Software Design, Modelling and Analysis in UML

Lecture 13: Core State Machines III

2016-12-15

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Content

• Recall: Transitions of UML State Machines
  • discard event,
  • dispatch event,
  • continue RTC,
  • environment interaction,
  • error condition.

• Example Revisited

• Initial States

• Rhapsody Demo III: Model Animation

• Create and Destroy Transformers
Recall: Transition Relation

From Core State Machines to LTS

**Definition.** Let \( \mathcal{F}_0 = ( \mathbb{T}_0, \mathbb{C}_0, \mathbb{V}_0, \mathbb{atr}_0, \mathbb{E} ) \) be a signature with signals (all classes in \( \mathbb{C}_0 \) active), \( \mathcal{D}_0 \) a structure of \( \mathcal{F}_0 \), and \( ( \mathbb{Eth}, \mathbb{ready}, \mathbb{⊕}, \mathbb{⊖}, \mathbb{[·]} ) \) an ether over \( \mathcal{F}_0 \) and \( \mathcal{D}_0 \). Assume there is one core state machine \( M_C \) per class \( C \in \mathbb{C} \).

We say, the state machines induce the following labelled transition relation on states \( S := ( \Sigma \mathcal{F}_0 \times \mathbb{Eth} ) \cup \{ \# \} \) with labels \( A := 2^{\mathbb{Eth}} \times 2^{(\mathbb{Eth} \cup \{ +, - \} ) \times \mathbb{Eth} } \times \mathbb{G}(\mathbb{C}) \):

1. \((\sigma, \varepsilon) \xrightarrow{\text{cons,Stud}}_{u} (\sigma', \varepsilon')\)
   if and only if
   (i) an event with destination \( u \) is discarded,
   (ii) an event is dispatched to \( u \), i.e. stable object processes an event, or
   (iii) run-to-completion processing by \( u \) continues,
   i.e. object \( u \) is not stable and continues to process an event,
   (iv) the environment interacts with object \( u \).

2. \(s \xrightarrow{\text{cons,∅}} \#\)
   if and only if
   (v) an error condition occurs during consumption of \( \text{cons} \), or
   \( s = \# \) and \( \text{cons} = \emptyset \).
(i) Discarding An Event

\[(\sigma, \varepsilon) \xrightarrow{\text{(cons, Snd)}} u \rightarrow (\sigma', \varepsilon')\]

if

- an \(E\)-event (instance of signal \(E\)) is ready in \(\varepsilon\) for object \(u\) of a class \(C\), i.e. if
  \[u \in \text{dom}(\sigma) \cap \mathcal{D}(C) \land \exists u_E \in \mathcal{D}(E) : u_E \in \text{ready}(\varepsilon, u)\]
- \(u\) is stable and in state machine state \(s\), i.e. \(\sigma(u)\text{(stable)} = 1\) and \(\sigma(u)\text{(st)} = s\),
- but there is no corresponding transition enabled (all transitions incident with current state of \(u\) either have other triggers or the guard is not satisfied)

\[\forall (s, F, expr, act, s') \in \rightarrow (SM_C) : F \neq E \lor I[expr](\sigma, u) = 0\]

and

- in the system configuration, stability may change, \(u_E\) goes away, i.e.
  \[\sigma' = \sigma[u \text{. stable} \mapsto b] \setminus \{u_E \mapsto \sigma(u_E)\}\]
  where \(b = 0\) if and only if there is a transition with trigger ‘_’ enabled for \(u\) in \((\sigma', \varepsilon')\).
- the event \(u_E\) is removed from the ether, i.e.
  \[\varepsilon' = \varepsilon \ominus u_E,\]
- consumption of \(u_E\) is observed, i.e.
  \[\text{cons} = \{u_E\}, \quad \text{Snd} = \emptyset.\]

Example: Discard

\[SM_C:\]

\[8_1 \xrightarrow{x > 0} 8_2\]

\[G[x > 0] / x := y \]

\[H / z := y / x\]

\[n\]

\[\sigma:\]

\[x = 1, z = 0, y = 2\]

\[\text{st} = s_1\]

\[\text{stable} = 1\]

\[\varepsilon:\]

\[(c, u_J : J),\]

\[(c, u_G : G)\]

\[\sigma':\]

\[x = 1, z = 0, y = 2\]

\[\text{st} = s_1\]

\[\text{stable} = 1\]
(ii) Dispatch

$$(\sigma, \varepsilon) \xrightarrow{\text{(cons, Snd)}} (\sigma', \varepsilon')$$

If

- $u \in \text{dom}(\sigma) \cap \mathcal{D}(C) \land \exists u_E \in \mathcal{D}(E) : u_E \in \text{ready}(\varepsilon, u)$
- $u$ is stable and in state machine state $s$, i.e. $\sigma(u)(\text{stable}) = 1$ and $\sigma(u)(\text{st}) = s$,
- a transition is enabled, i.e.

$$\exists (s, F, \text{expr}, \text{act}, s') \in (\mathcal{SM}_C) : F = E \land I[\text{expr}](\tilde{\sigma}, u) = 1$$

where $\tilde{\sigma} = \sigma[u.\text{params}_E \mapsto u_E]$.

and

- $(\sigma', \varepsilon')$ results from applying $t_{\text{act}}$ to $(\sigma, \varepsilon)$ and removing $u_E$ from the ether, i.e.

$$(\sigma'', \varepsilon') \in t_{\text{act}}[u](\tilde{\sigma}, \varepsilon \ominus u_E),
\sigma' = (\sigma''[u.\text{st} \mapsto s', u.\text{stable} \mapsto b, u.\text{params}_E \mapsto \emptyset])|_{\mathcal{D}(\varepsilon) \setminus u_E}$$

where $b$ depends (see (i))

- Consumption of $u_E$ and the side effects of the action are observed, i.e.

$\text{cons} = \{u_E\}, \quad \text{Snd} = \text{Obs}_{t_{\text{act}}}[u](\tilde{\sigma}, \varepsilon \ominus u_E)$.

---

Example: Dispatch

SM$_C$:

$$G[x > 0]/x := y \quad \lnot H/z := y/x$$

$\sigma$: $x = 1, z = 0, y = 2$

$\varepsilon$: $\{u_G : G\}$

$\sigma'$:

- $\sigma(u)(\text{stable}) = 1$, $\sigma(u)(\text{st}) = s$,
- $(\sigma'', \varepsilon') = t_{\text{act}}(\tilde{\sigma}, \varepsilon \ominus u_E)$
- $\sigma' = (\sigma''[u.\text{st} \mapsto s', u.\text{stable} \mapsto b, u.\text{params}_E \mapsto \emptyset])|_{\mathcal{D}(\varepsilon) \setminus u_E}$
- $\text{cons} = \{u_E\}$, $\text{Snd} = \text{Obs}_{t_{\text{act}}}[u](\tilde{\sigma}, \varepsilon \ominus u_E)$
(iii) Continue Run-to-Completion

\[(\sigma, \varepsilon) \xrightarrow{(\text{cons}, \text{Snd})} u \xrightarrow{} (\sigma', \varepsilon')\]

if

- there is an unstable object \(u\) of a class \(\mathcal{C}\), i.e.
  \[u \in \text{dom}(\sigma) \cap \mathcal{D}(\mathcal{C}) \land \sigma(u)(\text{stable}) = 0\]

and

- there is a transition without trigger enabled from the current state \(s = \sigma(u)(\text{st})\), i.e.
  \[\exists (s, \_ , \text{expr}, \text{act}, s') \in \rightarrow (S\mathcal{M}_C) : I[\text{expr}](\sigma, u) = 1\]

and

- \((\sigma', \varepsilon')\) results from applying \(t_{\text{act}}\) to \((\sigma, \varepsilon)\), i.e.
  \[(\sigma'', \varepsilon') \in t_{\text{act}}[u](\sigma, \varepsilon), \quad \sigma' = \sigma''[u.\text{st} \mapsto s', u.\text{stable} \mapsto b]\]

where \(b\) depends as before.

- Only the side effects of the action are observed, i.e. \(\text{cons} = \emptyset, \text{Snd} = \text{Obst}_t[u](\sigma, \varepsilon)\).

- \(\sigma' = \sigma[u.\text{stable} \mapsto 1], \varepsilon' = \varepsilon, \text{cons} = \emptyset, \text{Snd} = \emptyset, \) otherwise.

Example: Continue

\[\begin{align*}
\mathcal{S}\mathcal{M}_C: & \quad [x > 0]/x := x - 1; n! J \\
G\{x \geq 0]/x := y & \xrightarrow{} s_2 \\
H/z := y/x & \xrightarrow{} s_1 \\
\sigma: & \quad \begin{bmatrix}
\text{\text{c}} \in \mathcal{C} \\
x = 2, z = 0, y = 2 \\
\text{st} = s_2 \\
\text{stable} = 0
\end{bmatrix}
\end{align*}\]

\[\begin{align*}
\varepsilon': & \quad \begin{bmatrix}
\text{\text{c}} \in \mathcal{C} \\
x = \_ , z = \_ , y = \_ \\
\text{st} = \_ \\
\text{stable} = 0
\end{bmatrix}
\end{align*}\]
(iv) Environment Interaction

Assume that a set $E_{env} \subseteq E$ is designated as environment events and a set of attributes $V_{env} \subseteq V$ is designated as input attributes.

Then

$$\begin{align*}
(\sigma, \varepsilon) \xrightarrow{\text{cons}, \text{Snd}}_{env} (\sigma', \varepsilon')
\end{align*}$$

if either (!)

- an environment event $E \in E_{env}$ is spontaneously sent to an alive object $u \in \text{dom}(\sigma)$, i.e.

  $$\sigma' = \sigma \cup \{u_E \mapsto \{v_i \mapsto d_i | 1 \leq i \leq n\}, \quad \varepsilon' = \varepsilon \oplus (u, u_E)$$

  where $u_E \notin \text{dom}(\sigma)$ and $\text{atr}(E) = \{v_1, \ldots, v_n\}$.

- Sending of the event is observed, i.e. $\text{cons} = \emptyset, \text{Snd} = \{(u, E)\}$.

or

- Values of input attributes change freely in alive objects, i.e.

  $$\forall v \in V \forall u \in \text{dom}(\sigma) : \sigma'(u)(v) \neq \sigma(u)(v) \implies v \in V_{env}.$$  

  and no objects appear or disappear, i.e. $\text{dom}(\sigma') = \text{dom}(\sigma)$.

- $\varepsilon' = \varepsilon$.

Example: Environment

$$\begin{align*}
\text{SM}_{C}:
\begin{array}{c}
\text{stable} = 1 \\
\text{Int} = \{x, y, z\}
\end{array}
\end{align*}$$

$$\begin{align*}
\sigma &:\ C &:\ x = 0, z = 0, y = 2 \\
st = s_2 \\
\text{stable} = 1
\end{align*}$$

$$\begin{align*}
\varepsilon &:\ C &:\ x = , z = , y = \\
st = \\
\text{stable} =
\end{align*}$$

$$\begin{align*}
\sigma' &:\ \sigma \cup \{u_E \mapsto \{v_i \mapsto d_i | 1 \leq i \leq n\}, \\
u \in \text{dom}(\sigma)
\end{align*}$$

$$\begin{align*}
\varepsilon' &:\ \varepsilon \oplus u_E
\end{align*}$$
(v) Error Conditions

if, in (i), (ii), or (iii).
• $I[\text{expr}]$ is not defined for $\sigma$ and $u$, or
• $t_{\text{act}}[u]$ is not defined for $(\sigma, \varepsilon)$.

and
• $\text{cons} = \emptyset$, and $\text{Snd} = \emptyset$.

Examples:

• $\begin{array}{c}
E[\text{x/0}]/\text{act} \\
\sigma_1 \\
\sigma_2
\end{array}$

• $\begin{array}{c}
E[\text{true}]/\text{act} \\
\sigma_1 \\
\sigma_3
\end{array}$

• $\begin{array}{c}
E[\text{expr}]/\text{x := x/0} \\
\sigma_1 \\
\sigma_2
\end{array}$

Example: Error Condition

$SM_C$: $\begin{array}{c}
|x > 0]/\text{x := x - 1; n \! J} \\
\sigma_1 \\
\sigma_2
\end{array}$

$n$ $\begin{array}{c}
\text{I} \in \text{C} \\
\epsilon : \text{C} \\
\begin{array}{c}
\epsilon = x = 0, z = 0, y = 27 \\
st = s_2 \\
\text{stable} = 1
\end{array}
\end{array}$

$\text{E}$ $\begin{array}{c}
(c, u_H : H) \\
\text{epsilon}
\end{array}$

• $I[\text{expr}]$ not defined for $\sigma$ and $u$, or
• $t_{\text{act}}[u]$ is not defined for $(\sigma, \varepsilon)$.

$\text{cons} = \emptyset$, and
$\text{Snd} = \emptyset$. 

**Definition.** Let $A$ be a set of labels and $S$ a (not necessarily finite) set of states. We call

$$\rightarrow \subseteq S \times A \times S$$

a (labelled) transition relation.

Let $S_0 \subseteq S$ be a set of initial states. A (finite or infinite) sequence

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$$

with $s_i \in S$, $a_i \in A$ is called computation of the labelled transition system $(S, A, \rightarrow, S_0)$ if and only if

- **initiation**: $s_0 \in S_0$
- **consecution**: $(s_i, a_i, s_{i+1}) \in \rightarrow$ for $i \in \mathbb{N}_D$. 

---

**Example Revisited**

Transition Relation, Computation
Notions of Steps: The Step

**Note:** we call one evolution

\[(\sigma, \varepsilon) \xrightarrow{u} (\sigma', \varepsilon')\]

a step.

Thus in our setting, a step often\(^1\) directly corresponds to

one object (namely \(u\)) taking a single transition between regular states.

(We will extend the concept of "single transition" for hierarchical state machines.)

\(^1\): In case of dispatch and continue with enabled transition.

**That is:** We’re going for an interleaving semantics without true parallelism.
What is a run-to-completion step...

• **Intuition**: a maximal sequence of steps of one object, where the first step is a dispatch step, all later steps are continue steps, and the last step establishes stability (or object disappears).

**Note**: while one step corresponds to one transition in the state machine, a run-to-completion step is in general not syntactically definable: one transition may be taken multiple times during an RTC-step.

**Example**:

Let $(\sigma_0, \varepsilon_0) \xrightarrow{(\text{cons}_0, \text{Snd}_0)} \ldots \xrightarrow{(\text{cons}_{n-1}, \text{Snd}_{n-1})} (\sigma_n, \varepsilon_n), \quad n > 0,$

be a finite (!), non-empty, maximal, consecutive sequence such that

- $(\text{cons}_0, \text{Snd}_0)$ indicates dispatching to $u := u_0$ (by Rule (ii)), i.e. $\text{cons} = \{u_E\}, u_E \in \text{dom}(\sigma_0) \cap \mathcal{D}(E)$,

- if $u$ becomes stable or disappears, then in the last step, i.e.

\[ \forall i > 0 \bullet (\sigma_i(u)(\text{stable}) = 1 \lor u \notin \text{dom}(\sigma_i)) \implies i = n \]

Let $0 = k_1 < k_2 < \ldots < k_N < n$ be the maximal sequence of indices such that $u_{k_i} = u$ for $1 \leq i \leq N$. Then we call the sequence

$(\sigma_0(u) = \sigma_{k_1}(u), \sigma_{k_2}(u)\ldots, \sigma_{k_N}(u), \sigma_n(u))$

a (!) run-to-completion step of $u$ (from (local) configuration $\sigma_0(u)$ to $\sigma_n(u)$).
**Divergence**

We say, object \( u \) can diverge on reception \( \text{cons}_0 \) from (local) configuration \( \sigma_0(u) \) if and only if there is an infinite, consecutive sequence

\[
(\sigma_0, \varepsilon_0) \xrightarrow{\text{cons}_0, \text{Snd}_0} (\sigma_1, \varepsilon_1) \xrightarrow{\text{cons}_1, \text{Snd}_1} \ldots
\]

where \( u_i = u \) for infinitely many \( i \in \mathbb{N}_0 \) and \( \sigma_1(u)(\text{stable}) = 0, i > 0 \), i.e. \( u \) does not become stable again.

**Run-to-Completion Step: Discussion.**

Our definition of RTC-step takes a **global** and **non-compositional** view, that is:

- In the projection onto a single object we still see the effect of interaction with other objects.
- Adding classes (or even objects) may change the divergence behaviour of existing ones.
- Compositional would be:
  - the behaviour of a set of objects is determined by the behaviour of each object "in isolation".
  
Our semantics and notion of RTC-step doesn’t have this (often desired) property.
Our definition of RTC-step takes a **global** and **non-compositional** view, that is:

- In the projection onto a single object we still see the effect of interaction with other objects.
- Adding classes (or even objects) may change the divergence behaviour of existing ones.
- Compositional would be: the behaviour of a set of objects is determined by the behaviour of each object “in isolation”. Our semantics and notion of RTC-step doesn’t have this (often desired) property.

Can we give (syntactical) criteria such that any (global) run-to-completion step is an interleaving of local ones?

**Run-to-Completion Step: Discussion.**

**Maybe:** **Strict interfaces.**

(Proof left as exercise...)

- **(A):** Refer to private features only via “self”. (Recall that other objects of the same class can modify private attributes.)
- **(B):** Let objects only communicate by events, i.e. don’t let them modify each other’s local state via links **at all**.
Recall: a labelled transition system is \((S, A, \rightarrow, S_0)\).

We have

- \(S\): system configurations \((\sigma, \varepsilon)\)
- \(\rightarrow\): labelled transition relation \((\sigma, \varepsilon) \xrightarrow{(\text{cons}, \text{Snd}) \alpha} (\sigma', \varepsilon')\).

Wanted: initial states \(S_0\).
**Initial States**

**Recall:** a labelled transition system is $(S, A, \rightarrow, S_0)$. We have

- $S$: system configurations $(\sigma, \varepsilon)$
- $\rightarrow$: labelled transition relation $(\sigma, \varepsilon) \xrightarrow{(\text{cons}, \text{Snd})} (\sigma', \varepsilon')$.

**Wanted:** initial states $S_0$.

**Proposal:**

Require a (finite) set of *object diagrams* $\mathcal{OD}$ as part of a UML model $(\mathcal{C}, \mathcal{M}, \mathcal{OD})$.

And set

$$S_0 = \{ (\sigma, \varepsilon) \mid \sigma \in G^{-1}(\mathcal{OD}), \mathcal{OD} \in \mathcal{OD}, \varepsilon \text{ empty} \}.$$
**Semantics of UML Model (So Far)**

The *semantics* of the **UML model**

\[ M = (C \mathcal{D}, \mathcal{M}, \mathcal{O} \mathcal{D}) \]

where

- some classes in \( C \mathcal{D} \) are stereotyped as ‘signal’ (standard),
  some signals and attributes are stereotyped as ‘external’ (non-standard),
- there is a 1-to-1 relation between classes and state machines,
- \( \mathcal{O} \mathcal{D} \) is a set of object diagrams over \( C \mathcal{D} \),

is the **transition system** \( (S, A, \rightarrow, S_0) \) constructed on the previous slide(s).

The **computations** of \( M \) are the computations of \( (S, A, \rightarrow, S_0) \).

---

**OCL Constraints and Behaviour**

- Let \( M = (C \mathcal{D}, \mathcal{M}, \mathcal{O} \mathcal{D}) \) be a UML model.
- We call \( M \) **consistent** iff, for each OCL constraint \( expr \in Inv(C \mathcal{D}) \),
  \[ \sigma \models expr \] for each ”reasonable point” \((\sigma, \epsilon)\) of computations of \( M \).
  (Cf. tutorial for discussion of ”reasonable point”.)

**Note**: we could define \( Inv(\mathcal{M}) \) similar to \( Inv(C \mathcal{D}) \).
Let $M = (C \mathcal{D}, S \mathcal{M}, OD)$ be a UML model.

We call $M$ consistent iff, for each OCL constraint $expr \in Inv(C \mathcal{D})$, $\sigma \models expr$ for each "reasonable point" $(\sigma, \varepsilon)$ of computations of $M$.

(Cf. tutorial for discussion of "reasonable point".)

Note: we could define $Inv(S \mathcal{M})$ similar to $Inv(C \mathcal{D})$.

Pragmatics:

- In UML-as-blueprint mode, if $S \mathcal{M}$ doesn’t exist yet, then providing $M = (C \mathcal{D}, \emptyset, OD)$ is typically asking the developer to provide state machines $S \mathcal{M}$ such that $M' = (C \mathcal{D}, S \mathcal{M}, OD)$ is consistent.
  
  If the developer makes a mistake, then $M'$ is inconsistent.

- Not so common (but existing):
  
  If $S \mathcal{M}$ is given, then constraints are also considered when choosing transitions in the RTC-algorithm.
  
  In other words: even in presence of “mistakes”, the state machines in $S \mathcal{M}$ never move to inconsistent configurations.

---

Rhapsody Demo III: Model Animation
State Machines induce a labelled transition system.

There are five kinds of transitions in the LTS:

- discard, dispatch, continue, environment, error.

For now, we assume that all classes are active, thus steps of objects may interleave.

We distinguish steps and run-to-completion step.

Initial states can be characterised using object diagrams.

Missing transformers:

- Create: re-use identities vs. use fresh ones.
- Destroy: allow dangling references vs. clean up.

References
References
