Softwaretechnik

Lecture 10 (?): Live Sequence Charts and a Glimpse of UML Semantics

2009-06-08

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

The Languages of UML [OMG, 2007b, 684]

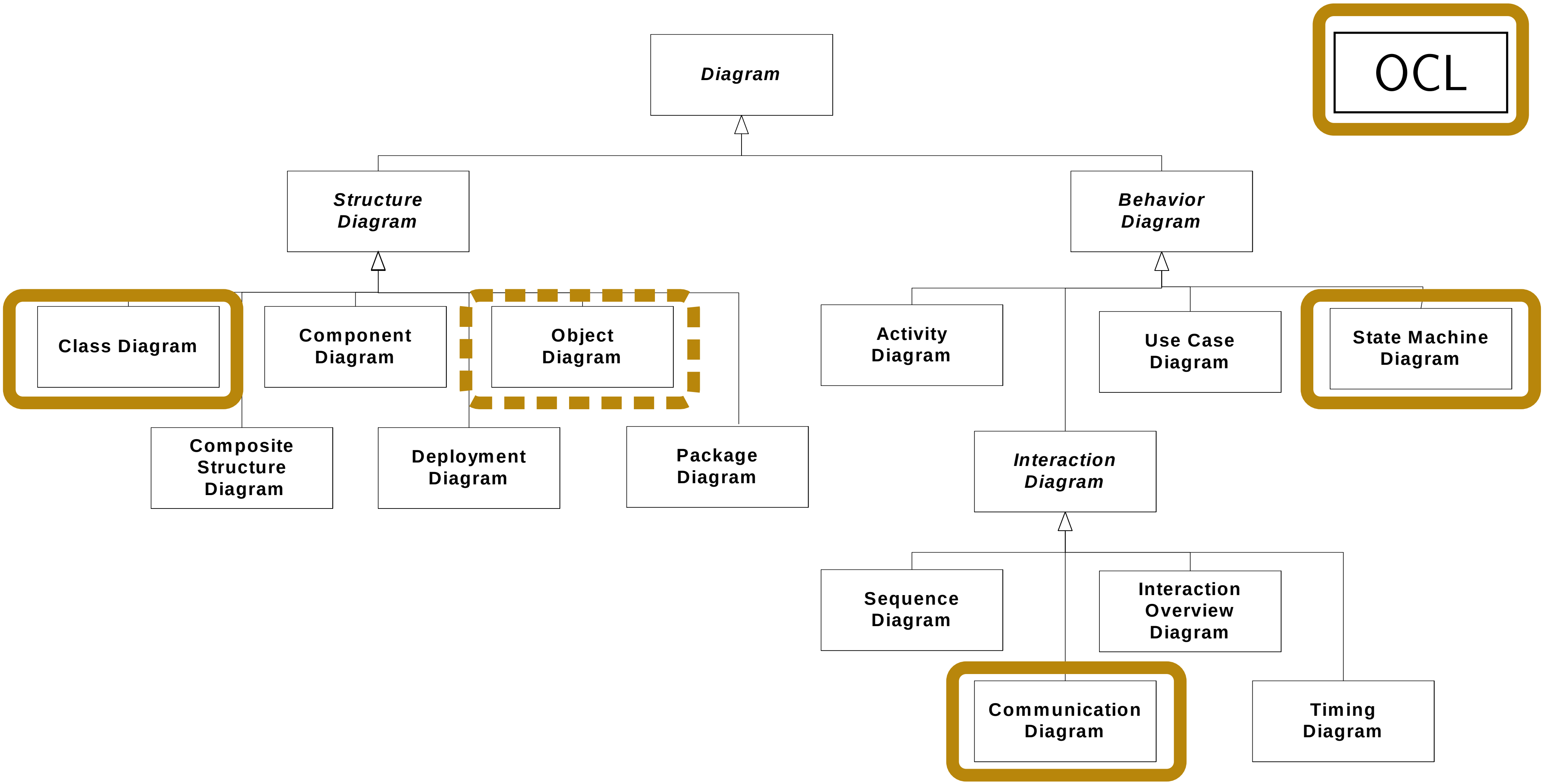


Figure A.5 - The taxonomy of structure and behavior diagram

The Languages of UML *[OMG, 2007b, 684]*

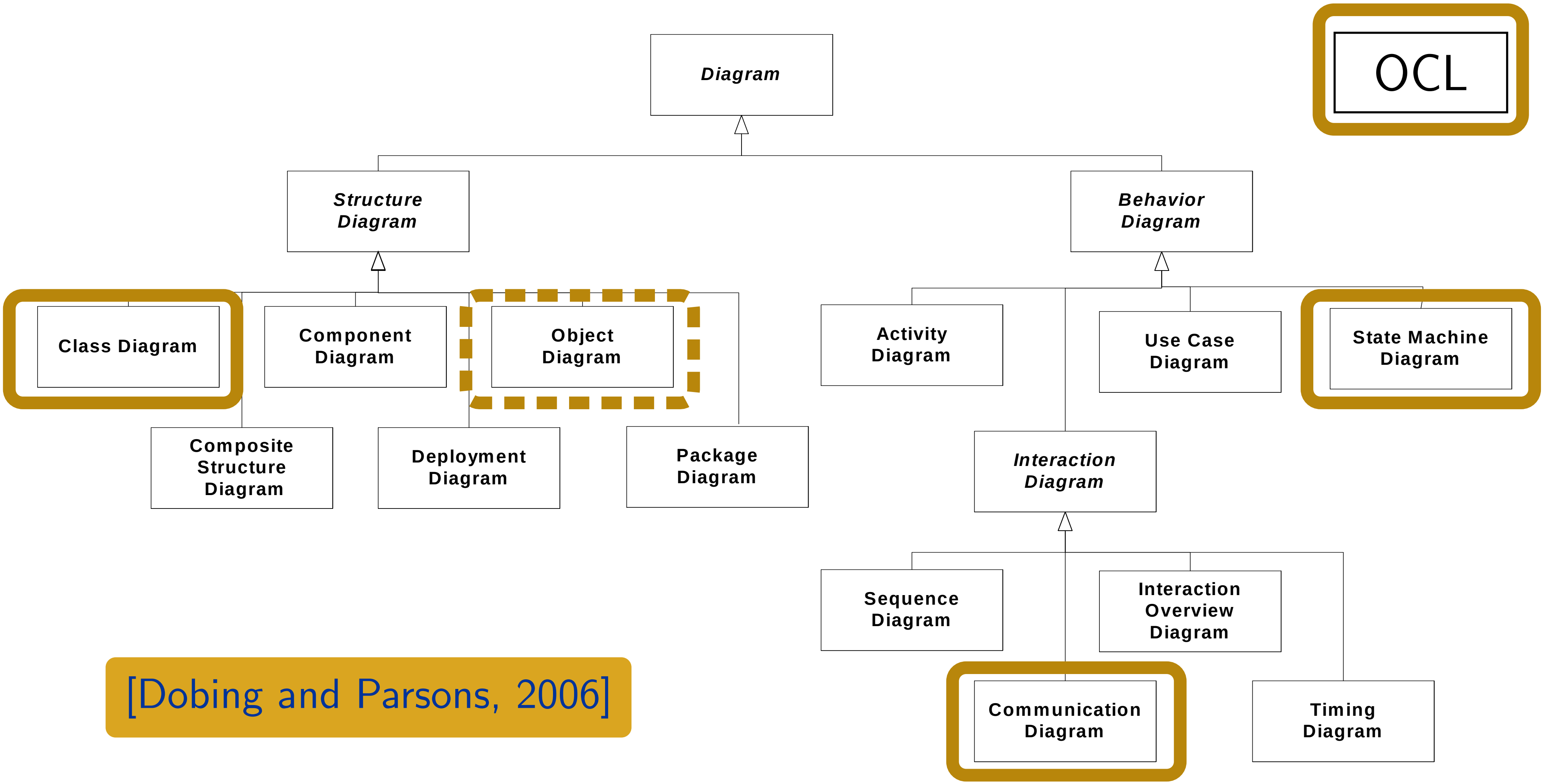


Figure A.5 - The taxonomy of structure and behavior diagram

The Languages of UML *[OMG, 2007b, 684]*

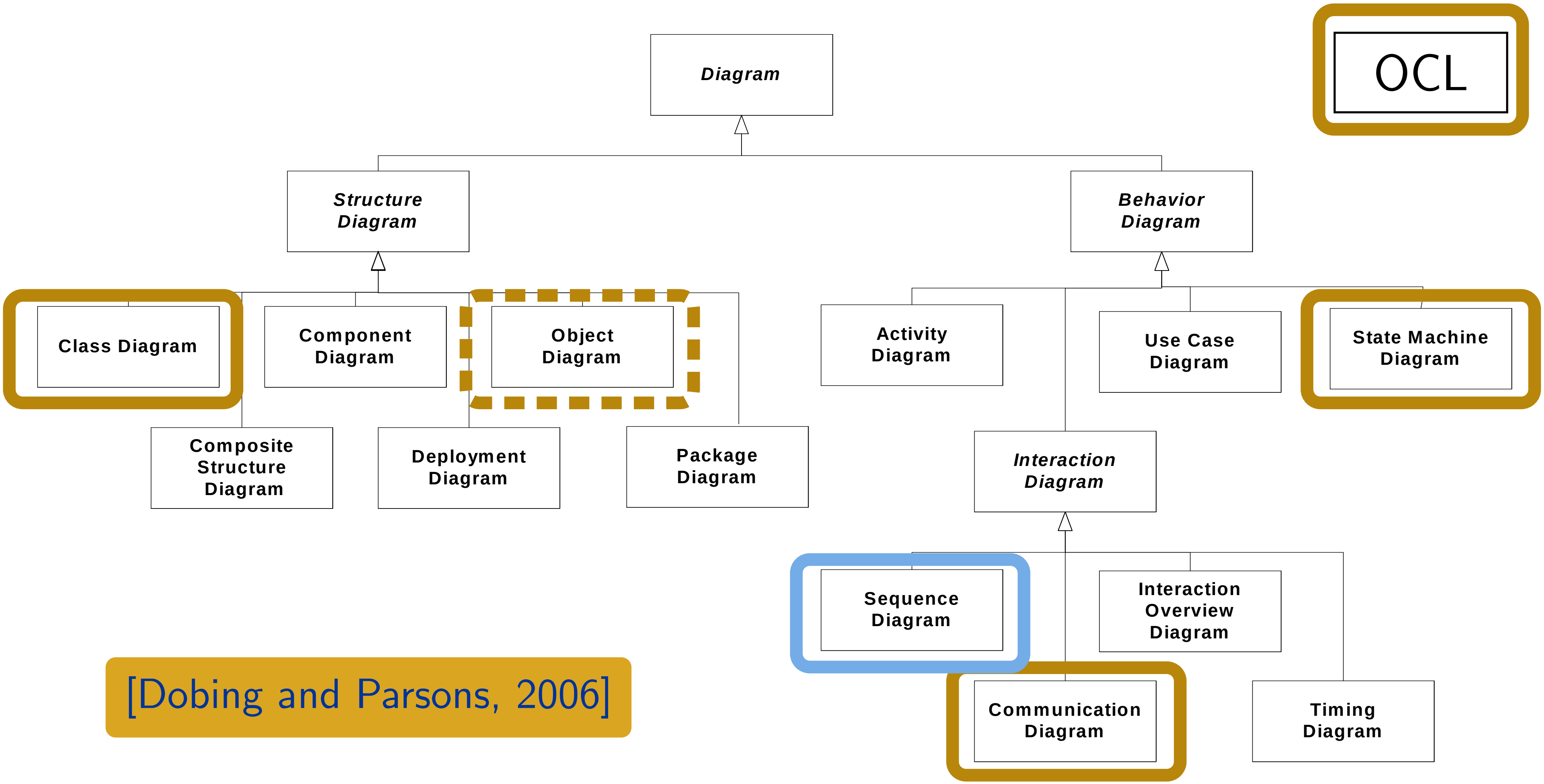


Figure A.5 - The taxonomy of structure and behavior diagram

What Can/Will We Do With It?

UML Mode *[Fowler, http://martinfowler.com/bliki]*

“[...] people differ about what should be in the UML because there are **differing fundamental views** about what the UML should be.

UML Mode *[Fowler, http://martinfowler.com/bliki]*

"[...] people differ about what should be in the UML because there are **differing fundamental views** about what the UML should be.

I came up with three primary classifications for thinking about the UML:

- **UmlAsSketch**,
- **UmlAsBlueprint**, and
- **UmlAsProgrammingLanguage**.

(Interestingly, Steve Mellor independently came up with the same classifications.)

UML Mode *[Fowler, http://martinfowler.com/bliki]*

"[...] people differ about what should be in the UML because there are **differing fundamental views** about what the UML should be.

I came up with three primary classifications for thinking about the UML:

- **UmlAsSketch**,
- **UmlAsBlueprint**, and
- **UmlAsProgrammingLanguage**.

(Interestingly, Steve Mellor independently came up with the same classifications.)

So when **someone else's view** of the UML seems **rather different** to yours, it may be because they use a different **UmlMode** to you."

One Extreme: UML As Sketch

"In this UmlMode, developers use the UML to
help communicate some aspects of a system. [...]

Sketches are also useful in **documents**,
in which case the focus is **communication rather than completeness**. [...]

The tools used for sketching are **lightweight drawing tools** and often

people are **not** too particular about **keeping** to every **strict rule** of the UML.

Most UML diagrams shown in **books**, such as mine, are sketches. "

The Other Extreme: UML As Programming Language

"**If** you can **detail** the UML **enough**,
and provide semantics for everything you need in software,
you can make the UML be your **programming language**.

Tools can take the UML diagrams you draw
and **compile** them into executable code.

The promise of this is that UML is a higher level language and
thus more productive than current programming languages. "

UML As Blueprint

"[...] In **forward engineering** the idea is that
blueprints are developed by a **designer**
whose job is to **build a detailed design**
for a **programmer** to code up.

That design should be **sufficiently complete**
that **all design decisions** are **laid out** and
the programming should follow as a pretty **straightforward activity**
that requires little thought. [...]

Blueprints require much **more sophisticated tools** than sketches in order to handle the details required for the task. [...] "

UML-as-Blueprint: Motivation

Wanted:

- Confirm **validity** early — are we developing what the customer wants?
- Preserve **consistency** — are there contradictions in the requirements?
- Establish **correctness** — is the design satisfying the requirements?
- Ensure **quality** — is the implementation following the design?

Claim: It's easier to

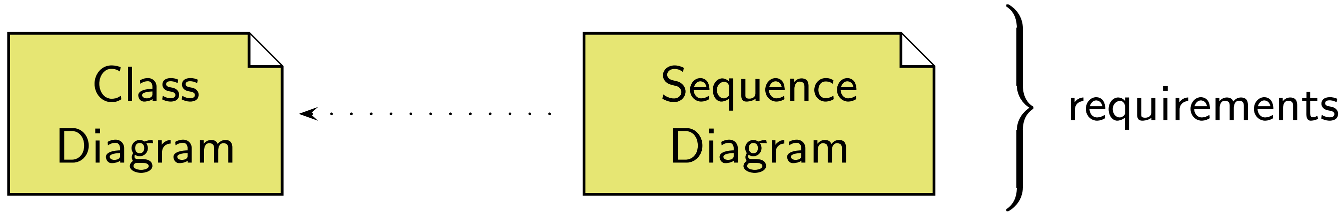
- **change**
- **find fundamental flaws** in

a (rather abstract) model than a more or less complete implementation.

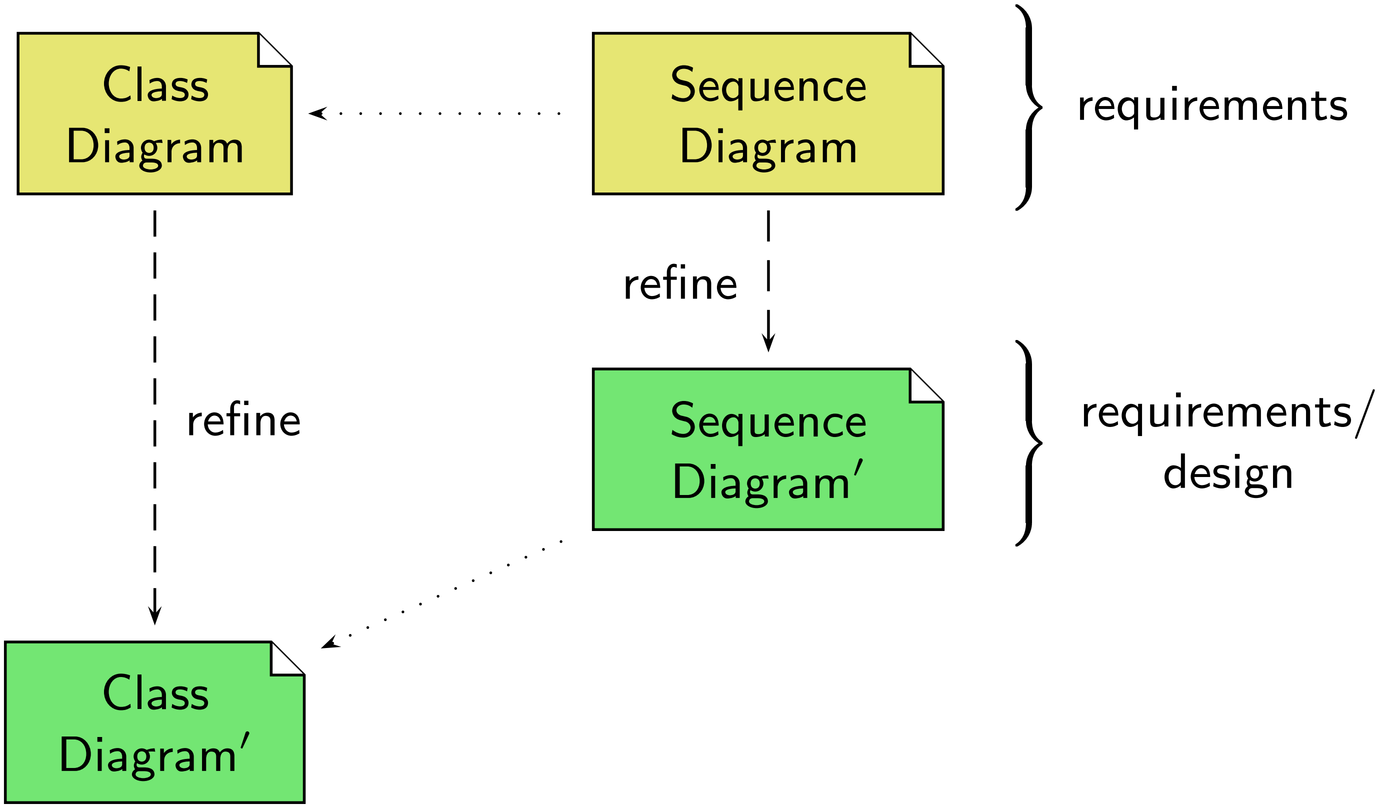
Thus: cost reduction (hopefully).

Note: also need unambiguous semantics.

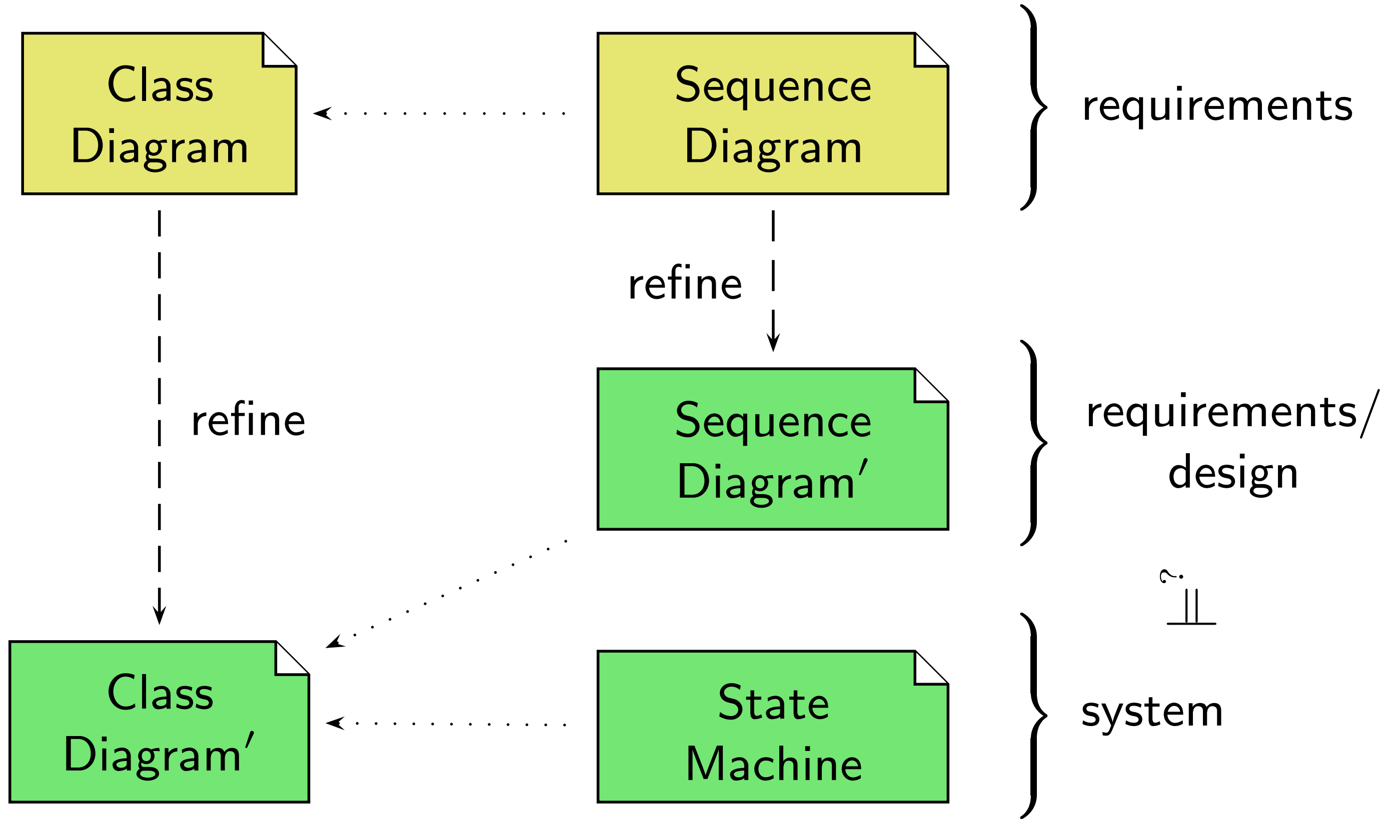
UML-as-Blueprint in Action: One Approach



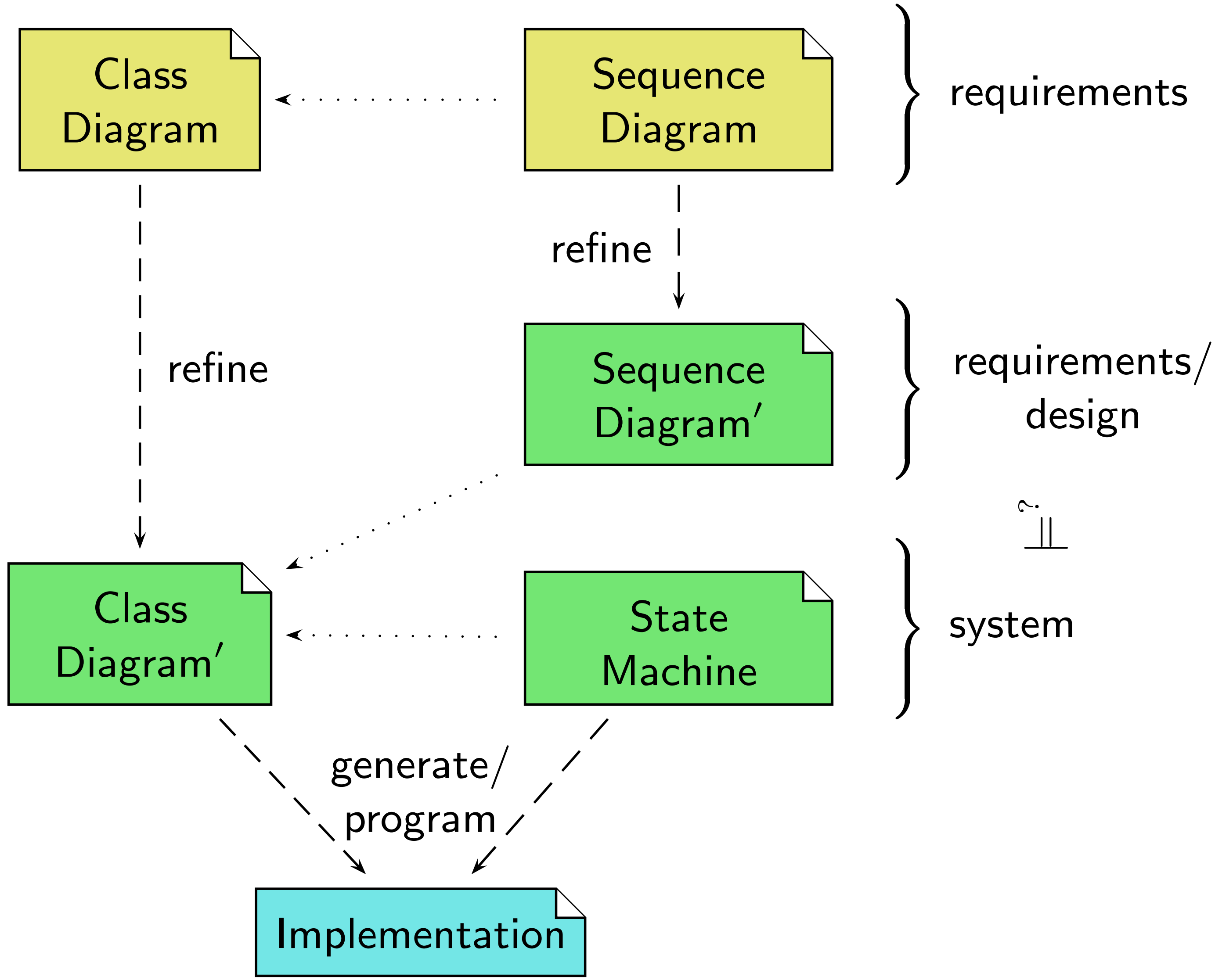
UML-as-Blueprint in Action: One Approach



UML-as-Blueprint in Action: One Approach



UML-as-Blueprint in Action: One Approach



What does "correct" mean exactly?

- **Given:** UML Model $\mathcal{M} = (\mathcal{CD}, \mathcal{OD}, \mathscr{SM}, \mathscr{I})$ with
  - class diagram $\mathcal{CD}$ (for simplicity: only one),
  - object diagram $\mathcal{OD}$ (giving initial configuration),
  - state-machines $\mathscr{SM}$ (for simplicity: one per class),
  - sequence diagrams $\mathscr{I}$, finitely many.

What does "correct" mean exactly?

- **Given:** UML Model $\mathcal{M} = (\mathcal{CD}, \mathcal{OD}, \mathscr{SM}, \mathscr{I})$ with
  - class diagram $\mathcal{CD}$ (for simplicity: only one),
  - object diagram $\mathcal{OD}$ (giving initial configuration),
  - state-machines $\mathscr{SM}$ (for simplicity: one per class),
  - sequence diagrams $\mathscr{I}$, finitely many.

- **Note:**
  - $\mathcal{M}_c := (\mathcal{CD}, \mathscr{SM}, \mathcal{OD})$ has a semantics:
    - the set $[\![\mathcal{M}_c]\!]$ of (**computed**) sequences of object diagrams over $\mathcal{CD}$, starting from object diagram $\mathcal{OD}$.
  - $\mathcal{M}_r := (\mathcal{CD}, \mathscr{I})$ has a semantics:
    - the set $[\![\mathcal{M}_r]\!]$ of (**accepted**) sequences of object diagrams over $\mathcal{CD}$.

What does "correct" mean exactly?

- **Given:** UML Model $\mathcal{M} = (\mathcal{CD}, \mathcal{OD}, \mathscr{SM}, \mathscr{I})$ with
  - class diagram $\mathcal{CD}$ (for simplicity: only one),
  - object diagram $\mathcal{OD}$ (giving initial configuration),
  - state-machines $\mathscr{SM}$ (for simplicity: one per class),
  - sequence diagrams $\mathscr{I}$, finitely many.

- **Note:**
  - $\mathcal{M}_c := (\mathcal{CD}, \mathscr{SM}, \mathcal{OD})$ has a semantics:
    - the set $[\![\mathcal{M}_c]\!]$ of (**computed**) sequences of object diagrams over $\mathcal{CD}$, starting from object diagram $\mathcal{OD}$.
  - $\mathcal{M}_r := (\mathcal{CD}, \mathscr{I})$ has a semantics:
    - the set $[\![\mathcal{M}_r]\!]$ of (**accepted**) sequences of object diagrams over $\mathcal{CD}$.

- **Correctness Problem:**
  - Are all computations produced by $\mathcal{M}_c$ accepted by the sequence diagrams?
  - **In other words**: Do all computations adhere to the requirements?
  - **In symbols**: $[\![\mathcal{M}_c]\!] \subseteq [\![\mathcal{M}_r]\!]$?

Possible Reasons for a Detected Incorrectness

- **ambiguous customer requirements**
  the sequence diagram author understood them **this way**,
  the state-machine author understood them **that way**

- **errors in design**
  the state-machine mistakenly doesn't do what it's author thinks/wishes it does

- **plain mistake**
  in one or the other

Having neither of these is (of course) desired.

Today

Plan

- Give **syntax** of **Live Sequence Charts** — a close relative of UML 2.0 Sequence Diagrams.

Plan

- Give **syntax** of **Live Sequence Charts** — a close relative of UML 2.0 Sequence Diagrams.

- Sketch how a formal semantics for UML models can be defined:
  - class diagrams characterise **system states** (object diagrams)
  - state machines describe how system states **evolve** into each other
  - sequence diagrams express **requirements** on sequences of evolving system states — define **semantics** in terms of Büchi Automata

Plan

- Give **syntax** of **Live Sequence Charts** — a close relative of UML 2.0 Sequence Diagrams.

- Sketch how a formal semantics for UML models can be defined:
  - class diagrams characterise **system states** (object diagrams)
  - state machines describe how system states **evolve** into each other
  - sequence diagrams express **requirements** on sequences of evolving system states — define **semantics** in terms of Büchi Automata

- Putting it all together to assess correctness.

Plan

- Give **syntax** of **Live Sequence Charts** — a close relative of UML 2.0 Sequence Diagrams.
- Sketch how a formal semantics for UML models can be defined:
  - class diagrams characterise **system states** (object diagrams)
  - state machines describe how system states **evolve** into each other
  - sequence diagrams express **requirements** on sequences of evolving system states — define **semantics** in terms of Büchi Automata
- Putting it all together to assess correctness.

```
\begin{advertisement}
```

For the full story see "Software Modelling, Design, and Analysis in UML".

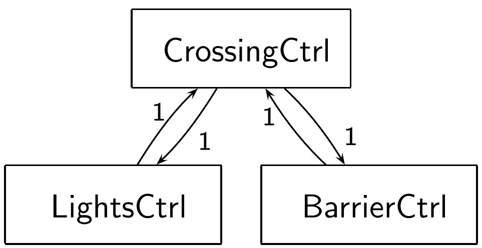
http://electures.informatik.uni-freiburg.de/
portal/web/guest/detail/-/modulnavigation/view/601/5217/

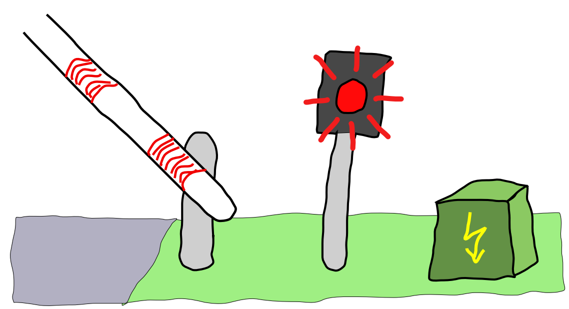
```
\end{advertisement}
```

Live Sequence Charts: Syntax

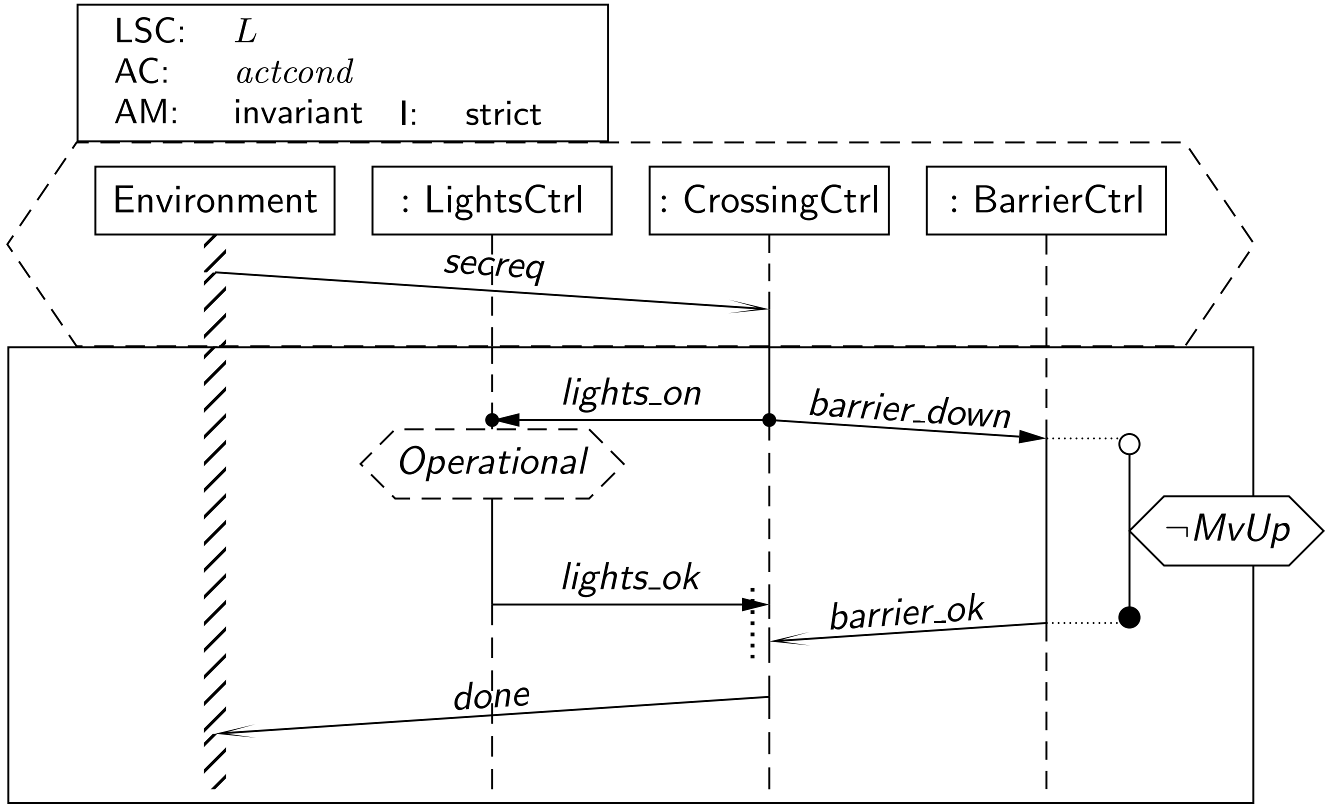
[Damm and Harel, 2001, Harel and Marelly, 2003, Klose, 2003]

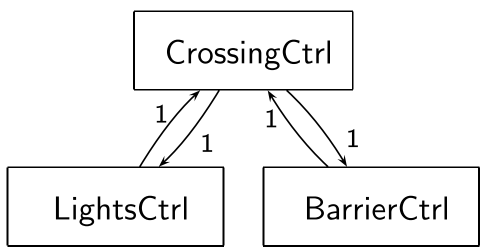
Concrete LSC Syntax by Example

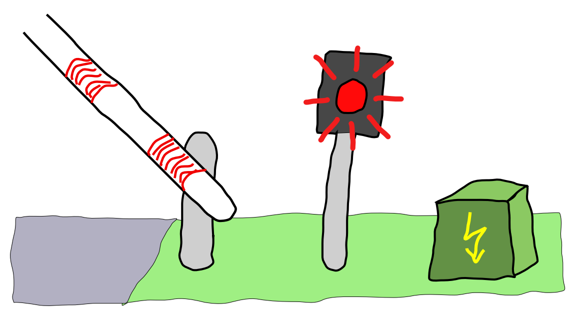




Concrete LSC Syntax by Example

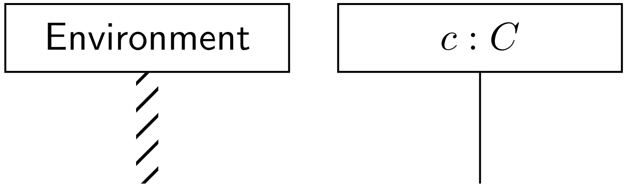






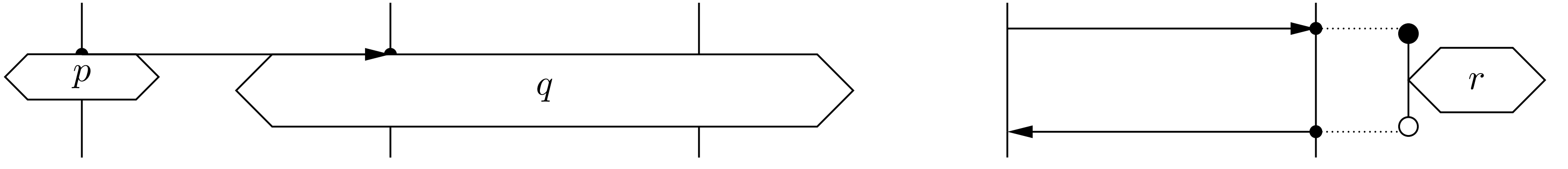
Building Blocks

- **Instance Lines:**



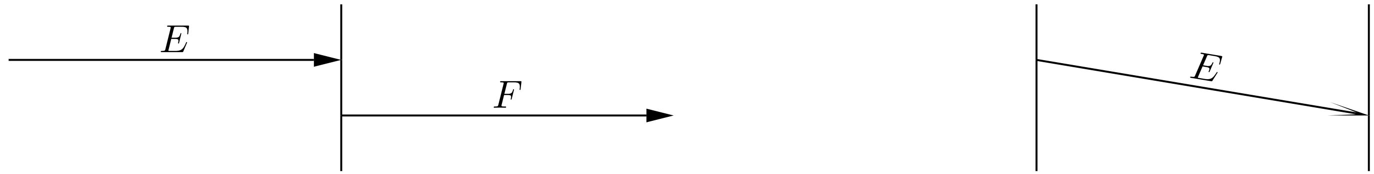
- **Messages:** (asynchronous or synchronous/instantaneous)

- **Conditions and Local Invariants:** (p, q, r e.g. OCL expressions)

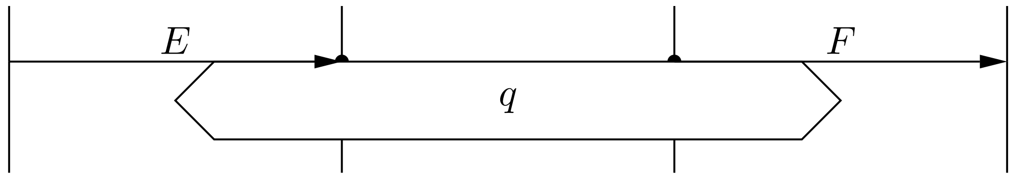


Intuitive Semantics: A Partial Order on Messages

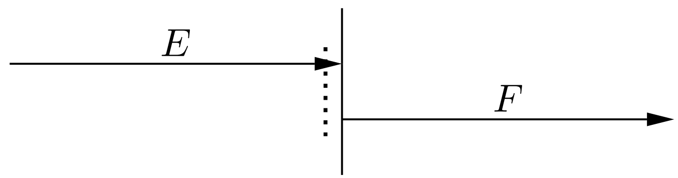
(i) **Strictly After:**



(ii) **Simultaneously:** (simultaneous region)



(iii) **Explicitly Unordered:** (co-region)



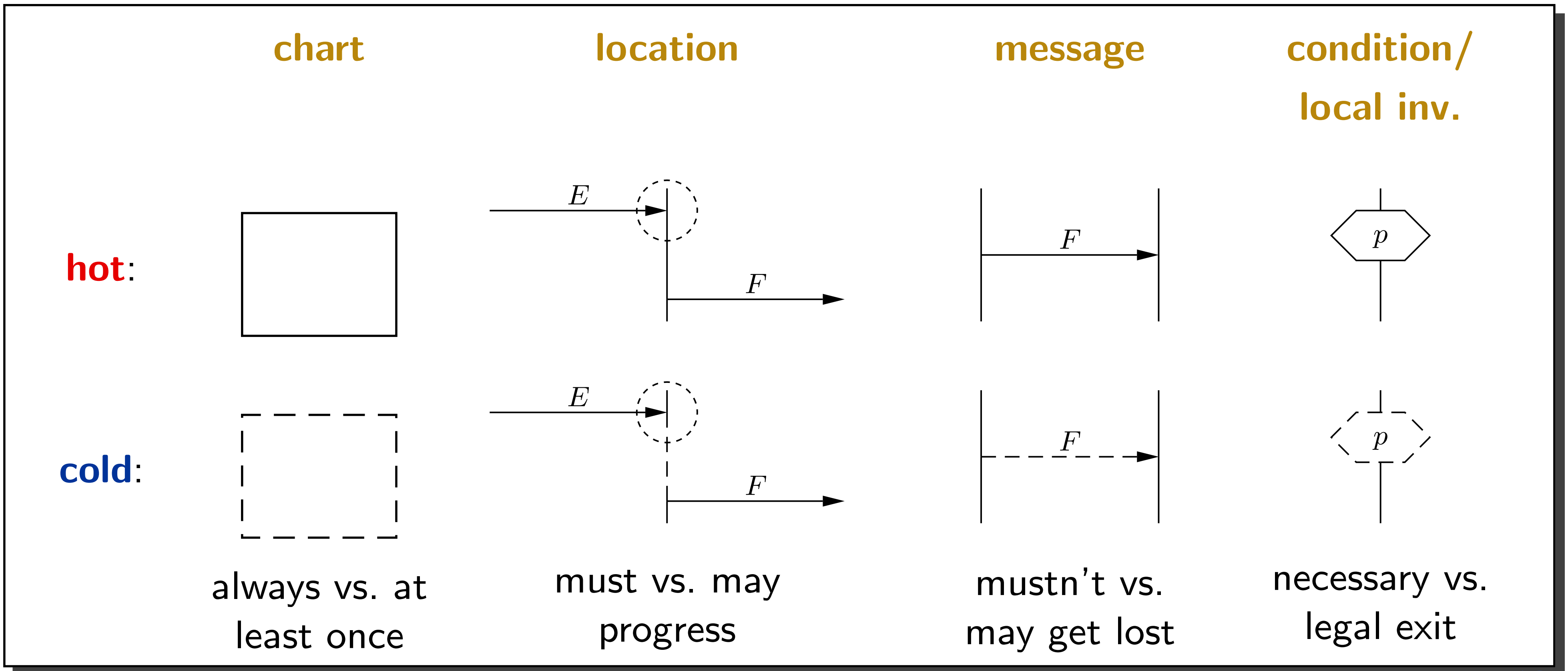
Intuition: A computation of $\mathcal{M}_c$ **violates** an LSC if the occurrence of some events doesn't adhere to partial order obtained as the **transitive closure** of (i) to (iii).

LSC Specialty: Modes

With LSCs,

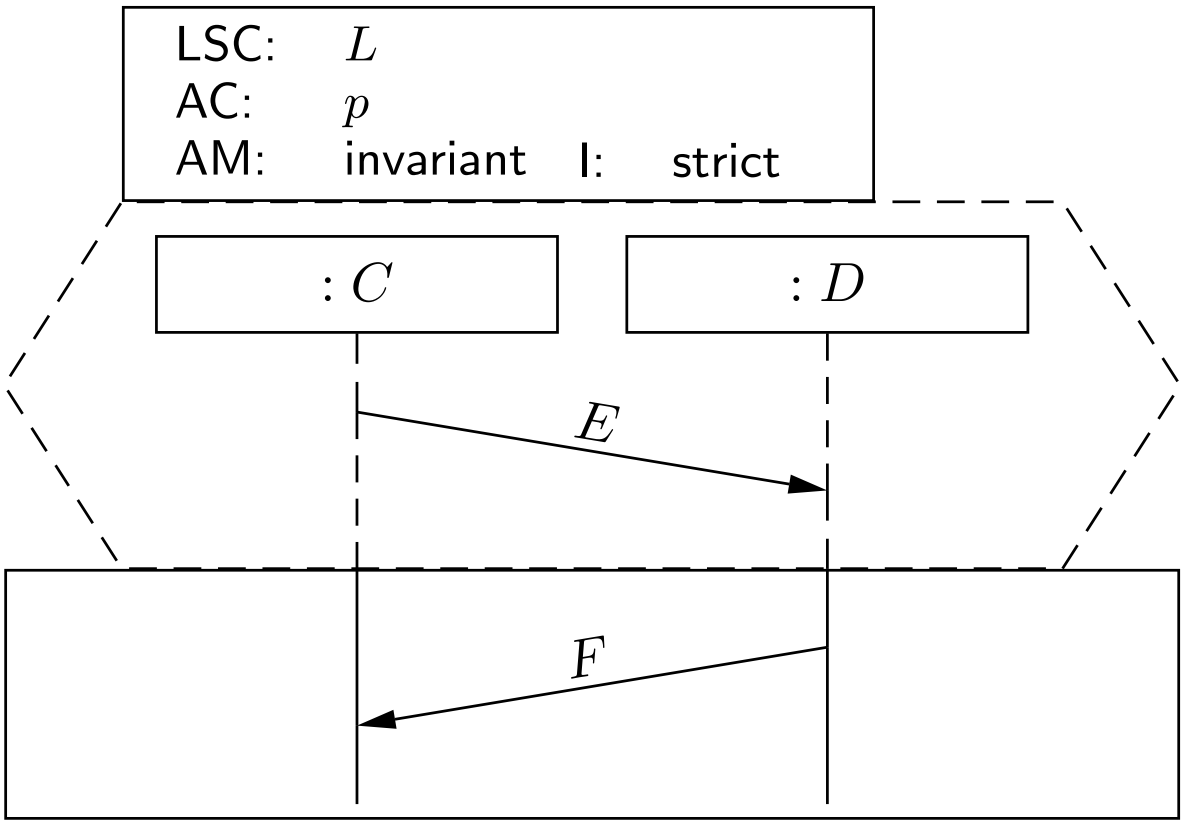
- whole charts,
- locations, and
- elements

have a **mode** — one of **hot** or **cold** (graphically indicated by outline).

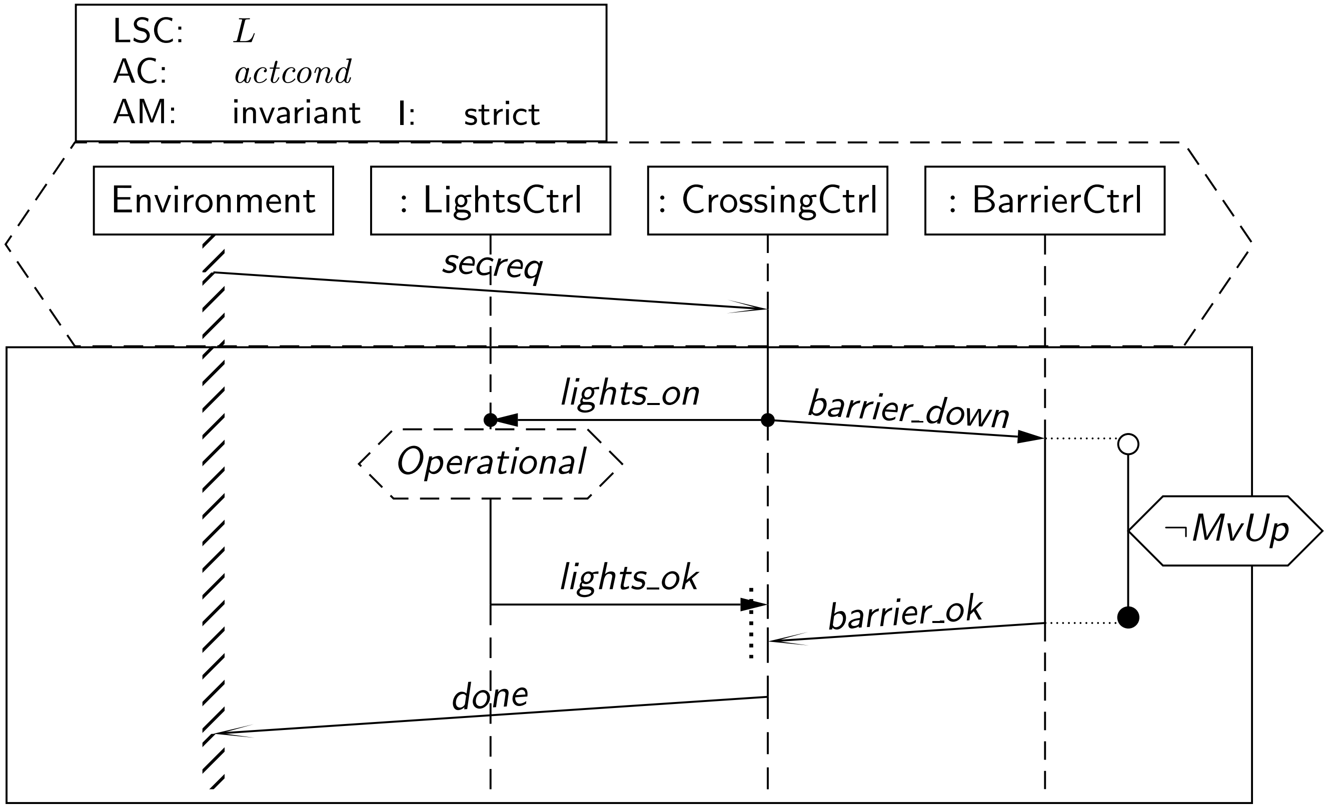


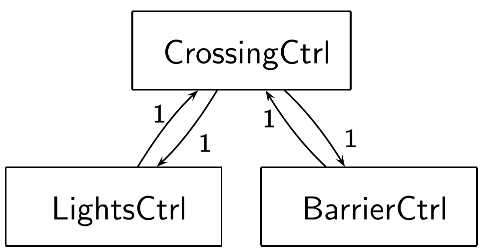
LSC Specialty: Activation

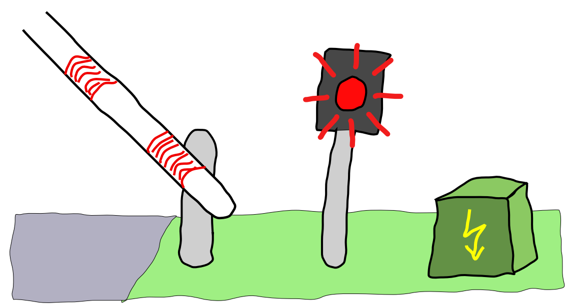
- One major defect of MSCs and SDs:
  they don't say **when** the scenario has to/may be observed.



Example Revisited: What Is Required?







UML Semantics: Approach

Approach: System vs. Requirements

Recall:

- (for us) a UML model is $\mathcal{M} = (\mathscr{CD}, \mathscr{OD}, \mathscr{SM}, \mathscr{I})$.
- And we set $\mathcal{M}_c := (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$ and $\mathcal{M}_r := (\mathscr{CD}, \mathscr{I})$.

What we want is

- on the one hand a **transition system**

$$M_{\mathcal{M}} = (S, s_0, \to)$$

  defined by $\mathcal{M}_c$ ("**programmed** behaviour of $\mathcal{M}$"), and

- on the other hand **one Büchi automaton**

$$A_L = (\Sigma, Q, q_0, \to, F)$$

  per LSC $L \in \mathscr{I}$ ("behaviour **requirements** of $\mathcal{M}$").

Approach: The Formal Relation

- Let *Ids* be a fixed set of (object) **identities**.
- $M_{\mathcal{M}} = (S, s_0, \to)$ **produces** a set $[\![\mathcal{M}_c]\!]$ of **computations** of the form

$$\pi = s_0 \xrightarrow{(cons_0, Snd_0)} s_1 \xrightarrow{(cons_1, Snd_1)} s_2 \dots$$

  where

  - $cons_i = \emptyset$ or $cons_i = \{(id, E)\}$ — object *id* **consumed** event E
  - $Snd_i \subseteq Ids \times E \times Ids$ — object id_1 **sent** event E to id_2

Approach: The Formal Relation

- Let *Ids* be a fixed set of (object) **identities**.
- $M_{\mathcal{M}} = (S, s_0, \rightarrow)$ **produces** a set $[\![\mathcal{M}_c]\!]$ of **computations** of the form

$$\pi = s_0 \xrightarrow{(cons_0, Snd_0)} s_1 \xrightarrow{(cons_1, Snd_1)} s_2 \ldots$$

  where

  - $cons_i = \emptyset$ or $cons_i = \{(\mathit{id}, E)\}$ — object *id* **consumed** event E
  - $Snd_i \subseteq \mathit{Ids} \times E \times \mathit{Ids}$ — object id_1 **sent** event E to id_2

- $A_L = (\Sigma, Q, q_0, \rightarrow, F)$ **accepts** a language $\mathcal{L}(A_L)$ of **words** of the form

$$\hat{\pi} = (s_0, (cons_0, Snd_0)), (s_1, (cons_1, Snd_1)), \ldots$$

Approach: The Formal Relation

- Let *Ids* be a fixed set of (object) **identities**.
- $M_\mathcal{M} = (S, s_0, \to)$ **produces** a set $[\![\mathcal{M}_c]\!]$ of **computations** of the form

$$\pi = s_0 \xrightarrow{(cons_0, Snd_0)} s_1 \xrightarrow{(cons_1, Snd_1)} s_2 \ldots$$

  where

  - $cons_i = \emptyset$ or $cons_i = \{(id, E)\}$ — object *id* **consumed** event E
  - $Snd_i \subseteq Ids \times E \times Ids$ — object id_1 **sent** event E to id_2

- $A_L = (\Sigma, Q, q_0, \to, F)$ **accepts** a language $\mathcal{L}(A_L)$ of **words** of the form

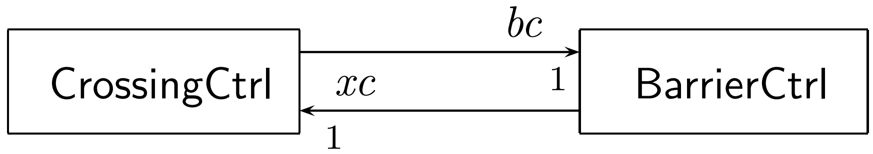
$$\hat{\pi} = (s_0, (cons_0, Snd_0)), (s_1, (cons_1, Snd_1)), \ldots$$

- We say $\mathcal{M}_c$ **satisfies** the **universal** LSC L with **invariant** activation and instance lines $i_1, \ldots, i_n$, denoted by $\mathcal{M}_c \models L$, if and only if

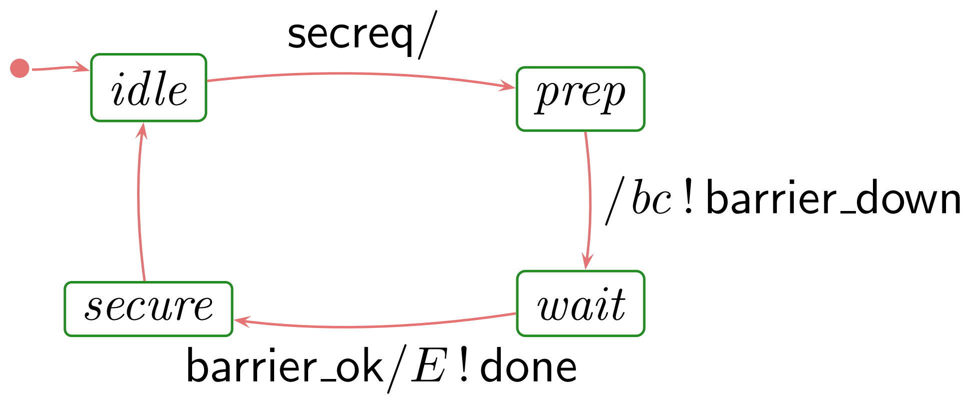
$$\forall\, \pi \in [\![\mathcal{M}_c]\!]\ \forall \text{type-cons. bindings } \theta = \{i_j \mapsto id_j \in Ids \mid 1 \leq j \leq n\}\ \forall\, k \in \mathbb{N}_0 :$$

$$\hat{\pi}/k \text{ activates } L \text{ under } \theta \implies \hat{\pi}/k \in \mathcal{L}_\theta(A_L)$$

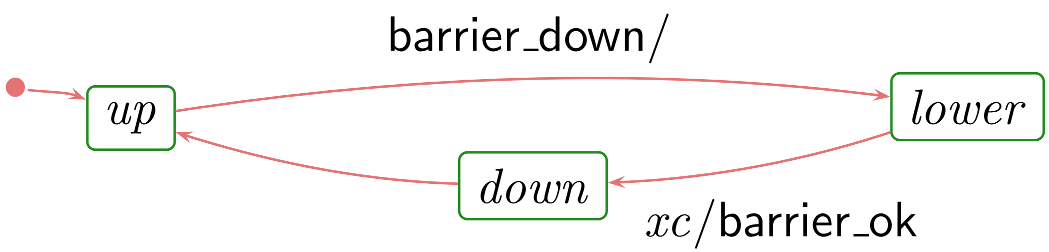
Approach: Example Model



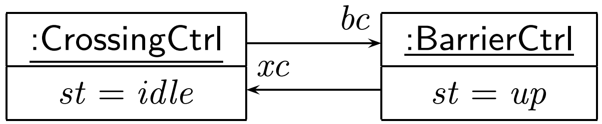
State-machine of **CrossingCtrl:**

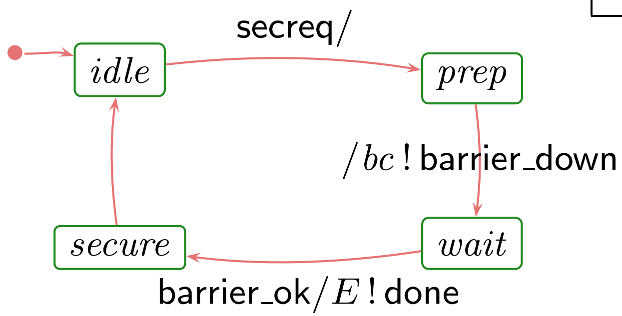


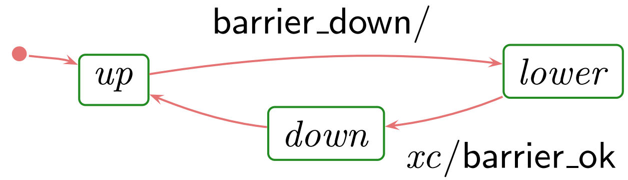
State-machine of **BarrierCtrl:**

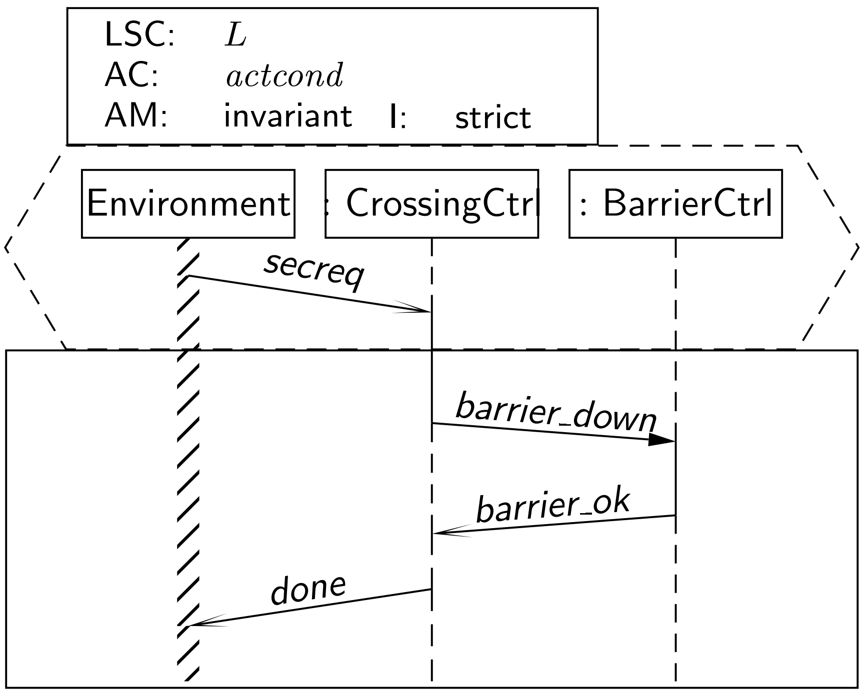


Example


:CrossingCtrl
st = idle
bc
xc
:BarrierCtrl
st = up


idle
secreq/
prep
/ bc ! barrier_down
wait
barrier_ok/ E ! done
secure


up
barrier_down/
lower
down
xc/barrier_ok


LSC: L
AC: actcond
AM: invariant I: strict
Environment
: CrossingCtrl
: BarrierCtrl
secreq
barrier_down
barrier_ok
done

Approach: All activation modes

$\mathcal{M}_c \models L$ with

- L **universal** (= hot), **invariant** if and only if

  $\forall\, \pi \in [\![\mathcal{M}_c]\!]\ \forall\, \theta\ \forall\, k \in \mathbb{N}_0 : \hat{\pi}/k$ activates L under $\theta \implies \hat{\pi}/k \in \mathcal{L}_\theta(A_L)$

Approach: All activation modes

$\mathcal{M}_c \models L$ with

- L **universal** $(=$ hot$)$, **invariant** if and only if

$$\forall\, \pi \in [\![\mathcal{M}_c]\!]\ \forall\, \theta\ \forall\, k \in \mathbb{N}_0 : \hat{\pi}/k \text{ activates } L \text{ under } \theta \implies \hat{\pi}/k \in \mathcal{L}_\theta(A_L)$$

- L **universal** $(=$ hot$)$, **initial** if and only if

$$\forall\, \pi \in [\![\mathcal{M}_c]\!]\ \forall\, \theta : \hat{\pi}/0 \text{ activates } L \text{ under } \theta \implies \hat{\pi}/0 \in \mathcal{L}_\theta(A_L)$$

Approach: All activation modes

$\mathcal{M}_c \models L$ with

- L **universal** $(=$ hot$)$, **invariant** if and only if

$$\forall\, \pi \in [\![\mathcal{M}_c]\!]\ \forall\, \theta\ \forall\, k \in \mathbb{N}_0 : \hat{\pi}/k \text{ activates } L \text{ under } \theta \implies \hat{\pi}/k \in \mathcal{L}_\theta(A_L)$$

- L **universal** $(=$ hot$)$, **initial** if and only if

$$\forall\, \pi \in [\![\mathcal{M}_c]\!]\ \forall\, \theta : \hat{\pi}/0 \text{ activates } L \text{ under } \theta \implies \hat{\pi}/0 \in \mathcal{L}_\theta(A_L)$$

- L **existential** $(=$ cold$)$, **invariant** if and only if

$$\exists\, \pi \in [\![\mathcal{M}_c]\!]\ \exists\, \theta\ \exists\, k \in \mathbb{N}_0 : \hat{\pi}/k \text{ activates } L \text{ under } \theta \implies \hat{\pi}/k \in \mathcal{L}_\theta(A_L)$$

- L **existential** $(=$ cold$)$, **initial** if and only if

$$\exists\, \pi \in [\![\mathcal{M}_c]\!]\ \exists\, \theta : \hat{\pi}/0 \text{ activates } L \text{ under } \theta \implies \hat{\pi}/0 \in \mathcal{L}_\theta(A_L)$$

We write $\mathcal{M}_c \models \mathcal{M}_r$ if and only if $\mathcal{M}_c \models L$ for all $L \in \mathscr{I}$.

So What's Missing?

Given:

- $\mathcal{M} = (\mathcal{CD}, \mathcal{OD}, \mathscr{SM}, \mathscr{I})$

Wanted:

- $M_{\mathcal{M}} = (S, s_0, \rightarrow)$
- $A_L = (\Sigma, Q, q_0, \rightarrow, F)$

Missing to complete the picture:

So What's Missing?

Given:

- $\mathcal{M} = (\mathscr{CD}, \mathscr{OD}, \mathscr{SM}, \mathscr{I})$

Wanted:

- $M_{\mathcal{M}} = (S, s_0, \rightarrow)$
- $A_L = (\Sigma, Q, q_0, \rightarrow, F)$

Missing to complete the picture:

- what are the system states S, s_0? — **object diagrams**
- when do we have $s \xrightarrow{(cons, Snd)} s'$? — **one object takes a state-machine transition**
- what is Q and Σ? — **cuts** of L
- when do we have $q \xrightarrow{\sigma} q'$? — **partial order** of L

UML Semantics: System States

System States

- Let $\mathcal{M} = (\mathcal{CD}, \mathcal{OD}, \mathscr{SM}, \mathscr{I})$ be a UML model.
- The class diagram $\mathcal{CD}$ describes a set of **complete object diagrams**.
- We call an object diagram **complete** if and only if
  - each attribute has a (type-consistent) value,
  - in particular the implicit attribute giving the current state-machine state (and whether the object is in the middle of a run-to-completion step or not),
  - each object obtains a unique name from a set *Ids*.
- In contrast:
  we call a (complete or partial) object diagram **legal** if and only if
  - all OCL constraints of the model are satisfied,
  - in particular multiplicities of links.

So we simply use

- for S the set of all complete object diagrams of $\mathcal{CD}$, and
- for s_0 the object diagram $\mathcal{OD}$ (it should thus be complete).

UML Semantics: Transition System

Evolution of System States

- Let s, s' be system states, *Ids* unique object names in object diagrams.
- Then $s \xrightarrow{(cons,Snd)} s'$ if
  - the **object** name $\mathit{id} \in \mathit{Ids}$ occurs in s (say it is of class C),
  - *id*'s current state-machine **state** is st,
  - there is a transition

$$st \xrightarrow{\text{trigger[guard]/[action]}} st'$$

  in the state-machine $\mathcal{SM}_C$ of C, which is **enabled**, that is,

    - either 'trigger' is empty and *id* is **not stable** ($cons = \emptyset$), or
      id is **stable** 'trigger' denotes a signal E, and an E-event is ready to be consumed in the receive buffer ($cons = \{(E, \mathit{id})\}$), and
    - expression 'guard' holds in s,

and

  - s' is (exactly) the **effect** of executing 'action' for *id* in s.

Evolution of System States

- Let s, s' be system states, *Ids* unique object names in object diagrams.
- Then $s \xrightarrow{(cons, Snd)} s'$ if
  - the **object** name $\mathit{id} \in \mathit{Ids}$ occurs in s (say it is of class C),
  - *id*'s current state-machine **state** is st,
  - there is an **enabled** transition $st \xrightarrow{\text{trigger[guard]/[action]}} st'$ in the state-machine $\mathcal{SM}_C$ of C,

  and

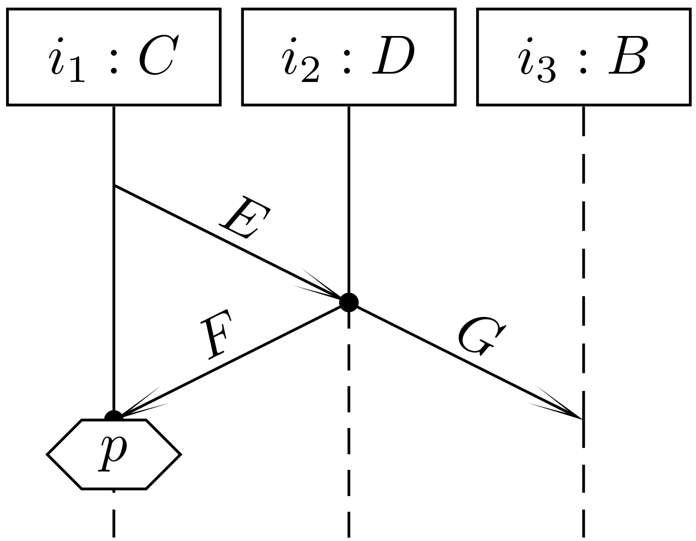
  - s' is (exactly) the **effect** of executing 'action' for *id* in s.

    That is, for instance,

    - **removal** of the consumed event from the input buffer,
    - **updating** attributes of object *id*, or other objects via links,
    - **creation** of new objects, **deletion** of *id* or other objects,
    - **sending** events $E_1, \dots, E_n$ to objects $\mathit{id}_1, \dots, \mathit{id}_n$, $n \geq 0$, resp.; then $Snd = \{(\mathit{id}, E_1, \mathit{id}_1), \dots, (\mathit{id}, E_1, \mathit{id}_1)\}$.

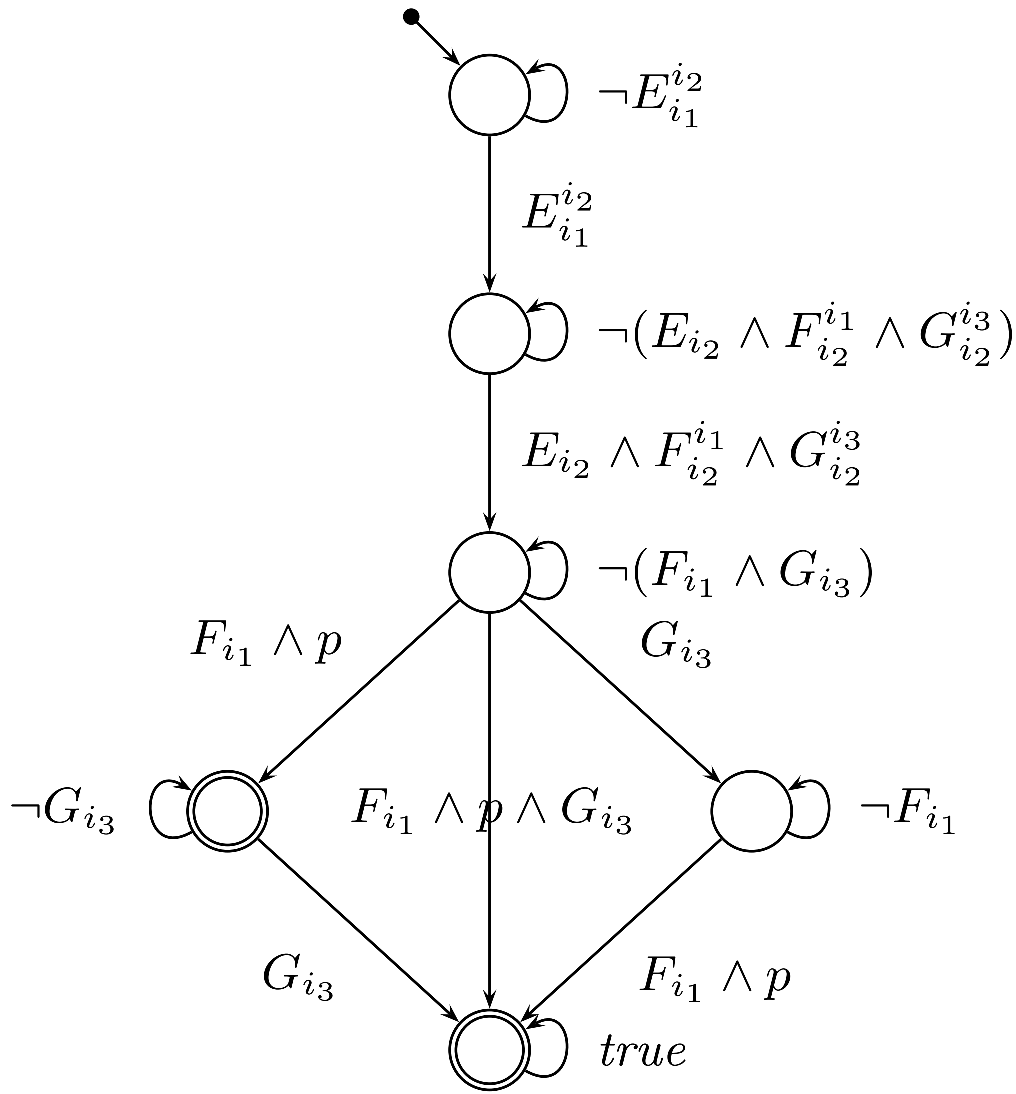
UML Semantics: Büchi Automaton

The (Symbolic) Büchi Automaton of an LSC (Example)



$A_L = (\Sigma, Q, q_0, \to, F)$:

- letters in Σ:
  E_i (i consumes an E),
  $E_i^{i'}$ (i sends E to i')
- states Q: cuts of LSC
- q_0: empty cut
- $q \to q'$: partial order on cuts, transitions labelled with prop. logic expressions over Σ
- F: cold cuts and final cut



The (Symbolic) Büchi Automaton of an LSC

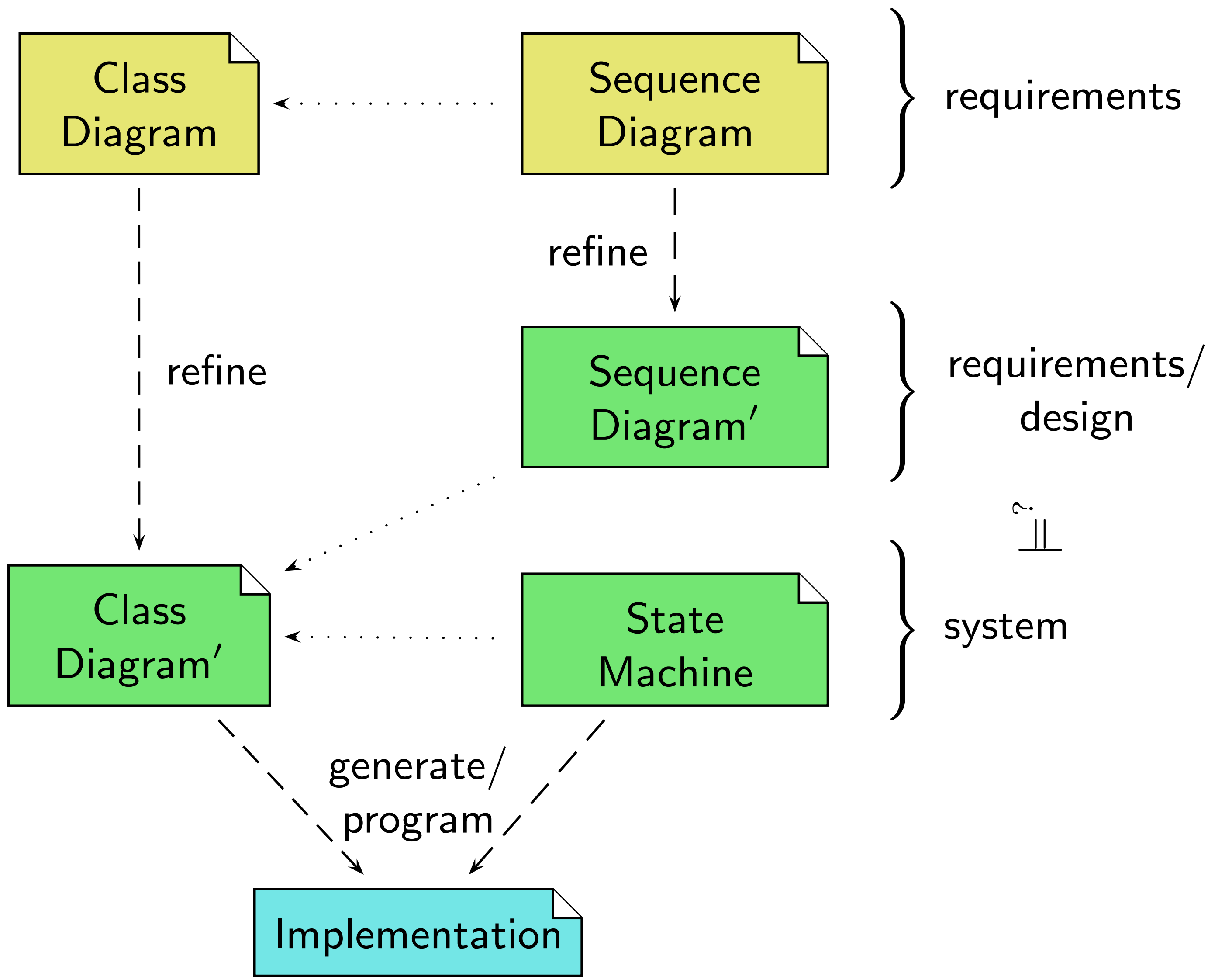
Not covered:

- treatment of pre-charts
- ...

See [Klose, 2003].

Summary

Recall Proposal



References

References

[Damm and Harel, 2001] Damm, W. and Harel, D. (2001). LSCs: Breathing life into Message Sequence Charts. *Formal Methods in System Design*, 19(1):45–80.

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

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

[Klose, 2003] Klose, J. (2003). *LSCs: A Graphical Formalism for the Specification of Communication Behavior*. PhD thesis, Carl von Ossietzky Universität Oldenburg.

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

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