Softwaretechnik / Software-Engineering

Lecture 9: Live Sequence Charts

2016-06-06

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

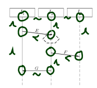
Albert-Ludwigs-Universität Freiburg, Germany

Topic Area Requirements Engineering: Content

Content

LSC Semantics

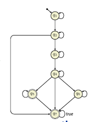
The Plan: A Formal Semantics for a Visual Formalism



concrete syntax
(diagram)

Locations (ℓ, E, ℓ')

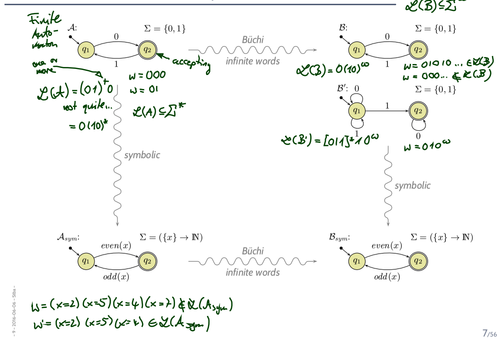
$((\mathcal{L}, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta)$
abstract syntax



semantics
(Büchi automaton)

Excursion: Symbolic Büchi Automata

From Finite Automata to Symbolic Büchi Automata



Symbolic Büchi Automata

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

$$\mathcal{B} = (\mathcal{C}_\mathcal{B}, Q, q_{ini}, \to, Q_F)$$

where

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

Run of TBA

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

$$w = \sigma_1, \sigma_2, \sigma_3, \dots \in (\Phi(\mathcal{C}_\mathcal{B}) \to \mathbb{B})^\omega$$

an infinite word, each letter is a valuation of $\Phi(\mathcal{C}_\mathcal{B})$.

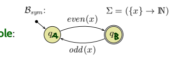
An infinite sequence

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

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 $(q_i, \psi_i, q_{i+1}) \in \to$ s.t. $\sigma_i \models \psi_i$.

Example:



The Language of a TBA

Definition.
We say TBA $\mathcal{B} = (\mathcal{C}_\mathcal{B}, Q, q_{ini}, \to, Q_F)$ **accepts** the word

$$w = (\sigma_i)_{i \in \mathbb{N}_0} \in (\Phi(\mathcal{C}_\mathcal{B}) \to \mathbb{B})^\omega$$

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

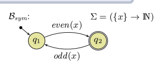
$$\varrho = (q_i)_{i \in \mathbb{N}_0}$$

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

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

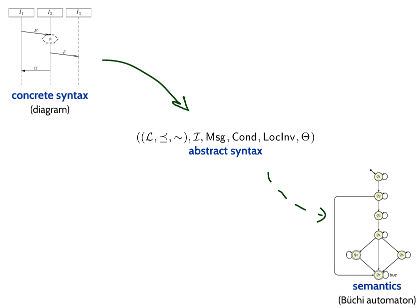
We call the set $Lang(\mathcal{B}) \subseteq (\Phi(\mathcal{C}_\mathcal{B}) \to \mathbb{B})^\omega$ of words that are accepted by $\mathcal{B}$ the **language of** $\mathcal{B}$.

Example:



LSC Semantics: TBA Construction

The Plan: A Formal Semantics for a Visual Formalism



LSC Semantics: It's in the Cuts!

Definition. Let $((\mathcal{L}, \preceq, \sim), \mathcal{I}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv}, \Theta)$ 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.
$$\forall I \in \mathcal{I} \bullet C \cap I \neq \emptyset.$$

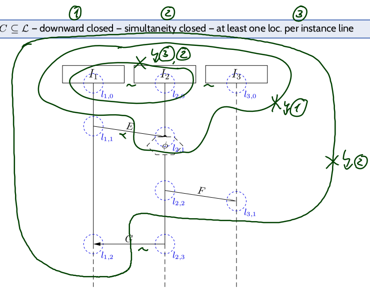
The temperature function is extended to cuts as follows:

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

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

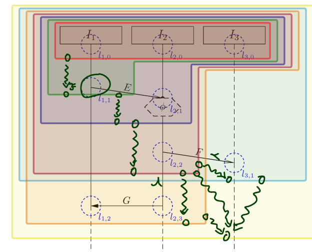
Cut Examples

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



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

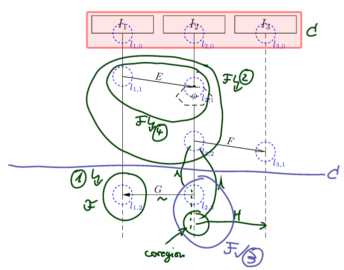
A set $\emptyset \neq \mathcal{F} \subseteq \mathcal{L}$ of locations is called fired-set $\mathcal{F}$ of cut C if and only if

- $C \cap \mathcal{F} = \emptyset$ 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,
$$\forall (l, E, l') \in \mathsf{Msg} \bullet l' \in \mathcal{F} \implies l \in C.$$

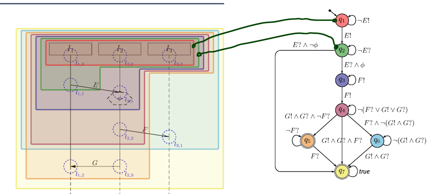
The cut $C' = C \cup \mathcal{F}$ is called direct successor of C via $\mathcal{F}$, denoted by $C \rightsquigarrow_{\mathcal{F}} C'$.

Successor Cut Example

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



Language of LSC Body: Example



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

- $\mathcal{C}_{\mathcal{B}} = \mathcal{C} \,\dot\cup\, \mathcal{E}_{!?}$, where $\mathcal{E}_{!?} = \{E!, E? \mid E \in \mathcal{E}\}$,
- Q is **the set of cuts** of $\mathscr{L}$, q_{ini} is the **instance heads** cut,
- $\rightarrow$ consists of loops, progress transitions (from $\rightsquigarrow_{\mathcal{F}}$), 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.

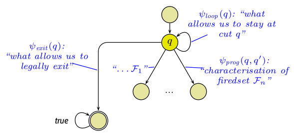
TBA Construction Principle

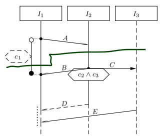
Recall: The TBA $\mathcal{B}(\mathscr{L})$ of LSC $\mathscr{L}$ is $(\mathcal{C}, Q, q_{ini}, \to, Q_F)$ with

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

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

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





TBA Construction Principle

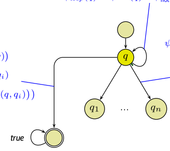
"Only" construct the transitions' labels:

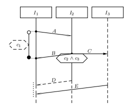
$$\to = \{(q, \psi_{loop}(q), q) \mid q \in Q\} \cup \{(q, \psi_{prog}(q, q'), q') \mid q \leadsto_\mathcal{F} q'\} \cup \{(q, \psi_{exit}(q), \mathcal{L}) \mid q \in Q\}$$

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

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

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





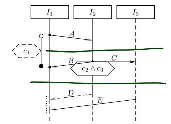
Loop Condition

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

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

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

  Local invariant $(l_o, \iota_0, \phi, l_1, \iota_1)$ is **active** at cut (!) q
  if and only if $l_0 \preceq l \prec l_1$ for some front location l of cut q or $l = l_1 \wedge \iota_1 = \bullet$.
- $\mathsf{Msg}(\mathcal{F}) = \{E! \mid (l, E, l') \in \mathsf{Msg},\ l \in \mathcal{F}\} \cup \{E? \mid (l, E, l') \in \mathsf{Msg},\ l' \in \mathcal{F}\}$
- $\mathsf{Msg}(\mathcal{F}_1, \ldots, \mathcal{F}_n) = \bigcup_{1 \le i \le n} \mathsf{Msg}(\mathcal{F}_i)$



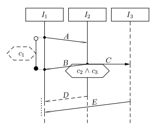
Progress Condition

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

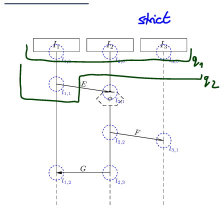
- $\psi^{\mathsf{Msg}}(q, q_i) = \bigwedge_{\psi \in \mathsf{Msg}(q_i \setminus q)} \psi \wedge \bigwedge_{j \ne i} \bigwedge_{\psi \in (\mathsf{Msg}(q_j \setminus q) \setminus \mathsf{Msg}(q_i \setminus q))} \neg\psi$
  $\wedge \underbrace{\left(strict \implies \bigwedge_{\psi \in (\mathcal{E}_{!?}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

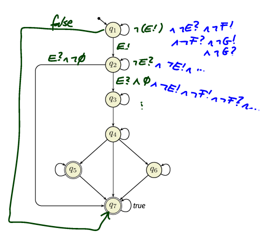
  Local invariant $(l_0, \iota_0, \phi, l_1, \iota_1)$ is **$\bullet$-active** at q if and only if
  - $l_0 \prec l \prec l_1$, or
  - $l = l_0 \wedge \iota_0 = \bullet$, or
  - $l = l_1 \wedge \iota_1 = \bullet$

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



Example



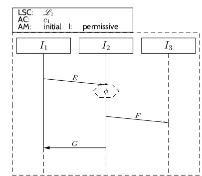


Full LSCs

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

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

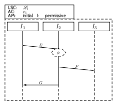
Concrete syntax:



Full LSCs

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

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

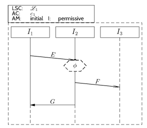


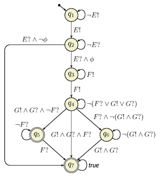
A **set of words** $W \subseteq (\mathcal{C} \to \mathbb{B})^\omega$ is **accepted** by $\mathscr{L}$ if and only if

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

where $ac = ac_0 \wedge \psi_{\mathsf{cold}}^{\mathsf{Cond}}(\emptyset, C_0) \wedge \psi^{\mathsf{Msg}}(\emptyset, C_0)$; C_0 is the minimal (or **instance heads**) cut.

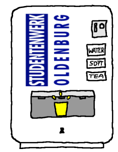
Full LSC Semantics: Example



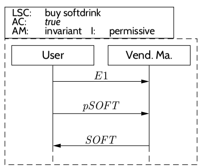


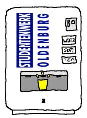
Example: Vending Machine

- **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.

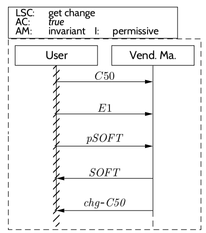


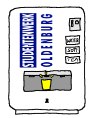
Example: Buy A Softdrink



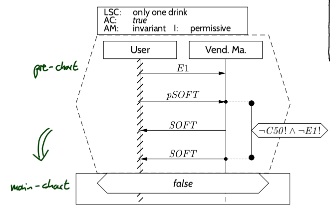


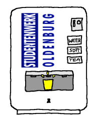
Example: Get Change



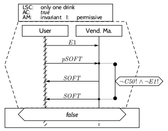


Anti-Scenarios: Don't Give Two Drinks





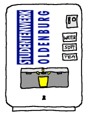
Pre-Charts

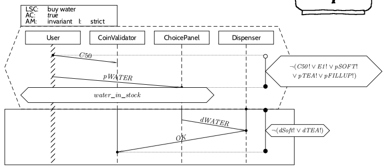


A **full LSC** $\mathscr{L} = (PC, MC, ac_0, am, \Theta_{\mathscr{L}})$ **actually** consist of

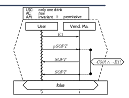
- **pre-chart** $PC = ((\mathcal{L}_P, \preceq_P, \sim_P), \mathcal{I}_P, \mathsf{Msg}_P, \mathsf{Cond}_P, \mathsf{LocInv}_P, \Theta_P)$ (possibly empty),
- **main-chart** $MC = ((\mathcal{L}_M, \preceq_M, \sim_M), \mathcal{I}_M, \mathsf{Msg}_M, \mathsf{Cond}_M, \mathsf{LocInv}_M, \Theta_M)$ (non-empty),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

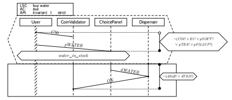
Universal LSC: Example





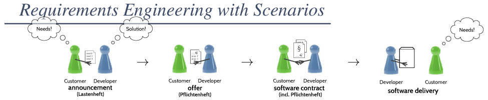
Pre-Charts Semantics





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

Requirements Engineering with Scenarios



One quite effective approach:

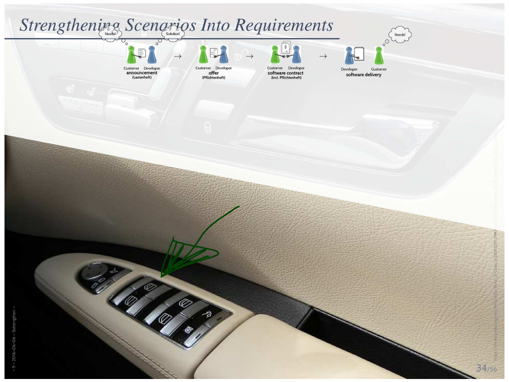
(i) **Approximate** the software requirements: ask for positive / negative **existential scenarios**.

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

That is:

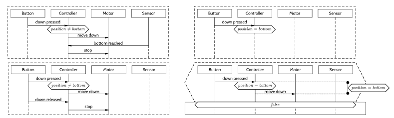
- Ask the customer to describe **example usages** of the desired system.
  In the sense of: "If the system is not at all able to do this, then it's not what I want."
  (→ positive use-cases, existential LSC)
- Ask the customer to describe behaviour that **must not happen** in the desired system.
  In 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

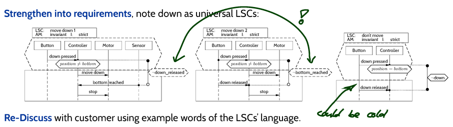


Strengthening Scenarios Into Requirements

- **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

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

Ludewig, J. and Lichter, H. (2013). *Software Engineering*. dpunkt.verlag, 3. edition.

Rupp, C. and die SOPHISTen (2014). *Requirements-Engineering und -Management*. Hanser, 6th edition.