6. Introduction to mCRL2

David Pereira José Proença

MScCCSE 2021/22

CISTER – ISEP
Porto, Portugal

https://cister-labs.github.io/ramde2122

mCRL2

`http://mcrl2.org`

- Formal specification language with an associated toolset
- Used for modelling, validating and verifying concurrent systems and protocols
- Tool suggestion: use `mcrl2ide` (not `mcrl2-gui`)

Recall CCS semantics

(act)
$$\frac{}{\alpha.P \xrightarrow{\alpha} P}$$

(sum-1)
$$\frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 + P_2 \xrightarrow{\alpha} P_1'}$$

(sum-2)
$$\frac{P_2 \xrightarrow{\alpha} P_2'}{P_1 + P_2 \xrightarrow{\alpha} P_2'}$$

(res)
$$\frac{P \xrightarrow{\alpha} P'}{P\backslash L \xrightarrow{\alpha} P'\backslash L} \quad \alpha \notin L$$

(rel)
$$\frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

(com1)
$$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

(com2)
$$\frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$$

(com3)
$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{P|Q \xrightarrow{\tau_a} P'|Q'}$$

CCS in mCRL2

Syntax (by example)

$$\begin{aligned}
a.\mathbf{0} &\rightarrow \texttt{a} \\
a.P &\rightarrow \texttt{a.P} \\
P_1 + P_2 &\rightarrow \texttt{P1 + P2} \\
P \backslash L &\rightarrow \textit{block}\texttt{(L,P)} \\
P[f] &\rightarrow \textit{rename}\texttt{(f,P)} \\
a.P | \bar{a}.Q &\rightarrow \textit{comm}\texttt{(\{a1}|\texttt{a2}\rightarrow\texttt{a\},a1.P} \parallel \texttt{a2.P)} \\
a.P | \bar{a}.Q \backslash \{a\} &\rightarrow \textit{block}\texttt{(\{a1,a2\},}\textit{comm}\texttt{(\{a1}|\texttt{a2}\rightarrow\texttt{a\},} \\
&\qquad \texttt{a1.P} \parallel \texttt{a2.Q))}
\end{aligned}$$

Processes in mCRL2

CCS in mCRL2 hiding communication

Syntax (by example)

$a.\mathbf{0} \rightarrow$ `a`

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

$P_1 + P_2 \rightarrow$ `P1 + P2`

$P \backslash L \rightarrow$ `block(L,P)`

$P[f] \rightarrow$ `rename(f,P)`

$a.P | \overline{a}.Q \rightarrow$ `hide({a},comm({a1|a2->a},a1.P || a2.P))`

$a.P | \overline{a}.Q \backslash \{a\} \rightarrow$ `hide({a},block({a1,a2},comm({a1|a2->a},`
`a1.P || a2.Q)))`

mCRL2

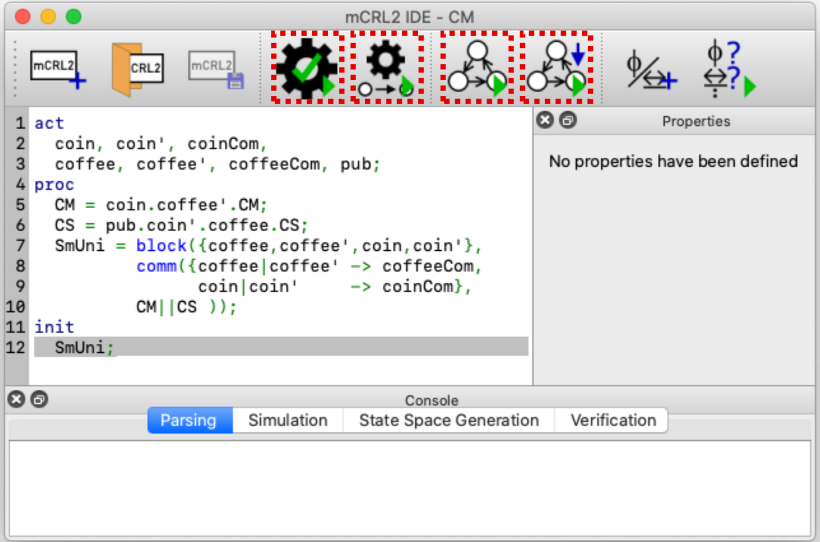
$$CM = \mathsf{coin}.\overline{\mathsf{coffee}}.CM$$

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

$$SmUni = (CM|CS)\backslash\{\mathsf{coin}, \mathsf{coffee}\}$$

```
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;
```

mCRL2 IDE



```
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;
```

No properties have been defined

Parse

Simulate

Visualize

Minimize & Visualize

Specifications *.mcrl2

```
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

Process Expressions

$$P = PE\ ;$$

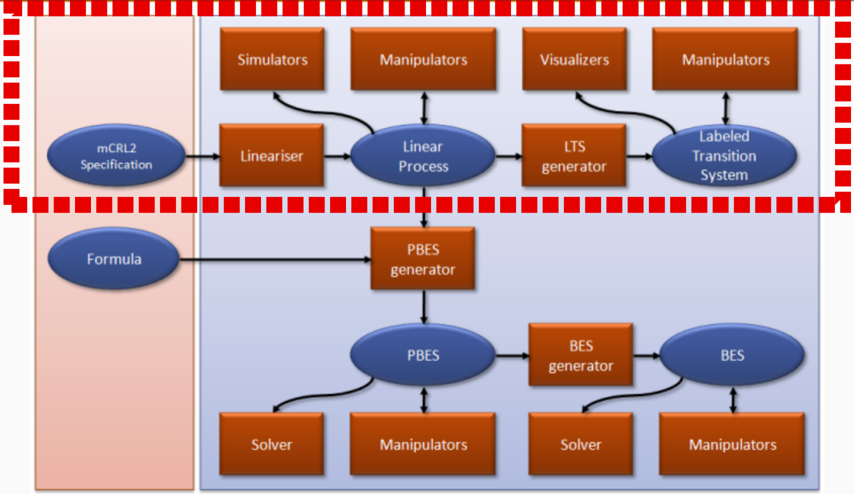
`a`	*Action*		
`a|b`	*Multi-action*		
`P`	*Process*	`block({a,b},PE)`	*Block*
`delta`	*Deadlock*	`allow({a,b},PE)`	*Allow*
`a(DataExpr)`	*Parameterized Act.*	`rename({a→b},PE)`	*Rename*
`P(DataExpr)`	*Parameterized Proc.*	`comm({a|b→c},PE)`	*Communicate*
`a.PE`	*Sequencing*	`sum m: Nat . PE`	*Gen. Choice*
`PE1+PE2`	*Choice*		
`PE1 || PE2`	*Parallel*		

Data Expressions

$P(exp)$

`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*

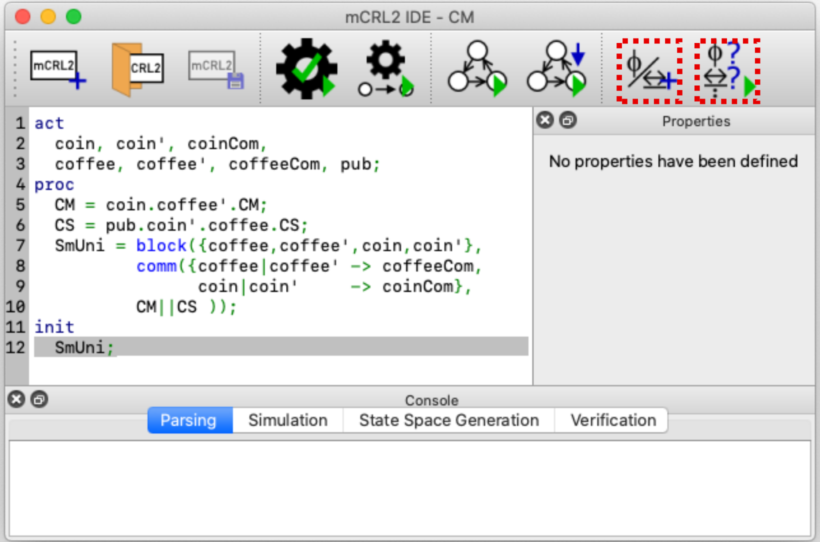
mCRL2 toolset overview



Assignment 1: https://cister-labs.github.io/ramde2122/assignments/a1-modelling.pdf

Logic and Verification

mCRL2 IDE



```
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;
```

No properties have been defined

Add properties

Verify properties

mCRL2 - modal logic

Syntax (simplified)

$$\phi = \texttt{true} \mid \texttt{false} \mid \texttt{forall x:T.}\phi \mid \texttt{exists x.:T}\phi$$
$$\mid \phi\ OP\ \phi \mid \texttt{!}\phi \mid \texttt{[}mod\texttt{]}\phi \mid \texttt{<}mod\texttt{>}\phi \mid \ldots$$

$$mod = \alpha \mid \texttt{nil} \mid mod\texttt{+}mod \mid mod\texttt{.}mod \mid mod\texttt{*} \mid mod\texttt{+}$$

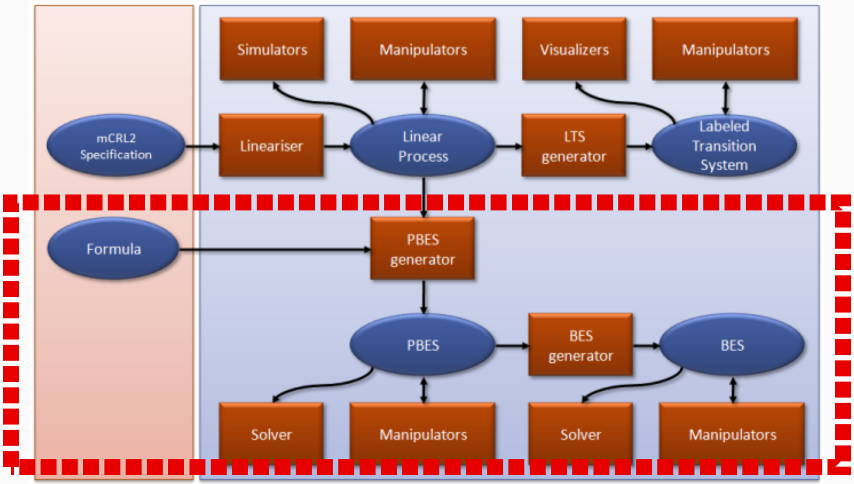
$$\alpha = \texttt{a(d)} \mid \texttt{a|b|c} \mid \texttt{true} \mid \texttt{false} \mid \alpha\ OP\ \alpha \mid \texttt{!}\alpha$$
$$\mid \texttt{forall x:T.}\alpha \mid \texttt{exists x:T.}\alpha \mid \ldots$$

where $T = \{Bool, Nat, Int, \ldots\}$ and $OP = \{\texttt{=>}, \texttt{\&\&}, \texttt{||}\}$

Example

"`[true*.a]<b>true`" means: *whenever an 'a' appears after any number of steps, it must be immediately followed by '*`b`*'.*

mCRL2 toolset overview



Assignment 2: https://cister-labs.github.io/ramde2122/assignments/a2-verification.pdf