Softwaretechnik / Software-Engineering

Lecture 09: Live Sequence Charts

2015-06-11

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
Contents & Goals

Last Lecture:

- Scenarios and Anti-Scenarios
- User Stories, Use Cases, Use Case Diagrams
- LSC: abstract and concrete syntax

This Lecture:

- Educational Objectives: Capabilities for following tasks/questions.
  - Which are the cuts and firedsets of this LSC?
  - Construct the TBA of a given LSC body.
  - Given a set of LSCs, which scenario/anti-scenario/requirement is formalised by them?
  - Formalise this positive scenario/anti-scenario/requirement using LSCs.

- Content:
  - Excursion: automata accepting infinite words
  - Cuts and Firedsets, automaton construction
  - existential LSCs, pre-charts, universal LSCs
  - Requirements Engineering: conclusions
Recall: LSC Body Syntax

LSC Body Example

- $\mathcal{L} : l_{1,0} \prec l_{1,1} \prec l_{1,2} \prec l_{1,3}, \ l_{1,2} \prec l_{1,4}, \ l_{2,0} \prec l_{2,1} \prec l_{2,2} \prec l_{2,3}, \ l_{3,0} \prec l_{3,1} \prec l_{3,2},$
  
  - $l_{1,1} \prec l_{2,1}, \ l_{2,2} \prec l_{1,2}, \ l_{2,3} \prec l_{1,3}, \ l_{3,2} \prec l_{1,4}, \ l_{2,2} \sim l_{3,1},$
  
- $\mathcal{I} = \{\{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}\}, \{l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}\}, \{l_{3,0}, l_{3,1}, l_{3,2}\}\},$

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

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

- $\text{LocInv} = \{(l_{1,1}, o, c_1, l_{1,2}, \bullet)\}$
LSC Semantics
Recall: decision tables

By the standard semantics, a decision table $T$ is software, 
\[ [T] = \{ \sigma_0 \xrightarrow{\alpha_1} \sigma_1 \xrightarrow{\alpha_2} \sigma_2 \cdots | \cdots \} \] is a set of computation paths.
• **Recall**: decision tables

• By the standard semantics, a decision table $T$ is **software**, $\llbracket T \rrbracket = \{ \sigma_0 \xrightarrow{\alpha_1} \sigma_1 \xrightarrow{\alpha_2} \sigma_2 \cdots | \cdots \}$ is a set of computation paths.

• **Recall**: Decision tables as software specification:
• **Recall**: decision tables

• By the standard semantics, a decision table \( T \) is **software**, 
\[ \left[ T \right] = \left\{ \sigma_0 \xrightarrow{\alpha_1} \sigma_1 \xrightarrow{\alpha_2} \sigma_2 \cdots \mid \cdots \right\} \] is a set of computation paths.

• **Recall**: Decision tables as software specification:

• We want **the same** for LSCs.
• **Recall**: decision tables

• By the standard semantics, a decision table $T$ is **software**, $\llbracket T \rrbracket = \{ \sigma_0 \xrightarrow{\alpha_1} \sigma_1 \xrightarrow{\alpha_2} \sigma_2 \cdots | \cdots \}$ is a set of computation paths.

• **Recall**: Decision tables as software specification:

---

• We want **the same** for LSCs.

• We will give a **procedure** to construct for each LSC $\mathcal{L}$ an **automaton** $B(\mathcal{L})$. The language (or semantics) of $\mathcal{L}$ is the set of comp. paths **accepted** by $B(\mathcal{L})$. Thus an LSC is also software.
• **Recall**: decision tables

• By the standard semantics, a decision table $T$ is **software**,
  \[
  [T] = \{ \sigma_0 \xrightarrow{\alpha_1} \sigma_1 \xrightarrow{\alpha_2} \sigma_2 \cdots | \cdots \} \text{ is a set of computation paths.}
  \]

• **Recall**: Decision tables as software specification:

  ![Diagram](image.png)

• **We want the same** for LSCs.

• **We will give a procedure** to construct for each LSC $\mathcal{L}$ an **automaton** $B(\mathcal{L})$. The language (or semantics) of $\mathcal{L}$ is the set of comp. paths **accepted** by $B(\mathcal{L})$. Thus an LSC is also software.

• **Problem**: computation paths may be infinite $\rightarrow$ Büchi acceptance.
Excursion: Symbolic Büchi Automata
From Finite Automata to Symbolic Büchi Automata

\[ \mathcal{A}: \Sigma = \{0, 1\} \]

\[ \mathcal{B}: \Sigma = \{0, 1\} \]

\[ \mathcal{B'}: \Sigma = \{0, 1\} \]

\[ A_{\text{sym}}: \Sigma = (\{x\} \rightarrow \mathbb{N}) \]

\[ B_{\text{sym}}: \Sigma = (\{x\} \rightarrow \mathbb{N}) \]

\[ \text{even}(x) \]

\[ \text{odd}(x) \]

\[ \text{Büchi infinite words} \]
Symbolic Büchi Automata

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

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

where

- \( C \) is a set of atomic propositions,
- \( Q \) is a finite set of states,
- \( q_{ini} \in Q \) is the initial state,
- \( \rightarrow \subseteq Q \times \Phi(C) \times Q \) is the finite transition relation.
  Each transition \((q, \psi, q') \in \rightarrow \) from state \( q \) to state \( q' \) is labelled with a formula \( \psi \in \Phi(C) \).
- \( Q_F \subseteq Q \) is the set of fair (or accepting) states.
**Definition.** Let $\mathcal{B} = (\mathcal{C}, Q, q_{ini}, \rightarrow, Q_F)$ be a TBA and

$$w = \sigma_1, \sigma_2, \sigma_3, \cdots \in (\mathcal{C} \rightarrow \mathbb{B})^\omega$$

an infinite word, each letter is a valuation of $\mathcal{C}_B$.

An infinite sequence

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

of states is called **run** of $\mathcal{B}$ 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:**

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

\begin{align*}
\begin{array}{c}
\text{even}(x) \\
\text{odd}(x)
\end{array}
\end{align*}

$\mathcal{B}_{sym}$ with

- $q_1 \xrightarrow{\text{even}(x)} q_2 \xrightarrow{\text{odd}(x)} q_1$
The Language of a TBA

Definition.
We say TBA $B = (C, Q, q_{ini}, \rightarrow, Q_F)$ accepts the word $w = (\sigma_i)_{i \in \mathbb{N}_0} \in (C \rightarrow B)\omega$ 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 $Lang(B) \subseteq (C \rightarrow B)\omega$ of words that are accepted by $B$ the language of $B$. 
run: $\rho = q_0, q_1, q_2, \ldots \in Q^\omega$ s.t. $\sigma_i \models \psi_i$, $i \in \mathbb{N}_0$. 
LSC Semantics: TBA Construction
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.
  \[
  \forall l, l' \in \mathcal{L} \bullet l' \in C \land l \preceq l' \implies l \in C,
  \]

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

- comprises at least one location per instance line, i.e.
  \[
  \forall I \in \mathcal{I} \bullet C \cap I \neq \emptyset.
  \]
**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.
  \[
  \forall l, l' \in \mathcal{L} \cdot l' \in C \land l \preceq l' \implies l \in C,
  \]
- is **closed** under **simultaneity**, i.e.
  \[
  \forall l, l' \in \mathcal{L} \cdot l' \in C \land l \sim l' \implies l \in C, \text{ and}
  \]
- comprises at least **one location per instance line**, i.e.
  \[
  \forall I \in \mathcal{I} \cdot C \cap I \neq \emptyset.
  \]

The temperature function is extended to cuts as follows:

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

that is, \(C\) is **hot** if and only if at least one of its maximal elements is hot.
\[ \emptyset \neq C \subseteq \mathcal{L} \quad \text{— downward closed} \quad \text{— simultaneity closed} \quad \text{— at least one loc. per instance line} \]
$\emptyset \neq C \subseteq \mathcal{L}$ — downward closed — simultaneity closed — at least one loc. per instance line
\[ \emptyset \neq C \subseteq \mathcal{L} \quad \text{— downward closed} \quad \text{— simultaneity closed} \quad \text{— at least one loc. per instance line} \]
$\emptyset \neq C \subseteq \mathcal{L}$ — downward closed — simultaneity closed — at least one loc. per instance line
\[ \emptyset \neq C \subseteq \mathcal{L} \quad \text{— downward closed} \quad \text{— simultaneity closed} \quad \text{— at least one loc. per instance line} \]
\[ \emptyset \neq C \subseteq \mathcal{L} \text{ — downward closed — simultaneity closed — at least one loc. per instance line} \]
∅ \neq C \subseteq \mathcal{L} — downward closed — simultaneity closed — at least one loc. per instance line
$\emptyset \neq C \subseteq \mathcal{L}$ — downward closed — simultaneity closed — at least one loc. per instance line
A Successor Relation on Cuts

The partial order “≤” and the simultaneity relation “∼” of locations induce a direct successor relation on cuts of \( \mathcal{L} \) as follows:

**Definition.**
Let \( C \subseteq \mathcal{L} \) bet a cut of LSC body \(((\mathcal{L}, \preceq, \sim), \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)\). A set \( \emptyset \neq \mathcal{F} \subseteq \mathcal{L} \) is called **fired-set** \( \mathcal{F} \) of \( C \) if and only if

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

The cut \( C' = C \cup \mathcal{F} \) is called **direct successor of \( C \)** via \( \mathcal{F} \), denoted by \( C \sim_{\mathcal{F}} C' \).
$C \cap \mathcal{F} = \emptyset$ — $C \cup \mathcal{F}$ is a cut — only direct $\prec$-successors — same instance line on front pairwise unordered — sending of asynchronous reception already in
\[ C \cap \mathcal{F} = \emptyset \quad \text{—} \quad C \cup \mathcal{F} \text{ is a cut — only direct} \preceq \text{-successors — same instance line on front pairwise unordered — sending of asynchronous reception already in} \]
The TBA $B(\mathcal{L})$ of LSC $\mathcal{L}$ over $C$ and $\mathcal{E}$ is $(C, Q, q_{ini}, \rightarrow, Q_F)$ with

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

- $Q$ is the set of cuts of $L$, $q_{ini}$ is the instance heads cut,
- $C = C \cup \{E!, E? \mid E \in E\}$,
- $\rightarrow$ consists of loops, progress transitions (from $\sim_F$), 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.

$$\rightarrow = \{(q, \ , q) \mid q \in Q\} \cup \{(q, \ , q') \mid q \sim_F q'\} \cup \{(q, \ , L) \mid q \in Q\}$$
Recall: The TBA $\mathcal{B}(\mathcal{L})$ of LSC $\mathcal{L}$ is $(\mathcal{C}, Q, q_{ini}, \rightarrow, Q_F)$ with

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

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

$$\rightarrow = \{(q, q) \mid q \in Q\} \cup \{(q, q') \mid q \Rightarrow F q'\} \cup \{(q, \mathcal{L}) \mid q \in Q\}$$
Recall: The TBA $B(\mathcal{L})$ of LSC $\mathcal{L}$ is $(\mathcal{C}, Q, q_{ini}, \rightarrow, Q_F)$ with

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

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

$$\rightarrow = \{(q, \psi_{\text{loop}}(q), q) \mid q \in Q\} \cup \{(q, \psi_{\text{prog}}(q, q'), q') \mid q \rightsquigarrow_{\mathcal{F}} q'\} \cup \{(q, \psi_{\text{exit}}(q), \mathcal{L}) \mid q \in Q\}$$
Recall: The TBA $\mathcal{B}(\mathcal{L})$ of LSC $\mathcal{L}$ is $(C, Q, q_{ini}, \rightarrow, Q_F)$ with

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

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

$$\rightarrow = \{(q, \psi_{\text{loop}}(q), q) | q \in Q\} \cup \{(q, \psi_{\text{prog}}(q, q'), q') | q \rightsquigarrow F q'\} \cup \{(q, \psi_{\text{exit}}(q), \mathcal{L}) | q \in Q\}$$
So in the following, we “only” need to construct the transitions’ labels:

\[ \rightarrow = \{(q, \psi_{\text{loop}}(q), q) \mid q \in Q\} \cup \{(q, \psi_{\text{prog}}(q, q'), q') \mid q \xrightarrow{\not\in F} q'\} \cup \{(q, \psi_{\text{exit}}(q), \mathcal{L}) \mid q \in Q\} \]

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

\[
\psi_{\text{exit}}(q) = \psi_{\text{loop}}(q) \land \neg \psi_{\text{LocInv}}(q) \lor \bigvee_{1 \leq i \leq n} \psi_{\text{prog}}(q, q_i) \land \neg \psi_{\text{LocInv}}(q, q_i) \lor \neg \psi_{\text{Cond}}(q, q_i)
\]

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

\[
\psi_{\text{hot}}_{\text{prog}}(q, q_n) = \psi_{\text{Msg}}(q, q_n) \land \psi_{\text{Cond}}(q, q_n) \land \psi_{\text{LocInv}}(q, q_n) \land \psi_{\text{LocInv, hot}}(q, q_n)
\]
Loop Condition

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

- $$\psi^{\text{Msg}}(q) = \neg \bigvee_{1 \leq i \leq n} \psi^{\text{Msg}}(q, q_i) \land (\text{strict} \implies \bigwedge_{\psi \in \mathcal{E}} \neg \mathcal{M}(\mathcal{L}) \neg \psi)$$

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

A location $$l$$ is called **front location** of cut $$C$$ if and only if $$\nexists l' \in \mathcal{L} \bullet l < l'$$.

Local invariant $$(l_0, \iota_0, \phi, l_1, \iota_1)$$ is **active** at cut (!) $$q$$ if and only if $$l_0 \preceq l \preceq l_1$$ for some front location $$l$$ of cut (!) $$q$$.

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

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


Progress Condition

\[ \psi_{\text{hot}}^{\text{prog}}(q, q_i) = \psi_{\text{msg}}^{\text{hot}}(q, q_n) \land \psi_{\text{hot}}^{\text{cond}}(q, q_n) \land \psi_{\text{hot}}^{\text{loc inv}}(q_n) \]

- \[\psi_{\text{msg}}^{\text{hot}}(q, q_i) = \bigwedge_{\psi \in \text{msg}(q_i \setminus q)} \psi \land \bigwedge_{j \neq i} \bigwedge_{\psi \in (\text{msg}(q_j \setminus q) \setminus \text{msg}(q_i \setminus q))} \neg \psi \land (\text{strict} \implies \bigwedge_{\psi \in (E !? \cap \text{msg}()) \setminus \text{msg}(F_i) \neg \psi) \]

- \[\psi_{\text{cond}}^{\text{hot}}(q, q_i) = \bigwedge_{\gamma = (L, \phi) \in \text{cond}, \ \Theta(\gamma) = \theta, \ L \cap (q_i \setminus q) \neq \emptyset \phi} \]

- \[\psi_{\text{loc inv}}^{\text{hot}}(q, q_i) = \bigwedge_{\lambda = (l, t, \phi, l', t') \in \text{loc inv}, \ \Theta(\lambda) = \theta, \ \lambda \bullet \text{-active at } q_i \phi} \]

Local invariant \((l_0, \iota_0, \phi, l_1, \iota_1)\) is \(\bullet\)-active at \(q\) if and only if:

- \(l_0 < l < l_1\), or
- \(l = l_0 \land \iota_0 = \bullet\), or
- \(l = l_1 \land \iota_1 = \bullet\)

for some front location \(l\) of cut (1) \(q\).
Example
Finally: The LSC Semantics

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

- **body** $((\mathcal{L}, \preceq, \sim), \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)$,
- **activation condition** $ac_0 \in \Phi(C)$, **strictness flag** strict (otherwise called permissive)
- **activation mode** $am \in \{\text{initial}, \text{invariant}\}$,
- **chart mode** existential ($\Theta_{\mathcal{L}} = \text{cold}$) or **universal** ($\Theta_{\mathcal{L}} = \text{hot}$).
Finally: The LSC Semantics

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

- **body** $((\mathcal{L}, \preceq, \sim), \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)$,
- **activation condition** $ac_0 \in \Phi(C)$, **strictness flag** strict (otherwise 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:

![Diagram of LSC semantics](attachment:image.png)
Finally: The LSC Semantics

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

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

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

\[
\begin{array}{c|c|c}
\Theta_\mathcal{L} & am = \text{initial} & am = \text{invariant} \\
\hline
\text{cold} &  &  \\
\hline
\text{hot} &  &  \\
\end{array}
\]

where \( ac = ac_0 \land \psi_{\text{cond}}^{\text{cold}}(\emptyset, C_0) \land \psi_{\text{msg}}(\emptyset, C_0) \); \( C_0 \) is the minimal (or **instance heads**) cut.
Finally: The LSC Semantics

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

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

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

<table>
<thead>
<tr>
<th>$\Theta_\mathcal{L}$</th>
<th>$am = \text{initial}$</th>
<th>$am = \text{invariant}$</th>
</tr>
</thead>
</table>
| cold                | $\exists w \in W \bullet w^0 \models ac \land$
                          | $w^0 \models \psi^\text{Cond}_{\text{hot}}(\emptyset, C_0) \land w/1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))$ |                        |
| hot                 |                      |                        |

where $ac = ac_0 \land \psi^\text{Cond}_{\text{cold}}(\emptyset, C_0) \land \psi^\text{Msg}(\emptyset, C_0)$; $C_0$ is the minimal (or **instance heads**) cut.
Finally: The LSC Semantics

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

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

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

<table>
<thead>
<tr>
<th>$\Theta_{\mathcal{L}}$</th>
<th>$am = \text{initial}$</th>
<th>$am = \text{invariant}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cold</strong></td>
<td>$\exists w \in W \bullet w^0</td>
<td>\models ac \land$</td>
</tr>
<tr>
<td></td>
<td>$w^0</td>
<td>\models \psi_{\text{Cond}}^{\text{hot}}(\emptyset, C_0) \land w/1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))$</td>
</tr>
<tr>
<td><strong>hot</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where $ac = ac_0 \land \psi_{\text{Cond}}^{\text{cold}}(\emptyset, C_0) \land \psi_{\text{Msg}}(\emptyset, C_0)$; $C_0$ is the minimal (or instance heads) cut.
A **full LSC** $\mathcal{L} = (((\mathcal{L}, \preceq, \sim), \mathcal{I}, \operatorname{Msg}, \operatorname{Cond}, \operatorname{LocInv}, \Theta), ac_0, am, \Theta \mathcal{L})$ consist of

- **body** $((\mathcal{L}, \preceq, \sim), \mathcal{I}, \operatorname{Msg}, \operatorname{Cond}, \operatorname{LocInv}, \Theta),$  
- **activation condition** $ac_0 \in \Phi(C)$, **strictness flag** $\mathbf{strict}$ (otherwise called **permissive**),
- **activation mode** $am \in \{\text{initial, invariant}\},$
- **chart mode** **existential** ($\Theta \mathcal{L} = \text{cold}$) or **universal** ($\Theta \mathcal{L} = \text{hot}$).

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

<table>
<thead>
<tr>
<th>$\Theta \mathcal{L}$</th>
<th>$am = \text{initial}$</th>
<th>$am = \text{invariant}$</th>
</tr>
</thead>
</table>
| **cold**            | $\exists w \in W \bullet w^0 \models ac \land$  
                      | $w^0 \models \psi^\text{Cond}_\text{hot} (\emptyset, C_0) \land w/1 \in \operatorname{Lang}(\mathcal{B}(\mathcal{L}))$ | $\exists w \in W \exists k \in \mathbb{N}_0 \bullet w^k \models ac \land$  
                      | $w^k \models \psi^\text{Cond}_\text{hot} (\emptyset, C_0) \land w/k + 1 \in \operatorname{Lang}(\mathcal{B}(\mathcal{L}))$ |
| **hot**             | $\forall w \in W \bullet w^0 \models ac \implies$  
                      | $w^0 \models \psi^\text{Cond}_\text{hot} (\emptyset, C_0) \land w/1 \in \operatorname{Lang}(\mathcal{B}(\mathcal{L}))$ |

where $ac = ac_0 \land \psi^\text{Cond}_{\text{cold}} (\emptyset, C_0) \land \psi^\text{Msg}_{\text{cold}} (\emptyset, C_0)$; $C_0$ is the minimal (or **instance heads**) cut.
Finally: The LSC Semantics

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

- **body** \( (((\mathcal{L}, \preceq, \sim), \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)) \),
- **activation condition** \( ac_0 \in \Phi(C) \), **strictness flag** **strict** (otherwise called **permissive**)  
- **activation mode** \( am \in \{\text{initial}, \text{invariant}\} \),
- **chart mode** **existential** \( (\Theta \mathcal{L} = \text{cold}) \) or **universal** \( (\Theta \mathcal{L} = \text{hot}) \).

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

<table>
<thead>
<tr>
<th>( \Theta \mathcal{L} )</th>
<th>( am = \text{initial} )</th>
<th>( am = \text{invariant} )</th>
</tr>
</thead>
</table>
| **cold** | \( \exists w \in W \cdot w^0 \models ac \land \)  
\( w^0 \models \psi_{\text{Cond}}^{\text{hot}}(\emptyset, C_0) \land w/1 \in \text{Lang}(\mathcal{B}(\mathcal{L})) \) | \( \exists w \in W \exists k \in \mathbb{N}_0 \cdot w^k \models ac \land \)  
\( w^k \models \psi_{\text{Cond}}^{\text{hot}}(\emptyset, C_0) \land w/k + 1 \in \text{Lang}(\mathcal{B}(\mathcal{L})) \) |
| **hot** | \( \forall w \in W \cdot w^0 \models ac \implies \)  
\( w^0 \models \psi_{\text{Cond}}^{\text{hot}}(\emptyset, C_0) \land w/1 \in \text{Lang}(\mathcal{B}(\mathcal{L})) \) | \( \forall w \in W \forall k \in \mathbb{N}_0 \cdot w^k \models ac \implies \)  
\( w^k \models \psi_{\text{Cond}}^{\text{hot}}(\emptyset, C_0) \land w/k + 1 \in \text{Lang}(\mathcal{B}(\mathcal{L})) \) |

where \( ac = ac_0 \land \psi_{\text{cold}}^{\text{Cond}}(\emptyset, C_0) \land \psi_{\text{Msg}}(\emptyset, C_0) \); \( C_0 \) is the minimal (or **instance heads**) cut.
Activation Condition

LSC: $L_1$
AC: $c_1$
AM: initial I: permissive

$I_1$  $I_2$  $I_3$

$E$  $F$  $G$
LSCs vs. Software
Let $S$ be a software with $\llbracket S \rrbracket = \{ \pi = \sigma_0 \xrightarrow{\alpha_1} \sigma_1 \xrightarrow{\alpha_2} \sigma_2 \cdots | \cdots \}. $

$S$ is called **compatible** with LSC $\mathcal{L}$ over $C$ and $\mathcal{E}$ is if and only if

- $\Sigma = (C \rightarrow \mathbb{B})$, i.e. the states are valuations of the conditions in $C$,
- $A \subseteq \mathcal{E}!\?, \text{ i.e. the events are of the form } E!, E\,$.
Let $S$ be a software with $\llbracket S \rrbracket = \{ \pi = \sigma_0 \xrightarrow{\alpha_1} \sigma_1 \xrightarrow{\alpha_2} \sigma_2 \cdots | \cdots \}$. $S$ is called **compatible** with LSC $\mathcal{L}$ over $C$ and $E$ is if and only if

- $\Sigma = (C \rightarrow \mathbb{B})$, i.e. the states are valuations of the conditions in $C$,
- $A \subseteq E!\?$, i.e. the events are of the form $E!, E?$.

Construct letters by joining $\sigma_i$ and $\alpha_{i+1}$ (viewed as a valuation of $E!, E?$):

$$w(\pi) = (\sigma_0 \cup \alpha_1), (\sigma_1 \cup \alpha_2), (\sigma_2 \cup \alpha_3), \ldots$$
Let $S$ be a software with $[S] = \{\pi = \sigma_0 \xrightarrow{\alpha_1} \sigma_1 \xrightarrow{\alpha_2} \sigma_2 \cdots | \cdots \}$. $S$ is called **compatible** with LSC $\mathcal{L}$ over $C$ and $E$ is if and only if

- $\Sigma = (C \rightarrow \mathbb{B})$, i.e. the states are valuations of the conditions in $C$,
- $A \subseteq \mathcal{E}!?$, i.e. the events are of the form $E!$, $E?$.

Construct letters by joining $\sigma_i$ and $\alpha_{i+1}$ (viewed as a valuation of $E!$, $E?$):

$$w(\pi) = (\sigma_0 \cup \alpha_1), (\sigma_1 \cup \alpha_2), (\sigma_2 \cup \alpha_3), \ldots$$

We say $S$ satisfies LSC $\mathcal{L}$ (e.g. universal, invariant), denoted by $S \models \mathcal{L}$, if and only if

$$\forall \pi \in [S] \forall k \in \mathbb{N}_0 \bullet w(\pi)^k \models ac \implies w(\pi)^k \models \psi_{\text{Cond}}^{\text{hot}}(\emptyset, C_0) \land w(\pi)/k + 1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))$$
Let $S$ be a software with $[S] = \{ \pi = \sigma_0 \xrightarrow{\alpha_1} \sigma_1 \xrightarrow{\alpha_2} \sigma_2 \cdots | \cdots \}$.  

$S$ is called **compatible** with LSC $\mathcal{L}$ over $C$ and $\mathcal{E}$ is if and only if

- $\Sigma = (C \rightarrow \mathbb{B})$, i.e. the states are valuations of the conditions in $C$,
- $A \subseteq \mathcal{E}!$, i.e. the events are of the form $E!$, $E?$.

Construct letters by joining $\sigma_i$ and $\alpha_{i+1}$ (viewed as a valuation of $E!$, $E?$):

$$ w(\pi) = (\sigma_0 \cup \alpha_1), (\sigma_1 \cup \alpha_2), (\sigma_2 \cup \alpha_3), \ldots $$

We say $S$ **satisfies** LSC $\mathcal{L}$ (e.g. universal, invariant), denoted by $S \models \mathcal{L}$, if and only if

$$ \forall \pi \in [S] \forall k \in \mathbb{N}_0 \bullet w(\pi)^k \models ac \implies w(\pi)^k \models \psi_{\text{Cond}}^{\text{hot}}(\emptyset, C_0) \land w(\pi)/k + 1 \in \text{Lang}(B(\mathcal{L})) $$

<table>
<thead>
<tr>
<th>$\Theta_{\mathcal{L}}$</th>
<th>$am = \text{initial}$</th>
<th>$am = \text{invariant}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cold</strong></td>
<td>$\exists w \in W \bullet w^0 \models ac \land$</td>
<td>$\exists w \in W \bullet w^0 \models ac \land$</td>
</tr>
<tr>
<td></td>
<td>$w^0 \models \psi_{\text{hot}}^{\text{Cond}}(\emptyset, C_0) \land w/1 \in \text{Lang}(B(\mathcal{L}))$</td>
<td>$w^k \models \psi_{\text{hot}}^{\text{Cond}}(\emptyset, C_0) \land w/k + 1 \in \text{Lang}(B(\mathcal{L}))$</td>
</tr>
<tr>
<td><strong>hot</strong></td>
<td>$\forall w \in W \bullet w^0 \models ac \implies$</td>
<td>$\forall w \in W \bullet w^0 \models ac \implies$</td>
</tr>
<tr>
<td></td>
<td>$w^0 \models \psi_{\text{hot}}^{\text{Cond}}(\emptyset, C_0) \land w/1 \in \text{Lang}(B(\mathcal{L}))$</td>
<td>$w^k \models \psi_{\text{hot}}^{\text{Cond}}(\emptyset, C_0) \land w/k + 1 \in \text{Lang}(B(\mathcal{L}))$</td>
</tr>
</tbody>
</table>
LSCs vs. Software

Let \( S \) be a software with \( [S] = \{ \pi = \sigma_0 \xrightarrow{\alpha_1} \sigma_1 \xrightarrow{\alpha_2} \sigma_2 \cdots | \cdots \} \).

\( S \) is called \textbf{compatible} with LSC \( \mathcal{L} \) over \( C \) and \( E \) is if and only if

- \( \Sigma = (C \rightarrow \mathbb{B}) \), i.e. the states are valuations of the conditions in \( C \),
- \( A \subseteq E_! \), i.e. the events are of the form \( E! \), \( E? \).

Construct letters by joining \( \sigma_i \) and \( \alpha_{i+1} \) (viewed as a valuation of \( E! \), \( E? \)):

\[
w(\pi) = (\sigma_0 \cup \alpha_1), (\sigma_1 \cup \alpha_2), (\sigma_2 \cup \alpha_3), \ldots
\]

We say \( S \) \textbf{satisfies} LSC \( \mathcal{L} \) (e.g. universal, invariant), denoted by \( S \models \mathcal{L} \), if and only if

\[
\forall \pi \in [S] \forall k \in \mathbb{N}_0 \cdot w(\pi)^k \models ac \implies w(\pi)^k \models \psi_{\text{hot}}^\text{Cond}(\emptyset, C_0) \land w(\pi)/k + 1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))
\]

<table>
<thead>
<tr>
<th>( \Theta_{\mathcal{L}} )</th>
<th>( am = \text{initial} )</th>
<th>( am = \text{invariant} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{cold}</td>
<td>\exists w \in W \cdot w^0 \models ac \land w^0 \models \psi_{\text{hot}}^\text{Cond}(\emptyset, C_0) \land w/1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))</td>
<td>\exists w \in W \exists k \in \mathbb{N}<em>0 \cdot w^k \models ac \land w^k \models \psi</em>{\text{hot}}^\text{Cond}(\emptyset, C_0) \land w/k + 1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))</td>
</tr>
<tr>
<td>\text{hot}</td>
<td>\forall w \in W \cdot w^0 \models ac \implies w^0 \models \psi_{\text{hot}}^\text{Cond}(\emptyset, C_0) \land w/1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))</td>
<td>\forall w \in W \forall k \in \mathbb{N}<em>0 \cdot w^k \models ac \implies w^k \models \psi</em>{\text{hot}}^\text{Cond}(\emptyset, C_0) \land w/k + 1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))</td>
</tr>
</tbody>
</table>

Software \( S \) \textbf{satisfies a set of} LSCs \( \mathcal{L}_1, \ldots, \mathcal{L}_n \) if and only if \( S \models \mathcal{L}_i \) for all \( 1 \leq i \leq n \).
One quite effective approach: try to **approximate** the requirements with positive and negative **scenarios**.

- Dear customer, please describe example usages of the desired system.
  
  “If the system is not at all able to do this, then it’s not what I want.”

- Dear customer, please describe behaviour that the desired system must not show.
  
  “If the system does this, then it’s not what I want.”

- From there on, refine and generalise: what about exceptional cases? what about corner-cases? etc.
Example: Buy A Softdrink

- LSC: buy softdrink
- AC: true
- AM: invariant I: permissive

User → Vend. Ma. → E1 → SPOT → SOFT
Example: Get Change

LSC: get change
AC: true
AM: invariant I: permissive

User ———— Vend. Ma.

C50

E1

pSOFT

SOFT

chg-C50
Example: Don’t Give Two Drinks
Example: Don’t Give Two Drinks

LSC: only one drink
AC: true
AM: invariant I: permissive

User

Vend. Ma.

$pSOFT$

$SOFT$

$C50! \land \neg E1!$

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

- **pre-chart** $PC = ((\mathcal{L}_P, \preceq_P, \sim_P), I_P, Msg_P, Cond_P, LocInv_P, \Theta_P)$ (possibly empty),
- **main-chart** $MC = ((\mathcal{L}_M, \preceq_M, \sim_M), I_M, Msg_M, Cond_M, LocInv_M, \Theta_M)$ (non-empty),
- **activation condition** $ac \in \Phi(C)$, **strictness flag** strict (otherwise called permissive)
- **activation mode** $am \in \{\text{initial, invariant}\}$,
- **chart mode** existential ($\Theta_\mathcal{L} = \text{cold}$) or universal ($\Theta_\mathcal{L} = \text{hot}$).
Pre-Charts Semantics

LSC: only one drink
AC: true
AM: invariant I: permissive

<table>
<thead>
<tr>
<th>ΘLLL</th>
<th>am = initial</th>
<th>am = invariant</th>
</tr>
</thead>
</table>
| cold  | \( \exists w \in W \exists m \in \mathbb{N}_0 \cdot w^0 \models ac \)  \\
|       | \( \land w^0 \models \psi_{hot}^{\text{Cond}}(\emptyset, C_0^P) \)  \\
|       | \( \land w/1, \ldots, w/m \in \text{Lang}(\mathcal{B}(PC)) \)  \\
|       | \( \land w^{m+1} \models \psi_{hot}^{\text{Cond}}(\emptyset, C_0^M) \)  \\
|       | \( \land w/m + 1 \in \text{Lang}(\mathcal{B}(MC)) \) | \( \exists w \in W \exists k < m \in \mathbb{N}_0 \cdot w^k \models ac \)  \\
|       | \( \land w^k \models \psi_{hot}^{\text{Cond}}(\emptyset, C_0^P) \)  \\
|       | \( \land w/k + 1, \ldots, w/m \in \text{Lang}(\mathcal{B}(PC)) \)  \\
|       | \( \land w^{m+1} \models \psi_{hot}^{\text{Cond}}(\emptyset, C_0^M) \)  \\
|       | \( \land w/m + 1 \in \text{Lang}(\mathcal{B}(MC)) \) |
| hot   | \( \forall w \in W \cdot w^0 \models ac \)  \\
|       | \( \land w^0 \models \psi_{hot}^{\text{Cond}}(\emptyset, C_0^P) \)  \\
|       | \( \land w/1, \ldots, w/m \in \text{Lang}(\mathcal{B}(PC)) \)  \\
|       | \( \land w^{m+1} \models \psi_{cold}^{\text{Cond}}(\emptyset, C_0^M) \)  \\
|       | \( \implies w^{m+1} \models \psi_{cold}^{\text{Cond}}(\emptyset, C_0^M) \)  \\
|       | \( \land w/m + 1 \in \text{Lang}(\mathcal{B}(MC)) \) | \( \forall w \in W \forall k \leq m \in \mathbb{N}_0 \cdot w^k \models ac \)  \\
|       | \( \land w^k \models \psi_{hot}^{\text{Cond}}(\emptyset, C_0^P) \)  \\
|       | \( \land w/k + 1, \ldots, w/m \in \text{Lang}(\mathcal{B}(PC)) \)  \\
|       | \( \land w^{m+1} \models \psi_{cold}^{\text{Cond}}(\emptyset, C_0^M) \)  \\
|       | \( \implies w^{m+1} \models \psi_{cold}^{\text{Cond}}(\emptyset, C_0^M) \)  \\
|       | \( \land w/m + 1 \in \text{Lang}(\mathcal{B}(MC)) \) |
Note: Scenarios 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!
Note: Scenarios 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.)
Strengthening Scenarios Into Requirements

\[(\Sigma \times A)^\omega\]

Customer

requirements analysis

Analyst
Strengthening Scenarios Into Requirements

Customer requirements analysis

Analyst requirements analysis
Universal LSC: Example

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

User — C50 — CoinValidator

CoinValidator — pWATER — water_in_stock — Dispenser

ChoicePanel

Dispenser — dWATER — OK
Universal LSC: Example

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

User ─── CoinValidator ─── ChoicePanel ─── Dispenser

¬(C50! ∨ E1! ∨ pSOFT! ∨ pTEA! ∨ pFILLUP!)

water_in_stock

C50

pWATER
dWATER

OK
Universal LSC: Example

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

User \rightarrow CoinValidator \rightarrow ChoicePanel \rightarrow Dispenser

\neg (C50 \lor E1 \lor pSOFT \lor pTEA \lor pFILLUP)

\neg (dSoft \lor dTEA)

water_in_stock

pWATER

C50

dWATER

OK
Shortcut: Forbidden Elements

\[
\neg (C50 \lor E1 \lor p\text{SOFT}! \lor p\text{TEA}! \lor p\text{FILLUP}!)
\]

\[
\neg (d\text{Soft}! \lor d\text{TEA}!)
\]
Modelling Idiom: Enforcing Order

LSC: \( L \)
AM: invariant \( I \): permissive

\[ I_1 \xrightarrow{E} I_2 \xrightarrow{E} I_3 \xrightarrow{F} I_4 \]

LSC: \( L \)
AM: invariant \( I \): permissive

\[ I_1 \xrightarrow{E} I_2 \xrightarrow{true} I_3 \xrightarrow{F} I_4 \]
A **requirements specification** should be

- **correct**
  - it correctly represents the wishes/needs of the customer,

- **complete**
  - all requirements (existing in somebody’s head, or a document, or . . . ) should be present,

- **relevant**
  - things which are not relevant to the project should not be constrained,

- **consistent, free of contradictions**
  - each requirement is compatible with all other requirements; otherwise the requirements are **not realisable**,  

- **neutral, abstract**
  - a requirements specification does not constrain the realisation more than necessary,

- **traceable, comprehensible**
  - the sources of requirements are documented, requirements are uniquely identifiable,

- **testable, objective**
  - the final product can **objectively** be checked for satisfying a requirement.
Requirements on LSC Specifications

- **correctness** is relative to “in the head of the customer” → still difficult;

- **complete**: we can at least define a kind of relative completeness in the sense of “did we cover all (exceptional) cases?”;

- **relevant** also not analyseable within LSCs;

- **consistency** can formally be analysed!

- **neutral/abstract** is relative to the realisation → still difficult;  
  But LSCs tend to support abstract specifications; specifying technical details is tedious.

- **traceable/comprehensible** are meta-properties, need to be established separately;

- a formal requirements specification, e.g. using LSCs, is immediately **objective/testable**.

For Decision Tables, we formally defined additional quality criteria:

- **uselessness/vacuity**,  

- **determinism** may be desired,  

- **consistency** wrt. domain model.

What about LSCs?
LSCs vs. MSCs
**LSCs vs. MSCs**

**Recall:** Most severe **drawbacks** of, e.g., MSCs:

- unclear **interpretation**: example scenario or invariant?
- unclear **activation**: what triggers the requirement?
- unclear **progress** requirement: must all messages be observed?
- **conditions** merely comments
- no means (in language) to express **forbidden scenarios**
Pushing It Even Further

(Harel and Marelly, 2003)
Requirements Engineering Wrap-Up
Recall: Software Specification Example

Alphabet:

- $M$ – dispense cash only,
- $C$ – return card only,
- $M$ $C$ – dispense cash and return card.

- **Customer 1** “don’t care”

\[
(M.C | C.M | \begin{array}{c} M \\ C \end{array})
\]

- **Customer 2** “you choose, but be consistent”

\[(M.C) \text{ or } (C.M)\]

- **Customer 3** “consider human errors”

\[(C.M)\]
Recall: Formal Software Development

Mmmh, Software!

\[ [\mathcal{S}_1] = \{(M.C, [\cdot]_1), (C.M, [\cdot]_1)\} \]

\[ [\mathcal{S}_2] = \{(M.T_{M.C}, [\cdot]_1), (C.T_{C.M}, [\cdot]_1)\} \]

\[ [S] = \{\sigma_0 \stackrel{\tau}{\rightarrow} \sigma_1 \stackrel{\tau}{\rightarrow} \sigma_2 \cdots, \ldots\} \]

Development Process/Project Management
Recall: Formal Software Development

Mmmh, Software!

\[ \mathcal{J}_1 = \{ (M.C, \cdot)_1, (C.M, \cdot)_1 \} \]

Requirements

- Elicit
- Formalise
- Verify
- Fix
- Validate

Development Process/Project Management

elicit req.(in.)

formalise req.(fo.)

verify req.(fo.)

fix req.(fo.)

validate req.(in.)

✔/✘

analyst
customer

analyst

analyst
customer

analyst
customer

analyst
customer

analyst
customer
One sometimes distinguishes:

- **Systems Engineering** (develop software for an embedded controller)
  Requirements typically stated in terms of *system observables* ("press WATER button"), needs to be mapped to terms of the software!

- **Software Engineering** (develop software which interacts with other software)
  Requirements stated in terms of the software.

We touched a bit of both, aimed at a general discussion.

- **Once again** (can it be mentioned too often?):
  Distinguish *domain elements* and *software elements* and (try to) keep them apart to avoid confusion.
Lehmann (Lehman, 1980; Lehman and Ramil, 2001) distinguishes three classes of software (my interpretation, my examples):

- **S-programs**: solve mathematical, abstract problems; can exactly (in particular formally) be specified; tend to be small; can be developed once and for all.
  
  **Examples**: sorting, compiler (!), compute $\pi$ or $\sqrt{\cdot}$, cryptography, textbook examples, …

- **P-programs**: solve problems in the real world, e.g. read sensors and drive actors, may be in feedback loop; specification needs domain model (cf. Bjørner (2006), “A tryptic software development paradigm”); formal specification (today) possible, in terms of domain model, yet tends to be expensive
  
  **Examples**: cruise control, autopilot, traffic lights controller, plant automatisation, …

- **E-programs**: embedded in socio-technical systems; in particular involve humans; specification often not clear, not even known; can grow huge; delivering the software induces new needs
  
  **Examples**: basically everything else; word processor, web-shop, game, smart-phone apps, …
(Rupp and die SOPHISTen, 2014)
References
References


