Automatic Translation
from OTS/CafeOBJ to OTS/Maude

Min Zhang

Prof. Kazuhiro Ogata
Japan Advanced Institute of Science and Technology (JAIST)

March, 2010
The main motivation to take full advantages of the verification facilities of both of CafeOBJ and Maude.

- **CafeOBJ**:
  - theorem proving by equational reasoning;
  - be capable of verifying systems with infinite states;
  - semi-automatic.

- **Maude**:  
  - model checking by search or an LTL model checker;
  - need finite or bounded state spaces;
  - fully automatic.
Specifications

- CafeOBJ: based on **equational logic**

**Implicit-state specification**

In the OTS/CafeOBJ specification, the state is specified *implicitly* in the sense that states and transitions are defined with equations.

- Maude: based on **rewriting logic**

**Explicit-state specification**

In the OTS/Maude specification, the state is *explicitly* characterized by a finite set of values and transitions are specified with rewriting rules.
Example: **QLOCK** revisited

**Representation of the initial state(s)**

- **CafeOBJ**:
  
  ```
  op init : -> Sys {constr} .
  -- initial state
  eq pc(init,I:Pid) = rm .
  eq queue(init) = empty .
  ```

- **Maude**:
  
  ```
  op init : -> State [ctor] .
  ```
Example: **QLOCK** revisited

**Representation of transition try**

- **CafeOBJ**: from \( S:Sys \) to \( \text{try}(S,I) \)

  \[
  \begin{align*}
  \text{eq} & \quad \text{c-try}(S,I) \quad = \quad (\text{pc}(S,I) = \text{wt} \quad \text{and} \quad \text{top}(\text{queue}(S)) = I) \ . \\
  \text{ceq} & \quad \text{pc}(\text{try}(S,I),J) \quad = \quad (\text{if} \ I = J \ \text{then} \ \text{cs} \ \text{else} \ \text{pc}(S,J) \ \text{fi}) \\
  & \quad \text{if} \ \text{c-try}(S,I) . \\
  \text{eq} & \quad \text{queue}(\text{try}(S,I)) \quad = \quad \text{queue}(S) . \\
  \text{ceq} & \quad \text{try}(S,I) \quad = \quad S \quad \text{if} \ \text{not} \ \text{c-try}(S,I) .
  \end{align*}
  \]

- **Maude**:

  \[
  \begin{align*}
  \text{crl} \quad [\text{try-I}] \quad (\text{pc} \ [I] : \text{wt}) \quad (\text{queue} : Q) \Rightarrow \\
  & \quad (\text{pc} \ [I] : \text{cs}) \quad (\text{queue} : Q) \quad \text{if} \ \text{top}(Q) = I . \\
  \text{rl} \quad [\text{try-I}] \quad (\text{pc} \ [I] : \text{wt}) \quad (\text{queue} : I \ Q) \Rightarrow \\
  & \quad (\text{pc} \ [I] : \text{cs}) \quad (\text{queue} : I \ Q) .
  \end{align*}
  \]
Two-step Translation: OTS/CafeOBJ to OTS/Maude

1. at the syntax level
   For a tight CafeOBJ module \( TM_S \) that specifies an OTS \( S \), there exists a corresponding functional module \( FM_S \) in Maude.
   \[ TM_S \rightarrow FM_S \]

2. at the semantics level
   \[ FM_S \rightsquigarrow SM_S \]
Translation of initial state(s)

1. choose a model $M$ of $TM_S$ s.t. $M_{Pid} = \{i, j\}$;
2. construct a concrete initial state:

   \[
   \begin{align*}
   pc [i] & : \text{metaReduce}(FM_S, pc(init,i)) \\
   pc [j] & : \text{metaReduce}(FM_S, pc(init,j)) \\
   \text{queue} & : \text{metaReduce}(FM_S, \text{queue}(init))
   \end{align*}
   \]
Translation of transition **try**

1. check the values involved in **try**:
   \[pc(S,I), pc(S,J)\text{ and } queue(S)\]

2. construct a corresponding state by introducing new variables
   \[(pc \ [I] : (L_I:Label))(pc \ [J] : (L_J:Label))(queue : (Q:Queue))\]

3. compute the successor state by reduction:
   \[(pc \ [I] : \text{metaReduce}(FM_S, pc(try(S,I),I))))\]
   \[(pc \ [J] : \text{metaReduce}(FM_S, pc(try(S,I),J))))\]
   \[(queue : \text{metaReduce}(FM_S, queue(try(S,I)))))\]

4. evaluate the condition: \(L_I = \text{wt} \text{ and } \text{top}(Q) = I\)

5. finalize the rewriting rule:
   \[crl \ [try_I] : (pc \ [I] : L_I) \ (pc \ [J] : L_J) \ (queue : Q) \Rightarrow\]
   \[(pc \ [I] : cs)\]
   \[(pc \ [J] : \text{if } I = J \text{ then } cs \text{ else } L_J \text{ fi}) \ (queue : Q)\]
   if \(L_I = \text{wt} \text{ and } \text{top}(Q) = I\).
Tool support and run on Full Maude

▶ The reflection of Maude is well suitable for the translation;
▶ A translator is implemented in Maude meta-programming.

run on Full Maude

minzhang@minzhang-laptop:~$ maude fm ots-maude
\||||||||||||||||||||/
--- Welcome to Maude ---
/||||||||||||||||||||\
Maude 2.4 built: Nov 6 2008 17:14:50
Copyright 1997-2008 SRI International
Fri Mar  5 00:27:49 2010

Full Maude 2.4i April 2nd 2009

An OTS/CafeOBJ to OTS/Maude Translator, Jun. 2009!

Maude>
The End

Have a good time in Japan ^_^