philosophers

Modelling the behaviour of the system as a whole:

$$
\begin{array}{l}
\textbf{init} \quad \nabla(\{\mathsf{lock}, \mathsf{free}, \mathsf{eat}\}, \\
\quad \Gamma(\{\mathsf{get}|\mathsf{up} \to \mathsf{lock}, \mathsf{put}|\mathsf{down} \to \mathsf{free}\}, \\
\quad\quad \mathsf{Phil}(p_1) \parallel \mathsf{Phil}(p_2) \parallel \mathsf{Phil}(p_3) \parallel \mathsf{Phil}(p_4) \parallel \mathsf{Phil}(p_5) \parallel \\
\quad\quad \mathsf{Fork}(f_1) \parallel \mathsf{Fork}(f_2) \parallel \mathsf{Fork}(f_3))) \parallel \mathsf{Fork}(f_4) \parallel \mathsf{Fork}(f_5) \\
\quad\quad) \\
\quad);
\end{array}
$$

Example: dining philosophers

Analysis with the mCRL2 toolset:

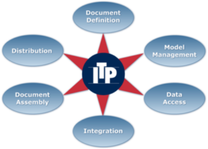
- Verification reveals traces to deadlock states:

$\mathsf{lock}(p_1, f_5)$	$\mathsf{lock}(p_5, f_5)$
$\mathsf{lock}(p_5, f_4)$	$\mathsf{lock}(p_4, f_4)$
$\mathsf{lock}(p_4, f_3)$	$\mathsf{lock}(p_3, f_3)$
$\mathsf{lock}(p_3, f_2)$	$\mathsf{lock}(p_2, f_2)$
$\mathsf{lock}(p_2, f_1)$	$\mathsf{lock}(p_1, f_1)$

- Traces can be validated by means of simulation

Industrial case studies

Selection of industrial case studies performed using the μCRL and mCRL2 toolsets:



Thank you for your attention

More information can be found on `mcrl2.org`.