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
Notions of Steps: The Step

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

Thus in our setting, a step directly corresponds to **one object** (namely \(u\)) takes a **single transition** 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:

Example:

\[ e \quad \frac{[x > 0]}{x := x - 1} \]

\[ \sigma: \quad \{ C \} \quad \sigma: \quad \{ C \} \quad \sigma: \quad \{ C \} \quad \sigma: \quad \{ C \} \quad \ldots \]

\[ \epsilon: \quad \text{for } u \quad \text{by } t \quad \text{by } t \text{ again} \]

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

Proposal: Let

\[(\sigma_0, \varepsilon_0) \xrightarrow{(\text{cons}_0, \text{Snd}_0)} u_0 \xrightarrow{} \ldots \xrightarrow{(\text{cons}_{n-1}, \text{Snd}_{n-1})} u_{n-1} \xrightarrow{} (\sigma_n, \varepsilon_n), \quad n > 0,\]

be a finite (1), non-empty, maximal, consecutive sequence such that

- object \( u \) is alive in \( \sigma_0 \),
- \( u_0 = u \) and \( (\text{cons}_0, \text{Snd}_0) \) indicates dispatching to \( u \), i.e. \( \text{cons} = \{(u, v \mapsto d)\} \),
- there are no receptions by \( u \) in between, i.e.

\[ \text{cons}_i \cap \{u\} \times \text{Evs}(\epsilon, \emptyset) = \emptyset, \quad i > 1, \]

- \( u_{n-1} = u \) and \( u \) is stable only in \( \sigma_0 \) and \( \sigma_n \), i.e.

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

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

\[ (\sigma_0(u) = \sigma_{k_1}(u), \sigma_{k_2}(u), \ldots, \sigma_{k_N}(u) = \sigma_{n-1}(u)) \]

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

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.
**Putting It All Together**

---

**The Missing Piece: Initial States**

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

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

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

**Proposal:**

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

And set

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

**Other Approach:** (used by Rhapsody tool) multiplicity of classes.
We can read that as an abbreviation for an object diagram.
The semantics of the UML model

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

where

- some classes in \( \mathcal{C} \mathcal{D} \) 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 \( \mathcal{C} \mathcal{D} \),

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

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


