mCRL2 toolset January 2009 release

Aad Mathijssen

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

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven

LaQuSo Lunch Presentation
Technische Universiteit Eindhoven

29th January 2009

mCRL2 toolset: analysis of system behaviour

Analysis of system behaviour:

- 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
  - *simulate* the model
  - *visualise* the model
- Verification: are we building the product right?
  - *verify requirements* on the model for all possible paths and configurations

mCRL2 toolset: goals

Goals of the mCRL2 toolset:

- Research:
  - Develop *techniques* for the analysis of system behaviour
  - Provide a *generic basis* for the analysis of system behaviour
- Education:
  - Teach model-based design, validation and verification
- Valorisation:
  - *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 for the analysis of system behaviour
- 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: January 2009 release

Released on Monday 26th of January, 2009.

Most important *improvements*:

- Exact rational numbers
- Unique representations of finite sets and bags
- Improved visualisation of directed graphs
- Distributed state space generation

New *experimental* tools (not available by default):

- Elimination of real numbers
- Graphical specification of behaviour
- Optimisation of Parameterised Boolean Equation Systems

Exact rational numbers

Rational numbers in mCRL2:

- New operations: $/$, $floor$, $ceil$, $round$
- No limitations on size

A process that divides by 2... 500 times:

act report:$\mathbb{R}$;
error;

proc $\mathsf{P}(n{:}\mathbb{N}, r{:}\mathbb{R}) = (n < 500) \rightarrow \mathsf{report}(r) \cdot \mathsf{P}(n+1, r/2)$
$+ \ (r/2 \geq r) \ \rightarrow \mathsf{error} \cdot \delta;$

init $\mathsf{P}(0, 1);$

State space generation:

- State space: 501 states and 500 transitions
- No error actions are found

Unique representations of finite sets

The wolf, the goat, and the cabbage:

Unique representations of finite sets

The wolf, the goat, and the cabbage:

Data type declarations:

sort $Item = \mathbf{struct}\ wolf \mid goat \mid cabbage;$
$Position = \mathbf{struct}\ left \mid right;$
$Shores = \mathbf{struct}\ shores(Set(Item), Set(Item));$

map $opp : Position \to Position;$
$items : Shores \times Position \to Set(Item);$
$update : Shores \times Position \times Item \to Shores;$

Unique representations of finite sets

The wolf, the goat, and the cabbage:

Data type definitions:

var $s, t : Set(Item); i : Item;$

eqn $opp(left) = right;$
$opp(right) = left;$
$items(shores(s, t), left) = s;$
$items(shores(s, t), right) = t;$
$update(shores(s, t), right, i) = shores(s \setminus \{i\}, t \cup \{i\});$
$update(shores(s, t), left, i) = shores(s \cup \{i\}, t \setminus \{i\});$

Unique representations of finite sets

The wolf, the goat, and the cabbage:

Actions:

act is_eaten:$Item$;
move:$Position$;
move:$Position \times Item$;
done;

Unique representations of finite sets

The wolf, the goat, and the cabbage:

Process:

$$
\begin{aligned}
\textbf{proc}\quad & \mathsf{WGC}(s{:}Shores, p{:}Position) = \\
& \quad (\{wolf, goat\} \subseteq items(s, opp(p))) \rightarrow \mathsf{is\_eaten}(goat) \cdot \delta \\
& + (\{goat, cabbage\} \subseteq items(s, opp(p))) \rightarrow \mathsf{is\_eaten}(cabbage) \cdot \delta \\
& + (\{wolf, goat\} \not\subseteq items(s, opp(p)) \wedge \{goat, cabbage\} \not\subseteq items(s, opp(p))) \rightarrow \\
& \quad (\mathsf{move}(opp(p)) \cdot \mathsf{WGC}(s, opp(p)) \\
& \qquad + \sum_{i:Item} (i \in items(s, p)) \rightarrow \\
& \qquad\quad \mathsf{move}(opp(p), i) \cdot \mathsf{WGC}(update(s, opp(p), i), opp(p)) \\
& \qquad + (items(s, right) \approx \{wolf, goat, cabbage\}) \rightarrow \mathsf{done} \\
& \quad); \\
\textbf{init}\quad & \mathsf{WGC}(shores(\{wolf, goat, cabbage\}, \emptyset), left);
\end{aligned}
$$

Unique representations of finite sets

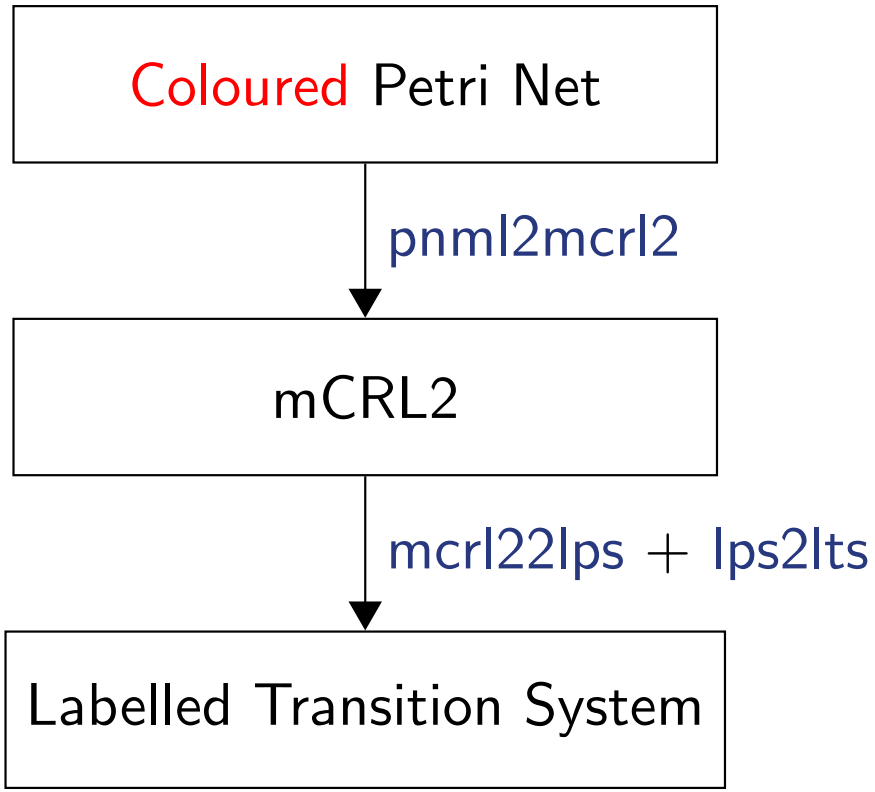
The wolf, the goat, and the cabbage:

State space generation:

- State space: 19 states and 44 transitions
- The following shortest solution is found:

move$(right, goat)$
move$(left)$
move$(right, wolf)$
move$(left, goat)$
move$(right, cabbage)$
move$(left)$
move$(right, goat)$
done

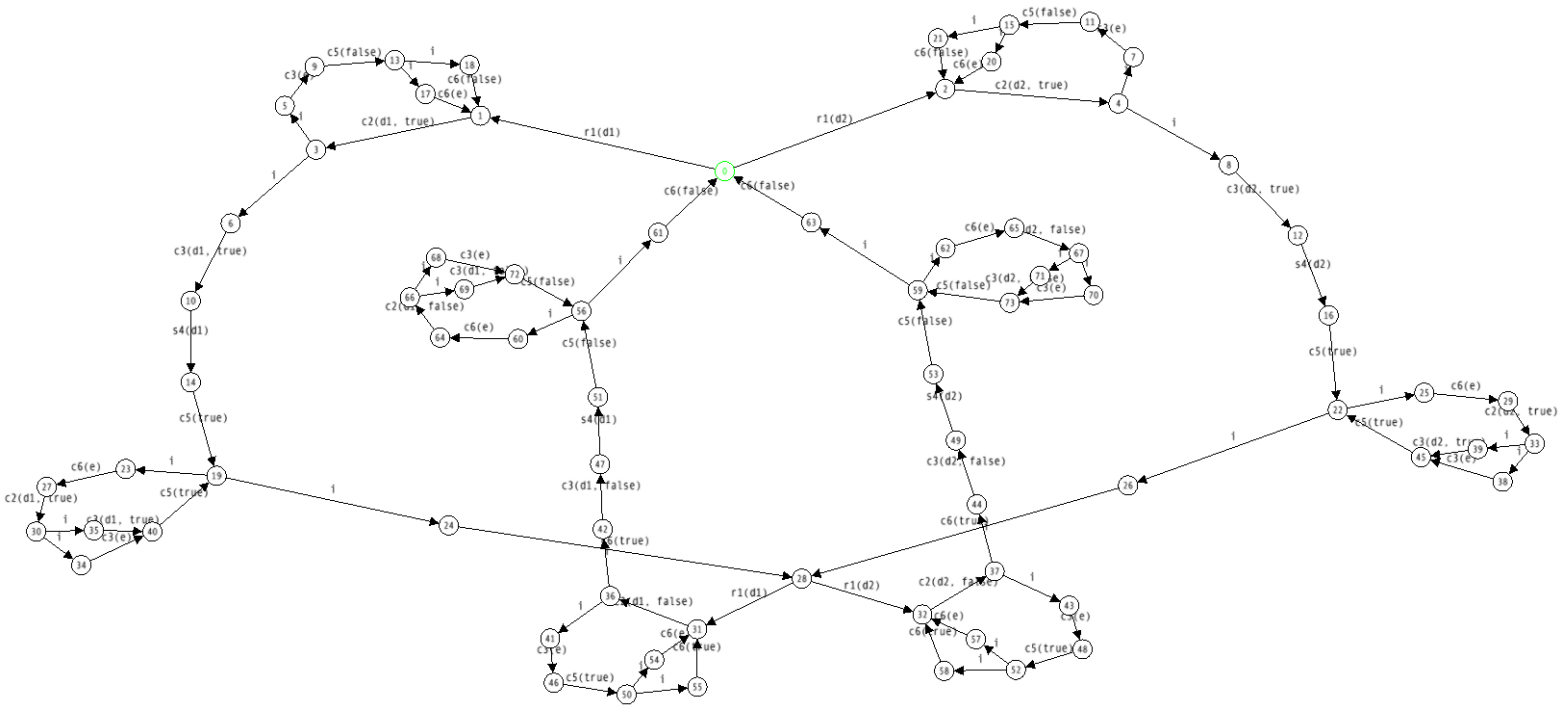
Unique representations of finite bags



Improved visualisation of directed graphs

Improvements of ltsgraph:

- OpenGL implementation
- dot file support



Distributed state space generation

Compatibility with the LTSmin toolset:

- Developed at the Formal Methods & Tools group, University of Twente
- Implements latest techniques for distributed instantiation of labelled transition systems
- Tools for mCRL2 state space generation:
  - Sequential: lps2lts-grey
  - Symbolic: lpsreach
  - Distributed: lps2lts-mpi
  - mCRL2 libraries provide next state/transition functionality

Elimination of real numbers (experimental)

A timed light switch:

$$
\begin{array}{ll}
\textbf{sort} & Light = \textbf{struct}\ off \mid on \mid bright; \\
\textbf{act} & \mathsf{press}, \mathsf{status}{:}Light; \\
\textbf{proc} & \mathsf{P}(r{:}\mathbb{R}, l{:}Light) = \\
 & \quad (l \approx off) \rightarrow \mathsf{press}(on) \cdot \mathsf{P}(0, on) \\
 & + \sum_{s:\mathbb{R}} (l \not\approx off \wedge 0 < s \wedge r + s < 5) \rightarrow \mathsf{press}(bright) \cdot \mathsf{P}(r + s, bright) \\
 & + \sum_{s:\mathbb{R}} (l \not\approx off \wedge 0 < s \wedge r + s \geq 5) \rightarrow \mathsf{press}(off) \cdot \mathsf{P}(0, off) \\
 & + \mathsf{status}(l) \cdot \mathsf{P}(r, l); \\
\textbf{init} & \mathsf{P}(0, off);
\end{array}
$$

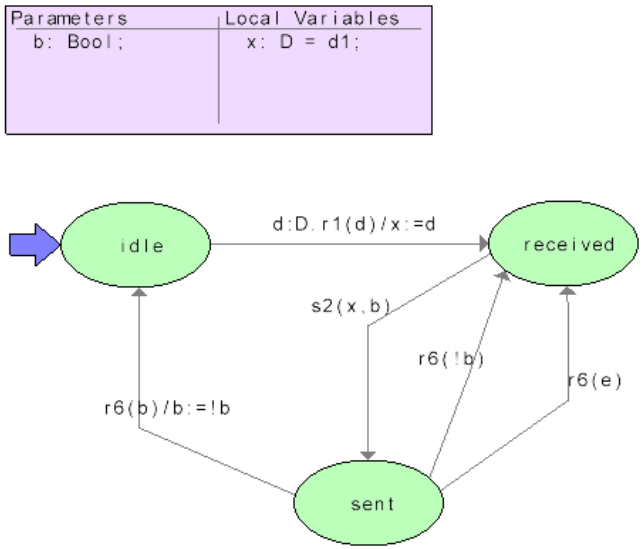
State space generation:

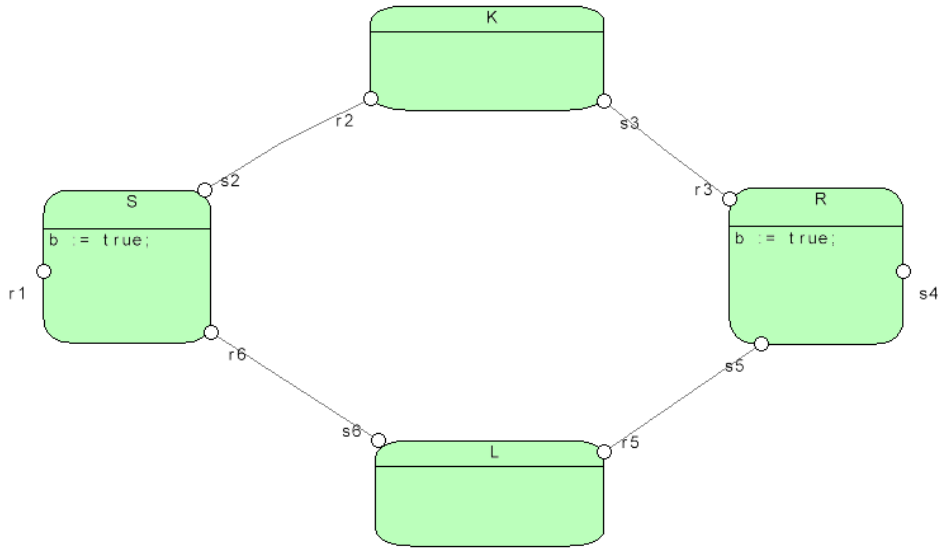
- Impossible directly
- Possible using lpsrealelm: 3 states and 8 transitions

Graphical modelling (experimental)

Graphical Process Editor (GraPE):

- graphical specification of sequential behaviour
- graphical specification of parallel behaviour





Thank you for your attention

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