From Core State Machines in DSR

Definition 14.7.5.3 (Core State Machines): Let $S$ be the signature of core
state machines. It contains the classes $\mathcal{C}$ and the signals $\mathcal{D}$, as defined
above. The core state machines are the objects $u \in S$ with $\mathcal{C} \cap \sigma \neq \emptyset
\sigma_0 = \cdot \mathcal{D}$ such that either

- $\sigma_0$ is the empty set, and
- $\sigma_0$ contains the signals of $\mathcal{D}$.

We call the core state machines $S^\mathcal{B}$ the binding set of core state
machines.

Assume there is one core state machine $u$, $z \in S_0 \cap \mathcal{C}$, $\mathcal{D}$, and
$\sigma_0 = \cdot \mathcal{D}$.

Example: Derealised

\begin{itemize}
  \item If $\mathcal{C}$ is a core state machine, then $\mathcal{D}$ is a core
  state machine.
\end{itemize}
and

\[ \sigma, \varepsilon \sigma \left( \tau \right) : \sigma, \varepsilon \] = \text{stable} \left( \sigma \right) \wedge \left( \sigma \uparrow \right) \in \text{dom} \left( \mathcal{E} \right) \]
Example: Error Condition
At all don't let them modify each other's local state via links: Let objects only communicate by events, i.e.

- Recall that other objects of the same class can modify private attributes.

Proof left as exercise...

**Strict interfaces:**

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

Our semantics and notion of RTC-step doesn't have this (often desired) property.

Compositional would be: the behaviour of a set of objects is determined by the behaviour of each object "in isolation".

Adding classes (or even objects) may change the divergence behaviour of existing ones.

In the projection onto a single object we still

- the effect of interaction with other objects.

View, that is:

- non-compositional and

- global

Our definition of RTC-step takes a

Run-to-Completion Step: Discussion.

- one transition may be taken multiple times during an RTC-step.

Example: while one step corresponds to one transition in the state machine, note

- a run-to-completion step is in general not syntactically definable

\[ u(\sigma_0) \rightarrow u(\sigma_1) \rightarrow \cdots \rightarrow u(\sigma_n) \]

\( n \geq 2 \)

- \( k \) be the maximal sequence of indices

\( i \) such that

\( 0 = i_0 < i_1 < \cdots < i_k \)

Then we call the sequence

\( \sigma_{i_0} \sigma_{i_1} \cdots \sigma_{i_k} \)

the maximal sequence (from (local) configuration \( u \))

\( u(\sigma_{i_0}) \rightarrow u(\sigma_{i_1}) \rightarrow \cdots \rightarrow u(\sigma_{i_k}) \)

such that

\[ u(\sigma_{i_0}) = u(\sigma_{i_1}) = \cdots = u(\sigma_{i_k}) \]

stable and

\[ u(\sigma_{i_0}) \not\rightarrow u(\sigma_{i_1}) \not\rightarrow \cdots \not\rightarrow u(\sigma_{i_k}) \]

stable

\[ u(\sigma_{i_0}) \not\rightarrow u(\sigma_{i_1}) \not\rightarrow \cdots \not\rightarrow u(\sigma_{i_k}) \]

\( u(\sigma_{i_0}) \not\rightarrow u(\sigma_{i_1}) \not\rightarrow \cdots \not\rightarrow u(\sigma_{i_k}) \)

\[ \sigma_{i_0} \sigma_{i_1} \cdots \sigma_{i_k} = u(\sigma_{i_0}) \rightarrow u(\sigma_{i_1}) \rightarrow \cdots \rightarrow u(\sigma_{i_k}) \]

\( \sigma_{i_0} \sigma_{i_1} \cdots \sigma_{i_k} \)

a run-to-completion step...?
Initial States

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

- $S$: system configurations $(\sigma, \varepsilon)$
- $\rightarrow$: labelled transition relation $(\sigma, \varepsilon)$

Wanted: initial states $S_0$.

Proposal: Require a (finite) set of object diagrams $OD$ as part of a UML model $(CD, SM, OD)$. And set $S_0 = \{ (\sigma, \varepsilon) | \sigma \in G^{-1}(OD), OD \in OD, \varepsilon \text{ empty} \}$.

Other Approach: (used by Rhapsody tool) multiplicity of classes (plus initialisation code).

Semantics of UML Model (So Far)

The semantics of the UML model $M = (CD, SM, OD)$ where:

- some classes in $CD$ 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,
- $OD$ is a set of object diagrams over $CD$, 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 $M = (CD, SM, OD)$ be a UML model.
- We call $M$ consistent iff, for each OCL constraint $expr \in Inv(CD)$, $\sigma|_\varepsilon = expr$ for each "reasonable point" $(\sigma, \varepsilon)$ of computations of $M$.

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

Note: we could define $Inv(SM)$ similarly to $Inv(CD)$.
Let $M = (C, D, S, M, O, D)$ be a UML model.

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

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

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

Pragmatics:

• In UML-as-blueprint mode, if $S, M$ doesn't exist yet, then providing $M = (C, D, \emptyset, O, D)$ is typically asking the developer to provide state machines $S, M$ such that $M' = (C, D, S, M, O, D)$ is consistent.

If the developer makes a mistake, then $M'$ is inconsistent.

• Not so common (but existing): If $S, M$ 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 $S, M$ 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
