Recall: Constructive vs. Reflective Descriptions

[Harel, 1997] proposes to distinguish constructive and reflective descriptions:

- "A language is constructive if it contributes to the dynamic semantics of the model. That is, its constructs contain information needed in executing the model or in translating it into executable code."
- "Other languages are reflective or assertive, and can be used by the system modeler to capture parts of the thinking that go into building the model – behavior included – to derive and present views of the model, statically or during execution, or to set constraints on behavior in preparation for verification."

Note: No sharp boundaries!
Recall: What is a Requirement?

– 20 – 2015-02-03 – Sreflective –

Recall:

• The semantics of the UML model $M = (C, D, S, M, O, D)$ is the transition system $(S, - →, S_0)$ constructed according to discard/dispatch/commence-rules.

• The computations of $M$, denoted by $\llbracket M \rrbracket$, are the computations of $(S, - →, S_0)$.

Now:

A reflective description tells what shall or shall not be computed. More formally: a requirement $\vartheta$ is a property of computations; something which is either satisfied or not satisfied by a computation $\pi = (\sigma_0, \varepsilon_0) \rightarrow \cdots \rightarrow (\sigma_1, \varepsilon_1) \rightarrow \cdots \in \llbracket M \rrbracket$, denoted by $\pi|\vartheta = \top$ and $\pi|\vartheta = \bot$, respectively.

Simplest case: OCL constraint.

Example

Building Blocks

Example: What Is Required?

Building Blocks
After having consumed both, CrossingCtrl may get lost if LightsCtrl is not operational when receiving that event, it shall reply with 'lights on' within 1–3 time units, during this time the BarrierCtrl shall reply with 'barrier lights on' and 'barrier ok' within 1–5 time units, (dispatch time not included) it shall not be in state 'MvUp', if LightsCtrl is not operational when receiving that event, it shall finally send 'lights down' to LightsCtrl and BarrierCtrl, barrier ok' may occur in any order.

The CrossingCtrl has consumed a 'secreq' event

Whenever the CrossingCtrl has consumed a 'secreq' event

Example: Modes

AC: Activation condition (AM: invariantI: strict)

LSC: LSC Specialty: Activation

Part of the scenario has as precondition

AM: invariantI: strict

Example: Modes

LSC: LSC Specialty: Modes

With LSCs, Conditions and Local Invariants:

• 1

Expr

∈ {1

Expr

∈ expr

∈ expr

∈ expr

∈ expr

L

L

L

L

CrossingCtrl

C

C

C

C

Environment

Intuitive Semantics: A Partial Order on Simclasses

Partial Order Requirements

Simultaneously:

(i) Expr ∈ Expr .

Strictly After:

(ii) Expr ∈ Expr .

Explicitly Unordered:

(iii) Expr ∈ Expr .

Example: Blocks

Example: Blocks

Building Blocks
One major defect of MSCs and SDs: they don't say when the scenario has to/may be observed.

LSCs: Activation condition (AC $\in$ Expr), activation mode (AM $\in \{\text{init}, \text{inv}\})$, and pre-chart.

Intuition: (universal case)

- given a computation $\pi$, whenever $\text{expr}$ holds in a configuration $(\sigma_i, \varepsilon_i)$ of $\xi$
  - which is initial, i.e. $k = 0$
  - and if the pre-chart is observed from $k$ to $k + n$
  - then the main-chart has to follow from $k + n + 1$.

Example: What Is Required?

LSC: $L$
AC: actcond
AM: invariant

Environment:

- LightsCtrl
- Operational $[1, 3]$
- CrossingCtrl
- $t(10)$
- BarrierCtrl $[1, 5]$
- secreq
- lights
- on
- barrier
- down
- lights
- ok
- barrier
- ok
- $\neg MvUp$
- done

CrossingCtrl $\leftarrow$ LightsCtrl $\leftarrow$ BarrierCtrl

• Whenever the CrossingCtrl has consumed a 'secreq' event
• then it shall finally send 'lights on' and 'barrier down' to LightsCtrl and BarrierCtrl,
• if LightsCtrl is not 'operational' when receiving that event, the rest of this scenario doesn't apply; maybe there's another LSC for that case.
• if LightsCtrl is 'operational' when receiving that event, it shall reply with 'lights ok' within 1–3 time units,
• the BarrierCtrl shall reply with 'barrier ok' within 1–5 time units, during this time (dispatch time not included) it shall not be in state 'MvUp',
• 'lights ok' and 'barrier ok' may occur in any order.
• After having consumed both, CrossingCtrl may reply with 'done' to the environment.
Towards Automata

Language

Loops

Loops

You are here.
Example: The Language of a Model

Signal and Attribute Expressions

\( M \) is the language of a UML model and \( \mathcal{C} \) a structure of \( (\Sigma, \mathcal{D}, \mathcal{S}_\text{-md}) \) which can be the initial and consecutive set computations of logical variables, \( \mathbf{X} \), over the words of a model. A tuple of initial and consecutive model expressions is denoted by \( \mathbf{X} \).
Let \( (\sigma, \text{cons}, \text{Snd}) \in \Sigma_{DS} \times \tilde{A} \) be a triple consisting of system state, consume set, and send set.

Let \( \beta : X \rightarrow D(C) \) be a valuation of the logical variables.

Then
- \( (\sigma, \text{cons}, \text{Snd}) \models \beta \) true
- \( (\sigma, \text{cons}, \text{Snd}) \models \beta \neg \psi \) if and only if not \( (\sigma, \text{cons}, \text{Snd}) \models \beta \psi \)
- \( (\sigma, \text{cons}, \text{Snd}) \models \beta \psi_1 \lor \psi_2 \) if and only if \( (\sigma, \text{cons}, \text{Snd}) \models \beta \psi_1 \) or \( (\sigma, \text{cons}, \text{Snd}) \models \beta \psi_2 \)
- \( (\sigma, \text{cons}, \text{Snd}) \models \beta \text{expr} \) if and only if \( I[\text{llbracket \text{expr} /rrbracket}(\sigma, \beta)] = 1 \)
- \( (\sigma, \text{cons}, \text{Snd}) \models \beta E!x,y \) if and only if \( \exists \vec{d} \cdot (\beta(x), (E, \vec{d}), \beta(y)) \in \text{Snd} \)
- \( (\sigma, \text{cons}, \text{Snd}) \models \beta E?x,y \) if and only if \( \exists \vec{d} \cdot (\beta(x), (E, \vec{d}), \beta(y)) \in \text{cons} \)

Observation: semantics of models keeps track of sender and receiver at sending and consumption time. We disregard the event identity.

Alternative: keep track of event identities.
Model Consistency wrt. Interaction

• We assume that the set of interactions $I$ is partitioned into two (possibly empty) sets of universal and existential interactions, i.e.

$$I = I_\forall \cup I_\exists.$$

Definition. A model $M = (C, D, S, M, O, D, I)$ is called consistent (more precise: the constructive description of behaviour is consistent with the reflective one) if and only if

$$\forall I \in I_\forall L(M) \subseteq L(I)$$

and

$$\forall I \in I_\exists L(M) \cap L(I) \neq \emptyset.$$

Interactions as Reflective Description

• In UML, reflective (temporal) descriptions are subsumed by interactions.

• A UML model $M = (C, D, S, M, O, D, I)$ has a set of interactions $I$.

• An interaction $I \in I$ can be (OMG claim: equivalently)

  • diagrammed as sequence diagram,
  • timing diagram,
  • communication diagram (formerly known as collaboration diagram).

Figure 14.26 - Sequence Diagram with time and timing concepts

Figure 14.27 - Communication diagram

Figure 14.30 - Compact Lifeline with States

Figure 14.31 - Timing Diagram with more than one Lifeline and with Messages

Figure 14.28 - Interaction Overview Diagram representing a High Level Interaction diagram
Lifelines

Subject: CallQueue

LifelineDurationConstraint


Communication Behavior

LSCs: A Graphical Formalism for the Specification of


[OMG, 2007b, 522]

[OMG, 2007b, 518]

[OMG, 2007b, 170]

Figure 9.12 - In the Observer collaboration two roles, a Subject and an Observer, collaborate to produce the desired behavior. Any instance playing the Subject role must possess the properties specified by CallQueue, and similarly for

Figure 14.26 - Sequence Diagram with time and timing concepts

Figure 14.28 - Interaction Overview Diagram representing a High Level Interaction diagram

Figure 14.30 - Compact Lifeline with States

We focuse on one specific issue: time. There are three distinct aspects of time in interaction diagrams:

1. The life of an object, represented by the lifeline. The start and end of the life is denoted by the first and last message of the lifeline. Additionally, the lifeline indicates the role's capacity.

2. The time at which the messages are exchanged. This is represented by the interlacing of messages in the sequence diagram.

3. The duration of the interaction. This is represented by the length of the sequence diagram.

We focuse on the lifeline, which is the most prominent feature of an interaction diagram. The lifeline's duration is a critical aspect of time in interaction diagrams. It is represented by the horizontal dimension of the lifeline, which is the range of messages exchanged between the roles.

Why Sequence Diagrams?

The lifeline's duration is a natural way to represent time in interaction diagrams. It provides a clear visual representation of the duration of the interaction, which is crucial for understanding the behavior of the system.

Summary

In this section, we have discussed the lifeline's role in representing time in interaction diagrams. We have also highlighted the importance of the lifeline's duration in understanding the behavior of the system. In the next section, we will discuss the impact of the lifeline's duration on the performance of the system.