# *Software Design, Modelling and Analysis in UML*

# *Lecture 21: Model-based Software Development*

*2017-02-06*

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

Albert-Ludwigs-Universität Freiburg, Germany

# Content

- Live Sequence Charts
  - Semantics
    - Full LSCs
      - Existential and Universal
      - Pre-Charts
      - Forbidden Scenarios
    - LSCs and Tests

- Model-Based/-Driven Software Engineering
  - Model Element Coverage of test cases
  - Model-based Testing

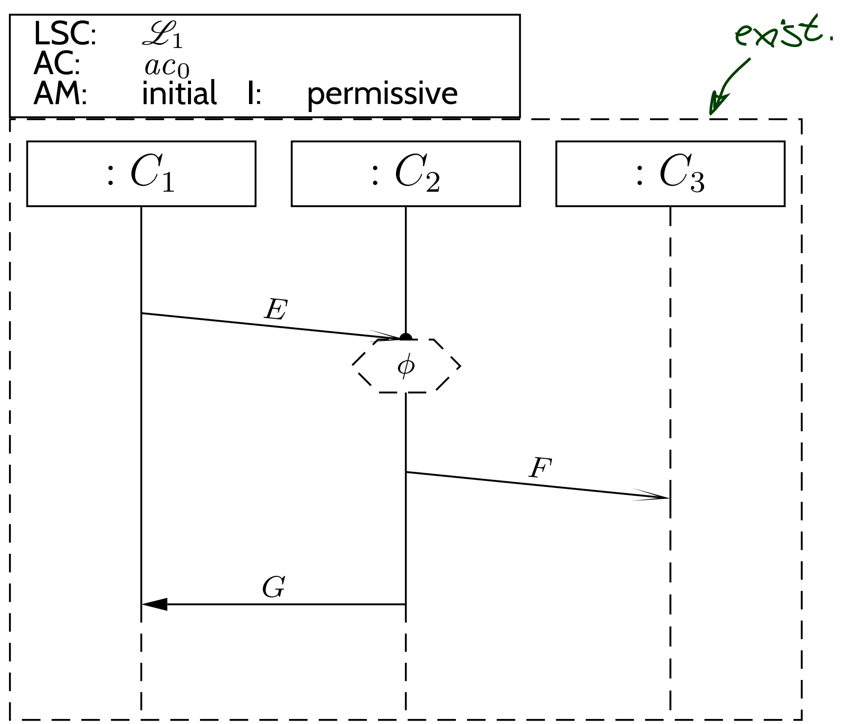
# *Live Sequence Charts — Full LSC Semantics*

# *Full LSCs*

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

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

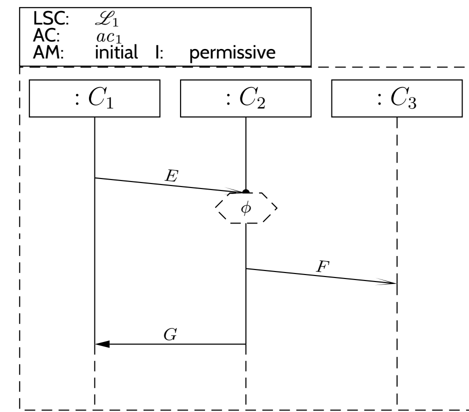
**Concrete syntax:**



# *Full LSCs*

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

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



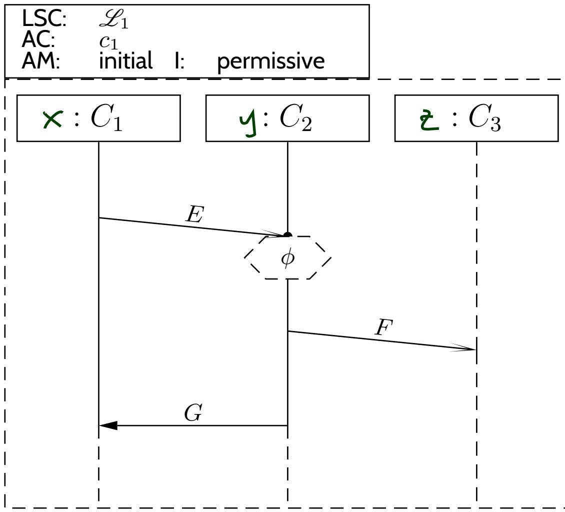

A **set of words** $W \subseteq (Expr_{\mathcal{B}} \to \mathbb{B})^\omega$ is **accepted** by $\mathscr{L}$ if and only if

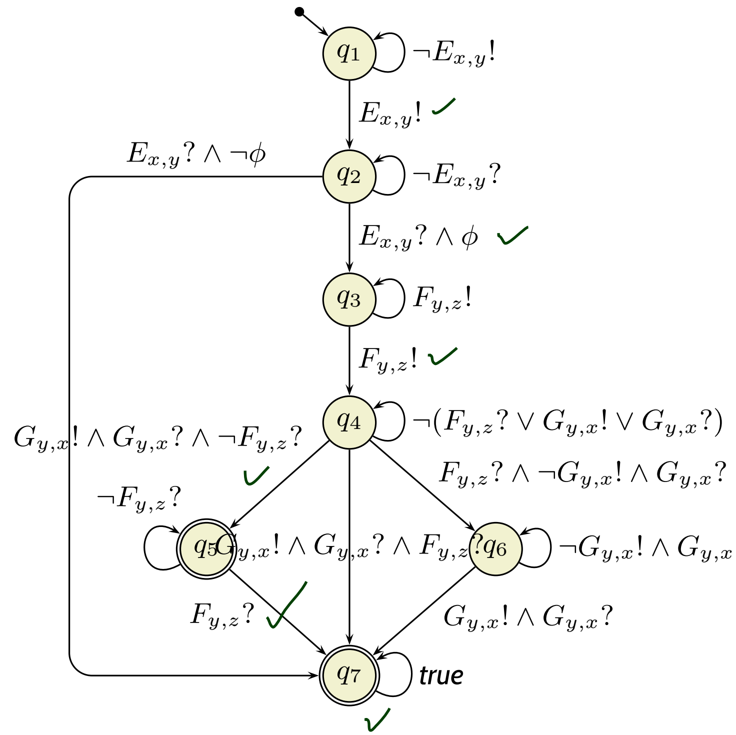
suffix of w starting at index 1

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

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

# *Full LSC Semantics: Example*




w:

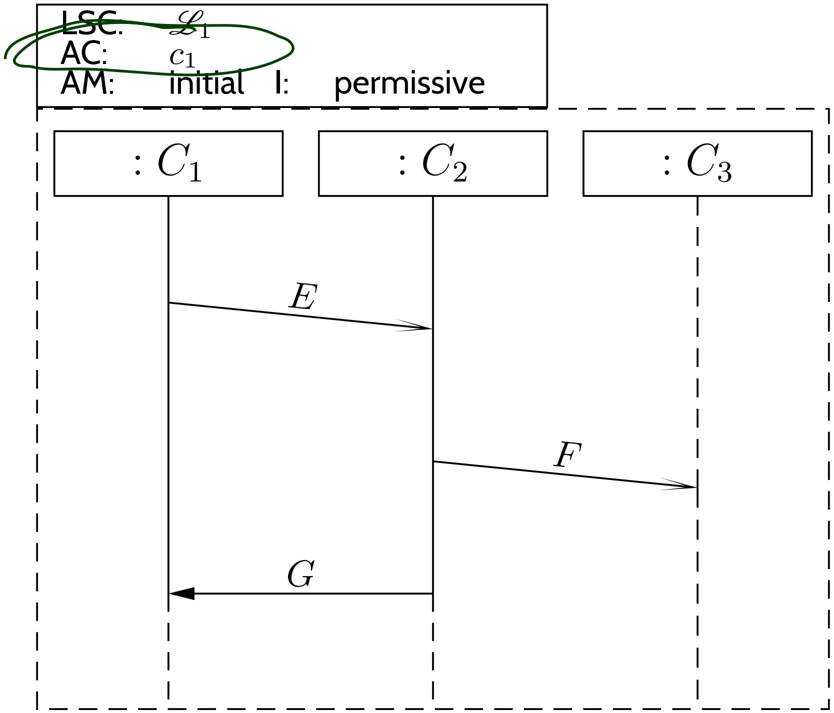
$$(\sigma,\varepsilon) \xrightarrow[u]{(cons,Snd)} \cdots \rightarrow (\sigma_0,\varepsilon_0) \xrightarrow[u_0]{(cons_0,Snd_0)} (\sigma_1,\varepsilon_1) \xrightarrow[c_1]{(cons_1,\{(:E,c_2)\})} (\sigma_2,\varepsilon_2) \xrightarrow[c_2]{(\{:E\},Snd_2)}$$

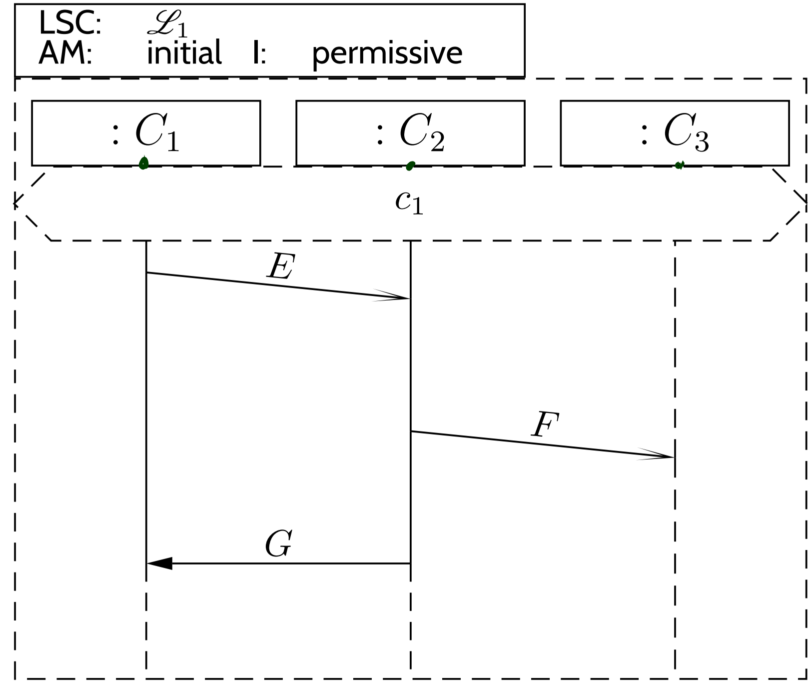
$$(\sigma_3,\varepsilon_3) \xrightarrow[c_2]{(cons_3,\{(:F,c_3)\})} (\sigma_4,\varepsilon_4) \xrightarrow[c_2]{(cons_4,\{(G(),c_1)\})} (\sigma_5,\varepsilon_5) \xrightarrow[c_3]{(\{:F\},Snd_5)} (\sigma_6,\varepsilon_6) \rightarrow \cdots$$

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

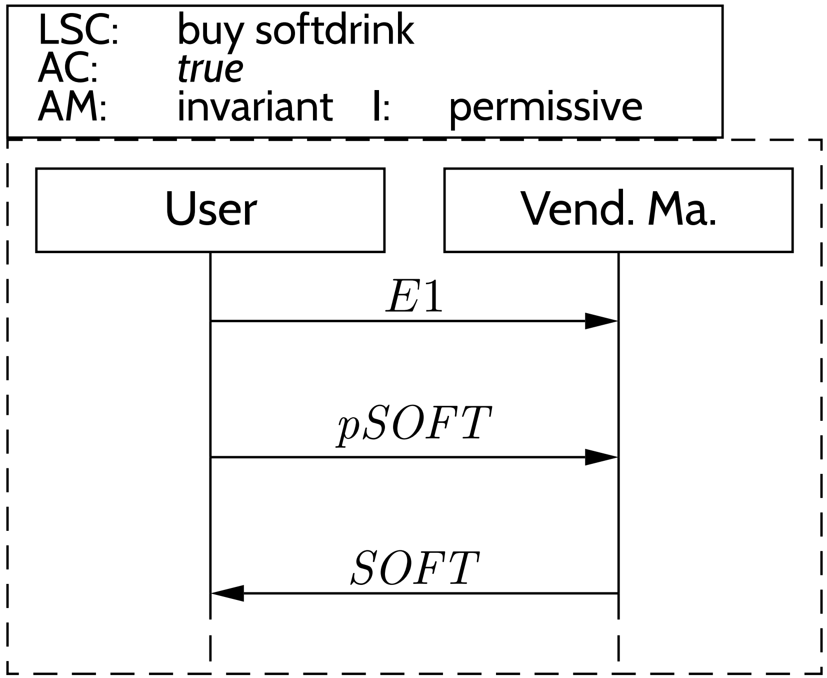
$\Rightarrow$ w is accepted by $\mathscr{L}_1$

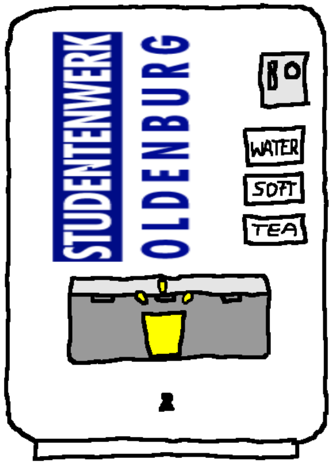
# Note: Activation Condition



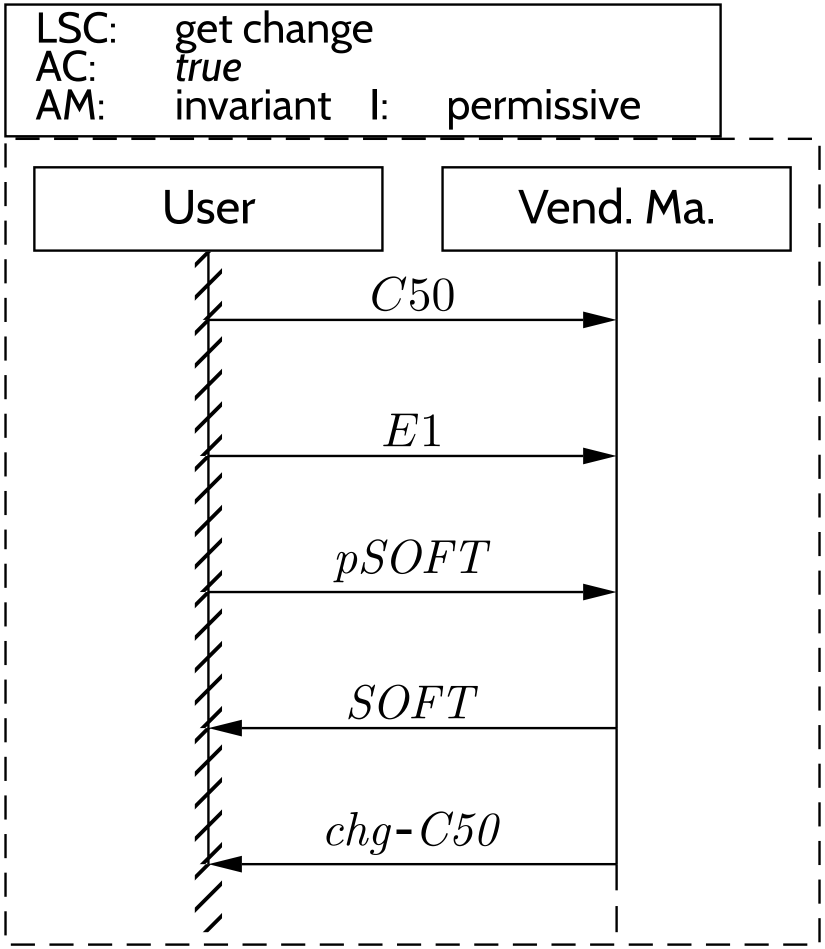



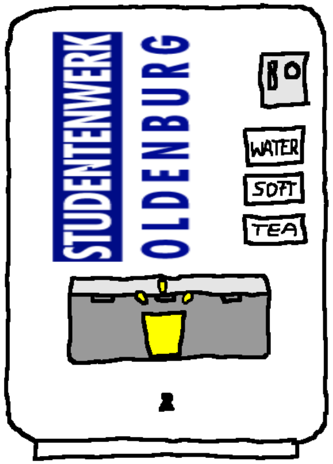
# *Existential LSC Example: Buy A Softdrink*



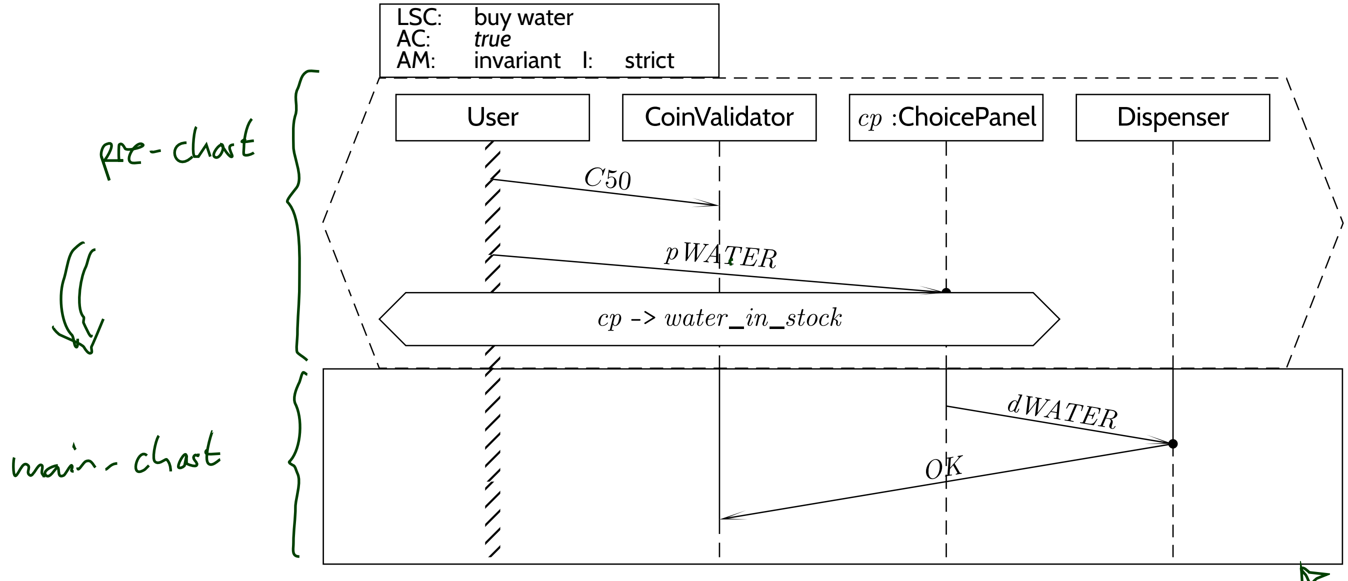
# Existential LSC Example: Get Change



# *Live Sequence Charts — Precharts*

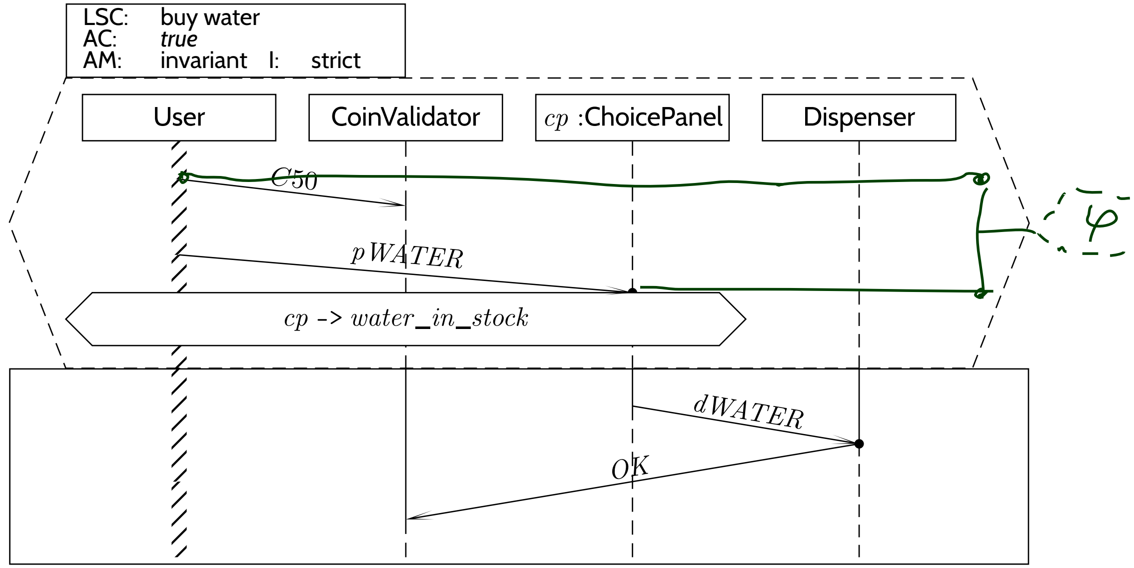
# *Pre-Charts*




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

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

# Pre-Charts Semantics

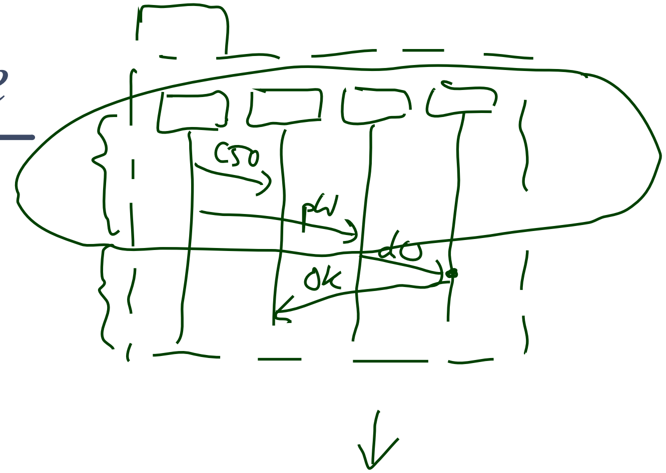


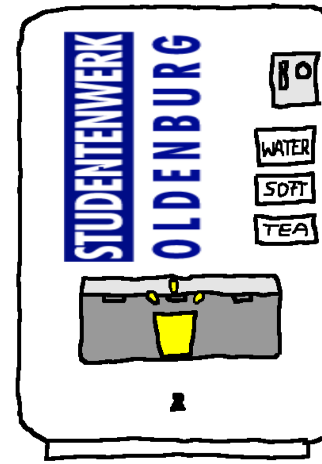

$w = w_0 w_1 \dots w_m w_{m+1} w_{m+2} w_{m+3} \dots$ ($w_0 \models ac$; $w_1 \dots w_m \in \mathcal{L}(\mathcal{B}(PC))$ ⇒ $w_{m+2} w_{m+3} \dots \in \mathcal{L}(\mathcal{B}(MC))$; $w_{m+1} = C_0^M$)

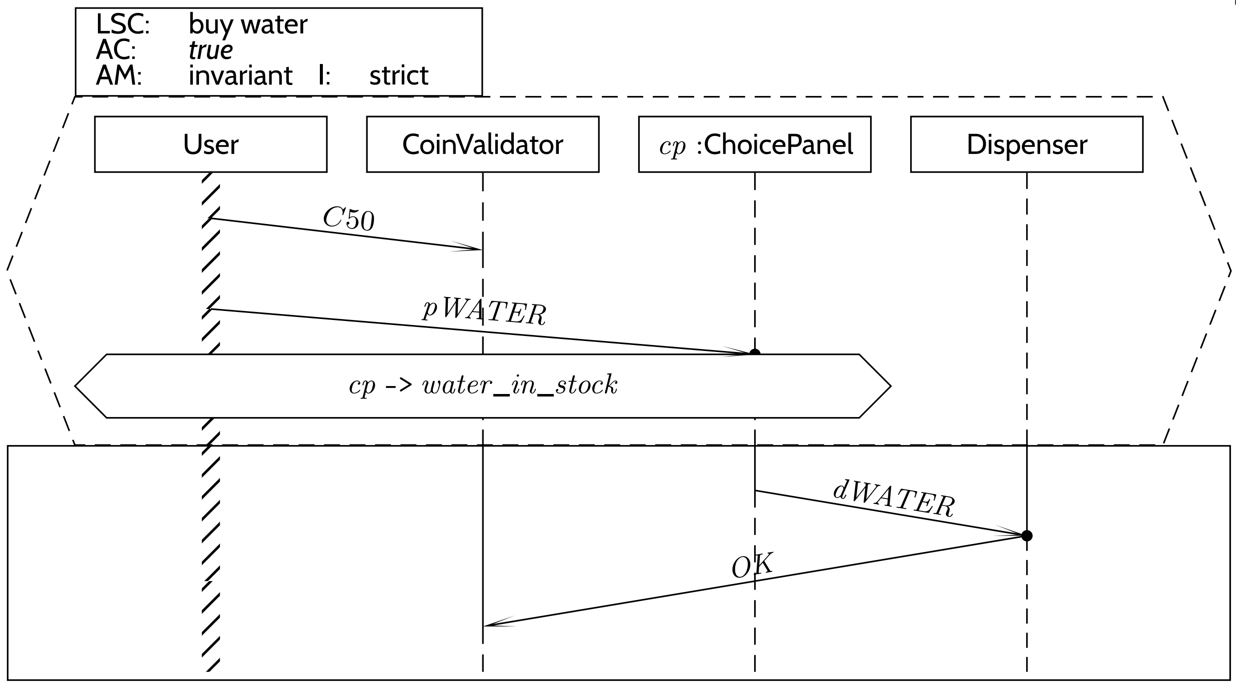
(*) usually: reaching state of maximal cut of pre-chart

| | $am$ = initial | $am$ = invariant |
|---|---|---|
| $\Theta_\mathcal{L}$ = cold | $\exists\, \beta\; \exists\, w \in W\; \exists\, m \in \mathbb{N}_0 \bullet$<br>$\wedge w^0 \models_\beta ac \wedge \neg\psi_{exit}(C_0^P) \wedge \psi_{prog}(\emptyset, C_0^P)$<br>$\wedge\, w^1, \dots, w^m \in \mathcal{L}_\beta(\mathcal{B}(PC))$<br>$\wedge\, w^{m+1} \models_\beta \neg\psi_{exit}(C_0^M)$<br>$\wedge\, w^{m+1} \models_\beta \psi_{prog}(\emptyset, C_0^M)$<br>$\wedge\, w/m+2 \in \mathcal{L}_\beta(\mathcal{B}(MC))$ | $\exists\, \beta\; \exists\, w \in W\; \exists\, k < m \in \mathbb{N}_0 \bullet$<br>$\wedge w^k \models_\beta ac \wedge \neg\psi_{exit}(C_0^P) \wedge \psi_{prog}(\emptyset, C_0^P)$<br>$\wedge\, w^{k+1}, \dots, w^m \in \mathcal{L}_\beta(\mathcal{B}(PC))$<br>$\wedge\, w^{m+1} \models_\beta \neg\psi_{exit}(C_0^M)$<br>$\wedge\, w^{m+1} \models_\beta \psi_{prog}(\emptyset, C_0^M)$<br>$\wedge\, w/m+2 \in \mathcal{L}_\beta(\mathcal{B}(MC))$ |
| $\Theta_\mathcal{L}$ = hot | $\forall\, \beta\; \forall\, w \in W\; \forall\, m \in \mathbb{N}_0 \bullet$<br>$\wedge w^0 \models_\beta ac \wedge \neg\psi_{exit}(C_0^P) \wedge \psi_{prog}(\emptyset, C_0^P)$<br>$\wedge\, w^1, \dots, w^m \in \mathcal{L}_\beta(\mathcal{B}(PC))$ (*)<br>$\wedge\, w^{m+1} \models_\beta \neg\psi_{exit}(C_0^M)$<br>$\implies w^{m+1} \models_\beta \psi_{prog}(\emptyset, C_0^M)$<br>$\wedge\, w/m+2 \in \mathcal{L}_\beta(\mathcal{B}(MC))$ | $\forall\, \beta\; \forall\, w \in W\; \forall\, k \leq m \in \mathbb{N}_0 \bullet$<br>$\wedge w^k \models_\beta ac \wedge \neg\psi_{exit}(C_0^P) \wedge \psi_{prog}(\emptyset, C_0^P)$<br>$\wedge\, w^{k+1}, \dots, w^m \in \mathcal{L}_\beta(\mathcal{B}(PC))$<br>$\wedge\, w^{m+1} \models_\beta \neg\psi_{exit}(C_0^M)$<br>$\implies w^{m+1} \models_\beta \psi_{prog}(\emptyset, C_0^M)$<br>$\wedge\, w/m+2 \in \mathcal{L}_\beta(\mathcal{B}(MC))$ |

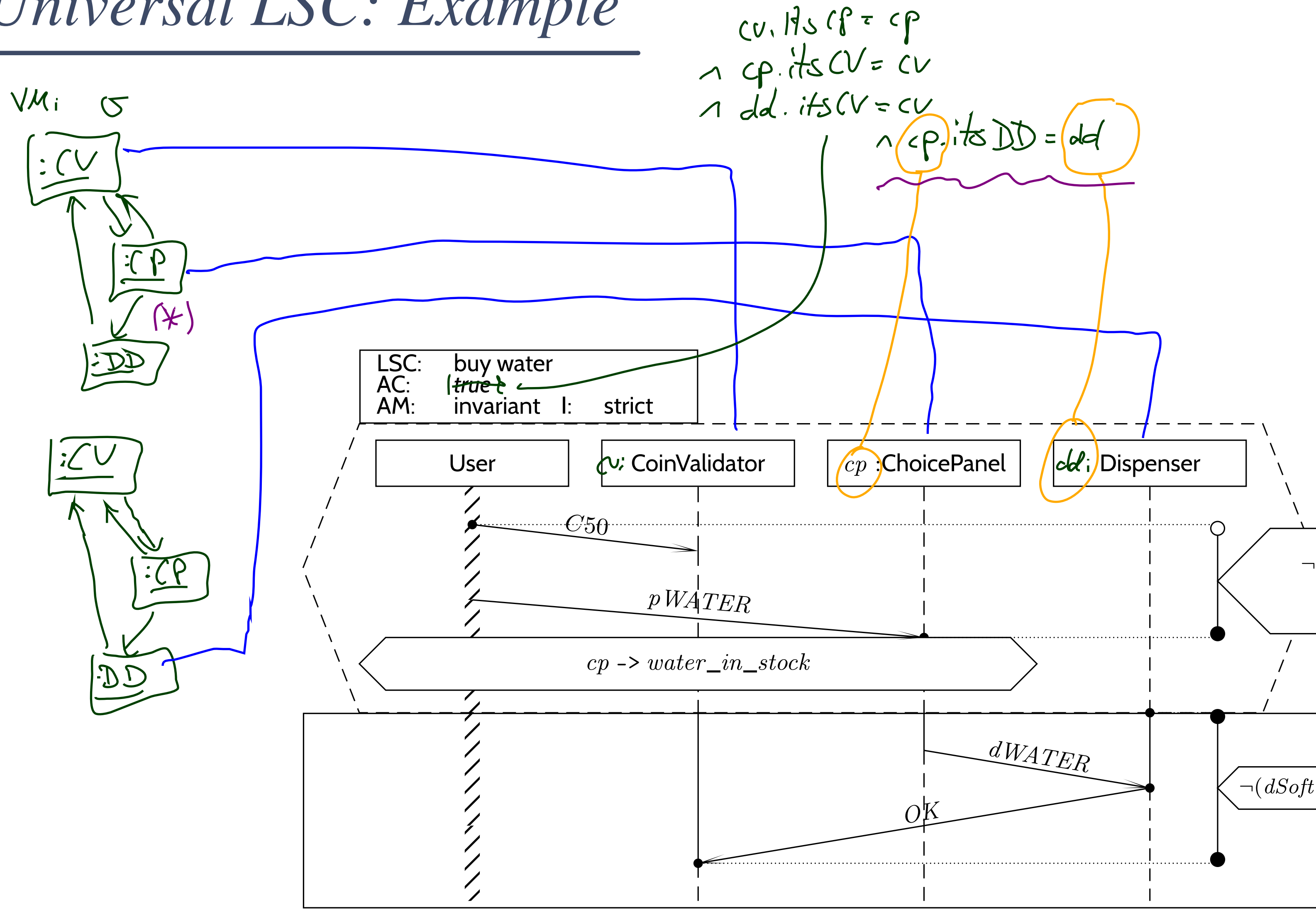
# Universal LSC: Example

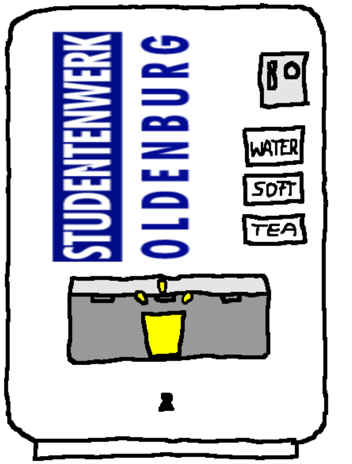


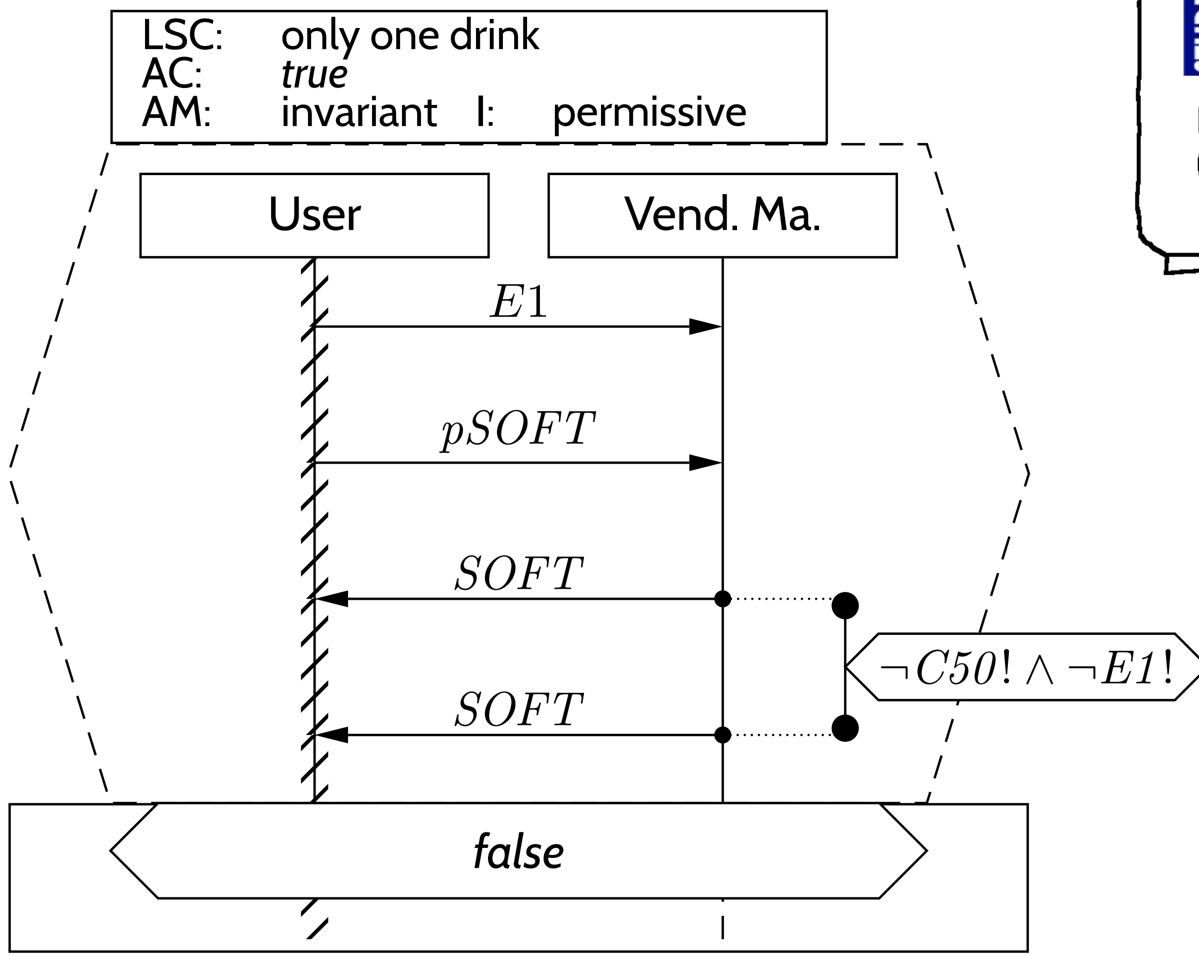



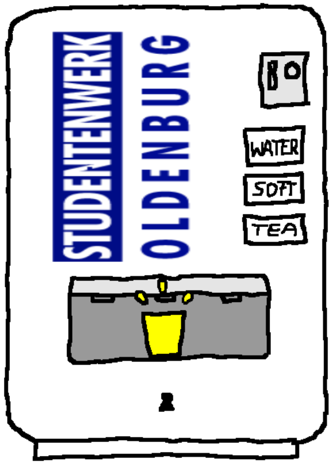
# Universal LSC: Example



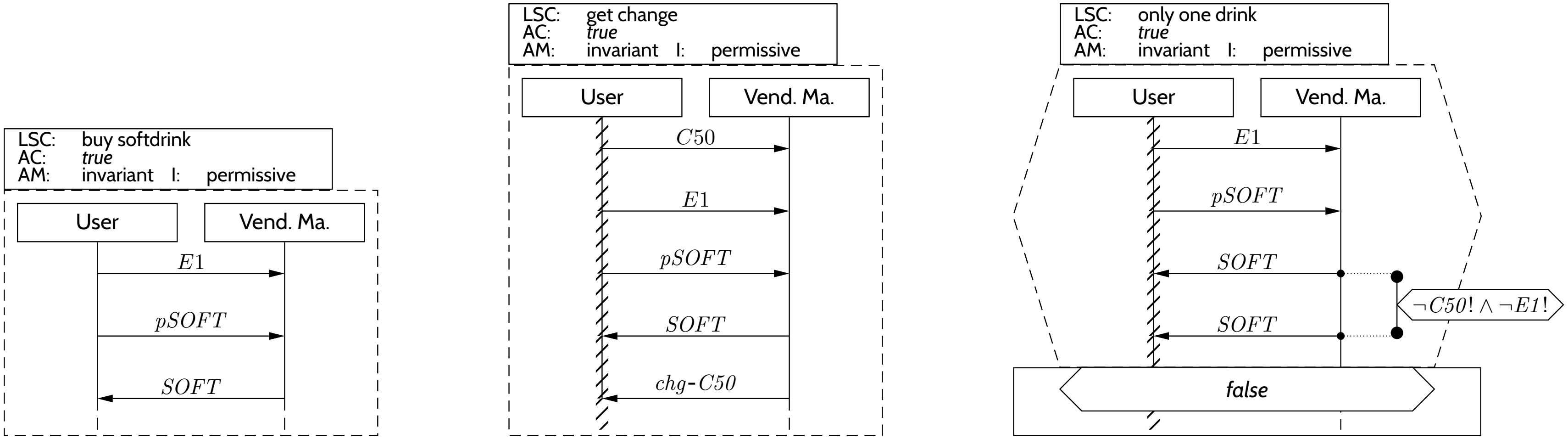



# Forbidden Scenario Example: Don't Give Two Drinks



# *Note: Sequence Diagrams and (Acceptance) Test*



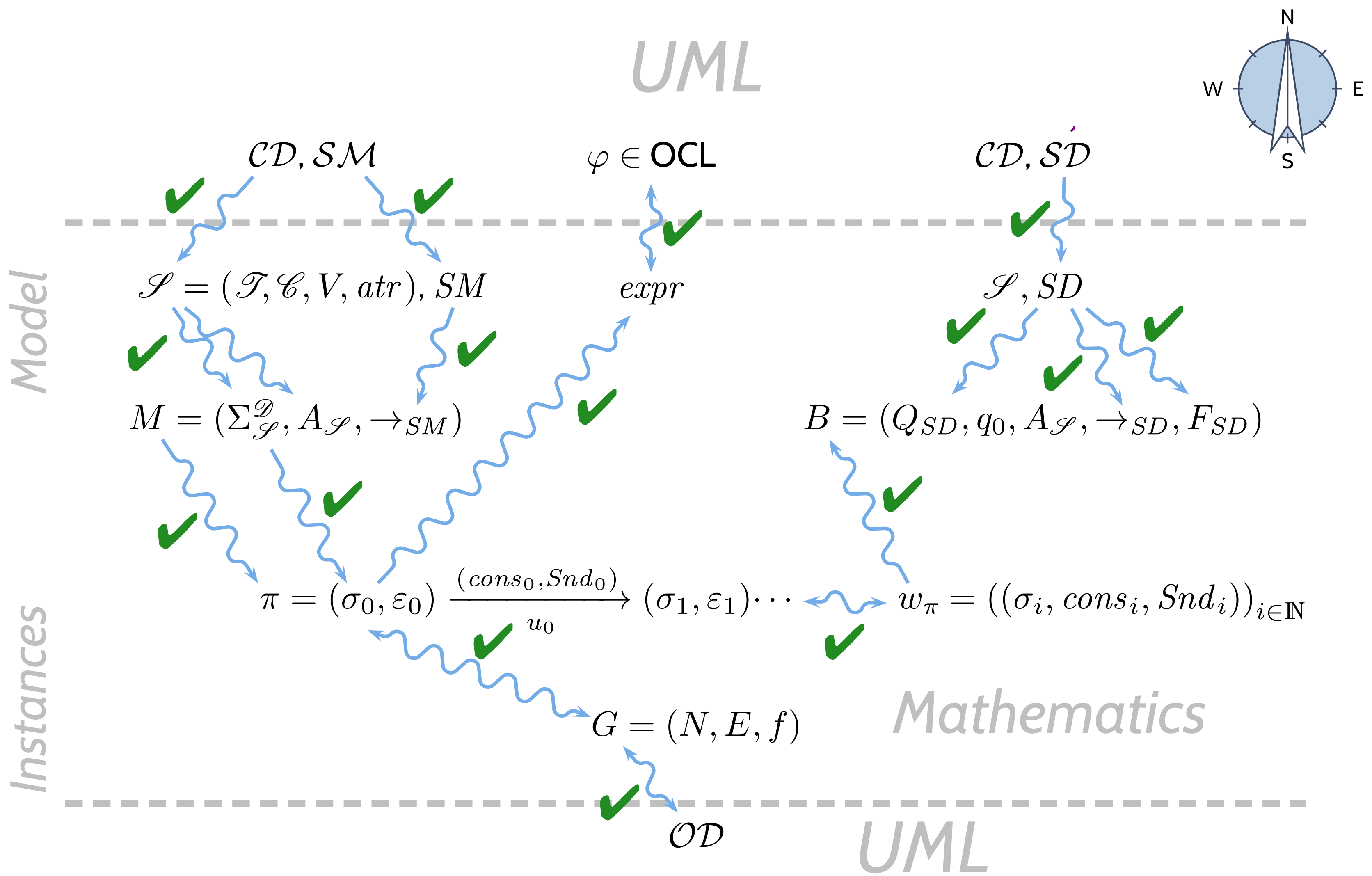

- **Existential** LSCs* may hint at **test-cases** for the **acceptance test**!

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

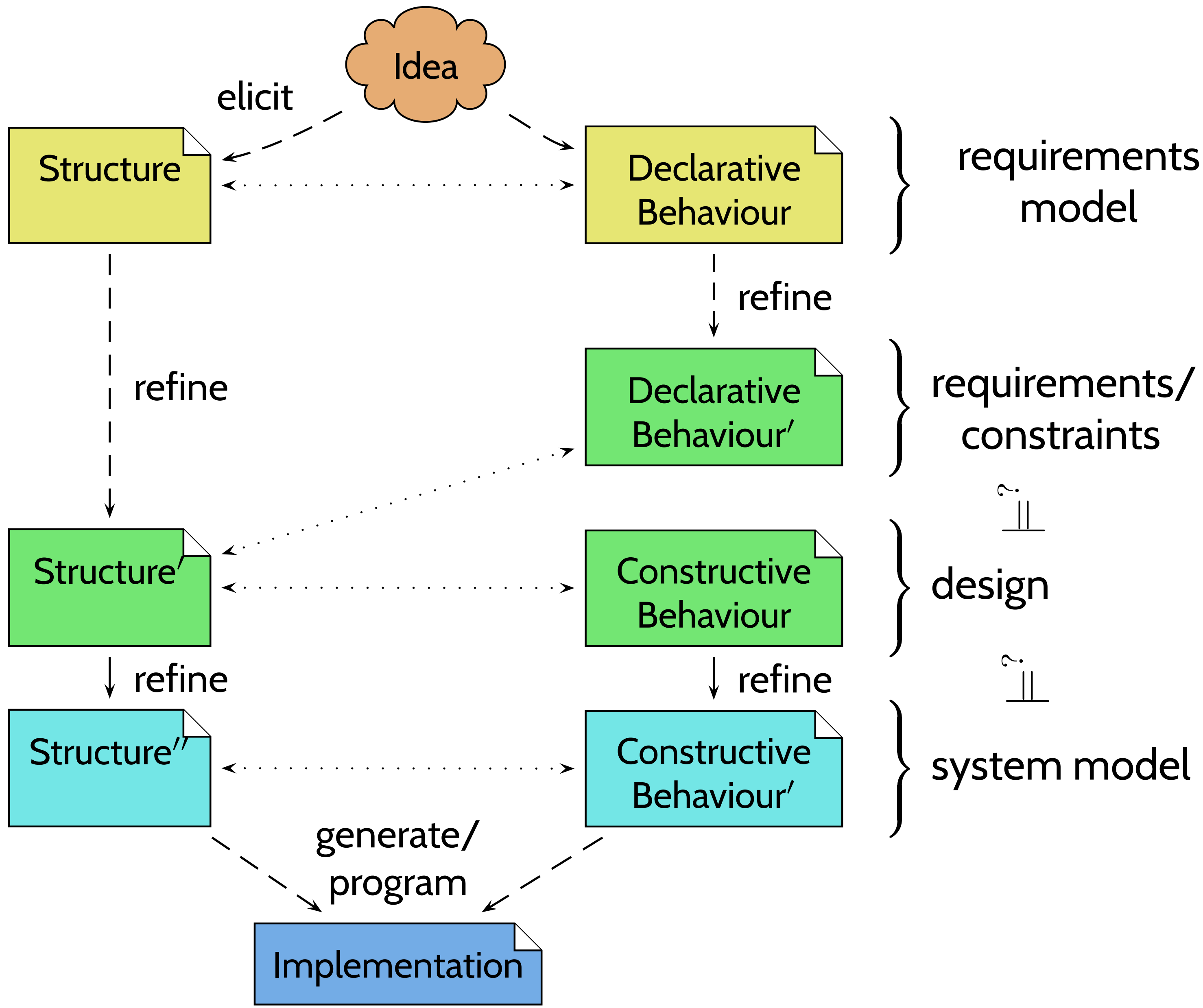
- **Universal** LSCs (and negative/anti-scenarios) in general need **exhaustive analysis**!

  (Because they require that the software never ever exhibits the unwanted behaviour.)

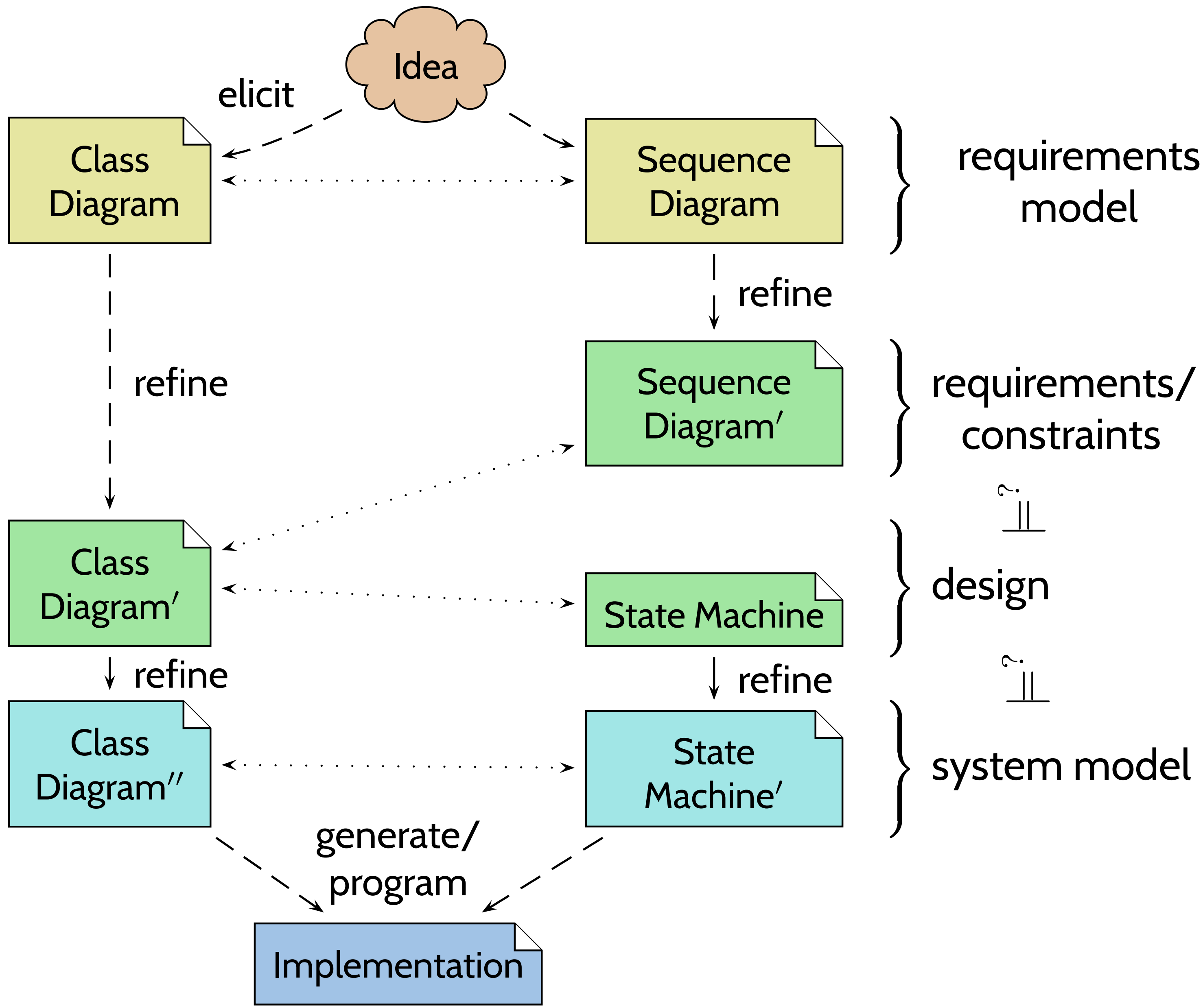
# Course Map


UML
$\mathcal{CD}, \mathcal{SM}$
$\varphi \in$ OCL
$\mathcal{CD}, \mathcal{SD}$
N
W
E
S
Model
$\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr), SM$
$expr$
$\mathscr{S}, SD$
$M = (\Sigma_{\mathscr{S}}^{\mathscr{D}}, A_{\mathscr{S}}, \to_{SM})$
$B = (Q_{SD}, q_0, A_{\mathscr{S}}, \to_{SD}, F_{SD})$
Instances
$\pi = (\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \cdots$
$w_\pi = ((\sigma_i, cons_i, Snd_i))_{i \in \mathbb{N}}$
Mathematics
$G = (N, E, f)$
$\mathcal{OD}$
UML

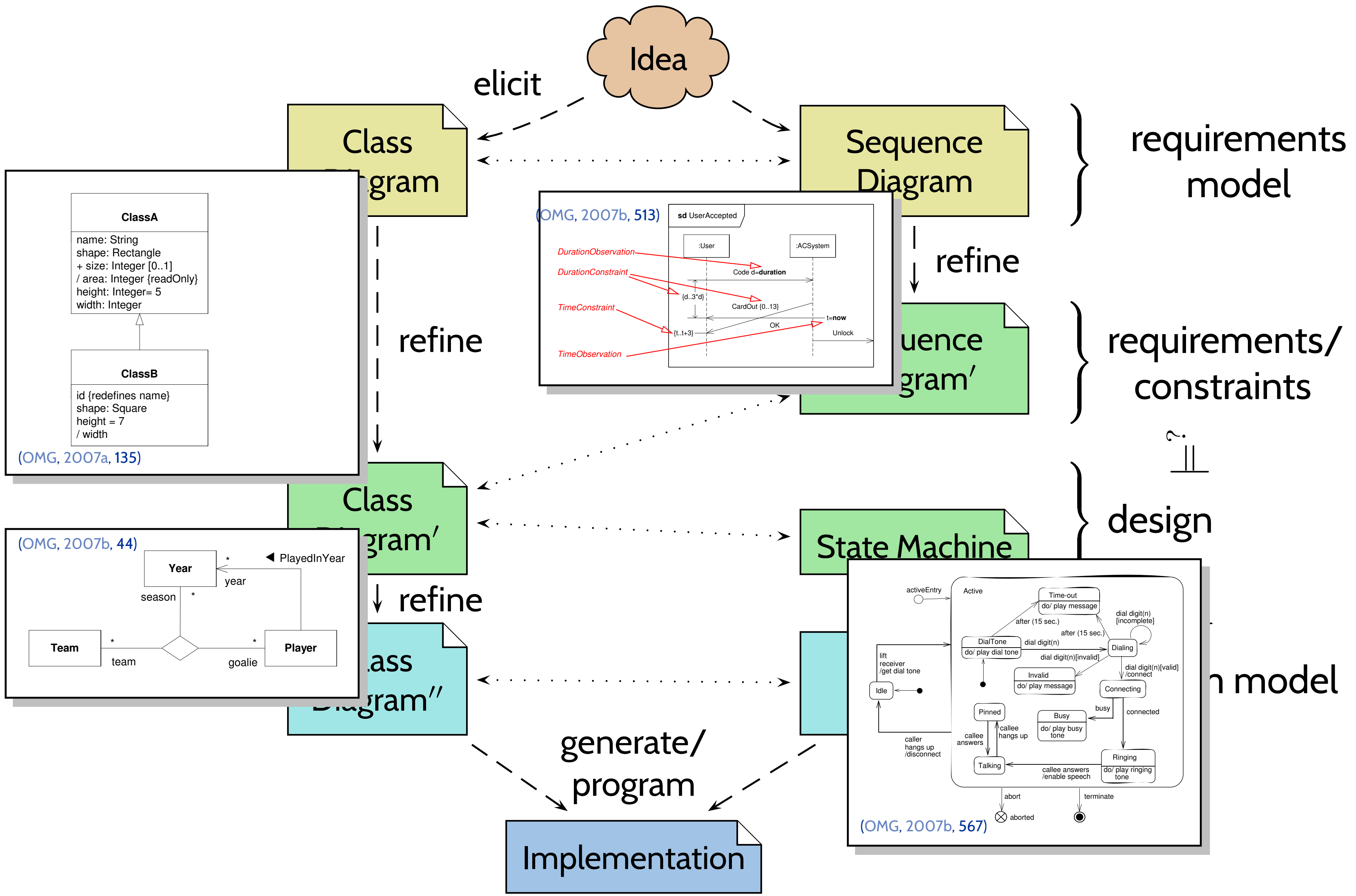
# *Model-Based/-Driven Software Engineering*


Idea
elicit
Structure
Declarative
Behaviour
requirements
model
refine
refine
Declarative
Behaviour′
requirements/
constraints
Structure′
Constructive
Behaviour
design
refine
refine
Structure′′
Constructive
Behaviour′
system model
generate/
program
Implementation

# Model-Driven Software Engineering with UML



# *Model-Driven Software Engineering with UML*



*Model-Based Testing*

# *Recall: Test Case*

**Definition.** A **test case** $T$ is a pair $(In, Soll)$ consisting of

- a description $In$ of sets of finite **input sequences**,
- a description $Soll$ of **expected outcomes**,

and an interpretation $[\![\cdot]\!]$ of these descriptions.

A **test execution** $\pi$, i.e. $((\pi^0, \dots, \pi^n) \downarrow \Sigma_{in}) \in In$ for some $n \in \mathbb{N}_0$, is called

- **successful** (or **positive**)
  if it discovered an error,
  i.e., if $\pi \notin [\![Soll]\!]$.

  (Alternative: test item $S$ **failed to pass test**; confusing: "test failed".)
- **unsuccessful** (or **negative**)
  if it did not discover an error,
  i.e., if $\pi \in [\![Soll]\!]$.

  (Alternative: test item $S$ **passed test**; okay: "test passed".)

# *Glass-Box Testing: Coverage*

- **Coverage** is a property of **test cases** and **test suites**.
- Execution $\pi$ of test case $T$ achieves $p\,\%$ **statement coverage** if and only if

$$p = cov_{stm}(\pi) := \frac{|\bigcup_{i \in \mathbb{N}_0} stm(\sigma_i)|}{|Stm_S|}, |Stm_S| \neq 0.$$

  Test case $T$ achieves $p\,\%$ **statement coverage** if and only if $p = \min_{\pi \text{ execution of } T} cov_{stm}(\pi)$.

- Execution $\pi$ of $T$ achieves $p\,\%$ **branch coverage** if and only if

$$p = cov_{cnd}(\pi) := \frac{|\bigcup_{i \in \mathbb{N}_0} cnd(\sigma_i)|}{|Cnd_S|}, |Cnd_S| \neq 0.$$

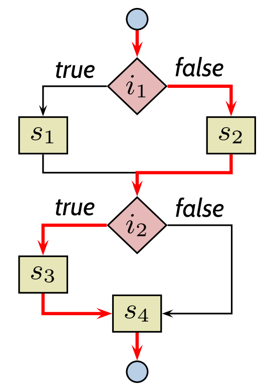
  Test case $T$ achieves $p\,\%$ **branch coverage** if and only if $p = \min_{\pi \text{ execution of } T} cov_{cnd}(\pi)$.

- **Define**: $p = 100$ for empty program.
- Statement/branch coverage canonically extends to test suite $\mathcal{T} = \{T_1, \dots, T_n\}$, e.g. given executions $\pi_1, \dots, \pi_n$, $\mathcal{T}$ achieves

$$p = \frac{|\bigcup_{1 \leq j \leq n} \bigcup_{i \in \mathbb{N}_0} stm(\pi_j^i)|}{|Stm_S|}, |Stm_S| \neq 0, \text{ statement coverage}.$$

# *Coverage Example*

int $f$( int $x$, int $y$, int $z$ )
{
$i_1$: if $(x > 100 \wedge y > 10)$
$s_1$: $z = z * 2$;
else
$s_2$: $z = z/2$;
$i_2$: if $(x > 500 \vee y > 50)$
$s_3$: $z = z * 5$;
$s_4$: ;
}

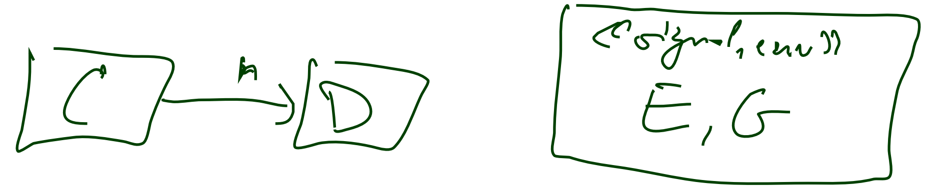



- **Requirement**: $\{$*true*$\}$ $f$ $\{$*true*$\}$ (no abnormal termination), i.e. $Soll = \Sigma^* \cup \Sigma^\omega$.

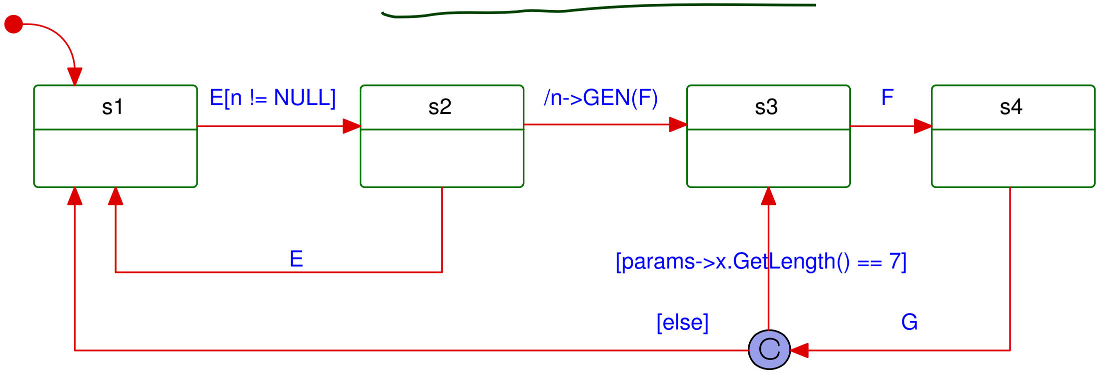
test suite coverage

| *In* $x, y, z$ | $i_1/t$ | $i_1/f$ | $s_1$ | $s_2$ | $i_2/t$ | $i_2/f$ | $c_1$ | $c_2$ | $s_3$ | $s_4$ | % stm | % cnd | $i_2$/% term |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $501, 11, 0$ | ✔ | | ✔ | | ✔ | | ✔ | | ✔ | ✔ | 75 | 50 | 25 |
| $501, 0, 0$ | | ✔ | | ✔ | ✔ | | ✔ | | ✔ | ✔ | 100 | 75 | 25 |
| $0, 0, 0$ | | ✔ | | ✔ | | ✔ | | | | ✔ | 100 | 100 | 75 |
| $0, 51, 0$ | | ✔ | | ✔ | ✔ | | | ✔ | | ✔ | 100 | 100 | 100 |

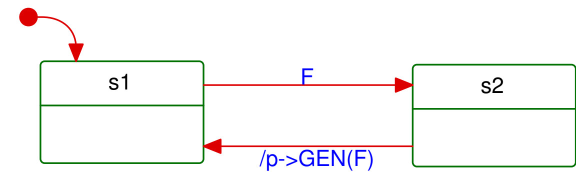
# Model-Element Coverage




State machine of $C$:




State machine of $D$:




**100 % Element coverage** of $C$'s state machine:

- a set of test cases (e.g. **Sequence Diagrams**) such that
- when conducting these test cases
  - **each state** of $C$ is **reached at least once**, (state coverage)
  - **each transition** of $C$ is **taken at least once**. (transition coverage)

In general: **State coverage of a set of test cases**

- number-of-states reached / number-of-states in state machine.

# *Excursion: Automatic Test Generation*

# *Model-based Testing*

- Given a **set of test cases** passing for the model,
- and an **implementation of the model** (maybe hand-written).

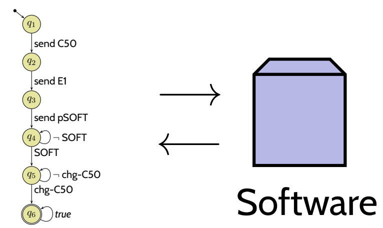
- **Execute the test cases on the implementation** (or the final system).

  This may need an appropriate **interpretation**. For example, if the test case says

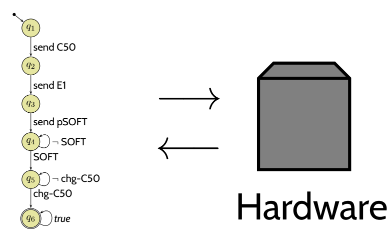
  - send "C50" to the CoinValidator,
  - rather insert a 50 Cent coin into the vending machine.

- If the vending machine **does not behave** according to the test,
  - then **there's something wrong** (wrong test conduction, wrong implementation, etc.).
- If the vending machine **does behave** according to the test,
  - then we know that **this scenario works** – not more.

# Vocabulary




- **Software-in-the-loop**:
  The final implementation is examined
  using a separate computer to simulate other system components.




- **Hardware-in-the-loop**:
  The final implementation is running on (prototype) hardware
  which is connected by its standard input/output interface (e.g. CAN-bus)
  to a separate computer which simulates other system components.

*References*

# References


Dobing, B. and Parsons, J. (2006). How UML is used. Communications of the ACM, 49(5):109–114.

OMG (2007a). Unified modeling language: Infrastructure, version 2.1.2. Technical Report formal/07-11-04.

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

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

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