Software Design, Modelling and Analysis in UML

Lecture 18: Live Sequence Charts II

2017-01-24

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

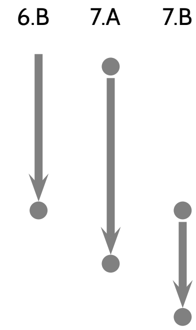
Albert-Ludwigs-Universität Freiburg, Germany

Content

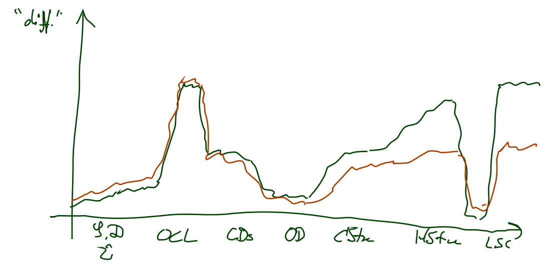
	Sem.	Model
LSC	not so simple	can be concise
HStm	not so simple	may be small
CStm	simple/short	can get large
C++	large	short
ASM	simple	may get large

The Plan

- Thu, 19. 1.: **Live Sequence Charts I**
  Firstly: State-Machines Rest, Code Generation
- Tue, 24. 1.: **Live Sequence Charts II**
- Thu, 26. 1.: **Live Sequence Charts III**
- Tue, 31. 1.: **Tutorial 7**
- Thu, 2. 2.: **Model Based/Driven SW Engineering**
- Mon, 6. 2.: **Inheritance**
- Tue, 7. 2.: **Meta-Modelling** + Questions



February, 17th: **The Exam.**



Constructive Behavioural Modelling in UML: Discussion

Pessimistic view: There are too many...

- **For instance**,
  - allow **absence of initial pseudo-states**
    object may then "be" in enclosing state without being in any substate;
    or assume one of the children states non-deterministically
  - (implicitly) **enforce determinism**, e.g.
    by considering the order in which things have been added to the CASE tool's repository,
    or some graphical order (left to right, top to bottom)
  - allow **true concurrency**
  - etc. etc.

  Exercise: Search the standard for "semantical variation point".

- Crane and Dingel (2007), e.g., provide an in-depth comparison of Statemate, UML, and Rhapsody state machines – the bottom line is:
  - **the intersection is not empty** (i.e. some diagrams mean the same to all three communities)
  - **none is the subset of another** (i.e. each pair of communities has diagrams meaning different things)

Optimistic view:

- tools exist with **complete and consistent** code generation.
- good modelling-guidelines can contribute to **avoiding misunderstandings**.

Reflective Descriptions of Behaviour

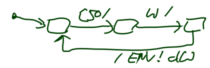
Constructive vs. Reflective Descriptions

Harel (1997) proposes to distinguish constructive and reflective descriptions:

- A constructive description tells us **how** things are computed:

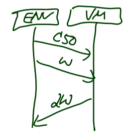
  "A language is ***constructive*** *if it contributes to the dynamic semantics of the model.*

  That is, its constructs contain information needed in executing the model or in translating it into executable code."

- A reflective description tells us **what** shall (or shall not) be computed:

  "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! (Would be too easy.)

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
  - **communication diagram** (formerly known as collaboration diagram),
  - **timing diagram**, or
  - **sequence diagram**.

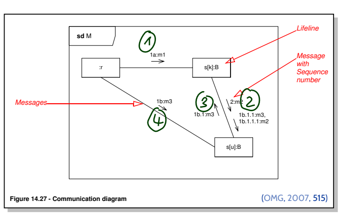


Figure 14.27 - Communication diagram (OMG, 2007, 515)

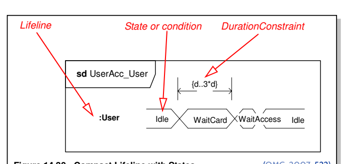


Figure 14.30 - Compact Lifeline with States (OMG, 2007, 522)

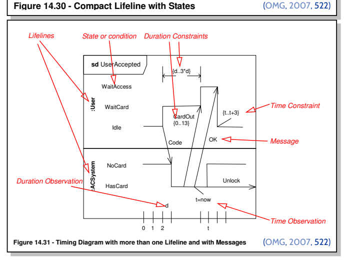


Figure 14.31 - Timing Diagram with more than one Lifeline and with Messages (OMG, 2007, 522)

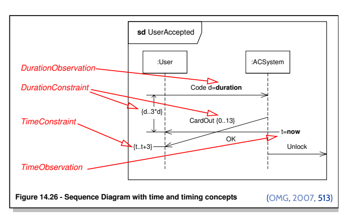
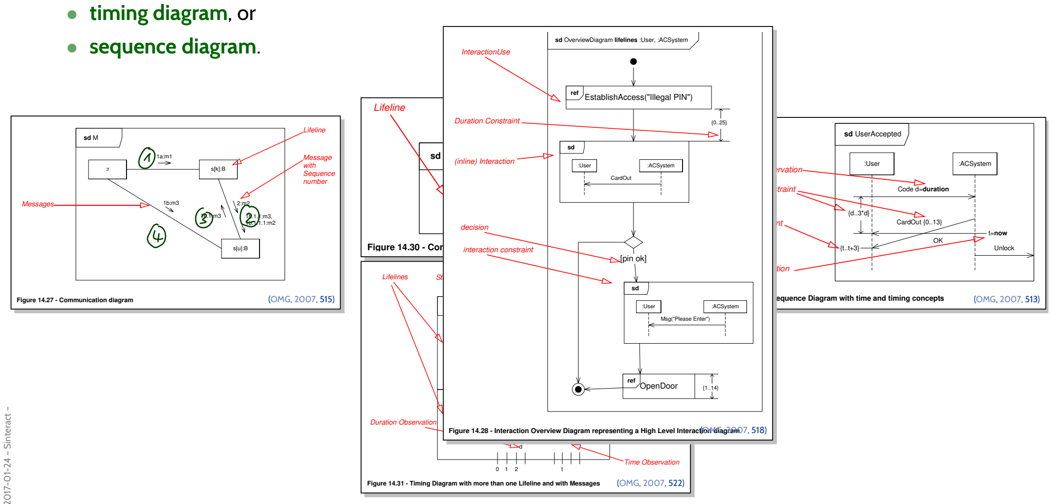


Figure 14.26 - Sequence Diagram with time and timing concepts (OMG, 2007, 513)

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
  - **communication diagram** (formerly known as collaboration diagram),
  - **timing diagram**, or
  - **sequence diagram**.



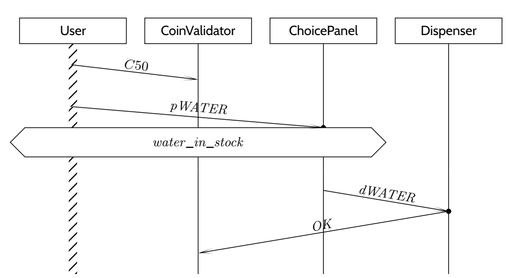
Why Sequence Diagrams?

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

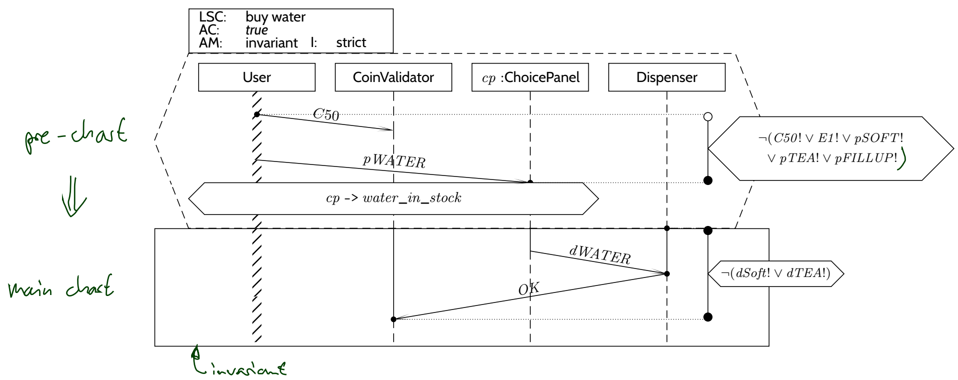
Most severe **drawbacks** of these formalisms:

- 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

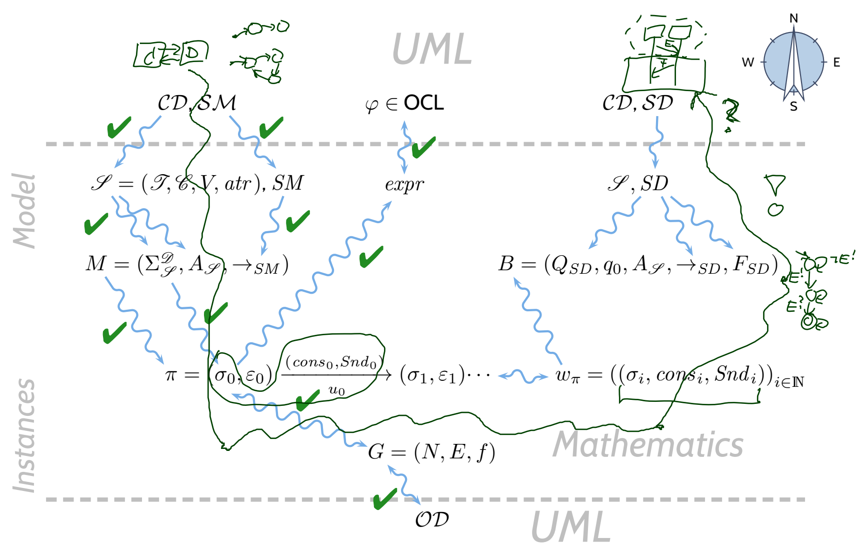


Hence: Live Sequence Charts

- **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.

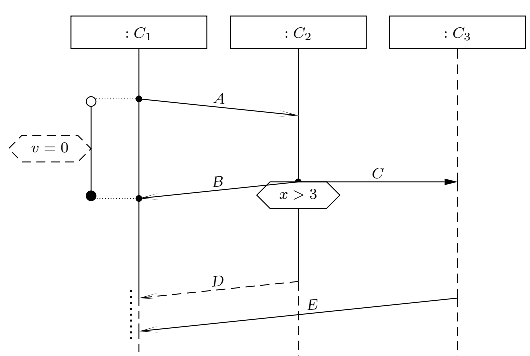


Course Map

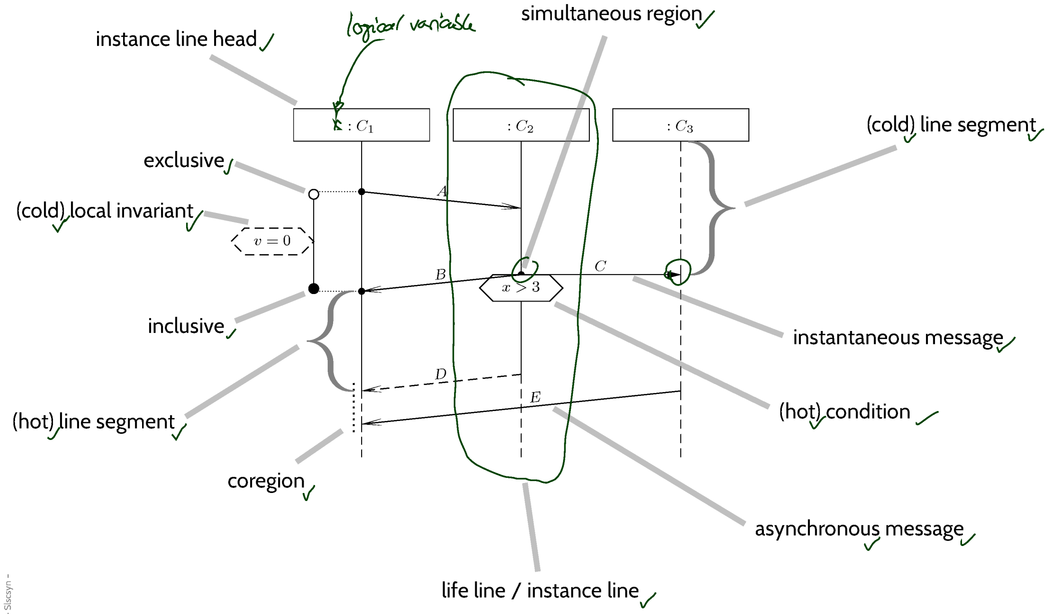


Live Sequence Charts — Syntax

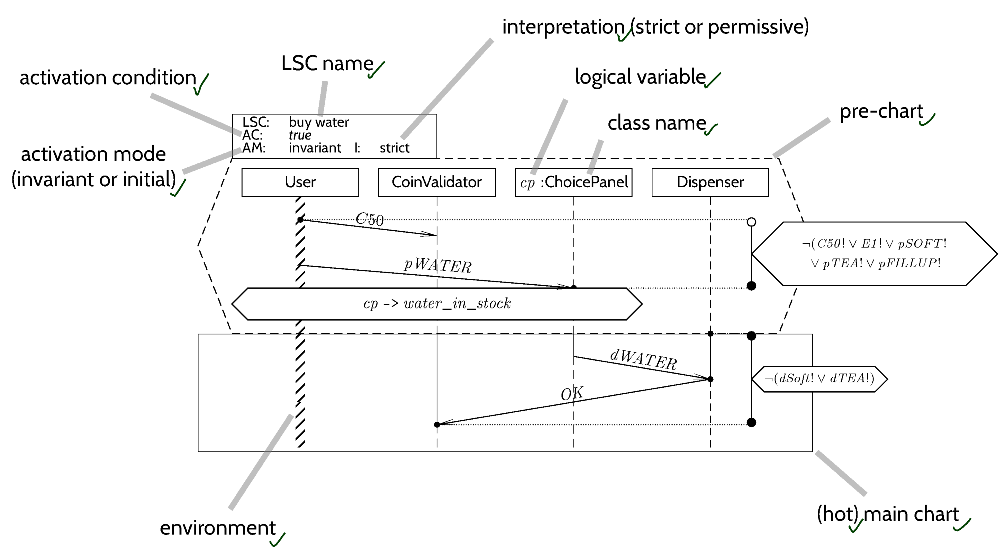
LSC Body Building Blocks



LSC Body Building Blocks



Full LSC Building Blocks for Later

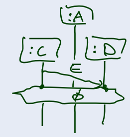


LSC Body: Abstract Syntax

Definition. [*LSC Body*]

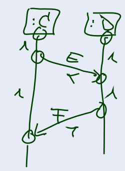
An LSC body over signature $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr, \mathscr{E})$ is a tuple

$$((L, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta)$$



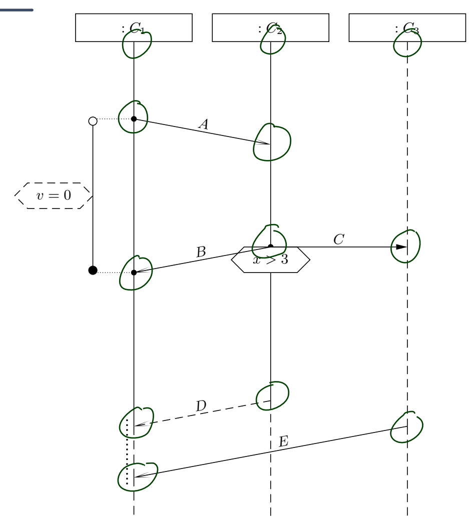
where

- L is a finite, non-empty of **locations** with
  - a **partial order** $\preceq \subseteq L \times L$,
  - a symmetric **simultaneity relation** $\sim \subseteq L \times L$ disjoint with $\preceq$, i.e. $\preceq \cap \sim = \emptyset$,
- $\mathcal{I} = \{I_1, \dots, I_n\}$ is a partitioning of L; elements of $\mathcal{I}$ are called **instance line**,
- $\mathsf{Msg} \subseteq L \times \mathscr{E} \times L$ is a set of **messages** with $(l, E, l') \in \mathsf{Msg}$ only if $(l, l') \in \prec \cup \sim$;
  message (l, E, l') is called **instantaneous** iff $l \sim l'$ and **asynchronous** otherwise,
- $\mathsf{Cond} \subseteq (2^L \setminus \emptyset) \times Expr_{\mathscr{S}}$ is a set of **conditions**
  with $(L, \phi) \in \mathsf{Cond}$ only if $l \sim l'$ for all $l \neq l' \in L$,
- $\mathsf{LocInv} \subseteq L \times \{\circ, \bullet\} \times Expr_{\mathscr{S}} \times L \times \{\circ, \bullet\}$ is a set of **local invariants**
  with $(l, \iota, \phi, l', \iota') \in \mathsf{LocInv}$ only if $l \prec l'$, $\circ$: exclusive, $\bullet$: inclusive,
- $\Theta : L \cup \mathsf{Msg} \cup \mathsf{Cond} \cup \mathsf{LocInv} \to \{\mathsf{hot}, \mathsf{cold}\}$
  assigns to each location and each element a **temperature**.



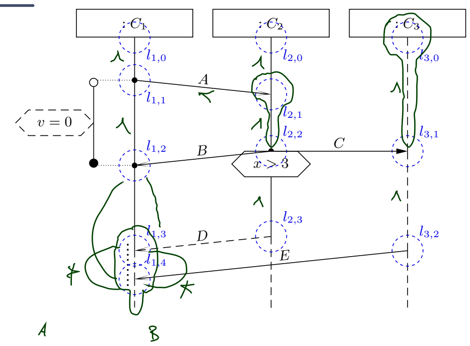
From Concrete to Abstract Syntax

- locations L,
- $\preceq \subseteq L \times L$, $\quad \sim \subseteq L \times L$
- $\mathcal{I} = \{I_1, \dots, I_n\}$,
- $\mathsf{Msg} \subseteq L \times \mathcal{E} \times L$,
- $\mathsf{Cond} \subseteq (2^L \setminus \emptyset) \times Expr_{\mathscr{S}}$
- $\mathsf{LocInv} \subseteq L \times \{\circ, \bullet\} \times Expr_{\mathscr{S}} \times L \times \{\circ, \bullet\}$,
- $\Theta : L \cup \mathsf{Msg} \cup \mathsf{Cond} \cup \mathsf{LocInv} \to \{\mathsf{hot}, \mathsf{cold}\}$.



From Concrete to Abstract Syntax

- locations L,
- $\preceq \subseteq L \times L, \quad \sim \subseteq L \times L$
- $\mathcal{I} = \{I_1, \dots, I_n\}$,
- Msg $\subseteq L \times \mathcal{E} \times L$,
- Cond $\subseteq (2^L \setminus \emptyset) \times Expr_{\mathscr{S}}$
- LocInv $\subseteq L \times \{\circ, \bullet\} \times Expr_{\mathscr{S}} \times L \times \{\circ, \bullet\}$,
- $\Theta : L \cup$ Msg $\cup$ Cond $\cup$ LocInv $\to \{$hot, cold$\}$.



$L = \{l_{1,0}, \dots, l_{1,4}, l_{2,0}, \dots, l_{2,3}, l_{3,0}, \dots, l_{3,2}\}$

$\prec = \{(l_{1,0}, l_{1,1}), (l_{1,1}, l_{1,2}), (l_{1,2}, l_{1,3}), (l_{1,2}, l_{1,4}), \dots, (l_{1,1}, l_{2,1}), (l_{2,2}, l_{1,2}), \dots\}$

$\preceq = \prec^*$

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

$Msg = \{(l_{1,1}, A, l_{2,1}), \dots\}$

$Cond = \{(\{l_{2,2}\}, x > 3)\}$

$LocInv = \{(l_{1,1}, \circ, v = 0, l_{1,2}, \bullet)\}$

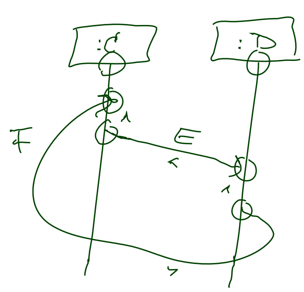
$l_{1,3} \mapsto cold, \; l_{1,4} \mapsto cold$

$\Theta = \{M \mapsto hot, C \mapsto hot, I \mapsto cold, l_{3,0} \mapsto cold, l_{2,1} \mapsto hot, \dots\}$

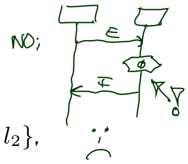
From Concrete to Abstract Syntax

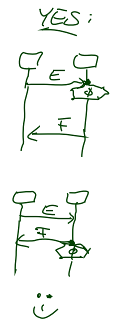
- locations L,
- $\preceq \subseteq L \times L, \quad \sim \subseteq L \times L$
- $\mathcal{I} = \{I_1, \dots, I_n\}$,
- Msg $\subseteq L \times \mathcal{E} \times L$,
- Cond $\subseteq (2^L \setminus \emptyset) \times Expr_{\mathscr{S}}$
- LocInv $\subseteq L \times \{\circ, \bullet\} \times Expr_{\mathscr{S}} \times L \times \{\circ, \bullet\}$,
- $\Theta : L \cup$ Msg $\cup$ Cond $\cup$ LocInv $\to \{$hot, cold$\}$.

NO:



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 : $l \in L$, or
  - a **local invariant**, i.e. $\exists\,(l_1,\iota_1,\phi,l_2,\iota_2) \in$ LocInv : $l \in \{l_1,l_2\}$,

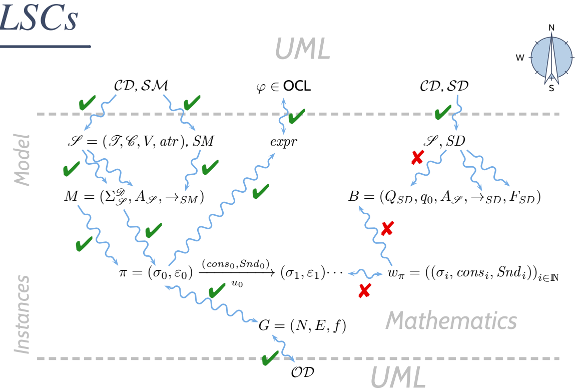
  then there is a location l' **simultaneous** to l, i.e. $l \sim l'$, which is the location of
  - an **instance head**, i.e. l' is minimal wrt. $\preceq$, or
  - a **message**, i.e.

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

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

Live Sequence Charts — Semantics

TBA-based Semantics of LSCs



Plan:

(i) Given an LSC $\mathscr{L}$ with body

$$((L, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta),$$

(ii) construct a TBA $\mathcal{B}_{\mathscr{L}}$, and

(iii) define language $\mathcal{L}(\mathscr{L})$ of $\mathscr{L}$ **in terms of** $\mathcal{L}(\mathcal{B}_{\mathscr{L}})$,
in particular taking activation condition and activation mode into account.

(iv) define language $\mathcal{L}(\mathcal{M})$ of a UML model.

- Then $\mathcal{M} \models \mathscr{L}$ (**universal**) if and only if $\mathcal{L}(\mathcal{M}) \subseteq \mathcal{L}(\mathscr{L})$.
  And $\mathcal{M} \models \mathscr{L}$ (**existential**) if and only if $\mathcal{L}(\mathcal{M}) \cap \mathcal{L}(\mathscr{L}) \neq \emptyset$.

Live Sequence Charts — TBA Construction

Formal LSC Semantics: It's in the Cuts!

Definition.
Let $((L, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta)$ be an LSC body.
A non-empty set $\emptyset \neq C \subseteq L$ is called a **cut** of the LSC body iff

- it is **downward closed**, i.e. $\forall\, l, l' \bullet l' \in C \land l \preceq l' \implies l \in C$,
- it is **closed** under **simultaneity**, i.e.

$$\forall\, l, l' \bullet l' \in C \land l \sim l' \implies l \in C, \text{ and}$$

- it comprises at least **one location per instance line**, i.e.

$$\forall\, i \in I\, \exists\, l \in C \bullet i_l = i.$$

Formal LSC Semantics: It's in the Cuts!

Definition.
Let $((L, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta)$ be an LSC body.
A non-empty set $\emptyset \neq C \subseteq L$ is called a **cut** of the LSC body iff

- it is **downward closed**, i.e. $\forall\, l, l' \bullet l' \in C \land l \preceq l' \implies l \in C$,
- it is **closed** under **simultaneity**, i.e.

$$\forall\, l, l' \bullet l' \in C \land l \sim l' \implies l \in C, \text{ and}$$

- it comprises at least **one location per instance line**, i.e.

$$\forall\, i \in I\, \exists\, l \in C \bullet i_l = i.$$

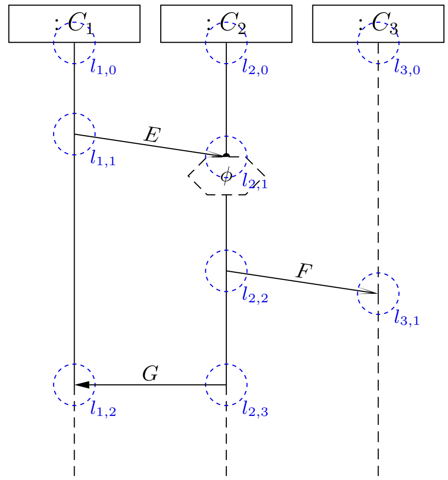
The **temperature function** is extended to cuts as follows:

$$\Theta(C) = \begin{cases} \mathsf{hot} & \text{, if } \exists\, l \in C \bullet (\nexists\, l' \in C \bullet l \prec l') \land \Theta(l) = \mathsf{hot} \\ \mathsf{cold} & \text{, otherwise} \end{cases}$$

that is, C is **hot** if and only if at least one of its maximal elements is hot.

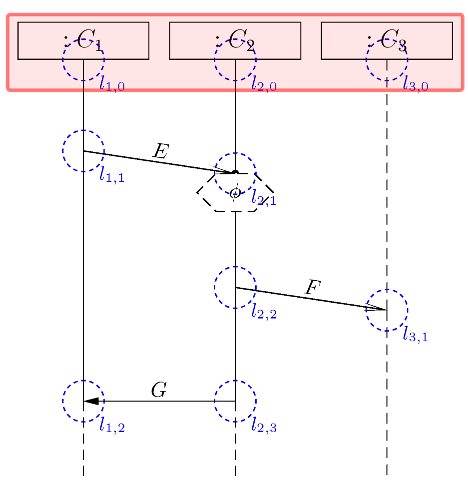
Cut Examples

$\emptyset \neq C \subseteq L$ – downward closed – simultaneity closed – at least one loc. per instance line



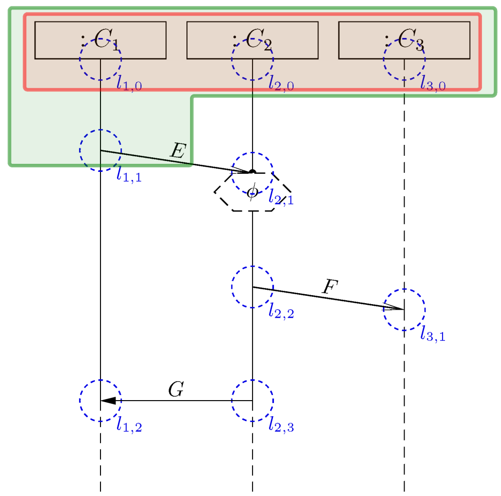
Cut Examples

$\emptyset \neq C \subseteq L$ – downward closed – simultaneity closed – at least one loc. per instance line



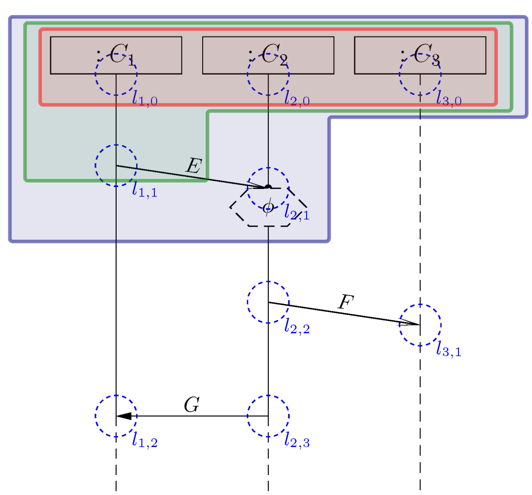
Cut Examples

$\emptyset \neq C \subseteq L$ – downward closed – simultaneity closed – at least one loc. per instance line



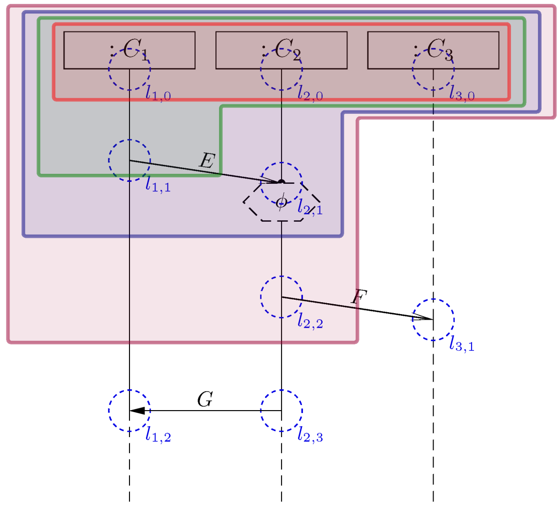
Cut Examples

$\emptyset \neq C \subseteq L$ – downward closed – simultaneity closed – at least one loc. per instance line



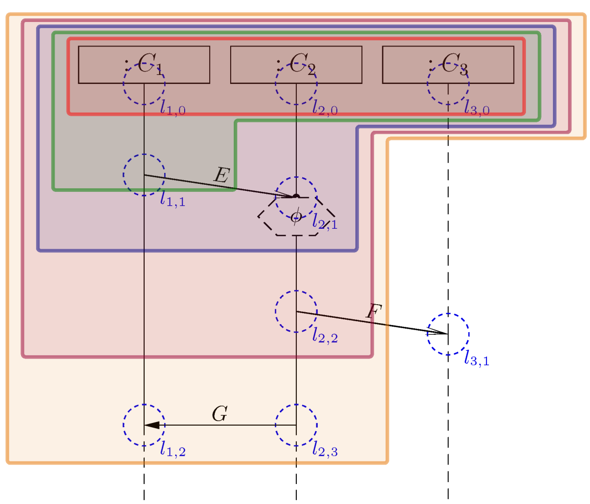
Cut Examples

$\emptyset \neq C \subseteq L$ – downward closed – simultaneity closed – at least one loc. per instance line



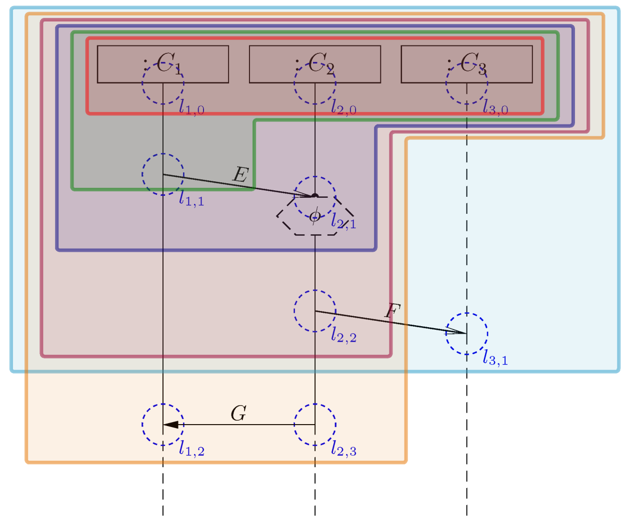
Cut Examples

$\emptyset \neq C \subseteq L$ – downward closed – simultaneity closed – at least one loc. per instance line



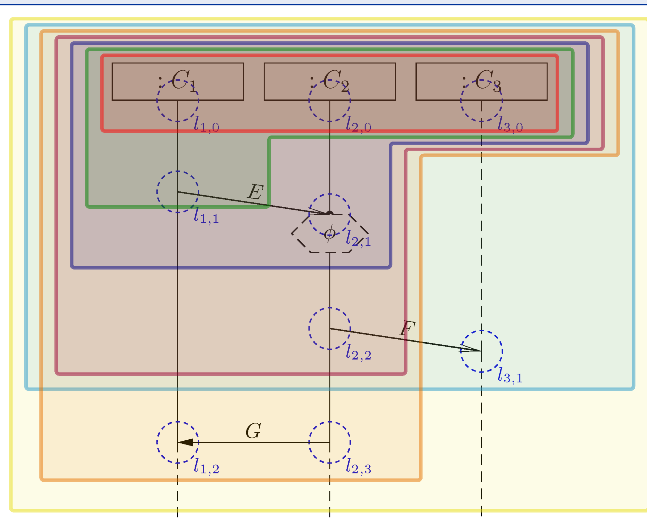
Cut Examples

$\emptyset \neq C \subseteq L$ – downward closed – simultaneity closed – at least one loc. per instance line



Cut Examples

$\emptyset \neq C \subseteq L$ – downward closed – simultaneity closed – at least one loc. per instance line



A Successor Relation on Cuts

The partial order "$\preceq$" and the simultaneity relation "$\sim$" of locations induce a **direct successor relation** on cuts of an LSC body as follows:

Definition.
Let $C \subseteq L$ bet a cut of LSC body $((L, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta)$.

A set $\emptyset \neq F \subseteq L$ of locations is called fired-set F of cut C if and only if

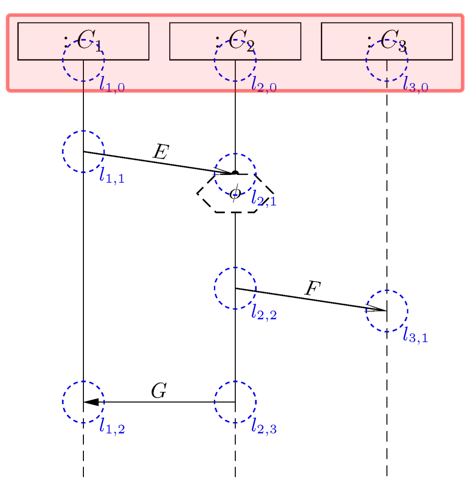
- $C \cap F = \emptyset$ and $C \cup F$ is a cut, i.e. F is closed under simultaneity,
- all locations in F are direct $\prec$-successors of the front of C, i.e.
$$\forall\, l \in F\, \exists\, l' \in C \bullet l' \prec l \wedge (\nexists\, l'' \in C \bullet l' \prec l'' \prec l),$$
- locations in F, that lie on the same instance line, are pairwise unordered, i.e.
$$\forall\, l \neq l' \in F \bullet (\exists\, I \in \mathcal{I} \bullet \{l, l'\} \subseteq I) \implies l \npreceq l' \wedge l' \npreceq l,$$
- for each asynchronous (!) message reception in F, the corresponding sending is already in C,
$$\forall\, (l, E, l') \in \mathsf{Msg} \bullet l' \in F \implies l \in C.$$

The cut $C' = C \cup F$ is called direct successor of C via F, denoted by $C \rightsquigarrow_F C'$.

- 18 - 2017-01-24 - Slsccutfire -

Successor Cut Example

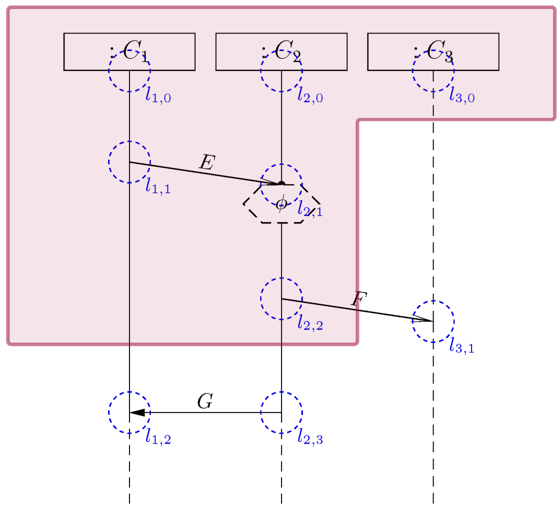
$C \cap F = \emptyset$ – $C \cup F$ is a cut – only direct $\prec$-successors – same instance line on front pairwise unordered – sending of asynchronous reception already in



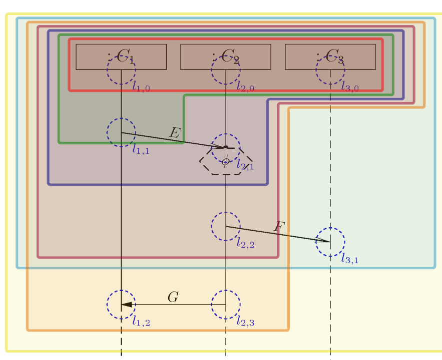
- 18 - 2017-01-24 - Slsccutfire -

Successor Cut Example

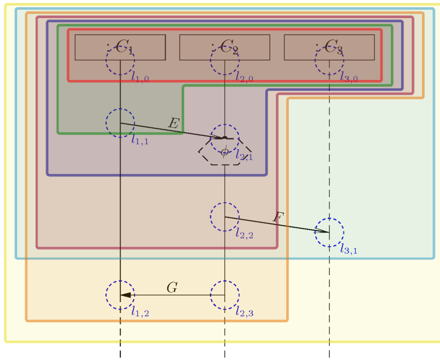
$C \cap F = \emptyset$ – $C \cup F$ is a cut – only direct $\prec$-successors – same instance line on front pairwise unordered – sending of asynchronous reception already in

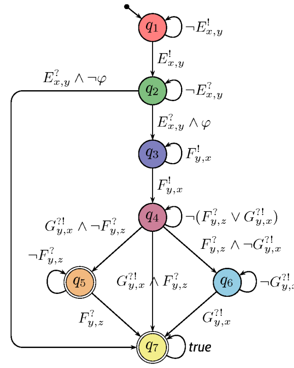


Language of LSC Body: Example

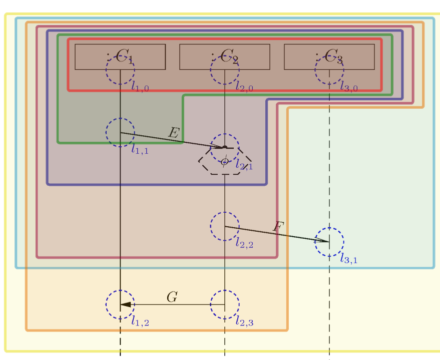


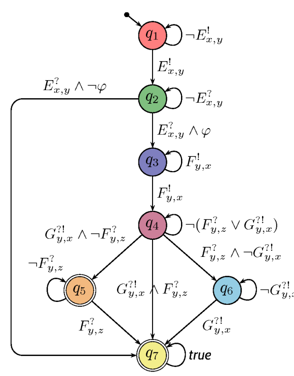
Language of LSC Body: Example





Language of LSC Body: Example





The TBA $\mathcal{B}_{\mathscr{L}}$ of LSC $\mathscr{L}$ over Φ and $\mathcal{E}$ is $(Expr_{\mathcal{B}}(X), X, Q, q_{ini}, \rightarrow, Q_F)$ with

- Q is **the set of cuts** of $\mathscr{L}$, q_{ini} is the **instance heads** cut,
- $Expr_{\mathcal{B}}(X) = Expr_{\mathscr{S}}(\mathscr{E}, X)$ (for considered signature $\mathscr{S}$),
- $\rightarrow$ consists of **loops**, **progress transitions** (by $\rightsquigarrow_F$), and **legal exits** (cold cond./local inv.),
- $Q_F = \{C \in Q \mid \Theta(C) = \text{cold} \vee C = L\}$ is the set of cold cuts and the maximal cut.

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:

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

  where $expr : Bool \in Expr_{\mathscr{S}}$, $E \in \mathscr{E}$, $x, y \in X$ (or keyword env).

- We use

$$\mathscr{E}_{!?}(X) := \{E^{!}_{x,y}, E^{?}_{x,y} \mid E \in \mathscr{E}, x, y \in X\}$$

  to denote the set of **event expressions** over $\mathscr{E}$ and X.

TBA Construction Principle

Recall: The TBA $\mathcal{B}(\mathscr{L})$ of LSC $\mathscr{L}$ is $(Expr_{\mathcal{B}}(X), X, Q, q_{ini}, \rightarrow, Q_F)$ with

- Q is **the set of cuts** of $\mathscr{L}$, q_{ini} is the **instance heads** cut,
- $Expr_{\mathcal{B}} = \Phi \,\dot\cup\, \mathscr{E}_{!?}(X)$,
- $\rightarrow$ consists of **loops**, **progress transitions** (from $\rightsquigarrow_F$), and **legal exits** (cold cond./local inv.),
- $F = \{C \in Q \mid \Theta(C) = \text{cold} \vee C = L\}$ is the set of cold cuts.

TBA Construction Principle

Recall: The TBA $\mathcal{B}(\mathscr{L})$ of LSC $\mathscr{L}$ is $(Expr_{\mathcal{B}}(X), X, Q, q_{ini}, \to, Q_F)$ with

- Q is **the set of cuts** of $\mathscr{L}$, q_{ini} is the **instance heads** cut,
- $Expr_{\mathcal{B}} = \Phi \mathbin{\dot\cup} \mathscr{E}_{!?}(X)$,
- $\to$ consists of **loops**, **progress transitions** (from $\rightsquigarrow_F$), and **legal exits** (cold cond./local inv.),
- $F = \{C \in Q \mid \Theta(C) = \text{cold} \vee C = L\}$ is the set of cold cuts.

So in the following, we "only" need to construct the transitions' labels:

$$\to = \{(q, \qquad, q) \mid q \in Q\} \cup \{(q, \qquad\qquad, q') \mid q \rightsquigarrow_F q'\} \cup \{(q, \qquad, L) \mid q \in Q\}$$

TBA Construction Principle

Recall: The TBA $\mathcal{B}(\mathscr{L})$ of LSC $\mathscr{L}$ is $(Expr_{\mathcal{B}}(X), X, Q, q_{ini}, \to, Q_F)$ with

- Q is **the set of cuts** of $\mathscr{L}$, q_{ini} is the **instance heads** cut,
- $Expr_{\mathcal{B}} = \Phi \mathbin{\dot\cup} \mathscr{E}_{!?}(X)$,
- $\to$ consists of **loops**, **progress transitions** (from $\rightsquigarrow_F$), and **legal exits** (cold cond./local inv.),
- $F = \{C \in Q \mid \Theta(C) = \text{cold} \vee C = L\}$ is the set of cold cuts.

So in the following, we "only" need to construct the transitions' labels:

$$\to = \{(q, \psi_{loop}(q), q) \mid q \in Q\} \cup \{(q, \psi_{prog}(q, q'), q') \mid q \rightsquigarrow_F q'\} \cup \{(q, \psi_{exit}(q), L) \mid q \in Q\}$$

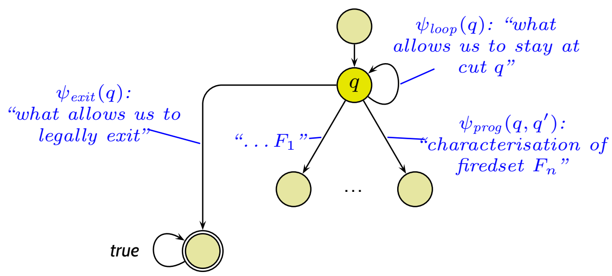
TBA Construction Principle

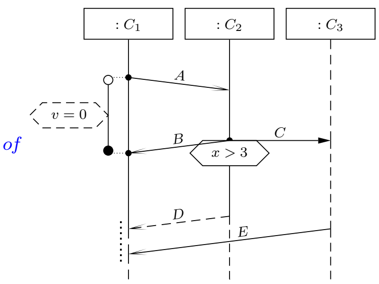
Recall: The TBA $\mathcal{B}(\mathscr{L})$ of LSC $\mathscr{L}$ is $(Expr_{\mathcal{B}}(X), X, Q, q_{ini}, \rightarrow, Q_F)$ with

- Q is **the set of cuts** of $\mathscr{L}$, q_{ini} is the **instance heads** cut,
- $Expr_{\mathcal{B}} = \Phi \mathbin{\dot\cup} \mathscr{E}_{!?}(X)$,
- $\rightarrow$ consists of **loops**, **progress transitions** (from $\rightsquigarrow_F$), and **legal exits** (cold cond./local inv.),
- $F = \{C \in Q \mid \Theta(C) = \text{cold} \vee C = L\}$ is the set of cold cuts.

So in the following, we "only" need to construct the transitions' labels:

$$\rightarrow = \{(q, \psi_{loop}(q), q) \mid q \in Q\} \cup \{(q, \psi_{prog}(q, q'), q') \mid q \rightsquigarrow_F q'\} \cup \{(q, \psi_{exit}(q), L) \mid q \in Q\}$$





TBA Construction Principle

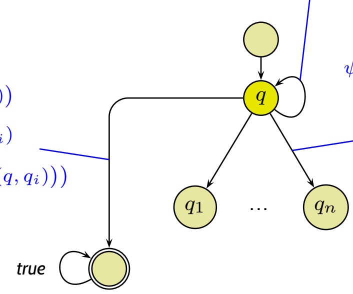
"Only" construct the transitions' labels:

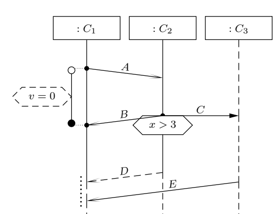
$$\rightarrow = \{(q, \psi_{loop}(q), q) \mid q \in Q\} \cup \{(q, \psi_{prog}(q, q'), q') \mid q \rightsquigarrow_F q'\} \cup \{(q, \psi_{exit}(q), L) \mid q \in Q\}$$

$$\psi_{loop}(q) = \overbrace{\psi^{\mathsf{Msg}}(q) \wedge \psi^{\mathsf{LocInv}}_{\mathsf{hot}}(q)}^{=:\psi^{\mathsf{hot}}_{loop}(q)} \wedge \psi^{\mathsf{LocInv}}_{\mathsf{cold}}(q)$$

$$\psi_{exit}(q) = (\psi^{\mathsf{hot}}_{loop}(q) \wedge \neg\psi^{\mathsf{LocInv}}_{\mathsf{cold}}(q)) \vee \bigvee_{1 \le i \le n} (\psi^{\mathsf{hot}}_{prog}(q, q_i) \wedge (\neg\psi^{\mathsf{LocInv},\bullet}_{\mathsf{cold}}(q, q_i) \vee \neg\psi^{\mathsf{Cond}}_{\mathsf{cold}}(q, q_i)))$$

$$\psi_{prog}(q, q_n) = \overbrace{\psi^{\mathsf{Msg}}(q, q_n) \wedge \psi^{\mathsf{Cond}}_{\mathsf{hot}}(q, q_n) \wedge \psi^{\mathsf{LocInv},\bullet}_{\mathsf{hot}}(q, q_n)}^{=:\psi^{\mathsf{hot}}_{prog}(q, q_n)} \wedge \psi^{\mathsf{Cond}}_{\mathsf{cold}}(q, q_n) \wedge \psi^{\mathsf{LocInv},\bullet}_{\mathsf{cold}}(q, q_n)$$





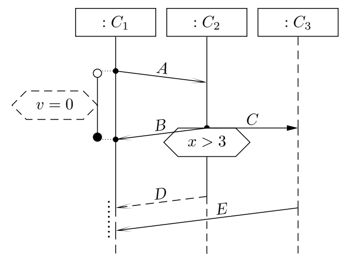
Loop Condition

$$\psi_{loop}(q) = \psi^{\mathsf{Msg}}(q) \wedge \psi^{\mathsf{LocInv}}_{\mathsf{hot}}(q) \wedge \psi^{\mathsf{LocInv}}_{\mathsf{cold}}(q)$$

- $\psi^{\mathsf{Msg}}(q) = \neg \bigvee_{1 \leq i \leq n} \psi^{\mathsf{Msg}}(q, q_i) \wedge \underbrace{\left(strict \implies \bigwedge_{\psi \in \mathsf{Msg}(L)} \neg\psi\right)}_{=:\psi_{\mathsf{strict}}(q)}$
- $\psi^{\mathsf{LocInv}}_{\theta}(q) = \bigwedge_{\ell=(l,\iota,\phi,l',\iota') \in \mathsf{LocInv},\ \Theta(\ell)=\theta,\ \ell \text{ active at } q} \phi$

  A location l is called **front location** of cut C if and only if $\nexists\, l' \in L \bullet l \prec l'$.

  Local invariant $(l_o, \iota_0, \phi, l_1, \iota_1)$ is **active** at cut (!) q
  if and only if $l_0 \preceq l \prec l_1$ for some front location l of cut q or $l_1 \in q \wedge \iota_1 = \bullet$.
- $\mathsf{Msg}(F) = \{E^{!}_{x_l, x_{l'}} \mid (l, E, l') \in \mathsf{Msg},\ l \in F\} \cup \{E^{?}_{x_l, x_{l'}} \mid (l, E, l') \in \mathsf{Msg},\ l' \in F\}$
- $x_l \in X$ is the logical variable associated with the instance line I which includes l, i.e. $l \in I$.
- $\mathsf{Msg}(F_1, \dots, F_n) = \bigcup_{1 \leq i \leq n} \mathsf{Msg}(F_i)$



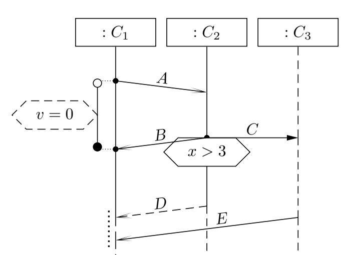
Progress Condition

$$\psi^{\mathsf{hot}}_{prog}(q, q_i) = \psi^{\mathsf{Msg}}(q, q_n) \wedge \psi^{\mathsf{Cond}}_{\mathsf{hot}}(q, q_n) \wedge \psi^{\mathsf{LocInv},\bullet}_{\mathsf{hot}}(q_n)$$

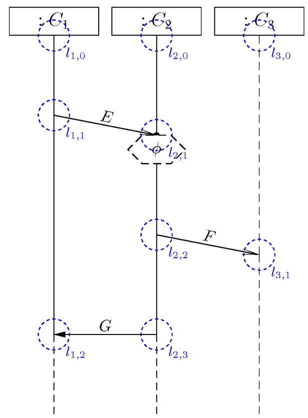
- $\psi^{\mathsf{Msg}}(q, q_i) = \bigwedge_{\psi \in \mathsf{Msg}(q_i \setminus q)} \psi \wedge \bigwedge_{j \neq i} \bigwedge_{\psi \in \mathsf{Msg}(q_j \setminus q) \setminus \mathsf{Msg}(q_i \setminus q)} \neg\psi$
  $\wedge \underbrace{\left(strict \implies \bigwedge_{\psi \in \mathsf{Msg}(L) \setminus \mathsf{Msg}(F_i)} \neg\psi\right)}_{=:\psi_{\mathsf{strict}}(q, q_i)}$
- $\psi^{\mathsf{Cond}}_{\theta}(q, q_i) = \bigwedge_{\gamma=(L,\phi) \in \mathsf{Cond},\ \Theta(\gamma)=\theta,\ L \cap (q_i \setminus q) \neq \emptyset} \phi$
- $\psi^{\mathsf{LocInv},\bullet}_{\theta}(q, q_i) = \bigwedge_{\lambda=(l,\iota,\phi,l',\iota') \in \mathsf{LocInv},\ \Theta(\lambda)=\theta,\ \lambda\ \bullet\text{-active at } q_i} \phi$

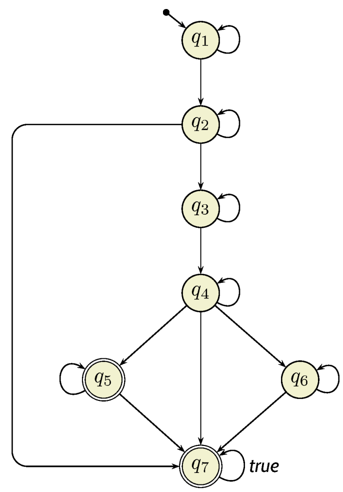
  Local invariant $(l_0, \iota_0, \phi, l_1, \iota_1)$ is **•-active** at q if and only if
  - $l_0 \prec l \prec l_1$, or
  - $l = l_0 \wedge \iota_0 = \bullet$, or
  - $l = l_1 \wedge \iota_1 = \bullet$

  for some front location l of cut (!) q.



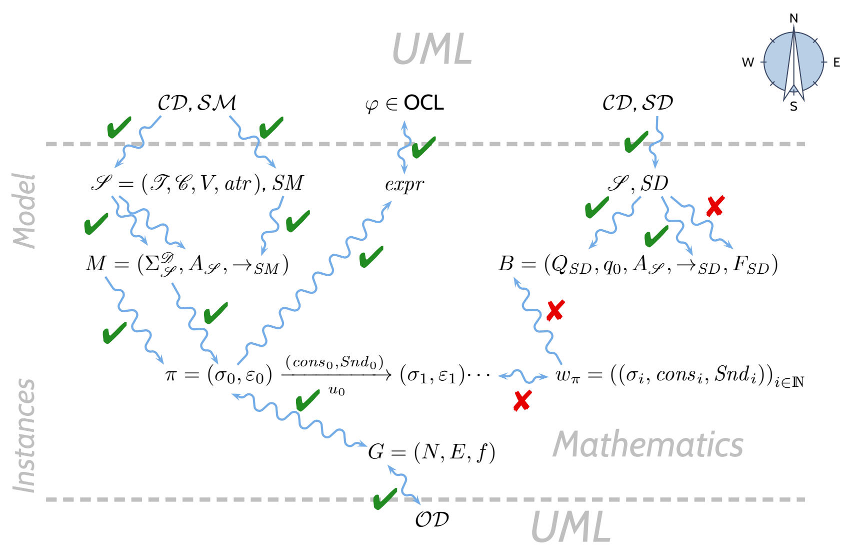
Example





Using logical variables x, y, z
for the instances lines
(from left to right).

Course Map

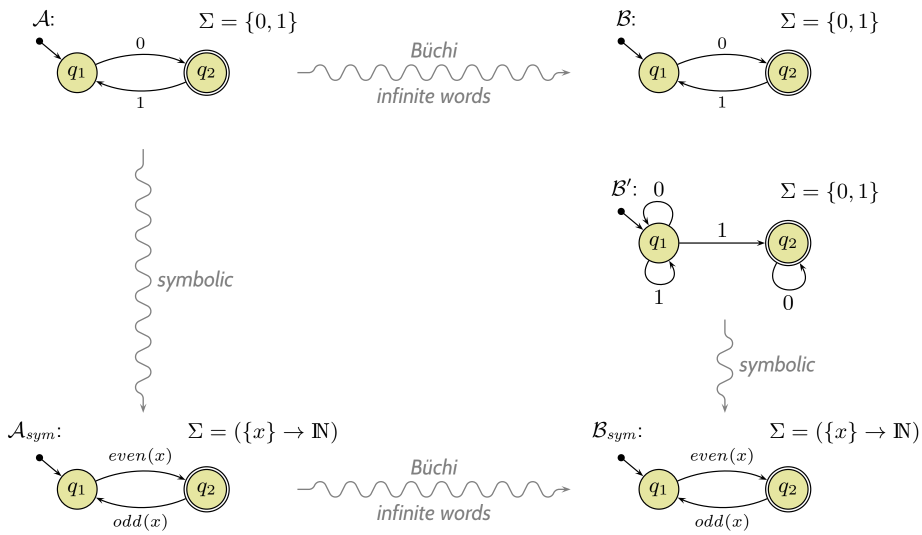


Tell Them What You've Told Them...

- Interactions can be **reflective** descriptions of behaviour, i.e.
  - describe **what** behaviour is (un)desired, without (yet) defining **how** to realise it.
- One visual formalism for interactions: **Live Sequence Charts**
  - locations in diagram **induce a partial order**,
  - instantaneous and aynchronous messages,
  - conditions and local invariants
- The **meaning** of an LSC is defined using TBAs.
  - **Cuts** become states of the automaton.
  - Locations induce a **partial order on cuts**.
  - Automaton-transitions and annotations correspond to a **successor relation** on cuts.
  - Annotations use **signal / attribute expressions**.
- **Later**:
  - TBA have **Büchi acceptance** (of infinite words (of a model)).
  - **Full LSC semantics**.
  - **Pre-Charts**.

Excursion: Büchi Automata

From Finite Automata to Symbolic Büchi Automata



Symbolic Büchi Automata

Definition. A **Symbolic Büchi Automaton** (TBA) is a tuple

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

where

- X is a set of logical variables,
- $Expr_{\mathcal{B}}(X)$ is a set of Boolean expressions over X,
- Q is a finite set of **states**,
- $q_{ini} \in Q$ is the initial state,
- $\to \subseteq Q \times Expr_{\mathcal{B}}(X) \times Q$ is the **transition relation**. Transitions (q, ψ, q') from q to q' are labelled with an expression $\psi \in Expr_{\mathcal{B}}(X)$.
- $Q_F \subseteq Q$ is the set of **fair** (or accepting) states.

Word

Definition. Let X be a set of logical variables and let $Expr_{\mathcal{B}}(X)$ be a set of Boolean expressions over X.

A set $(\Sigma, \cdot \models . \cdot)$ is called an **alphabet** for $Expr_{\mathcal{B}}(X)$ if and only if

- for each $\sigma \in \Sigma$,
  - for each expression $expr \in Expr_{\mathcal{B}}$, and
    - for each valuation $\beta : X \to \mathscr{D}(X)$ of logical variables,

$$\textbf{either} \quad \sigma \models_{\beta} expr \quad \textbf{or} \quad \sigma \not\models_{\beta} expr.$$

(σ **satisfies** (or does not satisfy) $expr$ under valuation β)

An **infinite sequence**

$$w = (\sigma_i)_{i \in \mathbb{N}_0} \in \Sigma^{\omega}$$

over $(\Sigma, \cdot \models . \cdot)$ is called **word** (for $Expr_{\mathcal{B}}(X)$).

– 18 – 2017-01-24 – Stba –

Run of TBA over Word

Definition. Let $\mathcal{B} = (Expr_{\mathcal{B}}(X), X, Q, q_{ini}, \to, Q_F)$ be a TBA and

$$w = \sigma_1, \sigma_2, \sigma_3, \dots$$

a word for $Expr_{\mathcal{B}}(X)$. An infinite sequence

$$\varrho = q_0, q_1, q_2, \dots \in Q^{\omega}$$

is called **run of $\mathcal{B}$ over** w under valuation $\beta : X \to \mathscr{D}(X)$ if and only if

- $q_0 = q_{ini}$,
- for each $i \in \mathbb{N}_0$ there is a transition

$$(q_i, \psi_i, q_{i+1}) \in \to$$

  such that $\sigma_i \models_{\beta} \psi_i$.

– 18 – 2017-01-24 – Stba –

Run of TBA over Word

Definition. Let $\mathcal{B} = (Expr_{\mathcal{B}}(X), X, Q, q_{ini}, \rightarrow, Q_F)$ be a TBA and

$$w = \sigma_1, \sigma_2, \sigma_3, \dots$$

a word for $Expr_{\mathcal{B}}(X)$. An infinite sequence

$$\varrho = q_0, q_1, q_2, \ldots \in Q^\omega$$

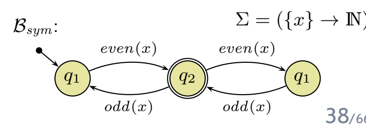
is called **run of $\mathcal{B}$ over** w under valuation $\beta : X \rightarrow \mathscr{D}(X)$ if and only if

- $q_0 = q_{ini}$,
- for each $i \in \mathbb{N}_0$ there is a transition

$$(q_i, \psi_i, q_{i+1}) \in \rightarrow$$

  such that $\sigma_i \models_\beta \psi_i$.

Example:



The Language of a TBA

Definition.
We say TBA $\mathcal{B} = (Expr_{\mathcal{B}}(X), X, Q, q_{ini}, \rightarrow, Q_F)$ **accepts** the word

$$w = (\sigma_i)_{i \in \mathbb{N}_0} \in (Expr_{\mathcal{B}} \rightarrow \mathbb{B})^\omega$$

if and only if $\mathcal{B}$ **has** a run

$$\varrho = (q_i)_{i \in \mathbb{N}_0}$$

over w such that fair (or accepting) states are **visited infinitely often** by ϱ, i.e., such that

$$\forall\, i \in \mathbb{N}_0\; \exists\, j > i : q_j \in Q_F.$$

We call the set $\mathcal{L}(\mathcal{B}) \subseteq (Expr_{\mathcal{B}} \rightarrow \mathbb{B})^\omega$ of words that are accepted by $\mathcal{B}$ the **language of** $\mathcal{B}$.

The Language of a TBA

Definition.
We say TBA $\mathcal{B} = (Expr_{\mathcal{B}}(X), X, Q, q_{ini}, \rightarrow, Q_F)$ **accepts** the word

$$w = (\sigma_i)_{i \in \mathbb{N}_0} \in (Expr_{\mathcal{B}} \rightarrow \mathbb{B})^\omega$$

if and only if $\mathcal{B}$ **has** a run

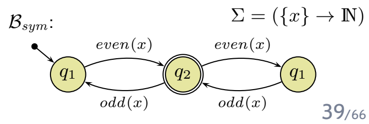
$$\varrho = (q_i)_{i \in \mathbb{N}_0}$$

over w such that fair (or accepting) states are **visited infinitely often** by ϱ,
i.e., such that

$$\forall\, i \in \mathbb{N}_0\; \exists\, j > i : q_j \in Q_F.$$

We call the set $\mathcal{L}(\mathcal{B}) \subseteq (Expr_{\mathcal{B}} \rightarrow \mathbb{B})^\omega$ of words that are accepted by $\mathcal{B}$ the **language of** $\mathcal{B}$.

Example:



Language of UML Model

The Language of a Model

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

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

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

The Language of a Model

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

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

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

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

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

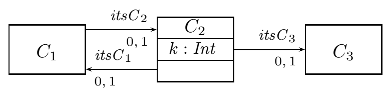
$$\mathcal{L}(\mathcal{M}) := \{(\sigma_i, u_i, cons_i, Snd_i)_{i \in \mathbb{N}_0} \in (\Sigma_{\mathscr{S}}^{\mathscr{D}} \times \tilde{A})^\omega \mid$$
$$\exists\, (\varepsilon_i)_{i \in \mathbb{N}_0} : (\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \dots \in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

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

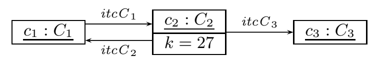
Example: Language of a Model

$$\mathcal{L}(\mathcal{M}) := \{(\sigma_i, u_i, cons_i, Snd_i)_{i \in \mathbb{N}_0} \mid \exists (\varepsilon_i)_{i \in \mathbb{N}_0} : (\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \cdots \in [\![\mathcal{M}]\!]\}$$

$\mathcal{CD}$:



σ_0:



$$(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} \cdots \rightarrow (\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \xrightarrow[c_1]{(cons_1, \{(:E, c_2)\})} (\sigma_2, \varepsilon_2) \xrightarrow[c_2]{(\{:E\}, Snd_2)}$$

$$(\sigma_3, \varepsilon_3) \xrightarrow[c_2]{(cons_3, \{(:F, c_3)\})} (\sigma_4, \varepsilon_4) \xrightarrow[c_2]{(cons_4, \{(G(), c_1)\})} (\sigma_5, \varepsilon_5) \xrightarrow[c_3]{(\{:F\}, Snd_5)} (\sigma_6, \varepsilon_6) \rightarrow \cdots$$

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

$$(\sigma_i, u_i, cons_i, Snd_i)_{i \in \mathbb{N}_0} \in \Sigma_{\mathscr{S}}^{\mathscr{D}} \times \mathscr{D}(\mathscr{C}) \times 2^{\mathscr{D}(\mathscr{E})} \times 2^{(\mathscr{D}(\mathscr{E}) \,\dot\cup\, \{*,+\}) \times \mathscr{D}(\mathscr{C})}$$

- The language $\mathcal{L}(\mathcal{M})$ of a UML model $\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$ is a word over the signature $\mathscr{S}(\mathscr{CD})$ induced by $\mathscr{CD}$ and $\mathscr{D}$, given structure $\mathscr{D}$.

Satisfaction of Signal and Attribute Expressions

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

Then

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

Satisfaction of Signal and Attribute Expressions

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

Then

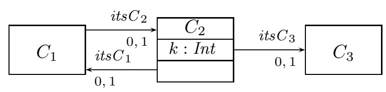
- $(\sigma, u, cons, Snd) \models_\beta$ ***true***
- $(\sigma, u, cons, Snd) \models_\beta \psi$ if and only if $I[\![\psi]\!](\sigma, \beta) = 1$
- $(\sigma, u, cons, Snd) \models_\beta \neg\psi$ if and only if not $(\sigma, cons, Snd) \models_\beta \psi$
- $(\sigma, u, cons, Snd) \models_\beta \psi_1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

Observation: we don't use all information from the computation path.

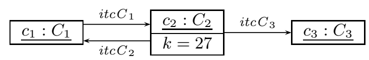
We could, e.g., also keep track of event identities between send and receive.

Example: Model Language and Signal / Attribute Expresions

$\mathcal{CD}$:



σ_0:



$$(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} \cdots \to (\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \xrightarrow[c_1]{(cons_1, \{(:E, c_2)\})} (\sigma_2, \varepsilon_2) \xrightarrow[c_2]{(\{:E\}, Snd_2)}$$

$$(\sigma_3, \varepsilon_3) \xrightarrow[c_2]{(cons_3, \{(:F, c_3)\})} (\sigma_4, \varepsilon_4) \xrightarrow[c_2]{(cons_4, \{(G(), c_1)\})} (\sigma_5, \varepsilon_5) \xrightarrow[c_3]{(\{:F\}, Snd_5)} (\sigma_6, \varepsilon_6) \to \cdots$$

- $\beta = \{x \mapsto c_1, y \mapsto c_2, z \mapsto c_3\}$
- $(\sigma_0, u_0, cons_0, Snd_0) \models_\beta y.k > 0$
- $(\sigma_0, u_0, cons_0, Snd_0) \models_\beta x.k > 0$
- $(\sigma_1, c_1, cons_1, \{(: E, c_2)\}) \models_\beta E^!_{x,y}$
- $(\sigma_1, c_1, cons_1, \{(: E, c_2)\}) \models_\beta F^!_{x,y}$
- $\cdots \models_\beta E^?_{x,y}$
- We set $(\sigma_4, c_2, cons_4, \{G(), c_1\}) \models_\beta G_{y,x}! \wedge G_{y,x}?$ (triggered operation or method call).

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 $\mathscr{S}$**.

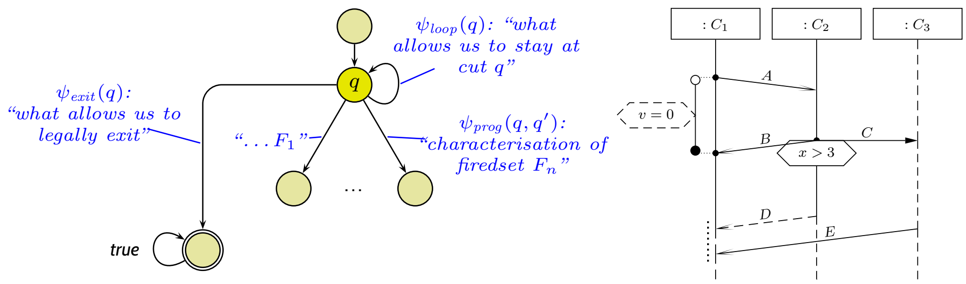
TBA Construction Principle

Recall: The TBA $\mathcal{B}(\mathscr{L})$ of LSC $\mathscr{L}$ is $(Expr_{\mathcal{B}}(X), X, Q, q_{ini}, \rightarrow, Q_F)$ with

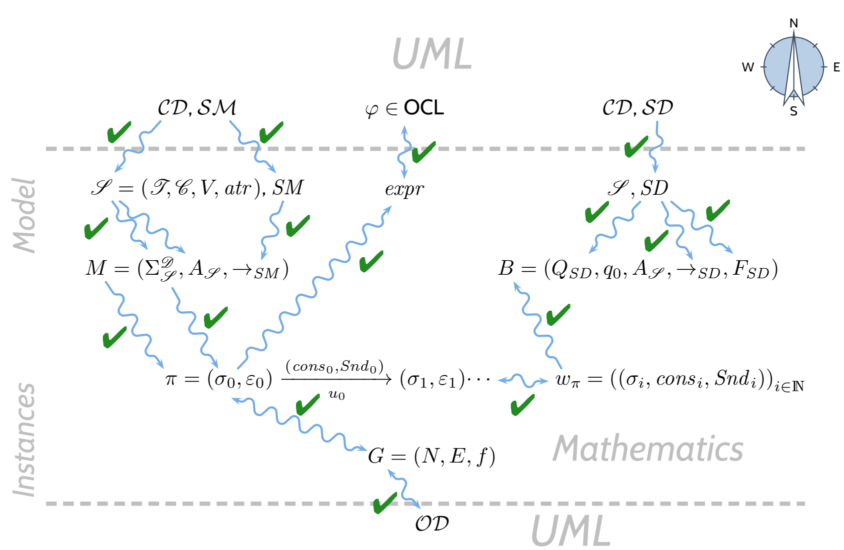
- Q is **the set of cuts** of $\mathscr{L}$, q_{ini} is the **instance heads** cut,
- $Expr_{\mathcal{B}} = \Phi \,\dot\cup\, \mathscr{E}_{!?}(X)$,
- $\rightarrow$ consists of **loops**, **progress transitions** (from $\rightsquigarrow_F$), and **legal exits** (cold cond./local inv.),
- $F = \{C \in Q \mid \Theta(C) = \text{cold} \vee C = L\}$ is the set of cold cuts.

So in the following, we "only" need to construct the transitions' labels:

$$\rightarrow = \{(q, \psi_{loop}(q), q) \mid q \in Q\} \cup \{(q, \psi_{prog}(q, q'), q') \mid q \rightsquigarrow_F q'\} \cup \{(q, \psi_{exit}(q), L) \mid q \in Q\}$$



Course Map



Live Sequence Charts — Semantics Cont'd

Full LSCs

A **full LSC** $\mathscr{L} = (((L, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta), ac_0, am, \Theta_{\mathscr{L}})$ consists of

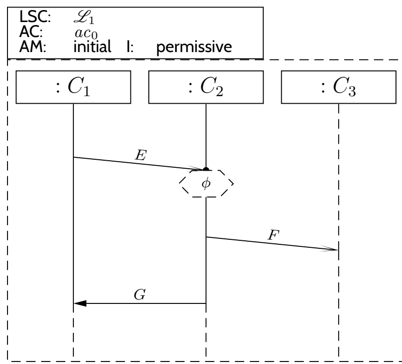
- **body** $((L, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta)$,
- **activation condition** $ac_0 \in Expr_{\mathscr{L}}$,
- **strictness flag** $strict$ (if *false*, $\mathscr{L}$ is called **permissive**)
- **activation mode** *am* $\in \{\text{initial}, \text{invariant}\}$,
- **chart mode** **existential** ($\Theta_{\mathscr{L}} = \text{cold}$) or **universal** ($\Theta_{\mathscr{L}} = \text{hot}$).

Full LSCs

A **full LSC** $\mathscr{L} = (((L, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta), ac_0, am, \Theta_{\mathscr{L}})$ consists of

- **body** $((L, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta)$,
- **activation condition** $ac_0 \in Expr_{\mathscr{S}}$,
- **strictness flag** $strict$ (if *false*, $\mathscr{L}$ is called **permissive**)
- **activation mode** *am* $\in$ {initial, invariant},
- **chart mode** **existential** ($\Theta_{\mathscr{L}}$ = cold) or **universal** ($\Theta_{\mathscr{L}}$ = hot).

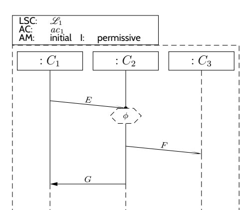
Concrete syntax:



Full LSCs

A **full LSC** $\mathscr{L} = (((L, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta), ac_0, am, \Theta_{\mathscr{L}})$ consists of

- **body** $((L, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta)$,
- **activation condition** $ac_0 \in Expr_{\mathscr{S}}$,
- **strictness flag** $strict$ (if *false*, $\mathscr{L}$ is called **permissive**)
- **activation mode** *am* $\in$ {initial, invariant},
- **chart mode** **existential** ($\Theta_{\mathscr{L}}$ = cold) or **universal** ($\Theta_{\mathscr{L}}$ = hot).

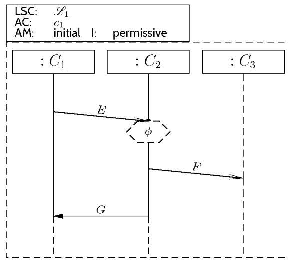


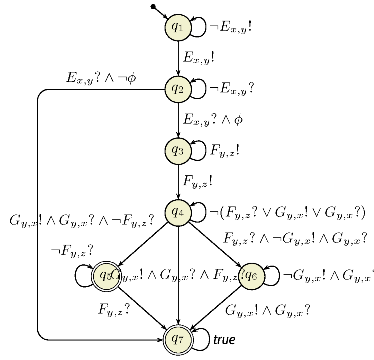
A **set of words** $W \subseteq (Expr_{\mathcal{B}} \to \mathbb{B})^\omega$ is **accepted** by $\mathscr{L}$ if and only if

$\Theta_{\mathscr{L}}$	am = initial	am = invariant
cold	$\exists\, w \in W \bullet w^0 \models ac \wedge \neg\psi_{exit}(C_0)$ $\wedge\, w^0 \models \psi_{prog}(\emptyset, C_0) \wedge w/1 \in \mathcal{L}(\mathcal{B}(\mathscr{L}))$	$\exists\, w \in W\, \exists\, k \in \mathbb{N}_0 \bullet w^k \models ac \wedge \neg\psi_{exit}(C_0)$ $\wedge\, w^k \models \psi_{prog}(\emptyset, C_0) \wedge w/k+1 \in \mathcal{L}(\mathcal{B}(\mathscr{L}))$
hot	$\forall\, w \in W \bullet w^0 \models ac \wedge \neg\psi_{exit}(C_0)$ $\implies w^0 \models \psi_{prog}(\emptyset, C_0) \wedge w/1 \in \mathcal{L}(\mathcal{B}(\mathscr{L}))$	$\forall\, w \in W\, \forall\, k \in \mathbb{N}_0 \bullet w^k \models ac \wedge \neg\psi_{exit}(C_0)$ $\implies w^k \models \psi^{\mathsf{Cond}}_{\mathsf{hot}}(\emptyset, C_0) \wedge w/k+1 \in \mathcal{L}(\mathcal{B}(\mathscr{L}))$

where C_0 is the minimal (or **instance heads**) cut.

Full LSC Semantics: Example

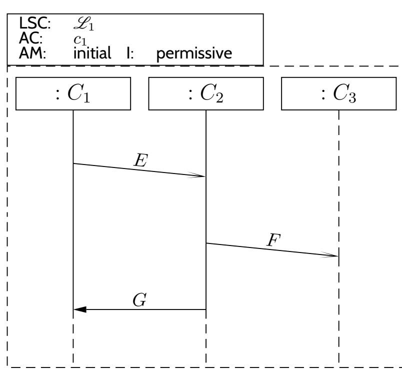


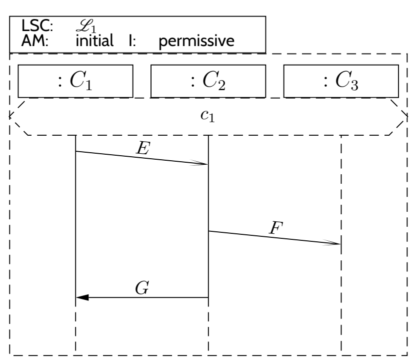


$$(\sigma,\varepsilon) \xrightarrow[u]{(cons,Snd)} \cdots \rightarrow (\sigma_0,\varepsilon_0) \xrightarrow[u_0]{(cons_0,Snd_0)} (\sigma_1,\varepsilon_1) \xrightarrow[c_1]{(cons_1,\{(:E,c_2)\})} (\sigma_2,\varepsilon_2) \xrightarrow[c_2]{(\{:E\},Snd_2)}$$

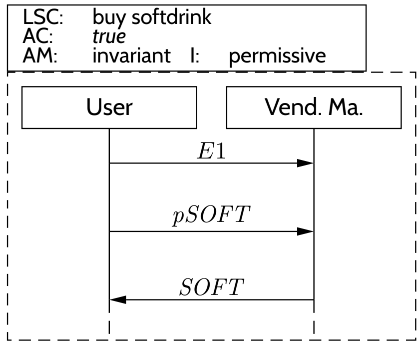
$$(\sigma_3,\varepsilon_3) \xrightarrow[c_2]{(cons_3,\{(:F,c_3)\})} (\sigma_4,\varepsilon_4) \xrightarrow[c_2]{(cons_4,\{(G(),c_1)\})} (\sigma_5,\varepsilon_5) \xrightarrow[c_3]{(\{:F\},Snd_5)} (\sigma_6,\varepsilon_6) \rightarrow \cdots$$

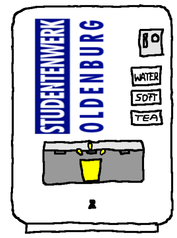
Note: Activation Condition



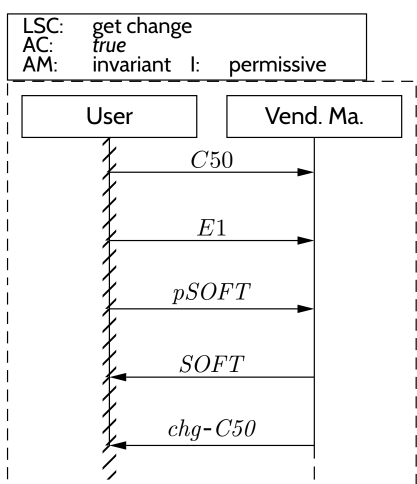


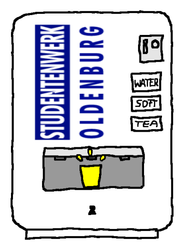
Existential LSC Example: Buy A Softdrink



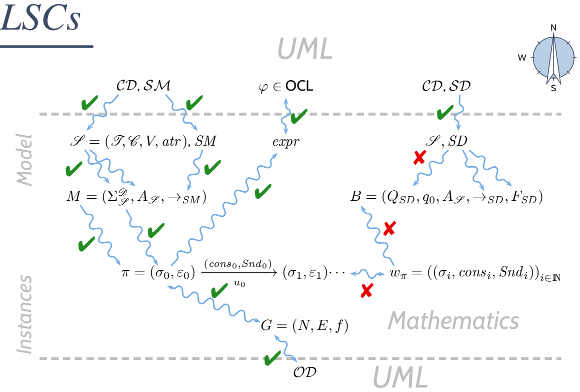


Existential LSC Example: Get Change





TBA-based Semantics of LSCs



Plan:

(i) Given an LSC $\mathscr{L}$ with body

$$((L, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta),$$

(ii) construct a TBA $\mathcal{B}_{\mathscr{L}}$, and

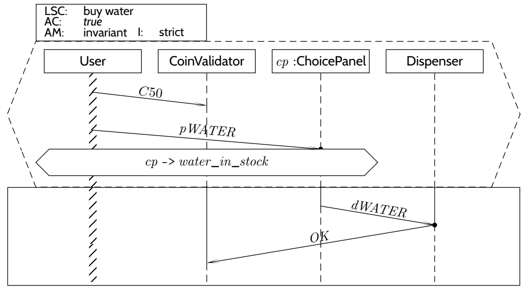
(iii) define language $\mathcal{L}(\mathscr{L})$ of $\mathscr{L}$ **in terms of** $\mathcal{L}(\mathcal{B}_{\mathscr{L}})$,
in particular taking activation condition and activation mode into account.

(iv) define language $\mathcal{L}(\mathcal{M})$ of a UML model.

- Then $\mathcal{M} \models \mathscr{L}$ (**universal**) if and only if $\mathcal{L}(\mathcal{M}) \subseteq \mathcal{L}(\mathscr{L})$.
  And $\mathcal{M} \models \mathscr{L}$ (**existential**) if and only if $\mathcal{L}(\mathcal{M}) \cap \mathcal{L}(\mathscr{L}) \neq \emptyset$.

Live Sequence Charts — Precharts

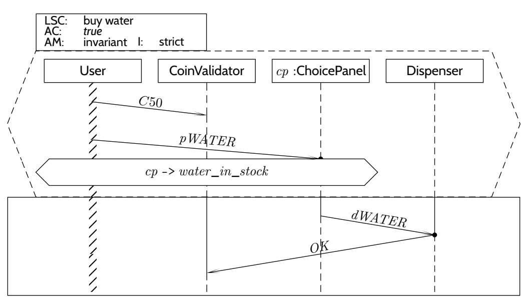
Pre-Charts



A **full LSC** $\mathscr{L} = (PC, MC, ac_0, am, \Theta_{\mathscr{L}})$ **actually** consist of

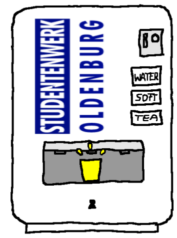
- **pre-chart** $PC = ((L_P, \preceq_P, \sim_P), \mathcal{I}_P, \mathscr{S}, \mathsf{Msg}_P, \mathsf{Cond}_P, \mathsf{LocInv}_P, \Theta_P)$ (possibly empty),
- **main-chart** $MC = ((L_M, \preceq_M, \sim_M), \mathcal{I}_M, \mathscr{S}, \mathsf{Msg}_M, \mathsf{Cond}_M, \mathsf{LocInv}_M, \Theta_M)$ (non-empty),
- **activation condition** $ac_0 : Bool \in Expr_{\mathscr{S}}$,
- **strictness flag** $strict$ (otherwise called **permissive**)
- **activation mode** *am* $\in$ {initial, invariant},
- **chart mode existential** ($\Theta_{\mathscr{L}}$ = cold) or **universal** ($\Theta_{\mathscr{L}}$ = hot).

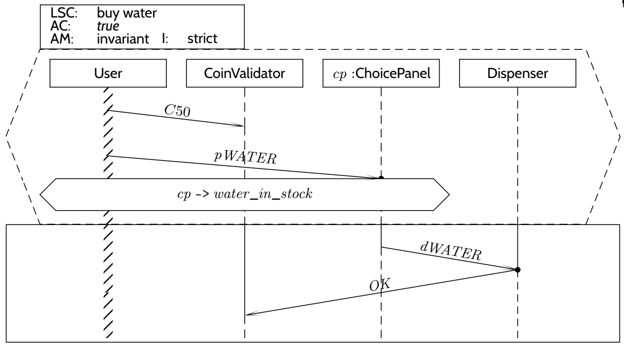
Pre-Charts Semantics



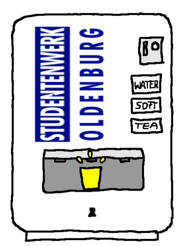
	am = initial	am = invariant
$\Theta_{\mathscr{L}}$ = cold	$\exists\, w \in W\ \exists\, m \in \mathbb{N}_0 \bullet$ $\wedge\, w^0 \models ac \wedge \neg\psi_{exit}(C_0^P) \wedge \psi_{prog}(\emptyset, C_0^P)$ $\wedge\, w^1, \dots, w^m \in \mathcal{L}(\mathcal{B}(PC))$ $\wedge\, w^{m+1} \models \neg\psi_{exit}(C_0^M)$ $\wedge\, w^{m+1} \models \psi_{prog}(\emptyset, C_0^M)$ $\wedge\, w/m+2 \in \mathcal{L}(\mathcal{B}(MC))$	$\exists\, w \in W\ \exists\, k < m \in \mathbb{N}_0 \bullet$ $\wedge\, w^k \models ac \wedge \neg\psi_{exit}(C_0^P) \wedge \psi_{prog}(\emptyset, C_0^P)$ $\wedge\, w^{k+1}, \dots, w^m \in \mathcal{L}(\mathcal{B}(PC))$ $\wedge\, w^{m+1} \models \neg\psi_{exit}(C_0^M)$ $\wedge\, w^{m+1} \models \psi_{prog}(\emptyset, C_0^M)$ $\wedge\, w/m+2 \in \mathcal{L}(\mathcal{B}(MC))$
$\Theta_{\mathscr{L}}$ = hot	$\forall\, w \in W\ \forall\, m \in \mathbb{N}_0 \bullet$ $\wedge\, w^0 \models ac \wedge \neg\psi_{exit}(C_0^P) \wedge \psi_{prog}(\emptyset, C_0^P)$ $\wedge\, w^1, \dots, w^m \in \mathcal{L}(\mathcal{B}(PC))$ $\wedge\, w^{m+1} \models \neg\psi_{exit}(C_0^M)$ $\implies w^{m+1} \models \psi_{prog}(\emptyset, C_0^M)$ $\wedge\, w/m+2 \in \mathcal{L}(\mathcal{B}(MC))$	$\forall\, w \in W\ \forall\, k \le m \in \mathbb{N}_0 \bullet$ $\wedge\, w^k \models ac \wedge \neg\psi_{exit}(C_0^P) \wedge \psi_{prog}(\emptyset, C_0^P)$ $\wedge\, w^{k+1}, \dots, w^m \in \mathcal{L}(\mathcal{B}(PC))$ $\wedge\, w^{m+1} \models \neg\psi_{exit}(C_0^M)$ $\implies w^{m+1} \models \psi_{prog}(\emptyset, C_0^M)$ $\wedge\, w/m+2 \in \mathcal{L}(\mathcal{B}(MC))$

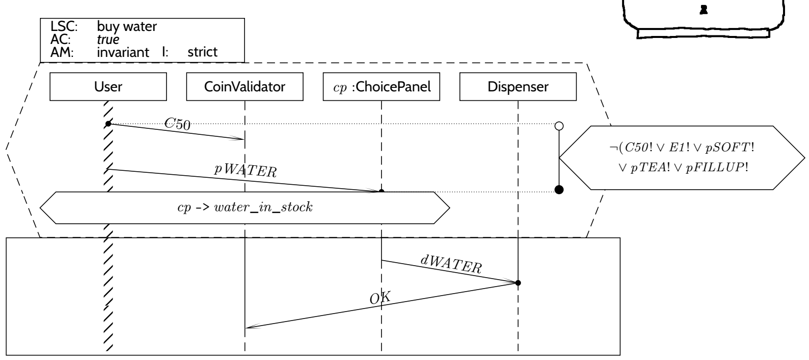
Universal LSC: Example



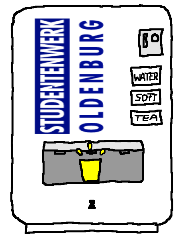


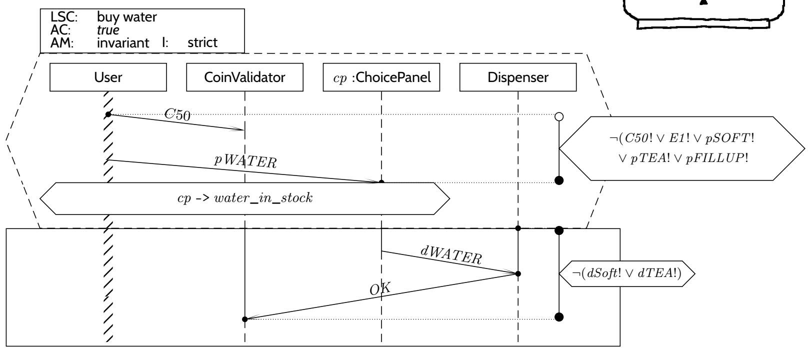
Universal LSC: Example



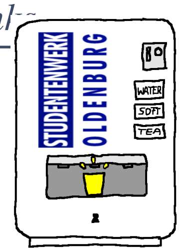


Universal LSC: Example

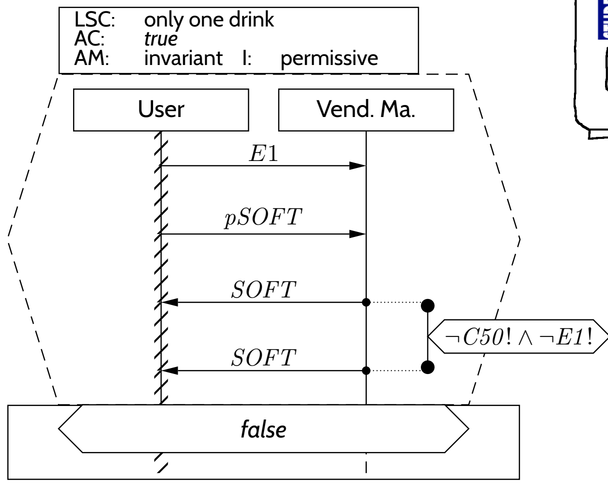


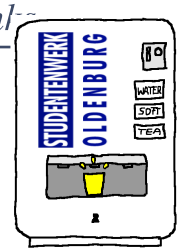


Forbidden Scenario Example: Don’t Give Two Drinks

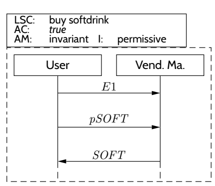


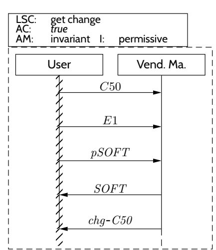
Forbidden Scenario Example: Don't Give Two Drinks

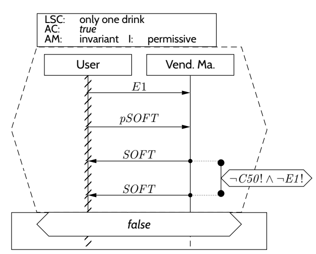




Note: Sequence Diagrams and (Acceptance) Test



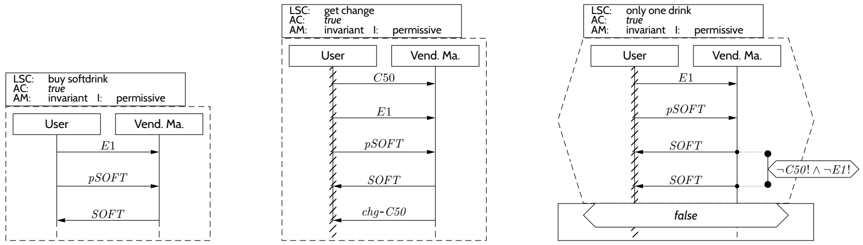




- **Existential** LSCs* may hint at **test-cases** for the **acceptance test**!

  (∗: as well as (positive) scenarios in general, like use-cases)

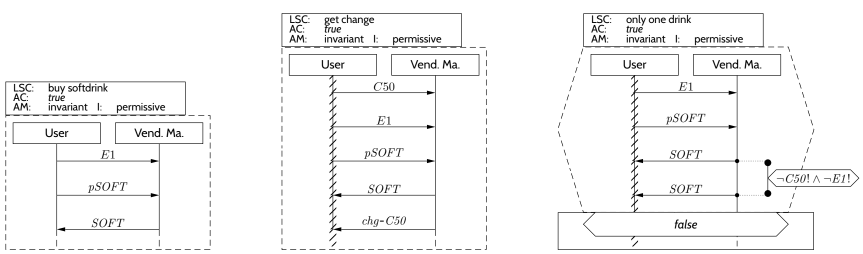
Note: Sequence Diagrams and (Acceptance) Test



- **Existential** LSCs* may hint at **test-cases** for the **acceptance test**!
  (∗: as well as (positive) scenarios in general, like use-cases)

- **Universal** LSCs (and negative/anti-scenarios) in general need **exhaustive analysis**!

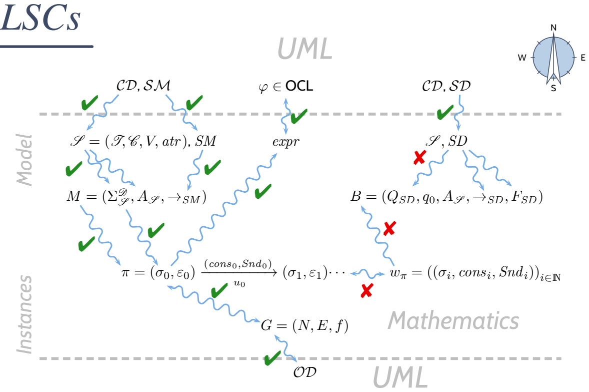
Note: Sequence Diagrams and (Acceptance) Test



- **Existential** LSCs* may hint at **test-cases** for the **acceptance test**!
  (∗: as well as (positive) scenarios in general, like use-cases)

- **Universal** LSCs (and negative/anti-scenarios) in general need **exhaustive analysis**!
  (Because they require that the software **never ever** exhibits the unwanted behaviour.)

TBA-based Semantics of LSCs



Plan:

(i) Given an LSC $\mathscr{L}$ with body

$$((L, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta),$$

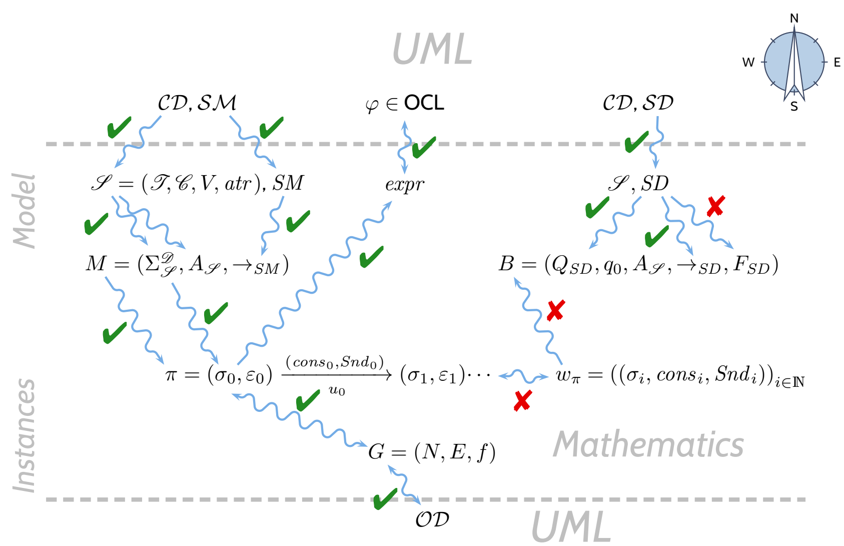
(ii) construct a TBA $\mathcal{B}_{\mathscr{L}}$, and

(iii) define language $\mathcal{L}(\mathscr{L})$ of $\mathscr{L}$ in terms of $\mathcal{L}(\mathcal{B}_{\mathscr{L}})$,
in particular taking activation condition and activation mode into account.

(iv) define language $\mathcal{L}(\mathcal{M})$ of a UML model.

- Then $\mathcal{M} \models \mathscr{L}$ (**universal**) if and only if $\mathcal{L}(\mathcal{M}) \subseteq \mathcal{L}(\mathscr{L})$.
  And $\mathcal{M} \models \mathscr{L}$ (**existential**) if and only if $\mathcal{L}(\mathcal{M}) \cap \mathcal{L}(\mathscr{L}) \neq \emptyset$.

Course Map



- **Büchi automata** accept **infinite words**
  - if there **exists is a run** over the word,
  - which visits an accepting state **infinitely often**.
- **The language of a model** is just a rewriting of **computations** into words over an alphabet.
- An LSC **accepts** a word (of a model) if

  Existential: at least on word (of the model) is accepted by the constructed TBA,

  Universion: all words (of the model) are accepted.
- Activation mode **initial** activates at system startup (only), **invariant** with each satisfied activation condition (or pre-chart).
- **Pre-charts** can be used to state **forbidden scenarios**.
- **Sequence Diagrams** can be useful for **testing**.

References

References

Crane, M. L. and Dingel, J. (2007). UML vs. classical vs. rhapsody statecharts: not all models are created equal. *Software and Systems Modeling*, 6(4):415–435.

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

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

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

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

OMG (2011a). Unified modeling language: Infrastructure, version 2.4.1. Technical Report formal/2011-08-05.

OMG (2011b). Unified modeling language: Superstructure, version 2.4.1. Technical Report formal/2011-08-06.

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.