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

- Reflective Descriptions of Behaviour
  - Interactions
  - A Brief History of Sequence Diagrams

- Live Sequence Charts
  - Abstract Syntax, Well-Formedness
  - Semantics
    - TBA Construction for LSC Body
    - Cuts, Firesets
    - Signal / Attribute Expressions
    - Loop / Progress Conditions
    - Excursion: Büchi Automata
  - Language of a Model
  - Full LSCs
    - Existential and Universal
    - Pre-Charts
    - Forbidden Scenarios
  - LSCs and Tests

- LS C not so simple can be concise
- NShA not so simple may be wrong
- CShA single short can get large
- ASH single may get big
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
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 \( M = (C, D, M, O, I) \) has a set of interactions \( I \).
- An interaction \( I \in I \) can be (OMG claim: equivalently) **dia grammed** as
  - communication diagram (formerly known as collaboration diagram),
  - timing diagram, or
  - sequence diagram.
In UML, reflective (temporal) descriptions are subsumed by interactions. A UML model $M = (C, D, S, O, I)$ has a set of interactions $I$.

An interaction $I \in I$ 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

\[ v = 0 \]
Full LSC Building Blocks for Later
Definition. [LSC Body]
An LSC body over signature \( \mathcal{S} = (\mathcal{F}, \mathcal{E}, V, atr, \delta) \) is a tuple
\[ ((L, \preceq, \sim), \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta) \]
where
- \( L \) is a finite, non-empty set 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. \( \preceq \cap \sim = \emptyset \),
- \( \mathcal{I} = \{I_1, \ldots, I_n\} \) is a partitioning of \( L \); elements of \( \mathcal{I} \) are called instance line,
- \( \text{Msg} \subseteq L \times \mathcal{E} \times L \) is a set of messages with \( (l, E, l') \in \text{Msg} \) only if \( (l, l') \in \prec \cup \sim \);
  message \( (l, E, l') \) is called instantaneous iff \( l \sim l' \) and asynchronous otherwise,
- \( \text{Cond} \subseteq (2^L \setminus \emptyset) \times \text{Expr}_S \) is a set of conditions
  with \( (L, \phi) \in \text{Cond} \) only if \( l \sim l' \) for all \( l \neq l' \in L \),
- \( \text{LocInv} \subseteq L \times \{\circ, \bullet\} \times \text{Expr}_S \times L \times \{\circ, \bullet\} \) is a set of local invariants
  with \( (l, i, \phi, l', i') \in \text{LocInv} \) only if \( l \sim l' \), \( \circ \): exclusive, \( \bullet \): inclusive,
- \( \Theta : L \cup \text{Msg} \cup \text{Cond} \cup \text{LocInv} \to \{\text{hot, cold}\} \)
  assigns to each location and each element a temperature.
locations $L$,
\[ \preceq \subseteq L \times L, \quad \sim \subseteq L \times L \]
$I = \{ I_1, \ldots, I_n \}$,
$\text{Msg} \subseteq L \times \mathcal{E} \times L$
$\text{Cond} \subseteq (2^L \setminus \emptyset) \times \text{Expr}$
$\text{LocInv} \subseteq L \times \{ \circ \cdot \} \times \text{Expr} \times L \times \{ \circ \cdot \}$
$\Theta : L \cup \text{Msg} \cup \text{Cond} \cup \text{LocInv} \rightarrow \{ \text{hot}, \text{cold} \}$.

\[ L = \{ \ell_{la}, \ldots, \ell_{lk}, \ell_{la}, \ldots, \ell_{lk} \} \]
\[ \preceq = \{ (\ell_{la}, \ell_{ka}), (\ell_{ka}, \ell_{ka}), (\ell_{ka}, \ell_{ka}), \ldots, (\ell_{la}, \ell_{ka}), (\ell_{ka}, \ell_{ka}) \} \]
\[ \sim = \{ (\ell_{la}, \ell_{ka}) \} \]
\[ \text{Msg} = \{ (\ell_{la}, A, \ell_{ka}), \ldots \} \]
\[ \text{Cond} = \{ \{ \ell_{la} \mid k > 3 \} \} \]
\[ \text{LocInv} = \{ (\ell_{la}, 0, \ell_{la, 2}, \bullet) \} \]
\[ \Theta = \{ \text{hot} \rightarrow \text{cold}, \ell_{la, 2} \rightarrow \text{cold}, A \rightarrow \text{hot}, \ldots \} \]
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. \( \exists (L, \phi) \in \text{Cond} : l \in L \), or
  - a **local invariant**, i.e. \( \exists (l_1, t_1, l_2) \in \text{LocInv} : l \in \{l_1, l_2\} \),

  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**
Plan:

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

\((L, \preceq, \sim), I, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)\),

(ii) construct a TBA $B_{\mathcal{L}}$, and

(iii) define language $\mathcal{L}(\mathcal{L})$ of $\mathcal{L}$ in terms of $\mathcal{L}(B_{\mathcal{L}})$,

in particular taking activation condition and activation mode into account.

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

Then $\mathcal{M} \models \mathcal{L}$ (universal) if and only if $\mathcal{L}(\mathcal{M}) \subseteq \mathcal{L}(\mathcal{L})$.

And $\mathcal{M} \models \mathcal{L}$ (existential) if and only if $\mathcal{L}(\mathcal{M}) \cap \mathcal{L}(\mathcal{L}) \neq \emptyset$.

---

**Live Sequence Charts — TBA Construction**
Definition.
Let \((L, \preceq, \sim), I, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)\) 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. \(\forall l, l' \in C \land l \preceq l' \implies l \in C\),
- it is **closed under simultaneity**, i.e.
  \[\forall l, l' \in C \land l \sim l' \implies l \in C,\]
- it comprises at least **one location per instance line**, i.e.
  \[\forall i \in I \exists l \in C \cdot il = i.\]

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

\[\Theta(C) = \begin{cases} 
  \text{hot} & \text{if } \exists l \in C \land (\exists l' \in C \cdot l \prec l') \land \Theta(l) = \text{hot} \\
  \text{cold} & \text{otherwise}
\end{cases}\]

that is, \(C\) is **hot** if and only if at least one of its maximal elements is hot.
$\emptyset \neq C \subseteq L$ – 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

\[ \emptyset \neq C \subseteq L \text{ - downward closed \ - simultaneity closed \ - at least one loc. per instance line} \]
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 \(((L, \preceq, \sim), I, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)\).

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.
  \[ \forall l \in F \exists l' \in C \; l' \prec l \land (\nexists l'' \in C \; l' \prec l'' \prec l) \],
- locations in \( F \), that lie on the same instance line, are pairwise unordered, i.e.
  \[ \forall l \neq l' \in F \; (\exists I \in I \; \{l, l'\} \subseteq I) \implies l \nless l' \land l' \nless l \],
- for each asynchronous (!) message reception in \( F \), the corresponding sending is already in \( C \),
  \[ \forall (l, E, l') \in \text{Msg} \; l' \in F \implies l \in C. \]

The cut \( C' = C \cup F \) is called direct successor of \( C \) via \( F \), denoted by \( C \succ_F C' \).

**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
**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**
The TBA $B_{\mathcal{L}}$ of LSC $\mathcal{L}$ over $\Phi$ and $\mathcal{L}$ is $(\text{Expr}_{\mathcal{L}}(X), X, Q_{\text{init}}, \rightarrow, Q_{F})$ with

- $Q$ is the set of cuts of $\mathcal{L}$, $q_{\text{init}}$ is the instance heads cut.
- $\text{Expr}_{\mathcal{L}}(X) = \text{Expr}_{\mathcal{S}}(\mathcal{E}, X)$ (for considered signature $\mathcal{S}$).
- $\rightarrow$ consists of loops, progress transitions (by $\rightarrow_{p}$), and legal exits (cold cond./local inv.).
- $Q_{F} = \{ C \in Q \mid \Theta(C) = \text{cold} \lor C = L \}$ is the set of cold cuts and the maximal cut.
Signal and Attribute Expressions

- Let $\mathcal{S} = (\mathcal{F}, \mathcal{E}, V, atr, \mathcal{E})$ be a signature and $X$ a set of logical variables.

- The signal and attribute expressions $\text{Expr}_\mathcal{S}(\mathcal{E}, X)$ are defined by the grammar:

$$\psi ::= \text{true} \mid \psi \mid E_{x,y}^1 \mid E_{x,y}^2 \mid \neg \psi \mid \psi_1 \lor \psi_2,$$

where $\text{expr} : \text{Bool} \in \text{Expr}_\mathcal{S}$, $E \in \mathcal{E}$, $x, y \in X$ (or keyword $\text{env}$).

- We use

$$\mathcal{E}_{\text{er}}(X) := \{E_{x,y}^1, E_{x,y}^2 \mid E \in \mathcal{E}, x, y \in X\}$$

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

TBA Construction Principle

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

- $Q$ is the set of cuts of $\mathcal{L}$, $q_{ini}$ is the instance heads cut,
- $\text{Expr}_B = \Phi \cup \mathcal{E}_{\text{er}}(X)$,
- $\rightarrow$ consists of loops, progress transitions from $\rightarrow_F$, and legal exits (cold cond./local inv.),
- $F = \{C \in Q \mid \Theta(C) = \text{cold} \lor C = L\}$ is the set of cold cuts.
Recall: The TBA $B(L)$ of LSC $L$ is $(\text{Expr}_B(X), X, Q, q_{ini}, \rightarrow, Q_F)$ with

- $Q$ is the set of cuts of $L$, $q_{ini}$ is the instance heads cut,
- $\text{Expr}_B = \Phi \cup \delta_U(X)$,
- $\rightarrow$ consists of loops, progress transitions [from $\rightarrow_f$], and legal exits [cold cond./local inv],
- $F = \{C \in Q \mid \Theta(C) = \text{cold} \lor C = L\}$ is the set of cold cuts.

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

$$\rightarrow = \{(q, q) \mid q \in Q\} \cup \{(q, q') \mid q \rightleftharpoons F q'\} \cup \{(q, L) \mid q \in Q\}$$
Recall: The TBA $B(\mathcal{L})$ of LSC $\mathcal{L}$ is $(\text{Expr}_R(X), X, Q, q_{out}, \rightarrow, Q_F)$ with

- $\mathcal{L}$ is the set of cuts of $\mathcal{L}$, $q_{out}$ is the instance heads cut,
- $\text{Expr}_R = \Phi \cup \delta_C(X)$,
- $\rightarrow$ consists of loops, progress transitions from $\rightarrow$ and legal exits (cold cond./local inv.),
- $F = \{C \in Q | \Theta(C) = \text{cold} \lor C = L\}$ is the set of cold cuts.

So in the following, we “only” need to construct the transitions’ labels:

$$\rightarrow = \{(q, \psi_{\text{loop}}(q), q) \mid q \in Q\} \cup \{(q, \psi_{\text{prop}}(q, q'), q') \mid q \rightarrow F q'\} \cup \{(q, \psi_{\text{exit}}(q), L) \mid q \in Q\}$$
Loop Condition

\[ \psi_{\text{loop}}(q) = \psi_{\text{Msg}}(q) \land \psi_{\text{LocInv}}(q) \land \psi_{\text{Cold}}(q) \]

- \( \psi_{\text{Msg}}(q) = \neg \bigvee_{1 \leq i \leq n} \psi_{\text{Msg}}(q, q_i) \land \left( \text{strict} \Rightarrow \bigwedge_{\psi \in \text{Msg}(L)} \neg \psi \right) \)

- \( \psi_{\text{LocInv}}(q) = \bigwedge_{l=(l,\ell,\phi,\ell',\iota) \in \text{LocInv}, \phi(l) = \phi, \ell \text{ active at } q} \phi \)

A location \( l \) is called \textit{front location} of cut \( C \) if and only if \( \exists l' \in L : l \prec l' \).

Local invariant \( (l_0, \iota_0, \phi, l_1, \iota_1) \) is \textit{active} at \( q \) if and only if \( l_0 \leq l < l_1 \) for some front location \( l \) of cut \( q \) or \( l_1 \in q \land \iota_1 = \ast \).

- \( \text{Msg}(F) = \{ E^{x_1, x_2} \mid (l, E, l') \in \text{Msg}, l \in F \} \cup \{ E^{x_1', x_2'} \mid (l, E, l') \in \text{Msg}, l' \in F \} \)

- \( x_1 \in X \) is the logical variable associated with the instance line \( I \) which includes \( l, i.e. l \in I \).

- \( \text{Msg}(F_1, \ldots , F_n) = \bigcup_{1 \leq i \leq n} \text{Msg}(F_i) \)

Progress Condition

\[ \psi_{\text{prog}}(q, q_{n}) = \psi_{\text{Msg}}(q, q_{n}) \land \psi_{\text{Cond}}(q, q_{n}) \land \psi_{\text{LocInv}}(q_{n}) \]

- \( \psi_{\text{Msg}}(q, q_{i}) = \bigwedge_{\psi \in \text{Msg}(q \cup q_i)} \phi \land \bigwedge_{j \neq i} \bigwedge_{\psi \in \text{Msg}(q \cup q_j)} \neg \psi \land \left( \text{strict} \Rightarrow \bigwedge_{\psi \in \text{Msg}(L) \setminus \text{Msg}(F_i)} \neg \psi \right) \)

- \( \psi_{\text{Cond}}(q, q_{i}) = \bigwedge_{\gamma=(L, \phi) \in \text{Cond}, \phi(\gamma) = \phi} \bigwedge_{\phi \in \text{Cond}, \phi(l) = \phi} \phi \)

- \( \psi_{\text{LocInv}}(q, q_{i}) = \bigwedge_{l=(l, \phi, \phi', l', \iota') \in \text{LocInv}, \phi(l) = \phi} \left( l \ast \text{active at } q \right) \phi \)

Local invariant \( (l_0, \iota_0, \phi, l_1, \iota_1) \) is \textit{active} at \( q \) if and only if
- \( l_0 < l < l_1, \text{ or } \)
- \( l = l_0 \land \iota_0 = \ast, \text{ or } \)
- \( l = l_1 \land \iota_1 = \ast \)

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

UML

Model

$\varphi \in \text{OCL}$

$M = (\Sigma_\varphi, A_\varphi, \rightarrow_{\text{SM}})$

$\pi = (\sigma_0, \varepsilon_0) \xrightarrow{(\text{cons}_0, \text{Snd}_0)} (\sigma_1, \varepsilon_1) \cdots$

$G = (N, E, f)$

Instances

$\mathbf{CD}, \mathbf{SM}$

$\mathbf{CD}, \mathbf{SD}$

$\mathbf{OD}$

Mathematics

UML

$\mathbf{UML}$
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*
Symbolic Büchi Automata

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

\[ B = (\text{Expr}_B(X), X, Q, q_{\text{ini}}, \rightarrow, Q_F) \]

where

- \( X \) is a set of logical variables,
- \( \text{Expr}_B(X) \) is a set of Boolean expressions over \( X \),
- \( Q \) is a finite set of states,
- \( q_{\text{ini}} \in Q \) is the initial state,
- \( \rightarrow \subseteq Q \times \text{Expr}_B(X) \times Q \) is the transition relation. Transitions \((q, \psi, q')\) from \( q \) to \( q' \) are labelled with an expression \( \psi \in \text{Expr}_B(X) \),
- \( Q_F \subseteq Q \) is the set of fair (or accepting) states.
**Definition.** Let $X$ be a set of logical variables and let $Expr_B(X)$ be a set of Boolean expressions over $X$.

A set $(\Sigma, \cdot \models \cdot)$ is called an **alphabet** for $Expr_B(X)$ if and only if

- for each $\sigma \in \Sigma$,
  - for each expression $expr \in Expr_B$, and
  - for each valuation $\beta : X \rightarrow \mathcal{P}(X)$ of logical variables,

    either $\sigma \models_\beta expr$ or $\sigma \not\models_\beta expr$.

(\sigma \text{ satisfies (or does not satisfy) } expr \text{ under valuation } \beta)

An infinite sequence $w = (\sigma_i)_{i \in \mathbb{N}_0} \in \Sigma^\omega$ over $(\Sigma, \cdot \models \cdot)$ is called **word** (for $Expr_B(X)$).

---

**Run of TBA over Word**

**Definition.** Let $B = (Expr_B(X), X, Q, q_{ini}, \rightarrow, Q_F)$ be a TBA and $w = \sigma_1, \sigma_2, \sigma_3, \ldots$ a word for $Expr_B(X)$. An infinite sequence $q = q_0, q_1, q_2, \ldots \in Q^\omega$ is called **run of $B$ over** $w$ under valuation $\beta : X \rightarrow \mathcal{P}(X)$ 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 $\sigma_i \models_\beta \psi_i$. 
Definition. Let $B = (\text{Expr}_B(X), X, Q, q_{\text{ini}}, \rightarrow, Q_F)$ be a TBA and

\[ w = \sigma_1, \sigma_2, \sigma_3, \ldots \]

a word for $\text{Expr}_B(X)$. An infinite sequence

\[ q = q_0, q_1, q_2, \ldots \in Q^\omega \]

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

- $q_0 = q_{\text{ini}}$.
- 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 $B = (\text{Expr}_B(X), X, Q, q_{\text{ini}}, \rightarrow, Q_F)$ accepts the word

\[ w = (\sigma_i)_{i \in \mathbb{N}_0} \in (\text{Expr}_B \rightarrow B)^\omega \]

if and only if $B$ has a run

\[ q = (q_i)_{i \in \mathbb{N}_0} \]

over $w$ such that fair (or accepting) states are visited infinitely often by $q$.

i.e., such that

\[ \forall i \in \mathbb{N}_0 \exists j > i : q_j \in Q_F. \]

We call the set $\mathcal{L}(B) \subseteq (\text{Expr}_B \rightarrow B)^\omega$ of words that are accepted by $B$ the language of $B$. 
The Language of a TBA

Definition.
We say TBA $B = (Expr_B(X), X, Q, q_{init}, \rightarrow, Q_F)$ accepts the word
$$w = (\sigma_i)_{i \in \mathbb{N}_0} \in (Expr_B \rightarrow B)^\omega$$
if and only if $B$ has a run
$$q = (q_i)_{i \in \mathbb{N}_0}$$
over $w$ such that fair (or accepting) states are visited infinitely often by $q$,
i.e., such that
$$\forall i \in \mathbb{N}_0 \exists j > i : q_j \in Q_F.$$

We call the set $L(B) \subseteq (Expr_B \rightarrow B)^\omega$ of words that are accepted by $B$ the language of $B$.

Example:

$B_{sym}$:

$\Sigma = (\{x\} \rightarrow \mathbb{N})$

Language of UML Model
Recall: A UML model \( M = (C, D, S, M) \) and a structure \( D \) denote a set \( \llbracket M \rrbracket \) of (initial and consecutive) computations of the form

\[
(\sigma_0, \varepsilon_0) \xrightarrow{a_0} (\sigma_1, \varepsilon_1) \xrightarrow{a_1} (\sigma_2, \varepsilon_2) \xrightarrow{a_2} \ldots
\]

where

\[
a_i = (\text{cons}_i, \text{Snd}_i, u_i) \in 2^{\mathcal{G}(E)} \times 2^{(\mathcal{G}(E) \cup \{*, +\})} \times \mathcal{G}(E).
\]

\( \llbracket M \rrbracket \) = \( \tilde{A} \).

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

**Definition.** Let \( M = (C, D, S, M) \) be a UML model and \( D \) a structure. Then

\[
\mathcal{L}(M) := \{ (\sigma_i, u_i, \text{cons}_i, \text{Snd}_i)_{i \in \mathbb{N}_0} \in (\Sigma_{\mathcal{G}(E)} \times \tilde{A})^{\omega} | \exists (\varepsilon_i)_{i \in \mathbb{N}_0} : (\sigma_0, \varepsilon_0) \xrightarrow{u_0} (\sigma_1, \varepsilon_1) \cdots \in \llbracket M \rrbracket \}
\]

is the language of \( M \).
**Example: Language of a Model**

\[
\mathcal{L}(M) := \{ (\sigma_i, u_i, \text{cons}_i, \text{Snd}_i) \mid i \in \mathbb{N}_0 \mid \exists (\varepsilon_i)_{i \in \mathbb{N}_0} : (\sigma_0, \varepsilon_0) \xrightarrow{u_0} (\sigma_1, \varepsilon_1) \cdots \in [M])
\]

\[
\begin{array}{c}
\text{CD:} \\
\begin{array}{c}
\begin{array}{c}
\text{C}_1 \quad \text{C}_2 \quad \text{C}_3
\end{array}
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\text{\sigma}_0: \\
\begin{array}{c}
\begin{array}{c}
\text{\text{cons}_0} \quad \text{\text{Snd}_0} \quad \text{C}_1 \quad \text{C}_2 \quad k = 27 \quad \text{C}_3
\end{array}
\end{array}
\end{array}
\]

**Words over Signature**

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

\[
(\sigma_i, u_i, \text{cons}_i, \text{Snd}_i)_{i \in \mathbb{N}_0} \in \Sigma_{\mathcal{S}} \times \mathcal{D}(\mathcal{I}) \times 2^{\mathcal{D}(\mathcal{E})} \times 2^{(\mathcal{D}(\mathcal{E}) \cup \{\ast, +\}) \times \mathcal{D}(\mathcal{E})}
\]

- The language \( \mathcal{L}(M) \) of a UML model \( M = (\mathcal{C} \mathcal{D}, \mathcal{I}, \mathcal{M}, \mathcal{C} \mathcal{D}) \) is a word over the signature \( \mathcal{S}(\mathcal{C} \mathcal{D}) \) induced by \( \mathcal{C} \mathcal{D} \) and \( \mathcal{D} \), given structure \( \mathcal{D} \).
Let $(\sigma, u, \text{cons}, \text{Snd}) \in \Sigma_{\mathcal{D}} \times \tilde{A}$ be a tuple consisting of system state, object identity, consume set, and send set.

Let $\beta : X \rightarrow \mathcal{D}(E)$ be a valuation of the logical variables.

Then

- $(\sigma, u, \text{cons}, \text{Snd}) \models_{\beta} \text{true}$
- $(\sigma, u, \text{cons}, \text{Snd}) \models_{\beta} \psi$ if and only if $I[\psi](\sigma, \beta) = 1$
- $(\sigma, u, \text{cons}, \text{Snd}) \models_{\beta} \neg \psi$ if and only if not $(\sigma, \text{cons}, \text{Snd}) \models_{\beta} \psi$
- $(\sigma, u, \text{cons}, \text{Snd}) \models_{\beta} \psi_1 \lor \psi_2$ if and only if $(\sigma, u, \text{cons}, \text{Snd}) \models_{\beta} \psi_1$ or $(\sigma, u, \text{cons}, \text{Snd}) \models_{\beta} \psi_2$
- $(\sigma, u, \text{cons}, \text{Snd}) \models_{\beta} E^1_{x,y}$ if and only if $\beta(x) = u \land \exists e \in \mathcal{D}(E) \cdot (e, \beta(y)) \in \text{Snd}$
- $(\sigma, u, \text{cons}, \text{Snd}) \models_{\beta} E^2_{x,y}$ if and only if $\beta(y) = u \land \text{cons} \subset \mathcal{D}(E)$

**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

\[ CD: \]

\[ \sigma_0: \]

\[ \beta = \{ x \mapsto c_1, y \mapsto c_2, z \mapsto c_3 \} \]

\[ (\sigma_0, u_0, cons_0, Snd_0) \models \beta \ y.k > 0 \]

\[ (\sigma_0, u_0, cons_0, Snd_0) \models \beta \ x.k > 0 \]

\[ (\sigma_1, c_1, cons_1, \{ (: E, c_2) \}) \models \beta \ E_{x,y}^1 \]

\[ (\sigma_1, c_1, cons_1, \{ (: E, c_2) \}) \models \beta \ F_{x,y} \]

\[ \cdots \models \beta \ E_{x,y}^2 \]

\[ \text{We set } (\sigma_4, c_2, cons_4, \{ G() \}, c_1) \models \beta \ G_{y,x}! \land G_{y,x}? \text{ (triggered operation or method call).} \]

---

**TBA over Signature**

**Definition.** A TBA

\[ B = (Exp_B(X), X, Q, q_{init}, \rightarrow, Q_F) \]

where \( Exp_B(X) \) is the set of signal and attribute expressions \( Exp_U(\mathcal{S}, X) \) over signature \( \mathcal{S} \) is called **TBA over** \( \mathcal{S} \).
Recall: The TBA $B(L)$ of LSC $L$ is $(\text{Expr}_B(X), X, Q, q_{\text{ini}}, \rightarrow, Q_F)$ with

- $Q$ is the set of cuts of $L$, $q_{\text{ini}}$ is the instance heads cut,
- $\text{Expr}_B = \Phi \cup \mathcal{E}$,
- $\rightarrow$ consists of loops, progress transitions $\text{from } \rightarrow F$ and legal exits (cold cond./local inv.),
- $F = \{ C \in Q \mid \Theta(C) = \text{cold} \lor C = L \}$ is the set of cold cuts.

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

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

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

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

Concrete syntax:

```
LSC: L1
AC: ac0
AM: initial
I: permissive
```

```
\[ \varphi : C_1 \quad \psi : C_2 \quad \Theta : C_3 \]
```

---

A set of words \( W \subseteq (\text{Expr}_B \rightarrow B)^\omega \) is accepted by \( \mathcal{L} \) if and only if

\[
\begin{align*}
\Theta_{\mathcal{L}} & \quad am = \text{initial} & am = \text{invariant} \\
\text{cold} & \quad \exists w \in W \cdot w^0 \models ac \land \neg \psi_{\text{exit}}(C_0) \\
& \quad \land w^0 \models \psi_{\text{prog}}(\emptyset, C_0) \land w/1 \in \mathcal{L}(B(\mathcal{L})) & \exists w \in W \exists k \in \mathbb{N}_0 \cdot w^k \models ac \land \neg \psi_{\text{exit}}(C_0) \\
& \quad \land w^k \models \psi_{\text{prog}}(\emptyset, C_0) \land w/k + 1 \in \mathcal{L}(B(\mathcal{L})) \\
\text{hot} & \quad \forall w \in W \cdot w^0 \models ac \land \neg \psi_{\text{exit}}(C_0) \\
& \quad \implies w^0 \models \psi_{\text{prog}}(\emptyset, C_0) \land w/1 \in \mathcal{L}(B(\mathcal{L})) & \forall w \in W \forall k \in \mathbb{N}_0 \cdot w^k \models ac \land \neg \psi_{\text{exit}}(C_0) \\
& \quad \implies w^k \models \psi^{\text{hot}}_{\text{exit}}(\emptyset, C_0) \land w/k + 1 \in \mathcal{L}(B(\mathcal{L}))
\end{align*}
\]

where \( C_0 \) is the minimal (or instance heads) cut.
Full LSC Semantics: Example

\[
\begin{align*}
(\sigma, \varepsilon) \xrightarrow{u} (\sigma_0, \varepsilon_0) & \quad (\sigma_0, \varepsilon_0) \xrightarrow{u_0} (\sigma_1, \varepsilon_1) \\
(\sigma_1, \varepsilon_1) \xrightarrow{c_1} (\sigma_2, \varepsilon_2) & \quad (\sigma_2, \varepsilon_2) \xrightarrow{c_2} (\sigma_3, \varepsilon_3) \\
(\sigma_3, \varepsilon_3) \xrightarrow{c_3} (\sigma_4, \varepsilon_4) & \quad (\sigma_4, \varepsilon_4) \xrightarrow{c_4} (\sigma_5, \varepsilon_5) \\
(\sigma_5, \varepsilon_5) \xrightarrow{c_5} (\sigma_6, \varepsilon_6) & \quad \ldots
\end{align*}
\]

Note: Activation Condition
Existential LSC Example: Buy A Softdrink

LSC: buy softdrink
AC: true
AM: invariant \(: \) permissive

User \(\xrightarrow{E1}\) Vend. Ma.

\(pSOFT\) \(\xrightarrow{}\) \(SOFT\)

Existential LSC Example: Get Change

LSC: get change
AC: true
AM: invariant \(: \) permissive

User \(\xrightarrow{C50}\) Vend. Ma.

\(E1\) \(\xrightarrow{}\) \(pSOFT\) \(\xrightarrow{}\) \(SOFT\) \(\xrightarrow{chg\cdot C50}\)
Plan:

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

\[
(\mathcal{L}, \preceq, \sim, I, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta),
\]

(ii) construct a TBA \( B_{\mathcal{L}} \), and

(iii) define language \( \mathcal{L}(\mathcal{L}) \) of \( \mathcal{L} \) in terms of \( \mathcal{L}(B_{\mathcal{L}}) \),

in particular taking activation condition and activation mode into account.

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

• Then \( \mathcal{M} \models \mathcal{L} \) (universal) if and only if \( \mathcal{L}(\mathcal{M}) \subseteq \mathcal{L}(\mathcal{L}) \).

And \( \mathcal{M} \models \mathcal{L} \) (existential) if and only if \( \mathcal{L}(\mathcal{M}) \cap \mathcal{L}(\mathcal{L}) \neq \emptyset \).
Pre-Charts

A full LSC $\mathcal{L} = (PC, MC, ac0, am, \Theta, \emptyset)$ actually consist of
- pre-chart $PC = ((L_P, \leq_P, \sim_P), \mathcal{T}_P, \mathcal{F}, \mathcal{M}(p, \theta_P), \text{Loc}\neg\neg_P, \Theta_P)$ (possibly empty).
- main-chart $MC = ((L_M, \leq_M, \sim_M), \mathcal{T}_M, \mathcal{F}, \mathcal{M}(m, \theta_M), \text{Loc}\neg\neg_M, \Theta_M)$ (non-empty).
- activation condition $ac0 : \text{Bool} \in \text{Expr } \emptyset$.
- strictness flag $\text{strict}$ (otherwise called permissive).
- activation mode $am \in \{\text{initial, invariant}\}$.
- chart mode existential $(\Theta, \emptyset = \text{cold})$ or universal $(\Theta, \emptyset = \text{hot})$.

Pre-Charts Semantics

$am = \text{initial}$

\[
\begin{align*}
\exists w \in W \forall m \in \mathbb{N}_0 \colon & \text{ } \\\\\\wedge w^0 = ac \wedge \neg \psi_{exit}(C_M^0) \wedge \psi_{inv}(0, C_M^0) \\
& \wedge w^1, \ldots, w^m \in \mathcal{L}(B(PC)) \\
& \wedge w^{m+1} = \neg \psi_{exit}(C_M^0) \\
& \wedge w^{m+1} = \psi_{inv}(0, C_M^0) \\
& \wedge w/m + 2 \in \mathcal{L}(B(MC))
\end{align*}
\]

$am = \text{invariant}$

\[
\begin{align*}
\exists w \in W \exists k < m \in \mathbb{N}_0 \colon & \text{ } \\\\\\wedge w^k = ac \wedge \neg \psi_{exit}(C_M^0) \wedge \psi_{inv}(0, C_M^0) \\
& \wedge w^{k+1}, \ldots, w^m \in \mathcal{L}(B(PC)) \\
& \wedge w^{m+1} = \neg \psi_{exit}(C_M^0) \\
& \wedge w^{m+1} = \psi_{inv}(0, C_M^0) \\
& \wedge w/m + 2 \in \mathcal{L}(B(MC))
\end{align*}
\]

$\Theta = \text{cold}$

\[
\begin{align*}
\forall w \in W \forall m \in \mathbb{N}_0 \colon & \text{ } \\\\\\wedge w^0 = ac \wedge \neg \psi_{exit}(C_M^0) \wedge \psi_{inv}(0, C_M^0) \\
& \wedge w^1, \ldots, w^m \in \mathcal{L}(B(PC)) \\
& \wedge w^{m+1} = \neg \psi_{exit}(C_M^0) \\
\Rightarrow & \wedge w^{m+1} = \psi_{inv}(0, C_M^0) \\
& \wedge w/m + 2 \in \mathcal{L}(B(MC))
\end{align*}
\]

$\Theta = \text{hot}$

\[
\begin{align*}
\forall w \in W \forall k \leq m \in \mathbb{N}_0 \colon & \text{ } \\\\\\wedge w^k = ac \wedge \neg \psi_{exit}(C_M^0) \wedge \psi_{inv}(0, C_M^0) \\
& \wedge w^{k+1}, \ldots, w^m \in \mathcal{L}(B(PC)) \\
& \wedge w^{m+1} = \neg \psi_{exit}(C_M^0) \\
\Rightarrow & \wedge w^{m+1} = \psi_{inv}(0, C_M^0) \\
& \wedge w/m + 2 \in \mathcal{L}(B(MC))
\end{align*}
\]
Universal LSC: Example

LSC: buy water
AC: true
AM: invariant I: strict

User CoinValidator cp ChoicePanel Dispenser

C50 WATER

cp -> water_in_stock

WATER

¬(C50 ∨ E1 ∨ pSOFT ∨ pTEA ∨ pFILLUP)
cp -> water_in_stock
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)
• **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.)
Plan:

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

\[ \left( \left( L, \preceq, \sim \right), I, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta \right) \]

(ii) construct a TBA $B_{\mathcal{L}}$, and

(iii) define language $\mathcal{L}(\mathcal{L})$ of $\mathcal{L}$ in terms of $\mathcal{L}(B_{\mathcal{L}})$,

in particular taking activation condition and activation mode into account.

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

• Then $M \models \mathcal{L}$ (universal) if and only if $\mathcal{L}(M) \subseteq \mathcal{L}(\mathcal{L})$.

And $M \models \mathcal{L}$ (existential) if and only if $\mathcal{L}(M) \cap \mathcal{L}(\mathcal{L}) \neq \emptyset$. 

---

Course Map
Büchi automata accept infinite words
- if there exists 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


