Softwaretechnik / Software-Engineering

Lecture 08: Scenarios and Use Cases

2015-06-08

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:
- Consistency, Completeness, etc. for Decision Tables.

This Lecture:
- Educational Objectives: Capabilities for following tasks/questions.
  - What is a scenario/an anti-scenario?
  - What is included in a use case? In a use case diagram?
  - What is the abstract syntax of this Live Sequence Chart (LSC)?
  - Which are the cuts and firedsets of this LSC?
  - Construct the TBA of a given LSC body.
  - Given a set of LSCs, which scenario/anti-scenario/requirement is formalised by them?
  - Formalise this positive scenario/anti-scenario/requirement using LSCs.

- Content:
  - Scenarios in Requirements Engineering
  - User Stories; Use Cases and Diagrams
  - Live Sequence Charts
You Are Here

Introduction
- Development Process, Metrics
  - L 1: 20.4., Mo
  - T 1: 23.4., Do
  - L 2: 27.4., Mo
  - L 3: 30.4., Do
  - L 4: 4.5., Mo
  - T 2: 7.5., Do
  - L 5: 11.5., Mo
  - - 14.5., Do
  - L 6: 18.5., Mo
  - L 7: 21.5., Do
  - - 25.5., Mo
  - - 28.5., Do
  - T 3: 1.6., Mo
  - - 4.6., Do
  - L 8: 8.6., Mo
  - L 9: 11.6., Do
  - L 10: 15.6., Mo
  - T 4: 18.6., Do
  - L 11: 22.6., Mo
  - L 12: 25.6., Do
  - L 13: 29.6., Mo
  - T 5: 2.7., Do
  - L 14: 6.7., Mo
  - L 15: 9.7., Do
  - L 16: 13.7., Mo
  - L 17: 16.7., Do
  - T 6: 20.7., Mo
  - L 18: 23.7., Do

Requirements Engineering

Architecture & Design

Constructive Models

Testing, Formal Verification

Invited Talks

Wrap-Up
Scenarios

Recall: The Crux of Requirements Engineering

One quite effective approach: try to approximate the requirements with positive and negative scenarios.

- Dear customer, please describe example usages of the desired system.
  “If the system is not at all able to do this, then it’s not what I want.”
- Dear customer, please describe behaviour that the desired system must not show.
  “If the system does this, then it’s not what I want.”
- From there on, refine and generalise: what about exceptional cases? what about corner-cases? etc.
Example: Vending Machine

- Positive scenario:
  (i) Insert one 1 euro and one 50 cent coin.
  (ii) Press the ‘softdrink’ button.
  (iii) Get a softdrink.
  (iv) Get 50 cent change.

- Negative scenario:
  (i) After switching on, insert no money.
  (ii) Press the ‘tea’ button.
  (iii) Get a tea.
  (iv) Get 100 € change.

Notations for Scenarios

- The idea of scenarios (sometimes without negative or anti-scenarios) (re-)occurs in many process models or software development approaches.

- First prominent recognition: OOSE (Jacobson, 1992)

- In the following, we will discuss two and a half notations (in increasing formality):
  - User Stories (part of Extreme Programming)
  - Use Cases and Use Case Diagrams (OOSE)
  - Sequence Diagrams (here: Live Sequence Charts (Damm and Harel, 2001))
**User Stories (Beck, 1999)**

“A User Story is a concise, written description of a piece of functionality that will be valuable to a user (or owner) of the software.”

Per user story, one file card with the user story, e.g. following the pattern:

- **As a [role] I want [something] so that [benefit].**

and in addition:

- **unique identifier** (e.g. unique number),
- **priority** (from 1 (highest) to 10 (lowest)) assigned by customer,
- **effort**, estimated by developers,
- back side of file card:
  (acceptance) **test case(s)** — how to tell whether the user story has been realised.
User Stories: Discussion

Proposed card layout (front side):

<table>
<thead>
<tr>
<th>priority</th>
<th>unique identifier, name</th>
<th>estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>As a [role] I want [something] so that [benefit].</td>
<td></td>
</tr>
</tbody>
</table>

✓ easy to create
✓ close contact to customer
✗ customers are usually not trained in writing requirements
✗ may get difficult to keep overview over whole system to be developed
✗ strong dependency on competent developers
✗ estimation of effort may be difficult
✗ not easy to cover non-functional requirements and restrictions

(Balzert, 2009)

Recall:

Natural Language Patterns

Natural language requirements can be written using A, B, C, D, E, F where

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>clarifies when and under what conditions the activity takes place</td>
</tr>
<tr>
<td>B</td>
<td>is MUST (obligation), SHOULD (wish), or WILL (intention); also: MUST NOT (forbidden)</td>
</tr>
<tr>
<td>C</td>
<td>is either “the system” or the concrete name of a (sub-)system</td>
</tr>
<tr>
<td>D</td>
<td>one of three possibilities:</td>
</tr>
<tr>
<td></td>
<td>• “does”, description of a system activity,</td>
</tr>
<tr>
<td></td>
<td>• “offers”, description of a function offered by the system to somebody,</td>
</tr>
<tr>
<td></td>
<td>• “is able if”, usage of a function offered by a third party, under certain conditions</td>
</tr>
<tr>
<td>E</td>
<td>extensions, in particular an object</td>
</tr>
<tr>
<td>F</td>
<td>the actual process word (what happens)</td>
</tr>
</tbody>
</table>

Example:

After office hours (= A), the system (= C) should (= B) offer to the operator (= D) a backup (= F) of all new registrations to an external medium (= E).
Use Cases
**Use Case: Definition**

**use case** — A sequence of interactions between an actor (or actors) and a system triggered by a specific actor, which produces a result for an actor.

(Jacobson, 1992)

- **participants**: the system and at least one actor,
- **actor**: an actor represents what interacts with the system.

  - An actor is a role, which a user or an external system may assume when interacting with the system under design.
  - Actors are not part of the system, thus they are **not described in detail**.
  - Actions of actors are **non-deterministic** (possibly constrained by domain model).

- A use case is triggered by a **stimulus** as input by the main actor.
- A use case is **goal oriented**, i.e. the main actor wants to reach a particular goal.
- A use case describes all interactions between the system and the participating actors that are needed to achieve the goal (or fail to achieve the goal for reasons).
- A use case ends when the desired goal is achieved, or when it is clear that the desired goal cannot be achieved.

**Use Case Example**

<table>
<thead>
<tr>
<th>name</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>goal</td>
<td>the client wants access to the ATM</td>
</tr>
<tr>
<td>pre-condition</td>
<td>the ATM is operational, the welcome screen is displayed, card and PIN of client are available</td>
</tr>
<tr>
<td>post-condition</td>
<td>client accepted, services of ATM are offered</td>
</tr>
<tr>
<td>post-cond. in exceptional case</td>
<td>access denied, card returned or withheld, welcome screen displayed</td>
</tr>
<tr>
<td>actors</td>
<td>client (main actor), bank system</td>
</tr>
<tr>
<td>open questions</td>
<td>none</td>
</tr>
</tbody>
</table>
| normal case | 1. client inserts card  
2. ATM read card, sends data to bank system  
3. bank system checks validity  
4. ATM shows PIN screen  
5. client enters PIN  
6. ATM reads PIN, sends to bank system  
7. bank system checks PIN  
8. ATM accepts and shows main menu |
| exception case | card not readable  
2a.1 ATM displays “card not readable”  
2a.2 ATM returns card  
2a.3 ATM shows welcome screen |
<table>
<thead>
<tr>
<th>open questions</th>
<th>normal case</th>
</tr>
</thead>
</table>
| none          | 1. client inserts card  
2. ATM read card, sends data to bank system  
3. bank system checks validity  
4. ATM shows PIN screen  
5. client enters PIN  
6. ATM reads PIN, sends to bank system  
7. bank system checks PIN  
8. ATM accepts and shows main menu |

| exception case 2a | card not readable  
2a.1 ATM displays “card not readable”  
2a.2 ATM returns card  
2a.3 ATM shows welcome screen |

| exception case 2b | card readable, but not ATM card |
| exception case 2c | no connection to bank system |
| exception case 3a | card not valid or disabled |
| exception case 5a | client cancels |
| exception case 5b | client doesn’t react within 5 s |
| exception case 6a | no connection to bank system |
| exception case 7a | first or second PIN wrong |
| exception case 7b | third PIN wrong |

*Use Case Diagrams*

(Ludewig and Lichter, 2013; V-Modell XT, 2006)
Use Case Diagrams

More notation:

- $\textit{extends}$
- $\textit{uses}$
- $\textit{include}$

Use Case Diagram: Bigger Examples

(Ludewig and Lichter, 2013)
Use Case Diagram: Bigger Examples

(V-Modell XT, 2006)

Sequence Diagrams
Recall: Vending Machine Example

- Positive scenario:
  (i) Insert one 1 euro and one 50 cent coin.
  (ii) Press the ‘softdrink’ button.
  (iii) Get a softdrink.
  (iv) Get 50 cent change.

- Negative scenario:
  (i) After switching on, insert no money.
  (ii) Press the ‘tea’ button.
  (iii) Get a tea.
  (iv) Get 100 € change.

<table>
<thead>
<tr>
<th>notation</th>
<th>understandable by customer</th>
<th>precision</th>
<th>complexity of definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>natural language</td>
<td>feels like ++</td>
<td>-</td>
<td>+++···</td>
</tr>
<tr>
<td>visual formalism</td>
<td>(+)</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>(temporal) logic</td>
<td>-</td>
<td>++</td>
<td>-</td>
</tr>
</tbody>
</table>

History

Most Prominent: Sequence Diagrams — with long history:

- Message Sequence Charts,
  standardized by the ITU in different versions (ITU Z.120, 1st edition: 1993),
  often accused to lack a formal semantics.
- Sequence Diagrams of UML 1.x (one of three main authors: I. Jacobson)

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 (in language) 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.

The Plan

LSCs as another formal software specification:
- LSC body abstract and concrete syntax,
- Excursion: Büchi automata (TBA),
- TBA construction: Cuts, Firedsets, Transitions,
- Language of an LSC body,
- Putting it all together:
  - Activation modes, Pre-charts, Existential and universal LSCs,
- Some bigger LSC examples,
- Discussion of MSC issues named on previous slide.

- Some concluding notes on the block Requirements Engineering.

- Next block: design, architecture...
**Definition.** [LSC Body] Let $E$ be a set of events and $C$ a set of atomic propositions. An LSC body over $E$ and $C$ is a tuple

$$(\mathcal{L}, \leq, \sim), I, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)$$

where

- $\mathcal{L}$ is a finite, non-empty of locations with
  - a partial order $\leq \subseteq \mathcal{L} \times \mathcal{L}$,
  - a symmetric simultaneity relation $\sim \subseteq \mathcal{L} \times \mathcal{L}$ disjoint with $\leq$, i.e. $\leq \cap \sim = \emptyset$,
- $I = \{I_1, \ldots, I_n\}$ is a partitioning of $\mathcal{L}$; elements of $I$ are called instance line,
- $\text{Msg} \subseteq \mathcal{L} \times E \times \mathcal{L}$ is a set of messages with $(l, E, l') \in \text{Msg}$ only if $(l, l') \in \llcorner \cup \sim$; message $(l, E, l')$ is called instantaneous iff $l \sim l'$ and asynchronous otherwise,
- $\text{Cond} \subseteq (\mathcal{L} \setminus \emptyset) \times \Phi(C)$ is a set of conditions with $(L, \phi) \in \text{Cond}$ only if $l \sim l'$ for all $l \neq l' \in L$,
- $\text{LocInv} \subseteq \mathcal{L} \times \{\circ, \bullet\} \times \Phi(C) \times \mathcal{L} \times \{\circ, \bullet\}$ is a set of local invariants with $(l, \iota, \phi, l', \iota') \in \text{LocInv}$ only if $l \llcorner l'$, $\circ$: exclusive, $\bullet$: inclusive,
- $\Theta : \mathcal{L} \cup \text{Msg} \cup \text{Cond} \cup \text{LocInv} \rightarrow \{\text{hot}, \text{cold}\}$ assigns to each location and each element a temperature.
LSC Body Example

- \(\mathcal{L}: l_{1,0} < l_{1,1} < l_{1,2} < l_{1,3}, \ l_{1,2} < l_{1,4}, \ l_{2,0} < l_{2,1} < l_{2,2} < l_{2,3}, \ l_{3,0} < l_{3,1} < l_{3,2},\)
- \(l_{1,1} < l_{2,1}, \ l_{2,2} < l_{1,2}, \ l_{2,3} < l_{1,3}, \ l_{3,2} < l_{1,4},\)

\(l_{2,2} \sim l_{3,1},\)

- \(\mathcal{I} = \{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}\}, \{l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}\}, \{l_{3,0}, l_{3,1}, l_{3,2}\}\),

- \(\text{Msg} = \{(l_{1,1}, A, l_{2,1}), (l_{2,2}, B, l_{1,2}), (l_{2,2}, C, l_{3,1}), (l_{2,3}, D, l_{1,3}), (l_{3,2}, E, l_{1,4})\}\)

- \(\text{Cond} = \{(l_{2,2}, c_2 \land c_3)\},\)

- \(\text{LocInv} = \{\{l_{1,1}, 0, c_1, l_{1,2}, \bullet\}\}\)

\(\Theta(C) = \text{col}\)
LSC Body Example

- $L: l_{1,0} < l_{1,1} < l_{1,2} < l_{1,3}, \; l_{1,2} < l_{1,4}, \; l_{2,0} < l_{2,1} < l_{2,2} < l_{2,3}, \; l_{2,1} < l_{2,2}, \; l_{2,2} < l_{1,2}, \; l_{2,3} < l_{1,3}, \; l_{3,0} < l_{3,1} < l_{3,2}, \; l_{1,1} < l_{2,1}, \; l_{2,2} < l_{1,2}, \; l_{2,3} < l_{1,3}, \; l_{3,2} < l_{1,4}, \; l_{2,2} \sim l_{3,1},$
- $I = \{(l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}), \{l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}\}, \{l_{3,0}, l_{3,1}, l_{3,2}\}\},$
- $Msg = \{(l_{1,1}, A, l_{2,1}), (l_{2,2}, l_{1,2}), (l_{2,3}, A, l_{3,1}), (l_{2,3}, D, l_{1,4}), (l_{3,2}, E, l_{1,4})\}$
- $Cond = \{(\{l_{2,3}\}, c_3 \land c_5)\},$
- $LocInv = \{(l_{1,1}, \neg, e_1, l_{1,2}, \bullet)\}$
**LSC Body Example**

- $L: l_{1,0} < l_{1,1} < l_{1,2} < l_{1,3}$, $l_{1,2} < l_{1,4}$, $l_{2,0} < l_{2,1} < l_{2,2} < l_{2,3}$, $l_{5,0} < l_{3,1} < l_{3,2}$, $l_{1,1} < l_{2,1}$, $l_{2,2} < l_{1,2}$, $l_{2,3} < l_{1,3}$, $l_{3,2} < l_{1,4}$, $l_{2,2} \sim l_{3,1}$.
- $I = \{(l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}), (l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}), (l_{3,0}, l_{3,1}, l_{3,2})\}$.
- $Msg = \{(l_{1,1}, A, l_{2,1}), (l_{2,2}, B, l_{1,2}), (l_{2,3}, C, l_{3,1}), (l_{2,3}, D, l_{1,3}), (l_{3,2}, E, l_{1,4})\}$.
- $Cond = \{\{(l_{2,3}, c_2 \land c_3)\}\}.$
- $LocInv = \{(l_{1,1}, \forall, c_1, l_{1,2}, \bullet)\}$.
LSC Body Example

- $L : l_{1,0} < l_{1,1} < l_{1,2} < l_{1,3}, \ l_{1,2} < l_{1,4}, \ l_{2,0} < l_{2,1} < l_{2,2} < l_{2,3}, \ l_{3,0} < l_{3,1} < l_{3,2}, \ l_{1,1} < l_{2,1}, \ l_{2,2} < l_{1,2}, \ l_{2,3} < l_{1,3}, \ l_{3,2} < l_{1,4}, \ l_{2,2} \sim l_{3,1}$.
- $I = \{(l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}), (l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}, l_{3,0}, l_{3,1}, l_{3,2})\}$.
- $Msg = \{(l_{1,1}, A, l_{2,1}), (l_{2,2}, B, l_{1,2}), (l_{2,2}, C, l_{3,1}), (l_{2,3}, D, l_{1,3}), (l_{3,2}, E, l_{1,4})\}$.
- $Cond = \{\{(l_{2,2}, c_2 \land c_3)\}\}$.
- $LocIn = \{(l_{1,1}, \gamma, c_1, l_{1,2}, \bullet)\}$

LSC Body Example

- $L : l_{1,0} < l_{1,1} < l_{1,2} < l_{1,3}, \ l_{1,2} < l_{1,4}, \ l_{2,0} < l_{2,1} < l_{2,2} < l_{2,3}, \ l_{3,0} < l_{3,1} < l_{3,2}, \ l_{1,1} < l_{2,1}, \ l_{2,2} < l_{1,2}, \ l_{2,3} < l_{1,3}, \ l_{3,2} < l_{1,4}, \ l_{2,2} \sim l_{3,1}$.
- $I = \{(l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}), (l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}, l_{3,0}, l_{3,1}, l_{3,2})\}$.
- $Msg = \{(l_{1,1}, A, l_{2,1}), (l_{2,2}, B, l_{1,2}), (l_{2,2}, C, l_{3,1}), (l_{2,3}, D, l_{1,3}), (l_{3,2}, E, l_{1,4})\}$.
- $Cond = \{\{(l_{2,2}, c_2 \land c_3)\}\}$.
- $LocIn = \{(l_{1,1}, \gamma, c_1, l_{1,2}, \bullet)\}$
LSC Body Example

- \( \mathcal{L} : l_{1,0} < l_{1,1} < l_{1,2} < l_{1,3}, \ l_{1,2} < l_{1,4}, \ l_{2,0} < l_{2,1} < l_{2,2} < l_{2,3}, \ l_{5,0} < l_{3,1} < l_{3,2}, \ l_{1,1} < l_{2,1}, \ l_{2,2} < l_{1,2}, \ l_{2,3} < l_{1,3}, \ l_{3,2} < l_{1,4}, \ l_{2,2} \sim l_{3,1}, \)

- \( \mathcal{I} = \{(l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}), \{l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}, l_{3,0}, l_{3,1}, l_{3,2}\}\}

- \( \text{Msg} = \{(l_{1,1}, A, l_{2,1}), (l_{2,2}, B, l_{1,2}), (l_{2,2}, C, l_{3,1}), (l_{2,3}, D, l_{1,3}), (l_{3,2}, E, l_{1,4})\}\)

- \( \text{Cond} = \{\{(l_{2,2}, c_0 \wedge c_2)\}\}\)

- \( \text{LocInv} = \{(l_{1,1}, c_3, l_{1,2}, \bullet)\}\)
LSC Body Example

- $\mathcal{L} : l_{1,0} < l_{1,1} < l_{1,2} < l_{1,3}, \ l_{1,2} < l_{1,4}, \ l_{2,0} < l_{2,1} < l_{2,2} < l_{2,3}, \ l_{3,0} < l_{3,1} < l_{3,2}, \ l_{1,1} < l_{2,1}, \ l_{2,2} < l_{1,2}, \ l_{2,3} < l_{1,3}, \ l_{3,2} < l_{1,4}, \ l_{2,2} \sim l_{3,1},$
- $I = \{(l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}), \ {l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}}, \ {l_{3,0}, l_{3,1}, l_{3,2}}\},$
- $\text{Msg} = \{(l_{1,1}, A, l_{2,1}), (l_{2,2}, B, l_{1,2}), (l_{2,2}, C, l_{3,1}), (l_{2,3}, D, l_{1,3}), (l_{3,2}, E, l_{1,4})\}$
- $\text{Cond} = \{(l_{3,2}, c_2 \wedge c_3)\}$
- $\text{LocInv} = \{(l_{1,1}, c_1, l_{1,2})\}$

And The Other Way Round

- $I = \{(l_{0,0}, l_{1,0}, l_{2,0}), \ {l_{0,0}, l_{1,1}, l_{2,1}}, \ {l_{0,0}, l_{1,2}, l_{2,2}}, \ {l_{0,0}, l_{1,3}, l_{2,3}}\}$
- $\text{Msg} = \{(l_{0,0}, A, l_{0,0}), \ {l_{0,0}, A, l_{0,1}}, \ {l_{0,0}, A, l_{0,2}}, \ {l_{0,0}, A, l_{0,3}}\}$
- $\text{Cond} = \{(l_{2,3}, c_2 \wedge c_3)\}$
- $\text{LocInv} = \{(l_{0,0}, c_1, l_{0,2}), \ (l_{0,0}, c_1, l_{0,3})\}$
Concrete vs. Abstract Syntax

- $L : l_{1,0} < l_{1,1} < l_{1,2} < l_{1,3}$, $l_{1,2} < l_{1,4}$, $l_{2,0} < l_{2,1} < l_{2,2} < l_{2,3}$, $l_{3,0} < l_{3,1} < l_{3,2}$, $l_{1,1} < l_{2,1}$, $l_{2,2} < l_{1,2}$, $l_{2,3} < l_{1,3}$, $l_{3,2} < l_{1,4}$, $l_{2,2} \sim l_{3,1}$,
- $I = \{\{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}\}, \{l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}\}, \{l_{3,0}, l_{3,1}, l_{3,2}\}\}$,
- $Msg = \{(l_{1,1}, A, l_{1,2}), (l_{2,2}, B, l_{1,2}), (l_{2,2}, C, l_{3,1}), (l_{2,3}, D, l_{1,2}), (l_{3,2}, E, l_{1,4})\}$
- $Cond = \{\{(l_{2,2}), c_2 \land c_3\}\}$,
- $LocInv = \{(l_{1,1}, \phi, l_{1,2}, \bullet)\}$

Well-Formedness

**Bondedness/no floating conditions**: (could be relaxed a little if we wanted to)

- For each location $l \in L$, if $l$ is the location of
  - a condition, i.e. $\exists (L, \phi) \in Cond : l \in \{l_1\}$, or
  - a local invariant, i.e. $\exists (l_1, l_1, \phi, l_2, \phi) \in LocInv : l \in \{l_1, l_2\}$, or

then there is a location $l'$ simultaneous to $l$, i.e. $l \sim l'$, which is the location of

- an instance head, i.e. $l'$ is minimal wrt. $\preceq$, or
- a message, i.e.

  $\exists (l_1, E, l_2) \in Msg : l \in \{l_1, l_2\}$.

**Note**: if messages in a chart are cyclic, then there doesn’t exist a partial order (so such charts don’t even have an abstract syntax).
References


