Software Design, Modelling and Analysis in UML

Lecture 15: ~~Hierarchical State Machines I~~
State Machines V

2014-01-13

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

- RTC-Rules: Discard, Dispatch, Commence.

This Lecture:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What does this State Machine mean? What happens if I inject this event?
  - Can you please model the following behaviour.
  - What is: initial state.
  - What does this **hierarchical** State Machine mean? What **may happen** if I inject this event?
  - What is: AND-State, OR-State, pseudo-state, entry/exit/do, final state, ...
- **Content:**
  - Step, RTC, Divergence
  - Putting It All Together
  - Rhapsody Demo
  - Hierarchical State Machines Syntax

Step and Run-to-completion Step

Notions of Steps: The Step

Note: we call one evolution $(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$ a **step**.

Thus in our setting, **a step directly corresponds** to

one object (namely u) takes **a single transition** in its statemachine between regular states.

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

That is: We're going for an interleaving semantics without true parallelism.

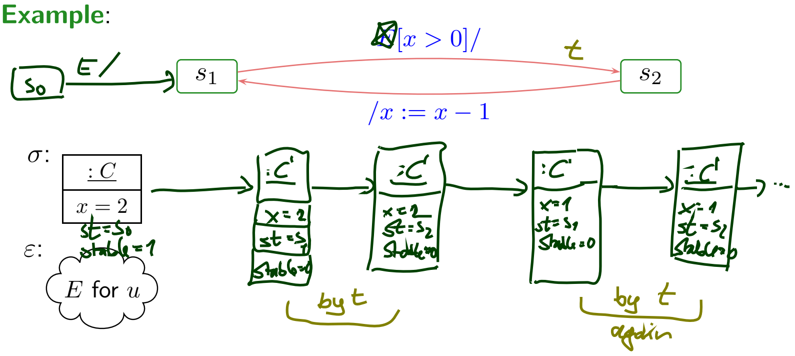
Notions of Steps: The Run-to-Completion Step

What is a **run-to-completion** step...?

- **Intuition**: a maximal sequence of steps, where the first step is a **dispatch** step and all later steps are **commence** steps.
- **Note**: 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:



Notions of Steps: The Run-to-Completion Step Cont'd

Proposal: Let

we cannot extend the sequence and still satisfy the properties below

other objects may do interleaving steps

$$(\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} \dots \xrightarrow[u_{n-1}]{(cons_{n-1}, Snd_{n-1})} (\sigma_n, \varepsilon_n), \quad n > 0,$$

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

$(\sigma_i, \varepsilon_i), (\sigma_{i+1}, \varepsilon_{i+1})$ are in transition relation

- object u is alive in σ_0,
- $u_0 = u$ and $(cons_0, Snd_0)$ indicates dispatching to u, i.e. $cons = \{(u, \vec{v} \mapsto \vec{d})\}$,
- there are no receptions by u in between, i.e.

$$cons_i \cap \{u\} \times Evs(\mathscr{E}, \mathscr{D}) = \emptyset, i > 1,$$

- $u_{n-1} = u$ and u is stable only in σ_0 and σ_n, i.e.

$$\sigma_0(u)(stable) = \sigma_n(u)(stable) = 1 \text{ and } \sigma_i(u)(stable) = 0 \text{ for } 0 < i < n,$$

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

$$(\sigma_0(u) =) \quad \sigma_{k_1}(u), \sigma_{k_2}(u) \dots, \sigma_{k_N}(u) \quad (= \sigma_{n-1}(u))$$

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

Divergence

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

$$(\sigma_0, \varepsilon_0) \xrightarrow{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \xrightarrow{(cons_1, Snd_1)} \dots$$

such that u doesn't become stable again.

- **Note**: disappearance of object not considered in the definitions. By the current definitions, it's neither divergence nor an RTC-step.

– 15 – 2014-01-13 – Sstmstep –

Run-to-Completion Step: Discussion.

What people may **dislike** on our definition of RTC-step is that it 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?

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**.

– 15 – 2014-01-13 – Sstmstep –

Putting It All Together

The Missing Piece: Initial States

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

- S: system configurations (σ, ε)
- $\to$: labelled transition relation $(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$.

Wanted: initial states S_0.

Proposal:
Require a (finite) set of **object diagrams** $\mathcal{OD}$ as part of a UML model

class diagrams, induce signature, and system states

$$(\mathscr{CD}, \mathscr{SM}, \mathscr{OD}).$$

object diagrams, induce system states, the initial ones

(statemachine diagrams, induce core state machines, and $\xrightarrow[u]{(cons, Snd)}$

And set

$$S_0 = \{(\sigma, \varepsilon) \mid \sigma \in G^{-1}(\mathcal{OD}), \mathcal{OD} \in \mathscr{OD}, \varepsilon \text{ empty}\}.$$

Other Approach: (used by Rhapsody tool) multiplicity of classes.
We can read that as an abbreviation for an object diagram.

Semantics of UML Model — So Far

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

$$\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$$

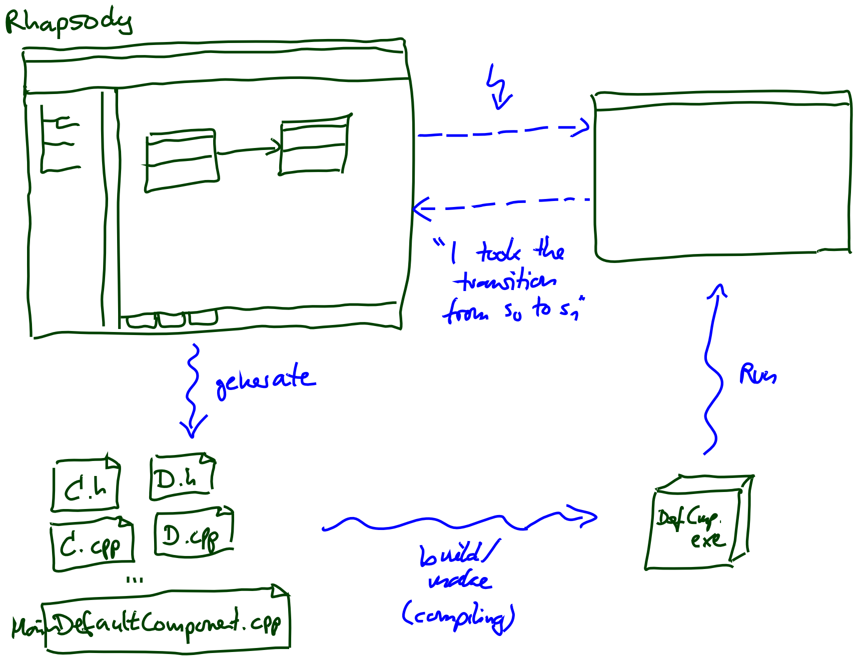
where

- some classes in $\mathscr{CD}$ 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,
- $\mathscr{OD}$ is a set of object diagrams over $\mathscr{CD}$,

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

The **computations of** $\mathcal{M}$ are the computations of $(S, \rightarrow, S_0)$.

Contemporary UML Modelling Tools


Rhapsody
"I took the transition from s0 to s1"
generate
Run
C.h
D.h
C.cpp
D.cpp
...
MainDefaultComponent.cpp
build/make
(compiling)
DefCmp.exe

References

References

[Crane and Dingel, 2007] Crane, M. L. and Dingel, J. (2007). UML vs. classical vs. rhapsody statecharts: not all models are created equal. *Software and Systems Modeling*, 6(4):415–435.

[Damm et al., 2003] Damm, W., Josko, B., Votintseva, A., and Pnueli, A. (2003). A formal semantics for a UML kernel language 1.2. IST/33522/WP 1.1/D1.1.2-Part1, Version 1.2.

[Fecher and Schönborn, 2007] Fecher, H. and Schönborn, J. (2007). UML 2.0 state machines: Complete formal semantics via core state machines. In Brim, L., Haverkort, B. R., Leucker, M., and van de Pol, J., editors, *FMICS/PDMC*, volume 4346 of *LNCS*, pages 244–260. Springer.

[Harel and Kugler, 2004] Harel, D. and Kugler, H. (2004). The rhapsody semantics of statecharts. In Ehrig, H., Damm, W., Große-Rhode, M., Reif, W., Schnieder, E., and Westkämper, E., editors, *Integration of Software Specification Techniques for Applications in Engineering*, number 3147 in LNCS, pages 325–354. Springer-Verlag.

[OMG, 2007] OMG (2007). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.

[Störrle, 2005] Störrle, H. (2005). *UML 2 für Studenten*. Pearson Studium.