

# 10. Introduction to mCRL2

---

David Pereira   José Proença

RAMDE 2021/2022

Requirements and Model-driven Engineering

CISTER – ISEP

Porto, Portugal

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

<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)

## Recall CCS semantics

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

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

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

## Syntax (by example)

$$a.0 \rightarrow a$$

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

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

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

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

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

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

## Processes in mCRL2

---

## Syntax (by example)

$$a.0 \rightarrow a$$

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

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

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

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

$$a.P \mid \bar{a}.Q \rightarrow \text{hide}(\{a\}, \text{comm}(\{a_1 \mid a_2 \rightarrow a\}, a_1.P \parallel a_2.P))$$

$$a.P \mid \bar{a}.Q \setminus \{a\} \rightarrow \text{hide}(\{a\}, \text{block}(\{a_1, a_2\}, \text{comm}(\{a_1 \mid a_2 \rightarrow a\}, a_1.P \parallel a_2.Q)))$$

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

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

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

**act**

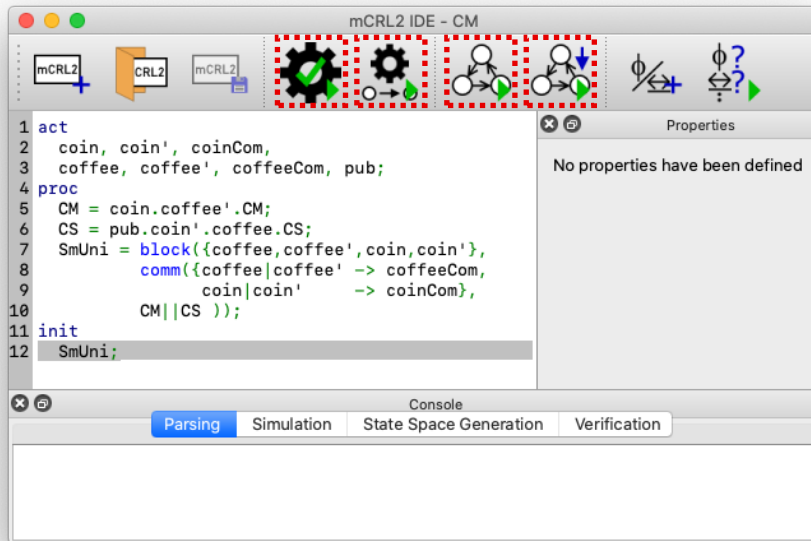
```
coin, coin', coinCom,
coffee, coffee', coffeeCom, pub;
```

**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;
```



Parse

Simulate

Visualize

Minimize &

Visualize



# Specifications \*.mcr12

## **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  $\rightarrow$  Int;

**var** x, y: Int;

## **eqn**

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

[https://mcr12.org/web/user\\_manual/language\\_reference/index.html](https://mcr12.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*

$P(\text{exp})$

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.*

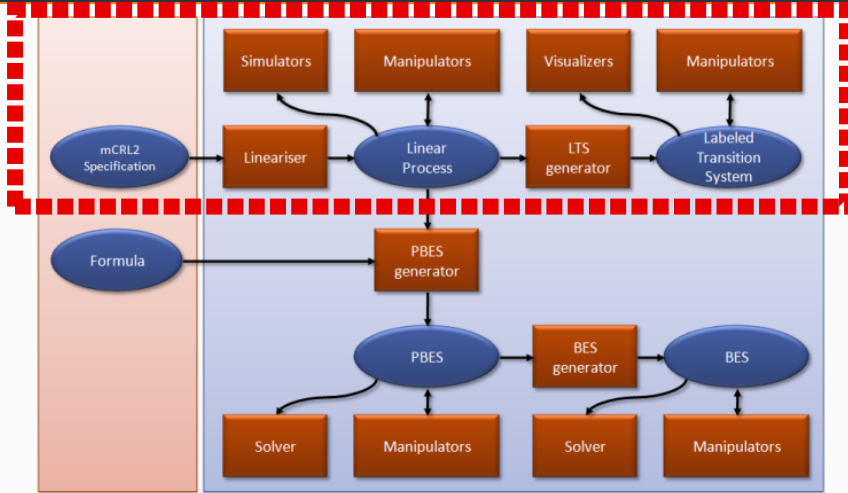
[exp, exp, ...] *List*

{exp, exp, ...} *Set*

{exp:2, exp:1, ...} *Bag*

lambda n:Nat . exp *Function*

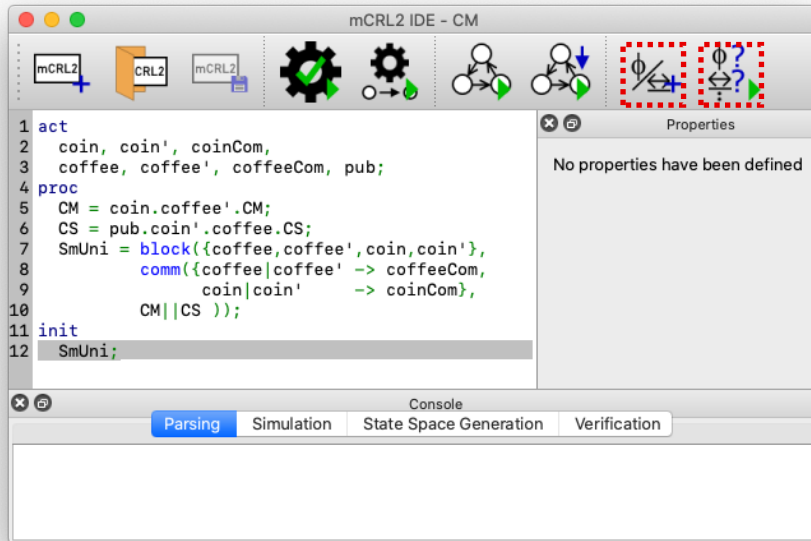
# mCRL2 toolset overview



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

# Logic and Verification

---



Add  
properties

Verify  
properties

## Syntax (simplified)

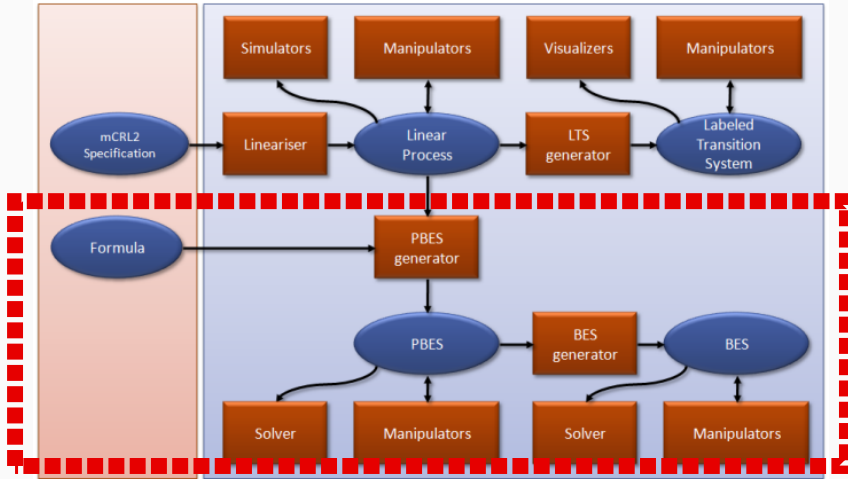
$$\phi = \text{true} \mid \text{false} \mid \text{forall } x:T.\phi \mid \text{exists } x:T.\phi \\ \mid \phi \text{ OP } \phi \mid !\phi \mid [\text{mod}]\phi \mid <\text{mod}>\phi \mid \dots$$
$$\text{mod} = \alpha \mid \text{nil} \mid \text{mod}+\text{mod} \mid \text{mod}.\text{mod} \mid \text{mod}^* \mid \text{mod}+$$
$$\alpha = a(d) \mid a|b|c \mid \text{true} \mid \text{false} \mid \alpha \text{ OP } \alpha \mid !\alpha \\ \mid \text{forall } x:T.\alpha \mid \text{exists } x:T.\alpha \mid \dots$$

where  $T = \{Bool, Nat, Int, \dots\}$  and  $OP = \{=>, \&\&, \parallel\}$

## Example

“ $[\text{true}^*.a]<b>\text{true}$ ” 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>