# *Software Design, Modelling and Analysis in UML*

# *Lecture 16: Hierarchical State Machines I*

*2015-01-15*

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

Albert-Ludwigs-Universität Freiburg, Germany

# *Contents & Goals*

**Last Lecture:**

- Missing transformers: create and destroy
- Step and run-to-completion (RTC) step, divergence

**This Lecture:**

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What does this State Machine mean? What happens if I inject this event?
  - Can you please model the following behaviour.
  - What does this **hierarchical** State Machine mean? What **may happen** if I inject this event?
  - What is: AND-State, OR-State, pseudo-state, entry/exit/do, final state, ...

- **Content:**
  - Putting it all together: UML model semantics (so far)
  - State Machines and OCL
  - Hierarchical State Machines Syntax
  - Initial and Final State

# *Putting It All Together*

# *The Missing Piece: Initial States*

**Recall**: a labelled transition system is $(S, \rightarrow, S_0)$. We **have**

- $S$: system configurations $(\sigma, \varepsilon)$
- $\rightarrow$: labelled transition relation $(\sigma, \varepsilon) \xrightarrow[u]{(cons,Snd)} (\sigma', \varepsilon')$.

**Wanted**: initial states $S_0$.

**Proposal**:
Require a (finite) set of **object diagrams** $\mathcal{OD}$ as part of a UML model

$$(\mathscr{CD}, \mathscr{SM}, \mathscr{OD}).$$

And set

$$S_0 = \{(\sigma, \varepsilon) \mid \sigma \in G^{-1}(\mathcal{OD}), \mathcal{OD} \in \mathscr{OD}, \varepsilon \text{ empty}\}.$$

**Other Approach**: (used by Rhapsody tool) multiplicity of classes.
We can read that as an abbreviation for an object diagram.

# *Semantics of UML Model — So Far*

The **semantics** of the **UML model**

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

where

- some classes in $\mathscr{CD}$ are stereotyped as 'signal' (standard), some signals and attributes are stereotyped as 'external' (non-standard),
- there is a 1-to-1 relation between classes and state machines,
- $\mathscr{OD}$ is a set of object diagrams over $\mathscr{CD}$,

is the **transition system** $(S, \to, S_0)$ constructed on the previous slide.

The **computations of** $\mathcal{M}$ are the computations of $(S, \to, S_0)$.

# *State Machines and OCL*

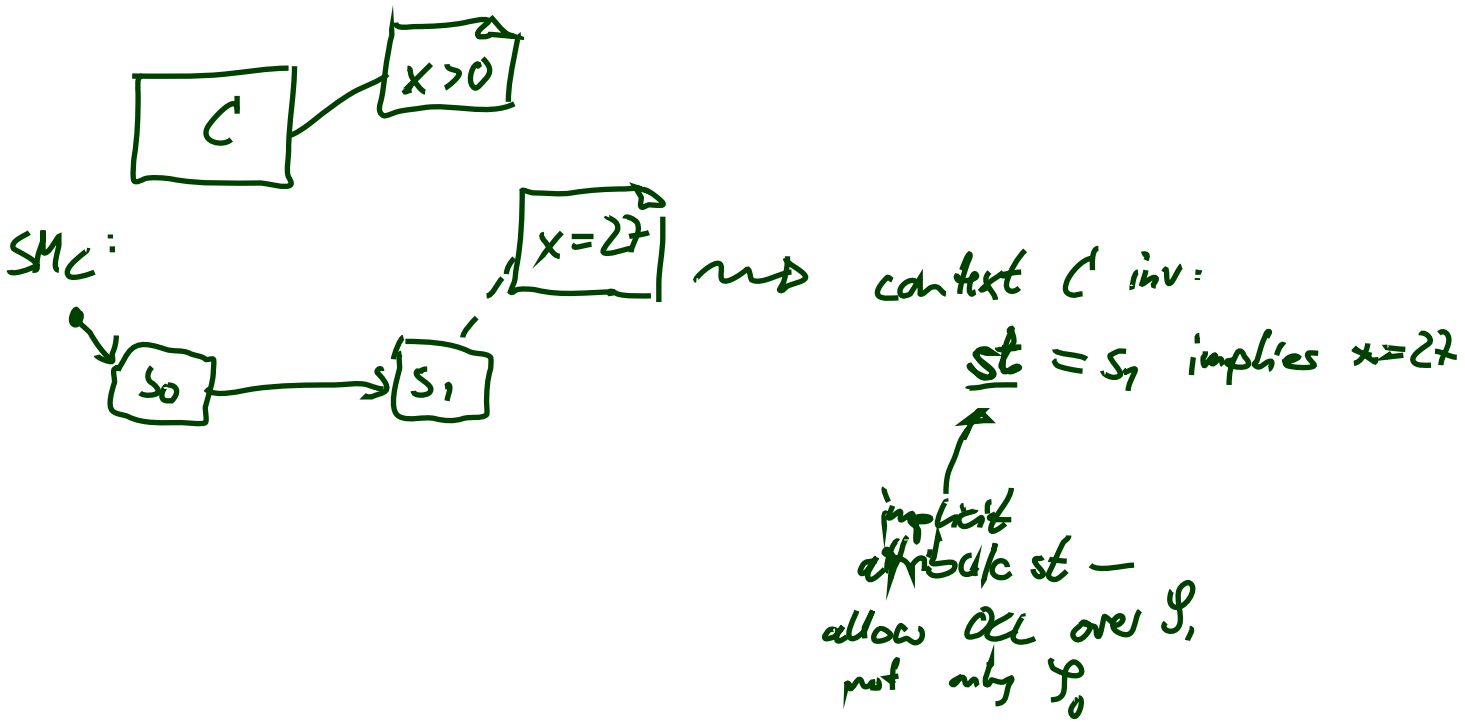
# OCL Constraints and Behaviour

- Let $\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$ be a UML model.
- We call $\mathcal{M}$ **consistent** iff, for each OCL constraint $expr \in \mathit{Inv}(\mathscr{CD}) \cup \mathit{Inv}(\mathscr{SM})$

$$\sigma \models expr \text{ for each “reasonable point” } (\sigma, \varepsilon) \text{ of computations of } \mathcal{M}.$$

(Cf. exercises and tutorial for discussion of “reasonable point”.)

**Note**: we could define $\mathit{Inv}(\mathscr{SM})$ similar to $\mathit{Inv}(\mathscr{CD})$.



# *OCL Constraints and Behaviour*

- Let $\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$ be a UML model.
- We call $\mathcal{M}$ **consistent** iff, for each OCL constraint $expr \in \mathit{Inv}(\mathscr{CD})$,

  $\sigma \models expr$ for each "reasonable point" $(\sigma, \varepsilon)$ of computations of $\mathcal{M}$.

  (Cf. exercises and tutorial for discussion of "reasonable point".)

**Note**: we could define $\mathit{Inv}(\mathscr{SM})$ similar to $\mathit{Inv}(\mathscr{CD})$.

→ OUR CHOICE: check for each $(\sigma,\varepsilon)$ in a computation (step granularity)

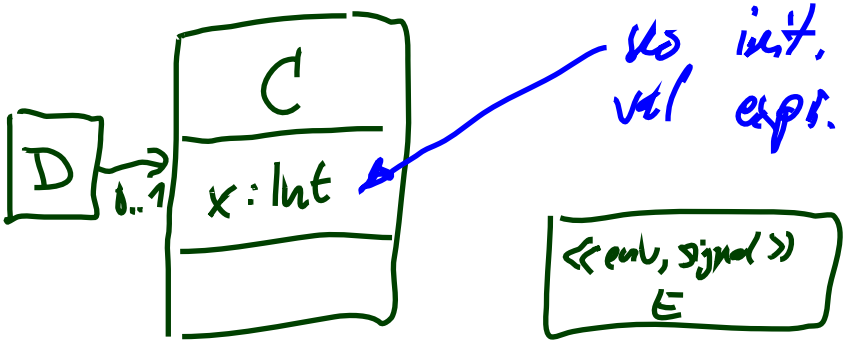
**Pragmatics**:

- In **UML-as-blueprint mode**, if $\mathscr{SM}$ doesn't exist yet, then $\mathcal{M} = (\mathscr{CD}, \emptyset, \mathscr{OD})$ is typically asking the developer to provide $\mathscr{SM}$ such that $\mathcal{M}' = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$ is consistent.

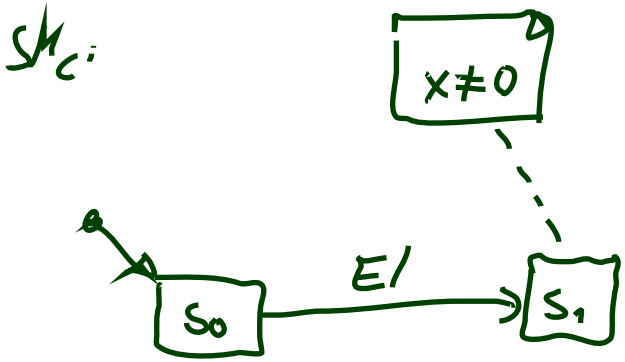
  If the developer makes a mistake, then $\mathcal{M}'$ is inconsistent.

- **Not common**: if $\mathscr{SM}$ is given, then constraints are also considered when choosing transitions in the RTC-algorithm. In other words: even in presence of mistakes, the $\mathscr{SM}$ never move to inconsistent configurations.

# Pragmatics: Example




no init. val expr.






(1) $\mathcal{M}$ is not consistent ("broken") because there is a comp. path leading to a $(\sigma, \varepsilon)$ s.t. $\sigma \not\models Inv(\mathcal{M})$

(2) "dear developer, I want
- states $s_0$, $s_1$
- $E$ must move me from $s_0$ to $s_1$
- in $s_1$, x must not be 0

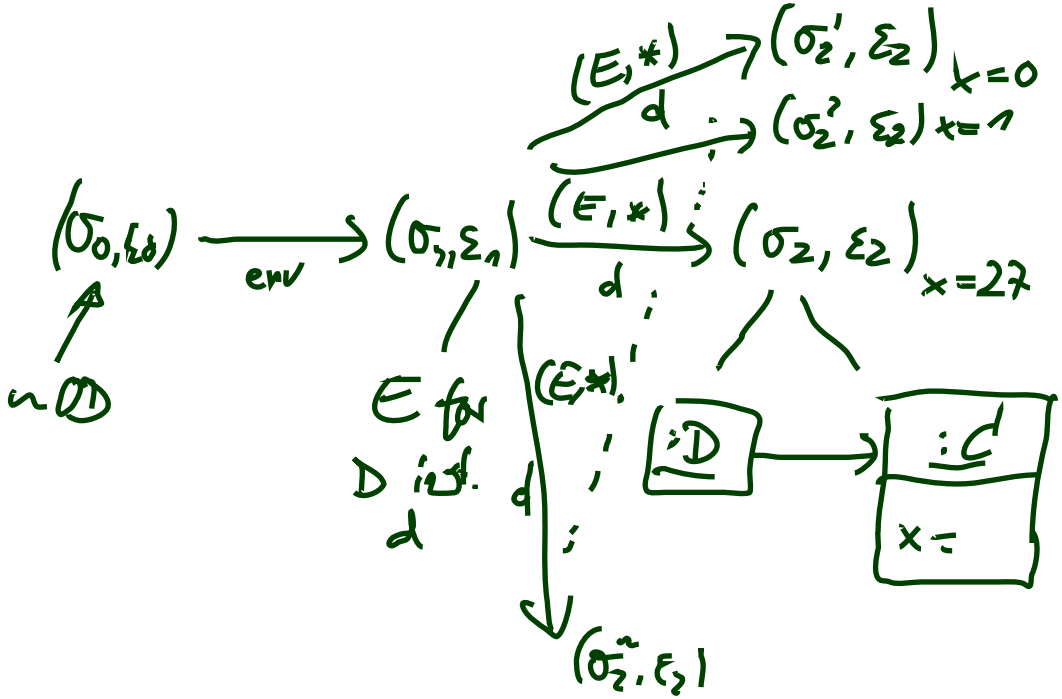
DO THAT!"

IN EACH SYSTEM STATE $\sigma$,

FOR EACH ALIVE OBJECT $v \in dom(\sigma)$, $v \in \mathcal{D}(C)$

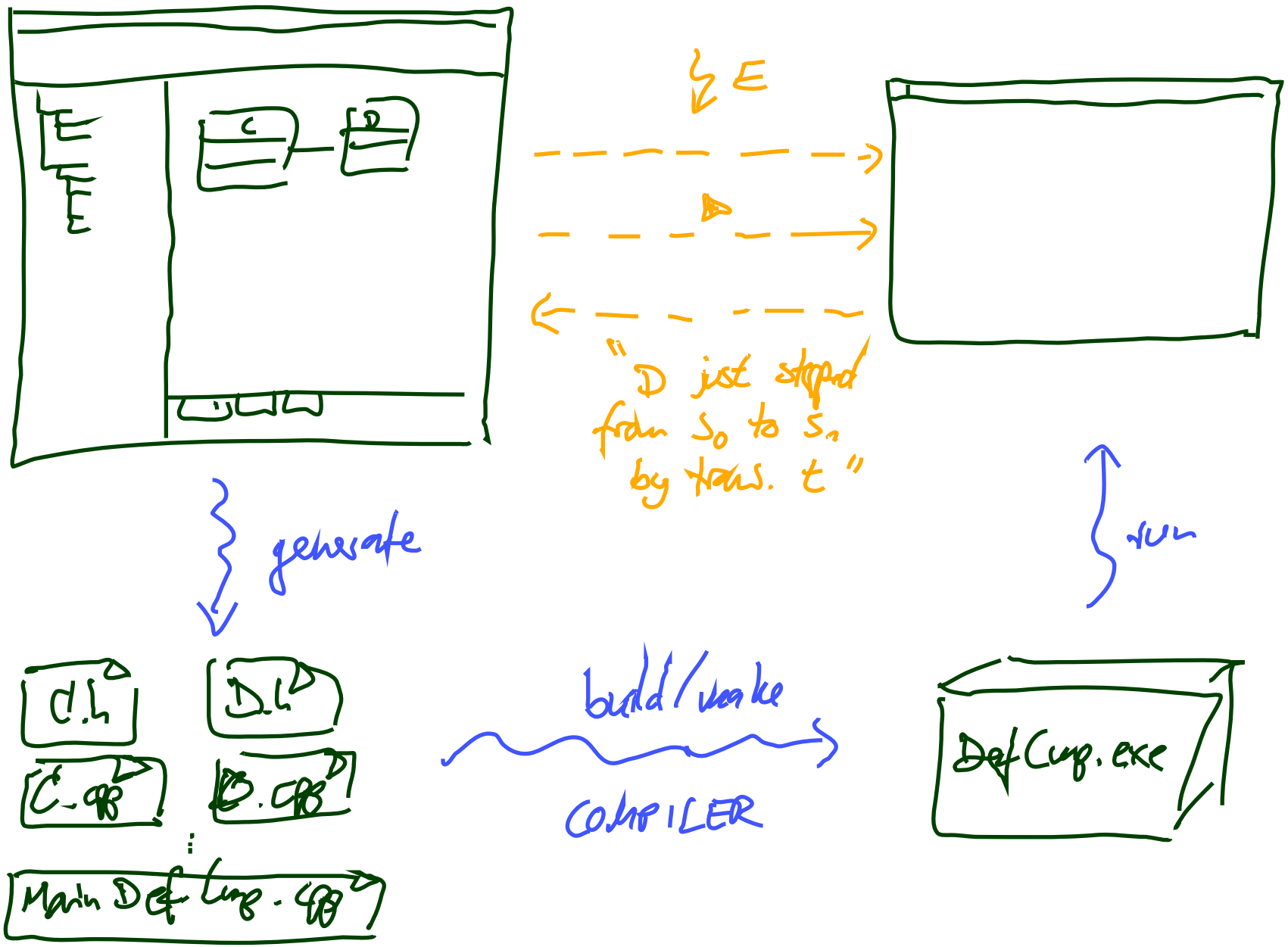
EACH OF $v$'s ATTRIBUTES HAS A (DEFINITE) VALUE!

$$\forall v \in alt(C) \bullet \sigma(v)(v) \in \mathcal{D}(type(v))$$



# *Rhapsody Demo II*

Rhapsody


E
"D just stepped from $S_0$ to $S_n$ by trans. t"
generate
run
C.h
D.h
C.cpp
D.cpp
Main DefCmp.cpp
build/make
COMPILER
DefCmp.exe