Software Design, Modelling and Analysis in UML

Lecture 20: Live Sequence Charts

2015-02-03

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

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

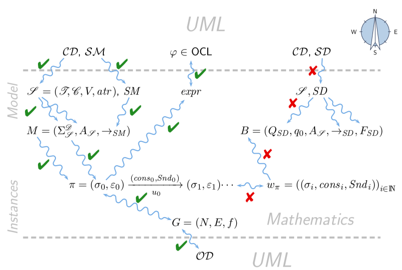
- Hierarchical State Machines completed.
- Behavioural feature (aka. methods).

This Lecture:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What does this LSC mean?
  - Are this UML model's state machines consistent with the interactions?
  - Please provide a UML model which is consistent with this LSC.
  - What is: activation, hot/cold condition, pre-chart, etc.?
- **Content:**
  - Reflective description of behaviour.
  - LSC concrete and abstract syntax.
  - LSC semantics.

You are here.

Course Map



Motivation: Reflective, Dynamic Descriptions of Behaviour

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."*

  A constructive description tells **how** things are computed (which can then be desired or undesired).
- *"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."*

  A reflective description tells **what** shall or shall not be computed.

Note: No sharp boundaries!

Recall: What is a Requirement?

Recall:

- The **semantics** of the **UML model** $\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$ is the **transition system** $(S, \to, S_0)$ constructed according to discard/dispatch/commence-rules.
- The **computations of** $\mathcal{M}$, denoted by $[\![\mathcal{M}]\!]$, are the computations of $(S, \to, S_0)$.

Now:

A reflective description tells **what** shall or shall not be computed.

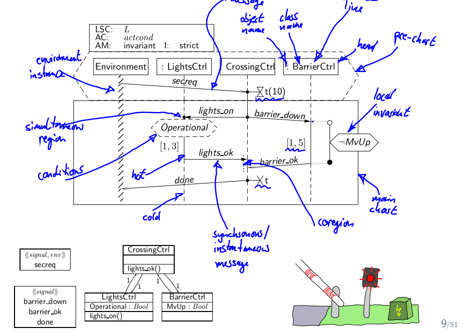
More formally: a requirement ϑ is a property of computations; something which is either satisfied or not satisfied by a computation

$$\pi = (\sigma_0, \varepsilon_0) \xrightarrow{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \xrightarrow{(cons_1, Snd_1)} \dots \in [\![\mathcal{M}]\!],$$

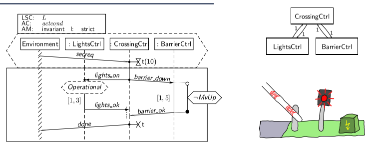
denoted by $\pi \models \vartheta$ and $\pi \not\models \vartheta$, resp.
Simplest case: OCL constraint.

Live Sequence Charts — Concrete Syntax

Example: Live Sequence Charts

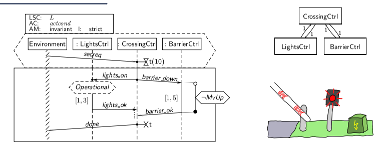


Example: What Is Required?

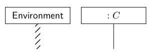


- **Whenever** the CrossingCtrl has consumed a 'secreq' event
- **then** it shall finally send 'lights_on' and 'barrier_down' to LightsCtrl and BarrierCtrl,
- if LightsCtrl **is not** 'operational' when receiving that event, the rest of this scenario doesn't apply; maybe there's another LSC for that 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.

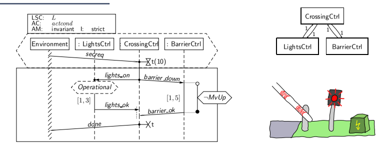
Building Blocks



- **Instance Lines:**

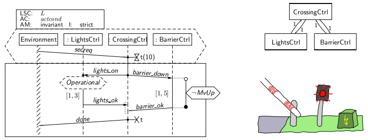


Building Blocks

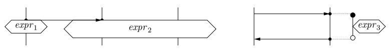


- **Messages:** (asynchronous or synchronous/instantaneous)

Building Blocks

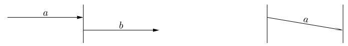


- **Conditions and Local Invariants:** $(expr_1, expr_2, expr_3 \in Expr_{\mathscr{S}})$

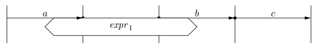


Intuitive Semantics: A Partial Order on Simclasses

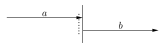
(i) **Strictly After:**



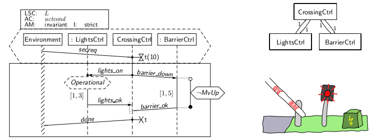
(ii) **Simultaneously:** (simultaneous region)



(iii) **Explicitly Unordered:** (co-region)



Partial Order Requirements



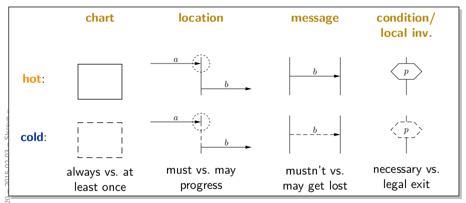
- **Whenever** the CrossingCtrl has consumed a 'secreq' event
- **then** it shall finally send 'lights_on' and 'barrier_down' to LightsCtrl and BarrierCtrl,
- if LightsCtrl **is not** 'operational' when receiving that event, the rest of this scenario doesn't apply; maybe there's another LSC for that 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.

LSC Specialty: Modes

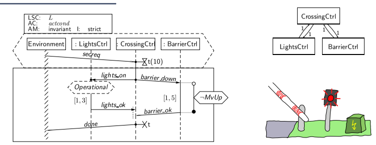
With LSCs,

- whole charts,
- locations, and
- elements

have a **mode** — one of **hot** or **cold** (graphically indicated by outline).



Example: Modes

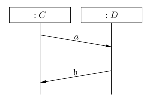


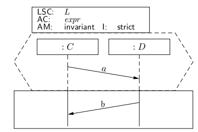
- **Whenever** the CrossingCtrl has consumed a 'secreq' event
- **then** it shall finally send 'lights_on' and 'barrier_down' to LightsCtrl and BarrierCtrl,
- if LightsCtrl **is not** 'operational' when receiving that event, the rest of this scenario doesn't apply; maybe there's another LSC for that 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.

LSC Specialty: Activation

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

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

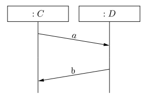


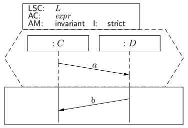


LSC Specialty: Activation

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

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



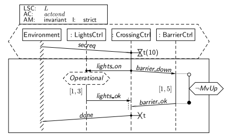


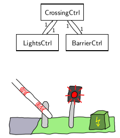
Intuition: (universal case)

- given a computation π, **whenever** $expr$ holds in a configuration $(\sigma_i, \varepsilon_i)$ of ξ
  - which is initial, i.e. $k = 0$, or (AM $= initial$)
  - whose k is not further restricted, (AM $= invariant$)

  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?

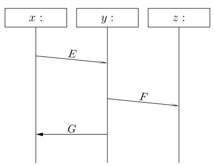




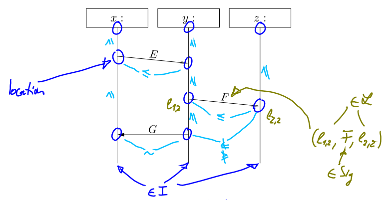
- **Whenever** the CrossingCtrl has consumed a 'secreq' event
- **then** it shall finally send 'lights_on' and 'barrier_down' to LightsCtrl and BarrierCtrl,
- if LightsCtrl **is not** 'operational' when receiving that event, the rest of this scenario doesn't apply; maybe there's another LSC for that 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.

Live Sequence Charts — Semantics in a Nutshell

Restricted Syntax



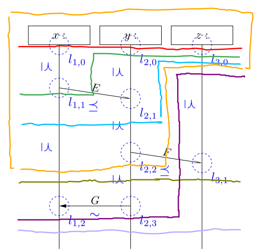
Restricted Abstract Syntax



$(I, (\mathscr{L}, \preceq), \sim, \mathscr{S}, \mathsf{Msg})$

instance lines, locations, partial, simultaneity, signature, messages

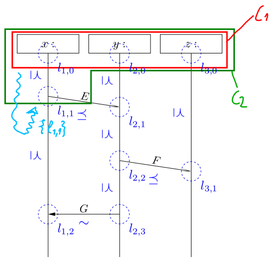
Cuts



A set $C \subseteq \mathscr{L}$ is called cut iff

- downward closed wrt. $\preceq$
- closed wrt. $\sim$
- at least one loc. per instance line (if more than one, then unordered)

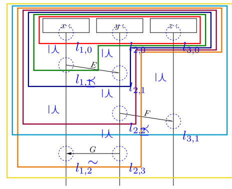
Firedsets

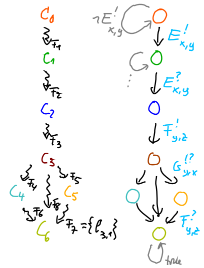


$C \rightsquigarrow_{\mathcal{F}} C'$ iff

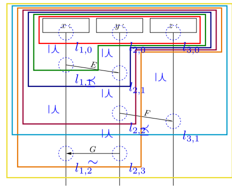
- $\mathcal{F} \neq \emptyset$
- $C' \setminus C = \mathcal{F}$
- for all event exceptions in $\mathcal{F}$, the sendings are in C
- direct successor: $\forall \ell \in \mathcal{F}\ \exists \ell' \in \mathcal{L} \bullet \ell' \preceq \ell \land \forall \ell'' \in \mathcal{L} \bullet \ell'' \preceq \ell \Rightarrow \ell'' \preceq \ell'$

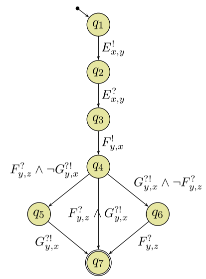
Towards Automata



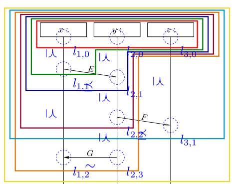


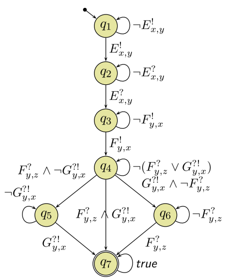
Alphabet — Progress Transitions



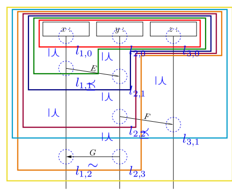


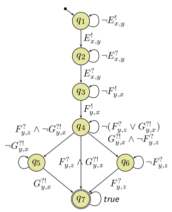
Loops





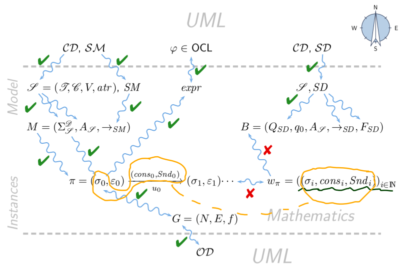
Language





You are here.

Course Map



Language of a Model

Words over Signature

Definition. Let $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr, \mathscr{E})$ be a signature and $\mathscr{D}$ a structure of $\mathscr{S}$. A **word** over $\mathscr{S}$ and $\mathscr{D}$ is an infinite sequence

$$(\sigma_i, cons_i, Snd_i)_{i \in \mathbb{N}_0} \in \left(\Sigma_{\mathscr{S}}^{\mathscr{D}} \times 2^{\mathscr{D}(\mathscr{C}) \times Evs(\mathscr{E}, \mathscr{D}) \times \mathscr{D}(\mathscr{C})} \times 2^{\mathscr{D}(\mathscr{C}) \times Evs(\mathscr{E}, \mathscr{D}) \times \mathscr{D}(\mathscr{C})}\right)^{\omega}.$$

The Language of a Model

Recall: A UML model $\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$ and a structure $\mathscr{D}$ denotes a set $[\![\mathcal{M}]\!]$ of (initial and consecutive) **computations** of the form

$$(\sigma_0, \varepsilon_0) \xrightarrow{a_0} (\sigma_1, \varepsilon_1) \xrightarrow{a_1} (\sigma_2, \varepsilon_2) \xrightarrow{a_2} \dots \text{ where}$$

$$a_i = (cons_i, Snd_i, u_i) \in \underbrace{2^{\mathscr{D}(\mathscr{C}) \times Evs(\mathscr{E}, \mathscr{D}) \times \mathscr{D}(\mathscr{C})} \times 2^{\mathscr{D}(\mathscr{C}) \times Evs(\mathscr{E}, \mathscr{D}) \times \mathscr{D}(\mathscr{C})}}_{=:\tilde{A}} \times \mathscr{D}(\mathscr{C}).$$

For the connection between models and interactions, we **disregard** the configuration of **the ether** and **who** made the step, and define as follows:

Definition. Let $\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$ be a UML model and $\mathscr{D}$ a structure. Then

$$\mathcal{L}(\mathcal{M}) := \{(\sigma_i, cons_i, Snd_i)_{i \in \mathbb{N}_0} \in (\Sigma_{\mathscr{S}}^{\mathscr{D}} \times \tilde{A})^{\omega} \mid \exists (\varepsilon_i, u_i)_{i \in \mathbb{N}_0} : (\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \dots \in [\![\mathcal{M}]\!]\}$$

is the **language** of $\mathcal{M}$.

Example: The Language of a Model

$$\mathcal{L}(\mathcal{M}) := \{(\sigma_i, cons_i, Snd_i)_{i \in \mathbb{N}_0} \in (\Sigma_{\mathscr{S}}^{\mathscr{D}} \times \tilde{A})^{\omega} \mid \exists (\varepsilon_i, u_i)_{i \in \mathbb{N}_0} : (\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \dots \in [\![\mathcal{M}]\!]\}$$

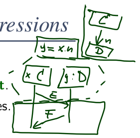
Signal and Attribute Expressions

- Let $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr, \mathscr{E})$ be a signature and X a set of logical variables,
- The signal and attribute expressions $Expr_{\mathscr{S}}(\mathscr{E}, X)$ are defined by the grammar:

$$\psi ::= \textit{true} \mid expr \mid E^{!}_{x,y} \mid E^{?}_{x,y} \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid E^{!?}_{x,y}$$

where $expr : Bool \in Expr_{\mathscr{S}}$, $E \in \mathscr{E}$, $x, y \in X$. ← set of variables

Satisfaction of Signal and Attribute Expressions



- Let $(\sigma, cons, Snd) \in \Sigma_{\mathscr{S}}^{\mathscr{D}} \times \tilde{A}$ be a triple consisting of **system state**, **consume set**, and **send set**.
- Let $\beta : X \to \mathscr{D}(\mathscr{C})$ be a valuation of the logical variables.

Then

- $(\sigma, cons, Snd) \models_\beta$ *true*
- $(\sigma, cons, Snd) \models_\beta \neg\psi$ if and only if not $(\sigma, cons, Snd) \models_\beta \psi$
- $(\sigma, cons, Snd) \models_\beta \psi_1 \vee \psi_2$ if and only if $(\sigma, cons, Snd) \models_\beta \psi_1$ or $(\sigma, cons, Snd) \models_\beta \psi_2$
- $(\sigma, cons, Snd) \models_\beta expr$ if and only if $I[\![expr]\!](\sigma, \beta) = 1$
- $(\sigma, cons, Snd) \models_\beta E^!_{x,y}$ if and only if $\exists \vec{d} \bullet (\beta(x), (E, \vec{d}), \beta(y)) \in Snd$
- $(\sigma, cons, Snd) \models_\beta E^?_{x,y}$ if and only if $\exists \vec{d} \bullet (\beta(x), (E, \vec{d}), \beta(y)) \in cons$

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.

TBA over Signature

Definition. A TBA

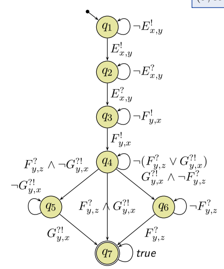
$$\mathcal{B} = (Expr_{\mathcal{B}}(X), X, Q, q_{ini}, \to, Q_F)$$

where $Expr_{\mathcal{B}}(X)$ is the set of **signal and attribute expressions** $Expr_{\mathscr{S}}(\mathscr{E}, X)$ over signature $\mathscr{S}$ is called **TBA over** $\mathscr{S}$.

- Any word over $\mathscr{S}$ and $\mathscr{D}$ is then a word for $\mathcal{B}$.
  (By the satisfaction relation defined on the previous slide; $\mathscr{D}(X) = \mathscr{D}(\mathscr{C})$.)
- Thus a TBA over $\mathscr{S}$ accepts words of models with signature $\mathscr{S}$.
  (By the previous definition of TBA.)

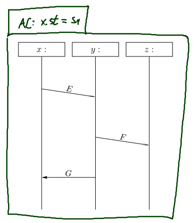
TBA over Signature Example

$(\sigma, cons, Snd) \models_\beta expr$ iff $I[\![expr]\!](\sigma, \beta) = 1$;
$(\sigma, cons, Snd) \models_\beta E^!_{x,y}$ iff $(\beta(x), (E, \vec{d}), \beta(y)) \in Snd$

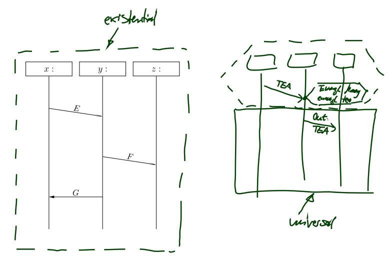


Activation, Chart Mode

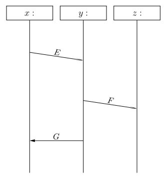
Activation Condition



Universal vs. Existential Charts

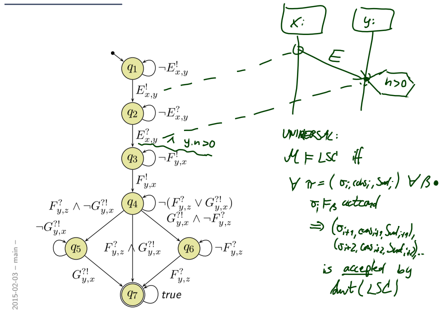


Prechart



Conditions

Conditions



Back to UML: Interactions

Model Consistency wrt. Interaction

- We assume that the set of interactions $\mathscr{I}$ is partitioned into two (possibly empty) sets of **universal** and **existential** interactions, i.e.

$$\mathscr{I} = \mathscr{I}_\forall \dot\cup \mathscr{I}_\exists.$$

Definition. A model

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

is called **consistent** (more precise: the constructive description of behaviour is consistent with the reflective one) if and only if

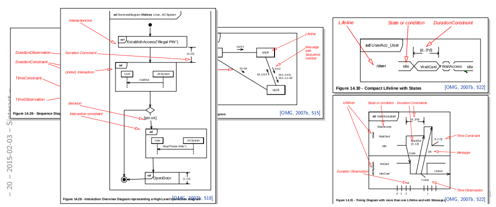
$$\forall \mathcal{I} \in \mathscr{I}_\forall : \mathcal{L}(\mathcal{M}) \subseteq \mathcal{L}(\mathcal{I})$$

and

$$\forall \mathcal{I} \in \mathscr{I}_\exists : \mathcal{L}(\mathcal{M}) \cap \mathcal{L}(\mathcal{I}) \neq \emptyset.$$

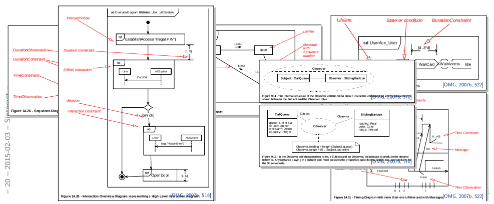
Interactions as Reflective Description

- In UML, reflective (temporal) descriptions are subsumed by **interactions**.
- A UML model $\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD}, \mathscr{I})$ has a set of interactions $\mathscr{I}$.
- An interaction $\mathcal{I} \in \mathscr{I}$ can be (OMG claim: equivalently) **diagrammed** as
  - **sequence diagram**, **timing diagram**, or
  - **communication diagram** (formerly known as collaboration diagram).



Interactions as Reflective Description

- In UML, reflective (temporal) descriptions are subsumed by **interactions**.
- A UML model $\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD}, \mathscr{I})$ has a set of interactions $\mathscr{I}$.
- An interaction $\mathcal{I} \in \mathscr{I}$ can be (OMG claim: equivalently) **diagrammed** as
  - **sequence diagram**, **timing diagram**, or
  - **communication diagram** (formerly known as collaboration diagram).



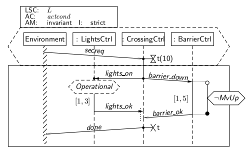
Why Sequence Diagrams?

Most Prominent: Sequence Diagrams — with **long history**:

- **Message Sequence Charts**, standardized by the ITU in different versions, often accused to lack a formal semantics.
- **Sequence Diagrams** of UML 1.x

Most severe **drawbacks** of these formalisms:

- unclear **interpretation**: example scenario or invariant?
- unclear **activation**: what triggers the requirement?
- unclear **progress** requirement: must all messages be observed?
- **conditions** merely comments
- no means to express **forbidden scenarios**



Thus: Live Sequence Charts

- **SDs of UML 2.x** address **some** issues, yet the standard exhibits unclarities and even contradictions [Harel and Maoz, 2007, Störrle, 2003]
- For the lecture, we consider **Live Sequence Charts** (LSCs) [Damm and Harel, 2001, Klose, 2003, Harel and Marelly, 2003], who have a common fragment with UML 2.x SDs [Harel and Maoz, 2007]
- **Modelling guideline**: stick to that fragment.

References

[Damm and Harel, 2001] Damm, W. and Harel, D. (2001). LSCs: Breathing life into Message Sequence Charts. *Formal Methods in System Design*, 19(1):45–80.

[Harel, 1997] Harel, D. (1997). Some thoughts on statecharts, 13 years later. In Grumberg, O., editor, *CAV*, volume 1254 of *LNCS*, pages 226–231. Springer-Verlag.

[Harel and Maoz, 2007] Harel, D. and Maoz, S. (2007). Assert and negate revisited: Modal semantics for UML sequence diagrams. *Software and System Modeling (SoSyM)*. To appear. (Early version in SCESM'06, 2006, pp. 13-20).

[Harel and Marelly, 2003] Harel, D. and Marelly, R. (2003). *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer-Verlag.

[Klose, 2003] Klose, J. (2003). *LSCs: A Graphical Formalism for the Specification of Communication Behavior*. PhD thesis, Carl von Ossietzky Universität Oldenburg.

[OMG, 2007a] OMG (2007a). Unified modeling language: Infrastructure, version 2.1.2. Technical Report formal/07-11-04.

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

[Störrle, 2003] Störrle, H. (2003). Assert, negate and refinement in UML-2 interactions. In Jürjens, J., Rumpe, B., France, R., and Fernandez, E. B., editors, *CSDUML 2003*, number TUM-I0323. Technische Universität München.