3. Introduction to mCRL2

José Proença

System Verification (CC4084) 2024/2025

CISTER – U.Porto, Porto, Portugal

https://fm-dcc.github.io/sv2425

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)
$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau_a} P'|Q'}$$

Processes in mCRL2

CCS in mCRL2

Syntax (by example)

$$a.\mathbf{0} \longrightarrow \texttt{a}$$

$$a.P \longrightarrow \texttt{a.P}$$

$$P_1 + P_2 \longrightarrow \texttt{P1 + P2}$$

$$P \backslash L \longrightarrow \texttt{block(L,P)}$$

$$P[f] \longrightarrow \texttt{rename(f,P)}$$

$$a.P|\overline{a}.Q \longrightarrow \texttt{comm(\{a1|a2->a\},a1.P||a2.Q)}$$

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

CCS in mCRL2 hiding communication

Syntax (by example)

$$a.\mathbf{0} \longrightarrow \texttt{a}$$

$$a.P \longrightarrow \texttt{a.P}$$

$$P_1 + P_2 \longrightarrow \texttt{P1 + P2}$$

$$P \backslash L \longrightarrow \texttt{block(L,P)}$$

$$P[f] \longrightarrow \texttt{rename(f,P)}$$

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

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

mCRL2

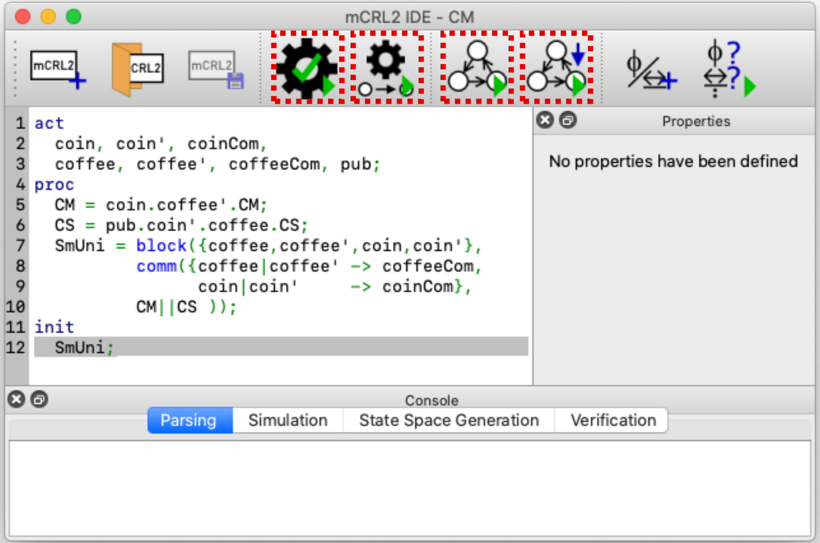
$$CM = \text{coin}.\overline{\text{coffee}}.CM$$

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

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

```
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;
```

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*
`delta`	*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->b},PE)`	*Rename*
`comm({a|b->c},PE)`	*Communicate*
`sum m: Nat . PE`	*Gen. Choice*

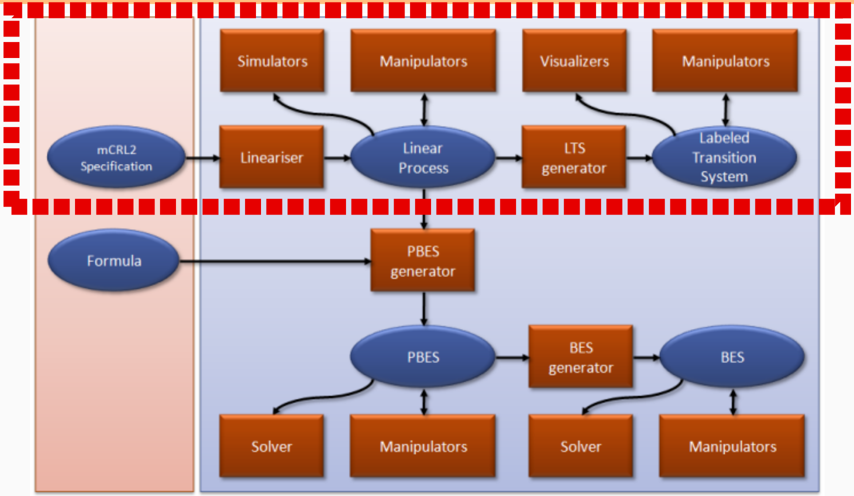
Data Expressions

$P(exp)$

`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



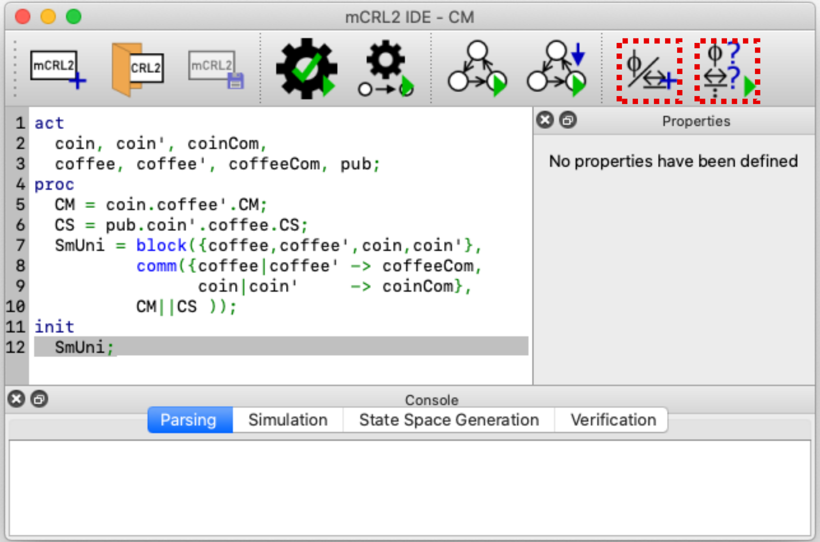
Assignment 1 (first part): tba

mCRL2 tutorial

`https://dcc-fm.github.io/sv2425/exercises/adventurers/adventurers-tutorial-mcrl2.zip`

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)

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

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

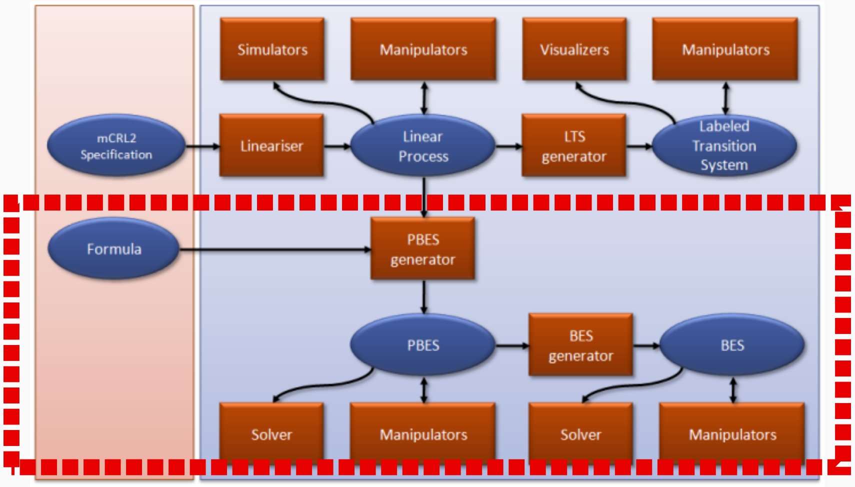
$$\alpha = \texttt{a(d)} \mid \texttt{a|b|c} \mid \texttt{true} \mid \texttt{false} \mid \alpha\ OP\ \alpha \mid \texttt{!}\alpha$$
$$\mid \texttt{forall x:T.}\alpha \mid \texttt{exists x:T.}\alpha \mid \ldots$$

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 1 (second part): tba