Formal Modelling of Communicating Systems

Jorge A. Pérez
JBI
University of Groningen

March 23, 2017

- The mCRL2 approach
- Process syntax / Contrast with CCS

The mCRL2 Approach

- mCRL2: "micro Common Representation Language 2"
  The "Dutch approach" to reactive, concurrent systems
- Axiomatic semantics (as opposed to the structured operational semantics of CCS)
- Extends Algebra of Communicating Processes (ACP) with data, time, and multi-actions
- Logic-based verification: the modal μ-calculus with data and time
- Supported by a robust toolset: `http://www.mcrl2.org`

Actions

- Denoted $a, b, c, \dots, \mathit{read}, \mathit{deliver}$ (as in CCS)
- Can be parametric on data - e.g., $\mathit{read}(7), \mathit{write}(\mathit{inc}, 2)$.
- As in CCS, most actions do not overlap in time...
- ...unless they are multi-actions, which occur at the same time.

Behavior

- Defined using LTS (as in CCS)
- It is useful to explicitly distinguish the starting and terminating states in an LTS:

$$A = (S, Act, \rightarrow, s, T)$$

- Trace and failure equivalences, in both strong and weak versions

Data Types

- Many built-in data types: Booleans, Naturals, and Reals, but also functions, lists, bags, sets, etc.
- Also supports user-defined types, following an equational treatment, based on constructors (elements), maps (operations on elements), and equations (rules)

Processes

- Building blocks for behavior, based upon multi-actions with data
- Operators combine behavior; axioms characterize their meaning
- Actions declared with their sort; they are atomic (no duration)

Multi-actions

- Collection of actions that occur at the same time
- Generated by the following syntax:

$$\alpha ::= \tau \mid a(\vec{d}) \mid \alpha|\beta$$

  where

  - τ is the empty multi-action
  - data parameters $\vec{d}$ can be omitted in $a(\vec{d})$

- Examples:
  - error
  - $\mathit{error}|\mathit{error}|\mathit{error}$
  - $\tau|\mathit{error}$
  - $\mathit{error}|\mathit{send}(\mathit{true})$
- Operations: $\alpha \sqsubseteq \beta$ (ordering), $\alpha \setminus \beta$ (removes actions in β from α), $\underline{\alpha}$ (removes data from α)

Sequential Composition, Choices, Conditionals

- An action a is a process that does a and then terminates
- Given processes p, q process $p \cdot q$ executes q once p computes
  This way, e.g., $a \cdot b \cdot c$ is a process (NB. no "zero" process)
- Choices $(+)$ are as in CCS, whereas sums $\sum_{d:D} p(d)$ generalize choices by considering a potentially infinite domain D
- An explicit form of deadlock, denoted δ, which prevents processes to terminate
- This way, e.g., choice process $a + b$ can terminate, whereas $a \cdot \delta + b \cdot \delta$ cannot terminate. Process $a + b \cdot \delta$ could terminate.
- Conditionals $c \rightarrow p \diamond q$ are as expected (c as a Boolean condition based on data)

Recursive Processes

Infinite behavior is handled using declarations, as in CCS.

This is easily seen in the following toy mCRL2 specification:

$$
\begin{array}{ll}
\textbf{act} & \mathit{set}, \mathit{alarm}, \mathit{reset}; \\
\textbf{proc} & P = \mathit{set}.Q; \\
 & Q = \mathit{reset}.P + \mathit{alarm}.Q; \\
\textbf{init} & P;
\end{array}
$$

This way, all actions and process variables are declared before use; the initial process behavior is stipulated under **init**.

Parallel Processes

- Parallel composition of processes p and q is denoted $p \parallel q$
- For conceptual and technical reasons, in mCRL2 process $p \parallel q$ is "decomposed" using two auxiliary operators:
  - In $p \llfloor q$ the first action must come from p
  - In $p \mid q$ the first action must occur simultaneously in p and q
- This decomposition is formalized by the axiom:

$$x \parallel y = x \llfloor y + y \llfloor x + x | y$$

  (In axioms, variables such as x, y stand for processes, and allow their manipulation)
- Notice that $a|b$ denotes both a multi-action AND a synchronization of two processes consisting of a single action

Process Communication

- Operator $\Gamma_C(p)$ takes some actions out of a multi-action and replaces them with a single action
- In $\Gamma_C(p)$, C is a set of allowed communications of the form

$$a_1|\cdots|a_n \to c \qquad (n > 1)$$

- Equality of data is relevant. Examples:
  - $\Gamma_{\{a|b\to c\}}(a(0)|b(0)) = c(0)$
  - $\Gamma_{\{a|b\to c\}}(a(0)|b(0)|d(0)) = c(0)|d(0)$
  - $\Gamma_{\{a|b\to c\}}(a(0)|b(1)) = a(0)|b(1)$
- Function $\gamma_C(\alpha)$ applies the communication described by C to multi-action α. Examples:
  - $\gamma_{\{a|b\to c\}}(a|a|b|c) = a|c|c$
  - $\gamma_{\{a|a\to a, b|c|d\to e\}}(a|b|a|d|c|a) = a|e|a$
- In C an action cannot occur in two LHSs of an allowed communication, nor a RHS can occur in an LHS.
  This way, e.g., $C = \{a|b \to c, a|d \to e\}$ is not allowed.

Allowed Actions

- Operator $\nabla_V(p)$ says which multi-actions from p are allowed to occur, with respect to the multi-actions in V. Data is ignored. Example:

$$\nabla_{\{a,a|b\}}(a|b + a + b) = a + a|b$$

- The empty multi-action τ cannot occur in V.
  Some axioms:

$$\begin{aligned} \nabla_V(\alpha) &= \alpha \quad \text{if } \underline{\alpha} \in V \cup \{\tau\} \\ \nabla_V(\alpha) &= \delta \quad \text{if } \underline{\alpha} \notin V \cup \{\tau\} \\ \nabla_V(x+y) &= \nabla_V(x) + \nabla_V(y) \end{aligned}$$

- Question: What would be the LTSs of
  - $(a \cdot b \parallel c \cdot d)$
  - $\Gamma_{\{a|c \to e, b|d \to f\}}(a \cdot b \parallel c \cdot d)$
  - $\nabla_{\{e,f\}}(\Gamma_{\{a|c \to e, b|d \to f\}}(a \cdot b \parallel c \cdot d))$

Blocking and Renaming

- The operator $\partial_B(p)$ has the opposite effect of $\nabla_V(p)$. The set B contains action names that are not allowed.
- A whole multi-action is blocked if one of its actions is in B. Example:

$$\partial_{\{b\}}(a(0)|b(true, 5)|c) = a(0)$$

- Some axioms:

$$\begin{array}{rcl}
\partial_B(\tau) & = & \tau \\
\partial_B(a(d)) & = & a(d) \quad \text{if } a \notin B \\
\partial_B(a(d)) & = & \delta \quad \text{if } a \in B \\
\partial_B(\alpha|\beta) & = & \partial_B(\alpha)|\partial_B(\beta)
\end{array}$$

- The operator $\rho_R(p)$, where set R contains renamings of the form $a \rightarrow b$, replaces every occurrence of a in p by b.

Hiding (and Prehiding)

- The hiding operator τ_I removes action names in I from multi-actions. Examples:
  - $\tau_{\{a\}}(a) = \tau$
  - $\tau_{\{a\}}(a|b) = b$
- The prehiding operator Υ_U postpones hiding of actions in U, using a special visible action *int*.
- Hiding and prehiding are therefore related:

$$\tau_{I \cup \{int\}}(x) = \tau_{\{int\}}(\Upsilon_I(x))$$