Translating Chi 2.0 to mCRL2

F.P.M. Stappers    M.A. Reniers

Dept. Mathematics and Computer Science,
Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{F.P.M.Stappers, M.A.Reniers}@tue.nl

ProSe -19/06/2008
Outline

1. Motivation
2. Introduction on Chi 2.0 and mCRL2
3. Semantical differences between Chi 2.0 to mCRL2
4. Projection from Chi 2.0 to mCRL2
5. Case Study: The turntable
6. Conclusion and Future work
Motivation

- Various disciplines are involved in the development of complex systems.
- Difficult integration trajectories.
- Models serve particular purpose.
Motivation

- **TWINS: Optimizing HW-SW Co Design Flow for Software Intensive System Development**

  **Focus**
  - Verification and validation of requirements and architecture models
  - Test-case generation
  - Hard-/software change and configuration management
  - Interdisciplinary ways to improve complex distributed and real-time embedded systems

- Opportunity to transform models for simulation to models for verification purposes.
Approach

- Explain Chi 2.0 and mCRL2
- Map Chi 2.0 process to mCRL2 process
- Based on Chi 2.0 process terms, a set of transformation rules are given for projecting Chi 2.0 models into mCRL2 models
Chi - Semantics

• Only consider un-timed Chi processes
• A Chi process is a triple \( \langle p, \sigma, E \rangle \), where
  • \( p \) is a process term
  • \( \sigma \) is a variable valuation \( \{x_1 \mapsto v_1, \ldots, x_n \mapsto v_n\} \)
  • \( E = (D, J, H) \)
• Action transition:

\[
\langle p, \sigma, E \rangle \xrightarrow{\sigma, l, W, \sigma'} \langle p', \sigma', E \rangle
\]

• \( l = \mathcal{L}_{\text{basic}} \cup \mathcal{L}_{\text{com}} \cup \{\tau\} \)
• \( W \) denotes the set of variables that is allowed to change
Chi - Syntax

- Syntax:

  \[ P ::= P_{\text{atom}} \mid P; \mid P \parallel P \mid P \mid *P \mid \partial_H(P) \]

  \[ P_{\text{atom}} ::= u \rightarrow a : W : r \]

  \[ \mid u \rightarrow h! e : W : r \]

  \[ \mid u \rightarrow h ? x : W : r \]

- Sugared Syntax:

  \[ b \rightarrow x := e \triangleq b \rightarrow \tau : \{x\} : x = e^{-} \]

  \[ b \rightarrow h! e \triangleq b \rightarrow h! e : \emptyset : \text{true} \]

  \[ b \rightarrow h ? x \triangleq b \rightarrow h ? x : \emptyset : \text{true} \]

  \[ b \rightarrow \text{skip} \triangleq b \rightarrow \tau : \emptyset : \text{true} \]
Chi example

A finite buffer, whereby the buffer size is denoted by $N$, can be specified as:

\[
\langle \text{chan } a, b : \text{item}, \\
\text{disc } N : \text{nat}, x : [\text{item}], x : \text{item} \\
, x = [], N = 5 \\
: \ast (\text{len}(x) < N \rightarrow a ? x : x := x ++ [x] \\
\mid \text{len}(x) > 0 \rightarrow b! \text{hd}(x) : x := \text{tl}(x) \\
) \\
\rangle
\]
mCRL2

- **Semantics:**
  A mCRL2 process is defined:
  
  $$ p \xrightarrow{a} p' $$

  - $p, p'$ are process terms
  - $a$ is a multi set of data parametrised action names

- **Syntax:**

  $$ P ::= \alpha \mid P + P \mid P \cdot P \mid P \parallel P \mid B \rightarrow P \mid \sum_{x:D} P \mid \nabla_A(P) \mid \partial_B(P) \mid \Gamma_V(P) \mid X(d) $$

  $$ \alpha ::= \tau \mid a(d) \mid \alpha \mid \alpha $$
mCRL2 - Example

This is a process with states 1, ..., $N$, where every state $i$ has transitions to the states 1, ..., $i + 1$.

\[
\text{act} \quad a; \\
\text{map} \quad N : \mathbb{N}^+; \\
\text{eqn} \quad N = 3; \\
\text{proc} \quad X(i : \mathbb{N}^+) = \\
\sum_{j : \mathbb{N}^+} (j \leq i + 1 \land j \leq N) \\
\quad \rightarrow a.X(j); \\
\text{init} \quad X(1);
\]
Approach

- Explain Chi 2.0 and mCRL2
- Map Chi 2.0 process to mCRL2 process
- Based on Chi 2.0 process terms, a set of transformation rules are given for projecting Chi 2.0 models into mCRL2 models
Solutions to the semantical differences 1/4

1. Chi 2.0 has model variables and assignments to them. mCRL2 does not facilitate a concept of model variables.

Store the value of the model variables in a separated process $M$

$$get_m \mid get_p \rightarrow \text{pre}$$

$$set_m \mid set_p \rightarrow \text{post}$$

Figure: Communication diagram between processes $M$ and the translation function $\mathcal{T}(p)$
Solutions to the semantical differences 2/4

Memory process description:

\[ M(\overset{\text{D}}{x}) = \sum_{d:D} \sum_{V \subseteq \{x\}} \left( \bigwedge_{y \in \{x\} \setminus V} x_y = d_y \right) \rightarrow \]

\[ \left( \text{get}_m(x) \mid \big|_{y \in V} \text{set}_m(y, d_y) \mid \big|_{y \in \{x\} \setminus V} \text{post}(y, x_y) \right) \]

\[ \cdot M(d) \]

where the notation \( \big|_{i \in I} p_i \) is inductively defined by:

\[ \big|_{i \in \emptyset} p_i = \tau \]

\[ \big|_{i \in I \cup \{k\}} p_i = p_k \mid \big|_{i \in I \setminus \{k\}} p_i \]
Chi 2.0 has separate labels for the action name, set of changing variables and value advertisement on the action transition. mCRL2 only has parametrised multi-actions. The separated labels are converted to a multi-action.

Example (separated labels to multi-action)

\[
\sigma, l, W, \sigma' \quad \Rightarrow \quad \text{pre}(\sigma)\mid l\mid \text{diff}(W)\mid_{x \in \sigma} \text{post}(x,v)
\]
Parallel composition in Chi 2.0 interleaves actions and corresponding send and receive actions are synchronised. In mCRL2 all actions are interleaved and combined into multi-actions.
To prevent the occurrence of multi-actions the restriction operator $\nabla_{AA}$\(^1\) is used.

\(^1\) $AA$ will be explained in translation of parallel composition operator
Approach

- Explain Chi 2.0 and mCRL2
- Map Chi 2.0 process to mCRL2 process
- Based on Chi 2.0 process terms, a set of transformation rules are given for projecting Chi 2.0 models into mCRL2 models
Translation Scheme

For a given Chi process \( \langle p, \sigma, E \rangle \) and some arbitrary fixed vector \( x \) that consists of precisely the model variables from the set \( D \), the corresponding mCRL2 process is given by

\[
\llbracket \langle p, \sigma, (D, J, H) \rangle \rrbracket = \partial_{B_M} (\Gamma_{C_M} (M(\sigma(x)) \parallel T_{x,J,H}(p)))
\]

where

- \( B_M = \{ \text{get}_m, \text{set}_m, \text{get}_p, \text{set}_p \} \)
- \( C_M = \{ \text{get}_m \mid \text{get}_p \rightarrow \text{pre}, \text{set}_m \mid \text{set}_p \rightarrow \text{post} \} \)
- The initial values for the variables are obtained from the valuation \( \sigma: \sigma(x) \) denotes the vector of values of variables from \( x \) in \( \sigma \) in the same order as these variables appear in \( x \).
- \( J \) denotes the set of jumping variables
- \( H \) denotes the set of channel names
Action update term

\[ T_{x,J,H}(u \rightarrow a : W : r) \]

\[ = \sum_{w:D} \sum_{s:D} (u[w/x] \land r[w/x^-, s/x]) \rightarrow \]

\[ get_p(w) \mid a \mid \text{diff}(W) \mid \bigl|_{x \in J \cup W} set_p(x, s_x) \]

Execution of a guarded action update \( u \rightarrow a : W : r \) involves:

- checking whether the guard \( u \) is satisfied w.r.t. the current values of the model variables, and

- find a new valuation of the model variables in \( J \cup W \), such that the reset predicate \( r \) is satisfied w.r.t. the \( \sigma \) and \( \sigma' \).
Action update term Example - Guarded Assignment

\[ T_{x,J,H}(\text{true} \rightarrow S := \text{false}) \]

\[ = \]

\[ T_{x,J,H}(\text{true} \rightarrow \tau : \{S\} : S = \text{false}) \]

\[ = \]

\[ \sum_{w:D} \ (\text{true} \land (S = \text{false})) \]

\[ \rightarrow get_p(w) \ | \ \tau \ | \ \text{diff}\ (\{S\}) \ | \ set_p(S, \text{false}) \]

Example (Within M, S \approx \text{true})

\[ \text{pre}(..., \text{true}, ...) \ | \ \tau \ | \ \text{diff}\ (\{S\}) \ | \ (... \ | \ \text{post}(S, \text{false}) \ | \ ...) \]

Diagram:

- Initial state: \( p \)
- Final state: \( p' \)

\( p \rightarrow p' \)
Send and Receive communication term

- **Send**
  \[
  T_{x,J,H}(u \rightarrow h! e : W : r) = \\
  \sum_{w:D} \sum_{s:D} (u[w/x] \land r[w/x^{-}, s/x]) \implies \\
  \text{get}_p(w) \mid \text{send}_h(e[w/x]) \mid \text{diff}(W) \mid \big|_{x \in J \cup W} \text{set}_p(x, s_x)
  \]

- **Receive**
  \[
  T_{x,J,H}(u \rightarrow h? x : W : r) = \\
  \sum_{w:D} \sum_{s:D} (u[w/x] \land r[w/x^{-}, s/x]) \implies \\
  \text{get}_p(w) \mid \text{recv}_h(s_x) \mid \text{diff}(W \cup \{x\}) \mid \big|_{y \in J \cup W \cup \{x\}} \text{set}_p(y, s_y)
  \]
Parallel composition

\[ \mathcal{T}_{x,J,H}(p \parallel q) = \nabla_{AA}(\Gamma_{C \cup D}(\mathcal{T}_{x,J,H}(p) \parallel \mathcal{T}_{x,J,H}(q))) \]

where

- \( AA = \{ get_p|a|diff|\big|_{i=0}^{n}set_p,\)
- \( get_p|a|diff|diff|\big|_{i=0}^{n}set_p \)
- \( a \in A_\chi, \ 0 \leq n \leq N \}

with \( N \) is the number of model variables and

- \( A_\chi = \{ send_h, recv_h, comm_h : h \in H \} \cup \mathcal{L}_{\text{basic}} \)
- \( C = \{ send_h|recv_h \rightarrow comm_h : h \in H \}; \)
- \( D = \{ get_p|get_p \rightarrow get_p, set_p|set_p \rightarrow set_p \} \)
Example communication

\[ T_{x,J,H}(h ! true \parallel h ? b) \]
\[ = \nabla_{AA}(\Gamma_{C \cup D}(T_{x,J,H}(h ! true : \emptyset : true) \parallel T_{x,J,H}(h ? b : \emptyset : true))) \]
\[ = \nabla_{AA}(\Gamma_{C \cup D}( \sum_{w:D} \sum_{s:D} get_p(w) \mid send_h(true) \mid diff(\emptyset) \mid \tau \\
\mid \sum_{w:D} \sum_{s:D} get_p(w) \mid recv_h(s_b) \mid diff(\{b\}) \mid set_p(b, s_b) \\
) ) \]

Example (Within \( M \), \( b \approx false \))

\( \text{pre}(false) \mid \text{comm}_h(b, true) \mid \text{diff}(\{\emptyset\}) \mid \text{diff}(\{b\}) \mid (\text{post}(b, true)) \)
Other operators

- **Sequential composition**
  \[ \mathcal{T}_{x,J,H}(p; q) = \mathcal{T}_{x,J,H}(p) \cdot \mathcal{T}_{x,J,H}(q) \]

- **Alternative composition**
  \[ \mathcal{T}_{x,J,H}(p\parallel q) = \mathcal{T}_{x,J,H}(p) + \mathcal{T}_{x,J,H}(q) \]

- **Iteration operator**
  \[ \mathcal{T}_{x,J,H}(\ast p) = X \]
  where \( X = \mathcal{T}_{x,J,H}(p) \cdot X \);

- **Channel encapsulation**
  \[ \mathcal{T}_{x,J,H}(\partial_{H'}(p)) = \partial_B(\mathcal{T}_{x,J,H}(p)) \]
  where \( B = \{\text{send}_h, \text{recv}_h : h \in H'\} \);
Turntable explained

Figure: Decomposition of the turntable system
The Chi model

The Chi model is adapted from the model used in:


Following modification are applied

- All hierarchical modelling is removed
- Time delays are removed
- Uninitialised values are initialised
- Multi-assignments are used where possible
- Meaningful names are introduced for the values that represent a configuration or state of a product
Obtaining the mCRL2 model

- Enumerated data types are modelled as structured sorts.

**Example**

```plaintext
enum configuration = {empty, addingSlot, removeSlot, filled} \Rightarrow
sort configuration = struct empty | addingSlot | removeSlot | filled;
```

- Chi process is translated with translation scheme
- To linearize the mCRL2 following adaptations have been made:
  - Removed local parallelism
  - Partitioned the memory process.
The mCRL2 model

- 500 Lines of mCRL2 code
- Number of states: 13863
- Number of labels: 34
- Number of transitions: 73432
Conclusion & Future work

Conclusions

- Provide an interchange media for transforming Chi 2.0 specifications into mCRL2 models.
- Established an isomorph relation between Chi 2.0 and mCRL2.

Future Work

- Tool support for translating Chi (current version has different implementation).
- Incorporate transformation rules for scopes, recursion variables and hierarchical modelling.
- Model check properties that are stated in terms of the values of model variables.
- Find a suitable time model.
The End