Towards Algebraic Specification of UML State Machines

Gefei Zhang
gefei.zhang@pst.ifi.lmu.de
Ludwig-Maximilians-Universität München

2010-03-05
UML State Machines:

- “The most popular language for modeling reactive systems” [Drusinsky 2006]
- Intuitive, easy to learn
- Only operational semantics published

⇒

- Algebraic specification with Maude
Hierarchic and concurrent
Basic: not considering state hierarchy and concurrency yet
Idea: following the Observational Transition System example (lecture 5).
Observable: active state
Mapping: states and events $\rightarrow$ constants

```
sorts Sys State Event .
op start : $\rightarrow$ Sys [ctor] .
ops init a b c : $\rightarrow$ State [ctor] .
ops abxy ba bc : $\rightarrow$ Event [ctor] .
```
Mapping: transitions $\rightarrow$ equations

\begin{align*}
\text{var } & S : \text{Sys} . \\
\text{var } & E : \text{Event} . \\
\text{vars } & S_1 \text{ S}_2 : \text{State} . \\
\text{eq } & \text{active(start)} = \text{init} . \\
\text{ceq } & \text{active(next(S,E))} = a \\
& \quad \text{if } \text{active(S)} = \text{init} \\
& \quad \text{or } (\text{active(S)} = b \text{ and } E = ba) . \\
\text{ceq } & \text{active(next(S,E))} = b \\
& \quad \text{if } \text{active(S)} = a \text{ and } E = ab . \\
\text{ceq } & \text{active(next(S,E))} = \text{final} \\
& \quad \text{if } \text{active(S)} = b \text{ and } E = bf .
\end{align*}
Observable: set of active states

```plaintext
op active : Sys -> StateSet .
ceq active(next(S,E)) = a,x,running
  if active(S) == init .
ceq active(next(S,E)) = (active(S) \ a), b
  if a in active(S) and not x in active(S) .
ceq active(next(S,E)) = b,y,running
  if active(S) == a,x,running and E == abxy .
```
Other elements of UML state machines:

- **Guards and effects**
  \[ \text{op env : Sys 'QID -> Nat} \]

- **Deferrable events**
  \[ \text{op deferred : Sys -> StateList} \]

- **Pseudostates** (forks, joins, junctions, ...)
  Elimination before translation
Other elements of UML state machines:

- Guards and effects
  \[ \text{op env : } \text{Sys } 'QID -> \text{Nat} \].

- Deferrable events
  \[ \text{op deferred : } \text{Sys } -> \text{StateList} \].

- Pseudostates (forks, joins, junctions, \ldots)
  Elimination before translation

- How to model non-determinism in Maude??
Thank you, JAIST!

Thanks for the excellent event!