Softwaretechnik / Software-Engineering

Lecture 9: Live Sequence Charts

Topic Area Requirements Engineering: Content
Content

- Excursion: Symbolic Büchi Automata
- LSC Semantics:
  - Cuts, Firedsets,
  - Automaton Construction
  - Full LSC (activation, chart mode)
- Pre-Charts
  - Requirements Engineering with scenarios
  - Strengthening scenarios into requirements
- Software, formally
  - Software specification
  - Requirements Engineering, formally
  - Software implements specification
- LSCs vs. Software
  - Software implements LSCs
  - Scenarios and tests
  - Play In/Play Out
- Requirements Engineering Wrap-Up

LSC Semantics
The Plan: A Formal Semantics for a Visual Formalism

Excursion: Symbolic Büchi Automata
Symbolic Büchi Automata

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

\[ B = (C_B, Q, q_{ini}, \rightarrow, Q_F) \]

where

- \( C_B \) is a set of atomic propositions,
- \( Q \) is a finite set of states,
- \( q_{ini} \in Q \) is the initial state,
- \( \rightarrow \subseteq Q \times \Phi(C_B) \times Q \) is the finite transition relation.

Each transition \((q, \psi, q') \in \rightarrow\) from state \( q \) to state \( q' \) is labelled with a formula \( \psi \in \Phi(C_B) \).
- \( Q_F \subseteq Q \) is the set of fair (or accepting) states.

\( A: \Sigma = \{0, 1\} \)

1. \( A \)
2. \( B \)
3. \( B' \)

A symbolic Büchi automaton \( A \) accepts the word \( w = 01010 \ldots \) if \( \mathcal{L}(A) \cap 2^\mathbb{N} \neq \emptyset \).

\( B \)

1. \( B \)
2. \( B' \)

A symbolic Büchi automaton \( B \) accepts the word \( w = 01010 \ldots \) if \( \mathcal{L}(B) \cap 2^\mathbb{N} \neq \emptyset \).

\( B' \)

1. \( B' \)
2. \( B'' \)

A symbolic Büchi automaton \( B' \) accepts the word \( w = 01010 \ldots \) if \( \mathcal{L}(B') \cap 2^\mathbb{N} \neq \emptyset \).
Run of TBA

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

$$w = \sigma_1, \sigma_2, \sigma_3, \ldots \in (\Phi(C_B) \rightarrow B)^\omega$$

an infinite word. Each letter is a valuation of $\Phi(C_B)$.

An infinite sequence

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

of states is called run of $B$ over $w$ if and only if

- $q_0 = q_{ini}$,
- for each $i \in \mathbb{N}_0$ there is a transition $(q_i, \psi_i, q_{i+1}) \in \rightarrow$ s.t. $\sigma_i \models \psi_i$.

Example:

$$\varrho = (q_2, \{x \rightarrow 2\}, \{x \rightarrow 4\})$$

$$\mathcal{L} = \{q_A, q_B, q_A \cdot q_B, \ldots\}$$

The Language of a TBA

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

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

if and only if $B$ has a run $\varrho = (q_i)_{i \in \mathbb{N}_0}$ over $w$ such that fair (or accepting) states are visited infinitely often by $\varrho$, i.e., such that

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

We call the set $\text{Lang}(B) \subseteq (\Phi(C_B) \rightarrow B)^\omega$ of words that are accepted by $B$ the language of $B$.

Example:

$$\mathcal{B}_{sym}: \Sigma = \{(x) \rightarrow \mathbb{N}\}$$

$$\varrho = (q_2) \in \mathbb{N}_0$$

$$\mathcal{L} = \{q_A, q_B, q_A \cdot q_B, \ldots\}$$
The Plan: A Formal Semantics for a Visual Formalism

(L, ≤, ∼), I, Msg, Cond, Lcinv, Θ

abstract syntax

concrete syntax
(diagram)

semantics
(Büchi automaton)
Definition. Let \((\mathcal{L}, \preceq, \sim, \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)\) be an LSC body.

A non-empty set \(\emptyset \neq C \subseteq \mathcal{L}\) is called a cut of the LSC body iff \(C\)

- is downward closed, i.e.
  \[\forall l, l' \in \mathcal{L} \mid l \preceq l' \implies l \in C,\]

- is closed under simultaneity, i.e.
  \[\forall l, l' \in \mathcal{L} \mid l \sim l' \implies l \in C,\]

- comprises at least one location per instance line, i.e.
  \[\forall I \in \mathcal{I} \mid C \cap I \neq \emptyset.\]

The temperature function is extended to cuts as follows:

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

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

Cut Examples

\(\emptyset \neq C \subseteq \mathcal{L}\) - 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 \(C \subseteq L\) be 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 a **fired-set** \(\mathcal{F}\) of cut \(C\) if and only if

- \(C \cap F = \emptyset\) and \(C \cup F\) is a cut, i.e. \(\mathcal{F}\) is closed under simultaneity,
- all locations in \(\mathcal{F}\) are direct \(\prec\)-successors of the front of \(C\), i.e.
  \[
  \forall l \in \mathcal{F} \exists l' \in C \cdot l \prec l' \land (\exists l'' \in C \cdot l' \prec l'')
  \]
- locations in \(\mathcal{F}\), that lie on the same instance line, are pairwise unordered, i.e.
  \[
  \forall l \neq l' \in \mathcal{F} \cdot (\exists I \in I \cdot \{l, l'\} \subseteq I) \implies l \not\preceq l' \land l' \not\preceq l,
  \]
- for each asynchronous message reception in \(\mathcal{F}\), the corresponding sending is already in \(C\),
  \[
  \forall (l, E, l') \in \text{Msg} \cdot l' \in \mathcal{F} \implies l \in C.
  \]

The cut \(C' = C \cup F\) is called a **direct successor of** \(C\) via \(\mathcal{F}\), denoted by \(C \rightsquigarrow_{\mathcal{F}} C'\).
**Successor Cut Example**

1. \( C \cap F = \emptyset \) if \( C \cup F \) is a cut – only direct \( \Leftarrow \) successors – same instance line on front pairwise unordered – sending of asynchronous reception already in.

2. \( F_{1,1} \) in coregion.

3. \( F_{1,2} \) in coregion.

4. \( F_{1,3} \) in coregion.

5. \( F_{1,4} \) in coregion.

**Language of LSC Body: Example**

The TBA \( B(\mathcal{L}) \) of LSC \( \mathcal{L} \) over \( C \) and \( \mathcal{E} \) is \( (C_{B}, Q, q_{init}, \rightarrow, Q_F) \) with

- \( C_{B} = C \cup \mathcal{E}_{init} \), where \( \mathcal{E}_{init} = \{ E!1, E?1 \mid E \in \mathcal{E} \} \).
- \( Q \) is the set of cuts of \( \mathcal{L} \). \( q_{init} \) is the instance heads cut,
- \( \rightarrow \) consists of loops, progress transitions (from \( \sim_{\rightarrow} \)), and legal exits (cold cond./local inv.),
- \( Q_F = \{ C \in Q \mid \emptyset(C) = \text{cold} \vee C = \mathcal{L} \} \) is the set of cold cuts and the maximal cut.
Recall: The TBA $B(L)$ of LSC $L$ is $(C, Q, q_{ini}, \rightarrow, F)$ with

- $Q$ is the set of cuts of $L$, $q_{ini}$ is the instance heads cut,
- $C = C_B \sqcup E$,
- $\rightarrow$ consists of loops, progress transitions (from $\rightarrow, \gamma$), 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:

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

"Only" construct the transitions' labels:

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

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

- \( \psi_{\text{Msg}}(q) = \neg \bigvee_{1 \leq i \leq n} \psi_{\text{Msg}}(q, q_i) \land (\text{strict} \implies \bigwedge_{\psi \in E_i \land \text{Msg}}(c)) \)

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

A location \( l \) is called a front location of cut \( C \) if and only if it active at \( l < l' \).

Local invariant \( (l_0, \epsilon_0, \phi, l_1, \epsilon_1) \) is active at \( l \) if and only if \( l_0 \leq l < l_1 \) for some front location \( l \) of cut \( q \) or \( l = l_1 \land \epsilon_1 = 0 \).

- \( \text{Msg}(\mathcal{F}) = \{ E! | (l, E, l') \in \text{Msg}, l \in \mathcal{F} \} \cup \{ E? | (l, E, l') \in \text{Msg}, l' \in \mathcal{F} \} \)

- \( \text{Msg}((\mathcal{F}_1, \ldots, \mathcal{F}_n) = \bigcup_{1 \leq i \leq n} \text{Msg}(\mathcal{F}_i) \)

Progress Condition

\[ \psi_{\text{Prog}}^\text{bat}(q, q_i) = \psi_{\text{Msg}}(q, q_n) \land \psi_{\text{Cond}}^\text{bat}(q, q_n) \land \psi_{\text{LocInv}}(q, \bullet) \]

- \( \psi_{\text{Msg}}^\text{bat}(q, q_i) = \bigwedge_{\psi \in \text{Msg}}(q, q_i) \land \bigwedge_{\psi \neq \psi \in \text{Msg}}(q, q_i) \land \text{Msg}(q, q_i) \land \neg \psi \)

- \( \psi_{\text{Cond}}^\text{bat}(q, q_i) = \bigwedge_{\gamma = (L, \phi) \in \text{Cond}, \Theta(\gamma) = \theta, \gamma(q_i) \neq \emptyset} \psi \)

- \( \psi_{\text{LocInv}}^\text{bat}(q, q_i) = \bigwedge_{\lambda = (l, \phi, l', \phi') \in \text{LocInv}, \Theta(\lambda) = \theta, \lambda \text{ active at } q, \lambda} \psi \)

Local invariant \( (l_0, \epsilon_0, \phi, l_1, \epsilon_1) \) is \( \bullet \)-active at \( q \) if and only if

- \( l_0 < l < l_1 \), or
- \( l = l_0 \land \epsilon_0 = 0 \), or
- \( l = l_1 \land \epsilon_1 = 0 \)

for some front location \( l \) of cut \( l \).
**Full LSCs**

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

- **body** \( ((\mathcal{L}, \leq, \sim), I, Msg, Cond, LocInv, \Theta) \).
- **activation condition** \( ac_0 \in \Phi(C) \).
- **strictness flag** \( strict \) (if false, \( \mathcal{L} \) is permissive)
- **activation mode** \( am \in \{initial, invariant\} \).
- **chart mode** \( existential (\Theta_{\geq} = cold) \) or \( universal (\Theta_{\geq} = hot) \).

**Concrete syntax:**

![Concrete syntax diagram](image)
Full LSCs

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

- **body** $((\mathcal{L}, \preceq, \sim), \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)$,
- **activation condition** $\text{ac}_0 \in \Phi(\mathcal{C})$,
- **strictness flag** $\text{strict}$ (if false, $\mathcal{L}$ is permissive)
- **activation mode** $\text{am} \in \{\text{initial}, \text{invariant}\}$
- **chart mode** $\Theta L = \text{cold}$ or $\text{universal}$ ($\Theta L = \text{hot}$).

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

- cold
  - $\exists w \in W \bullet w^0 \models \text{ac} \land \\
  - w^n \models \psi_{\text{Cond}}(\emptyset, C_0) \land w/1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))$
  - $\exists w \in W \exists k \in \mathbb{N}_0 \bullet w^k \models \text{ac} \land \\
  - w^k \models \psi_{\text{Cond}}(\emptyset, C_0) \land w/k + 1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))$

- hot
  - $\forall w \in W \bullet w^0 \models \text{ac} \Rightarrow \\
  - w^n \models \psi_{\text{Cond}}(\emptyset, C_0) \land w/1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))$
  - $\forall w \in W \forall k \in \mathbb{N}_0 \bullet w^k \models \text{ac} \Rightarrow \\
  - w^k \models \psi_{\text{Cond}}(\emptyset, C_0) \land w/k + 1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))$

where $ac = \text{ac}_0 \land \psi_{\text{Cond}}(\emptyset, C_0) \land \psi_{\text{Msg}}(\emptyset, C_0); C_0$ is the minimal (or instance heads) cut.

Full LSC Semantics: Example
Example: Vending Machine

- **Positive scenario**: Buy a Softdrink
  (i) Insert one 1 euro coin.
  (ii) Press the 'softdrink' button.
  (iii) Get a softdrink.

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

- **Negative scenario**: A Drink for Free
  (i) Insert one 1 euro coin.
  (ii) Press the 'softdrink' button.
  (iii) Do not insert any more money.
  (iv) Get two softdrinks.

Example: Buy A Softdrink
Example: Get Change

Anti-Scenarios: Don’t Give Two Drinks
A full LSC $\mathcal{L} = (PC, MC, ac_0, am, \Theta_{\mathcal{L}})$ actually consist of

- activation condition $ac_0 \in \Phi(C)$,
- strictness flag strict (if false, $\mathcal{L}$ is permissive)
- activation mode $am \in \{\text{initial, invariant}\}$,
- chart mode existential ($\Theta_{\mathcal{L}} = \text{cold}$) or universal ($\Theta_{\mathcal{L}} = \text{hot}$).

Universal LSC: Example
### Requirements Engineering with Scenarios

One quite effective approach:

(i) **Approximate** the software requirements: ask for positive / negative existential scenarios.

(ii) **Refine** result into universal scenarios (and validate them with customer).

That is:

- Ask the customer to describe example usages of the desired system.
  - In the sense of: "If the system is not at all able to do this, then it’s not what I want.”
  - (→ positive use-cases, existential LSC)
- Ask the customer to describe behaviour that **must not happen** in the desired system.
  - In the sense of: "If the system does this, then it’s not what I want.”
  - (→ negative use-cases, LSC with pre-chart and hot-false)
- Investigate preconditions, side-conditions, exceptional cases and corner-cases.
  - (→ extend use-cases, refine LSCs with conditions or local invariants)
- Generalise into universal requirements, e.g. universal LSCs.
- Validate with customer using new positive / negative scenarios.
Strengthening Scenarios Into Requirements

- Ask customer for (pos./neg.) scenarios. note down as existential LSCs:

- Strengthen into requirements, note down as universal LSCs:

- Re-Discuss with customer using example words of the LSCs’ language.
Tell Them What You’ve Told Them...

- **Live Sequence Charts** (if well-formed)
  - have an abstract syntax.
  - From an abstract syntax, mechanically construct its **TBA**.
- A **universal LSC** is **satisfied** by a software $S$ if and only if
  - all words induced by the computation paths of $S$
  - are accepted by the LSC’s TBA.
- An **existential LSC** is **satisfied** by a software $S$ if and only if
  - there is a word induced by a computation path of $S$
  - which is accepted by the LSC’s TBA.
- **Pre-charts** allow us to specify
  - anti-scenarios (“this must not happen”),
  - activation interactions.
- **Method**:
  - discuss (anti-)scenarios with customer,
  - generalise into universal LSCs and re-validate.

References
References

