

3. Introduction to mCRL2

José Proença

System Verification (CC4084) 2025/2026

CISTER – U.Porto, Porto, Portugal

<https://fm-dcc.github.io/sv2526>



CISTER - Research Centre in
Real-Time & Embedded
Computing Systems

<http://mcr12.org>

- Formal [specification language](#) with an associated toolset
- Used for [modelling](#), [validating](#) and [verifying](#) concurrent systems and protocols
- Tool suggestion: use [mcr12ide](#) (not mcr12-gui)

$$\begin{array}{c}
 \text{(act)} \\
 \hline
 \alpha.P \xrightarrow{\alpha} P
 \end{array}
 \qquad
 \begin{array}{c}
 \text{(sum-1)} \\
 \hline
 \frac{P_1 \xrightarrow{\alpha} P'_1}{P_1 + P_2 \xrightarrow{\alpha} P'_1}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{(sum-2)} \\
 \hline
 \frac{P_2 \xrightarrow{\alpha} P'_2}{P_1 + P_2 \xrightarrow{\alpha} P'_2}
 \end{array}$$

$$\begin{array}{c}
 \text{(res)} \\
 \hline
 \frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \quad \alpha \notin L
 \end{array}
 \qquad
 \begin{array}{c}
 \text{(rel)} \\
 \hline
 \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}
 \end{array}$$

$$\begin{array}{c}
 \text{(com1)} \\
 \hline
 \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{(com2)} \\
 \hline
 \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{(com3)} \\
 \hline
 \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{P|Q \xrightarrow{\tau_a} P'|Q'}
 \end{array}$$

Processes in mCRL2

Syntax (by example)

$$a.0 \rightarrow a$$

$$a.P \rightarrow a.P$$

$$P_1 + P_2 \rightarrow P_1 + P_2$$

$$P \setminus L \rightarrow \text{block}(L, P)$$

$$P[f] \rightarrow \text{rename}(f, P)$$

$$a.P \mid \bar{a}.Q \rightarrow \text{comm}(\{a1|a2 \rightarrow a\}, a1.P \mid a2.Q)$$

$$a.P \mid \bar{a}.Q \setminus \{a\} \rightarrow \text{block}(\{a1, a2\}, \text{comm}(\{a1|a2 \rightarrow a\}, \\ a1.P \mid a2.Q))$$

Syntax (by example)

$$a.0 \rightarrow a$$

$$a.P \rightarrow a.P$$

$$P_1 + P_2 \rightarrow P_1 + P_2$$

$$P \setminus L \rightarrow \text{block}(L, P)$$

$$P[f] \rightarrow \text{rename}(f, P)$$

$$a.P \mid \bar{a}.Q \rightarrow \text{hide}(\{a\}, \text{comm}(\{a1|a2 \rightarrow a\}, a1.P \mid a2.Q))$$

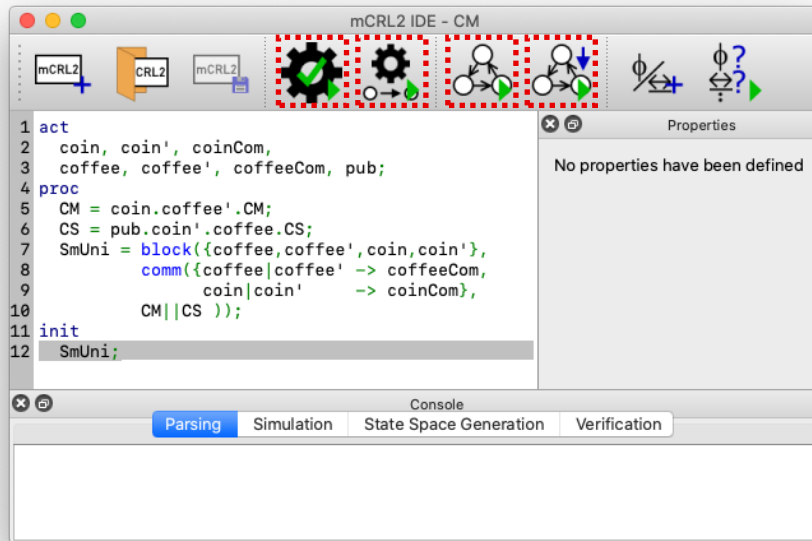
$$a.P \mid \bar{a}.Q \setminus \{a\} \rightarrow \text{hide}(\{a\}, \text{block}(\{a1, a2\}, \text{comm}(\{a1|a2 \rightarrow a\}, a1.P \mid a2.Q)))$$

$$CM = \text{coin}.\overline{\text{coffee}}.CM$$

$$CS = \text{pub}.\overline{\text{coin}}.\text{coffee}.CS$$

$$SmUni = (CM|CS) \setminus \{\text{coin}, \text{coffee}\}$$

```
act
  coin, coin', coinCom,
  coffee, coffee', coffeeCom, pub;
proc
  CM = coin.coffee'.CM;
  CS = pub.coin'.coffee.CS;
  SmUni = block({coffee,coffee',coin,coin'},
    comm({coffee|coffee' -> coffeeCom,
          coin|coin'      -> coinCom},
    CM||CS ));
init
  SmUni;
```



Parse

Simulate

Visualize

Minimize &

Visualize


```
act
  action1, action2, ...;
  action3, action4 : Type;

proc
  P1 = ...;
  P2(x: Bool) = ...;
    % Process expression

init
  SmUni;
```

```
sort List = struct
  empty | cons(A,List);

map sum2: Int # Int -> Int;

var x, y: Int;

eqn
  sum2(x,y) = (x+y) * (x+y);
  % Data patterns & expressions
```

https://mcrl2.org/web/user_manual/language_reference/index.html

$$P = PE ;$$

a *Action*

$a|b$ *Multi-action*

P *Process*

δ *Deadlock*

$a(DataExpr)$ *Parameterized Act.*

$P(DataExpr)$ *Parameterized Proc.*

$a.PE$ *Sequencing*

$PE1 + PE2$ *Choice*

$PE1 || PE2$ *Parallel*

$block(\{a,b\},PE)$ *Block*

$allow(\{a,b\},PE)$ *Allow*

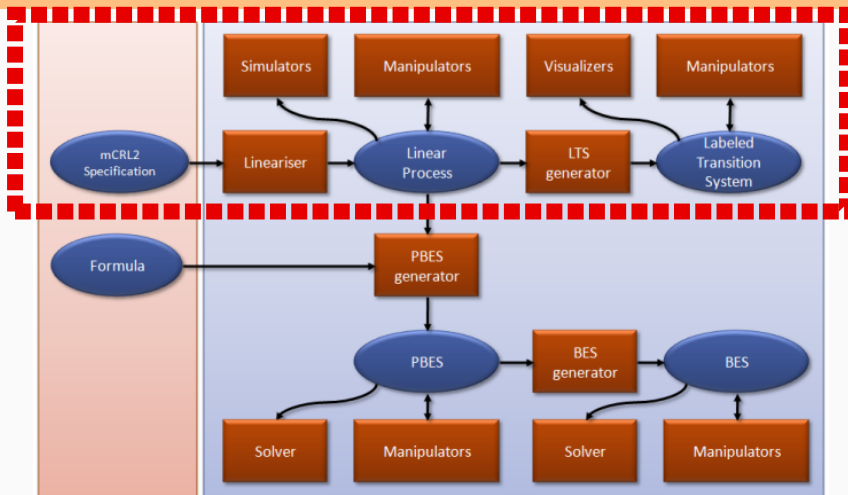
$rename(\{a \rightarrow b\},PE)$ *Rename*

$comm(\{a|b \rightarrow c\},PE)$ *Communicate*

$\sum m: Nat . PE$ *Gen. Choice*

$exp \rightarrow PE1 \triangleleft PE2$ *If-then-else*

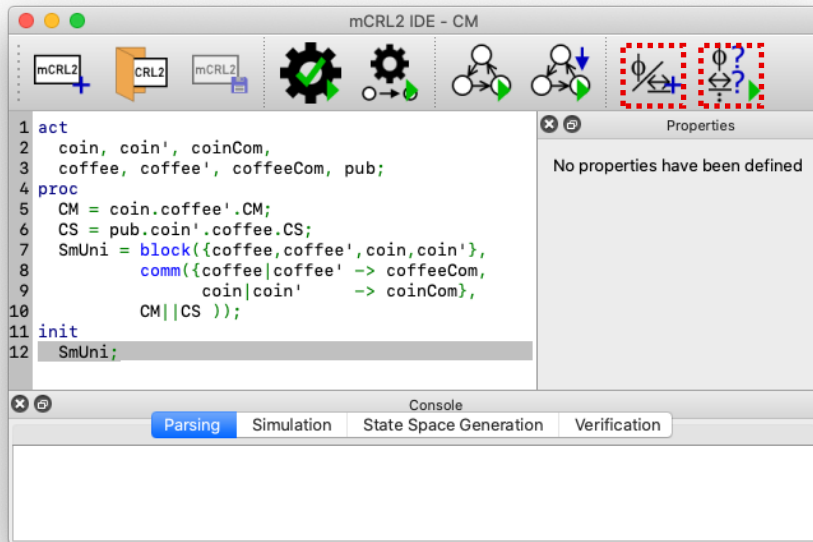
$P(\text{exp})$
true *Boolean*42 *Pos, Nat, Int, Real*exp! *Not*exp && exp *And*exp || exp *Or*exp => exp *Implies*forall n:Nat . exp *For all*exists n:Nat . exp *Exists*exp + exp *Sum*max(exp, exp) *And*exp mod exp *Remainder of div.*[exp, exp, ...] *List*{exp, exp, ...} *Set*{exp:2, exp:1, ...} *Bag*lambda n:Nat . exp *Function*



Assignment 1: TBA

```
https://fm-dcc.github.io/sv2526/exercises/  
adventurers/adventurers-tutorial-mcrl2.zip
```

Logic and Verification



Add
properties

Verify
properties

Syntax (simplified)

$$\phi = \text{true} \mid \text{false} \mid \text{forall } x:T.\phi \mid \text{exists } x:T\phi \\ \mid \phi \text{ OP } \phi \mid !\phi \mid [\text{expr}]\phi \mid \langle \text{expr} \rangle \phi \mid \dots$$

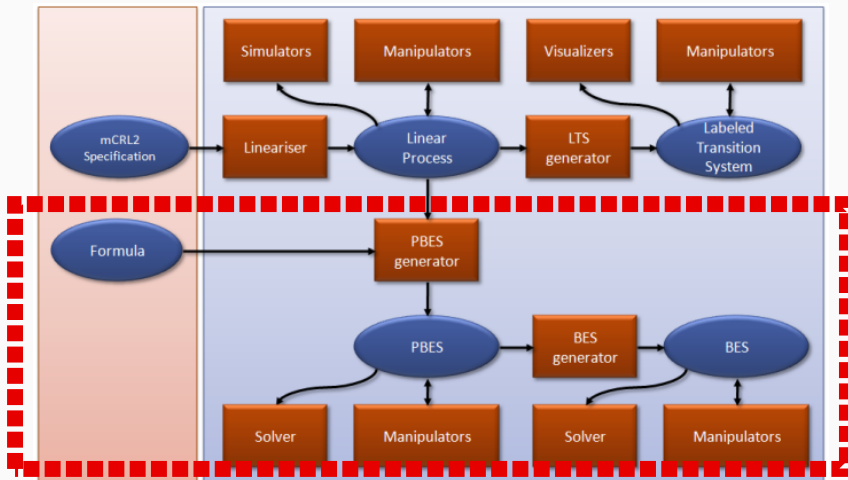
$$\text{expr} = \alpha \mid \text{nil} \mid \text{expr} + \text{expr} \mid \text{expr}.\text{expr} \mid \text{expr}^* \mid \text{expr} +$$

$$\alpha = a(d) \mid a|b|c \mid \text{true} \mid \text{false} \mid \alpha \text{ OP } \alpha \mid !\alpha \\ \mid \text{forall } x:T.\alpha \mid \text{exists } x:T.\alpha \mid \dots$$

where $T = \{Bool, Nat, Int, \dots\}$ and $OP = \{=\!, \&\&, ||\}$

Example

“ $[\text{true}^*.a]\langle b \rangle \text{true}$ ” means: *whenever an ‘a’ appears after any number of steps, it must be immediately followed by ‘b’.*



Assignment 1 (now everything):

<https://fm-dcc.github.io/sv2526/exercises/mcrl2-assignment.pdf>