

Software Design, Modelling and Analysis in UML

Lecture 2: Semantical Model

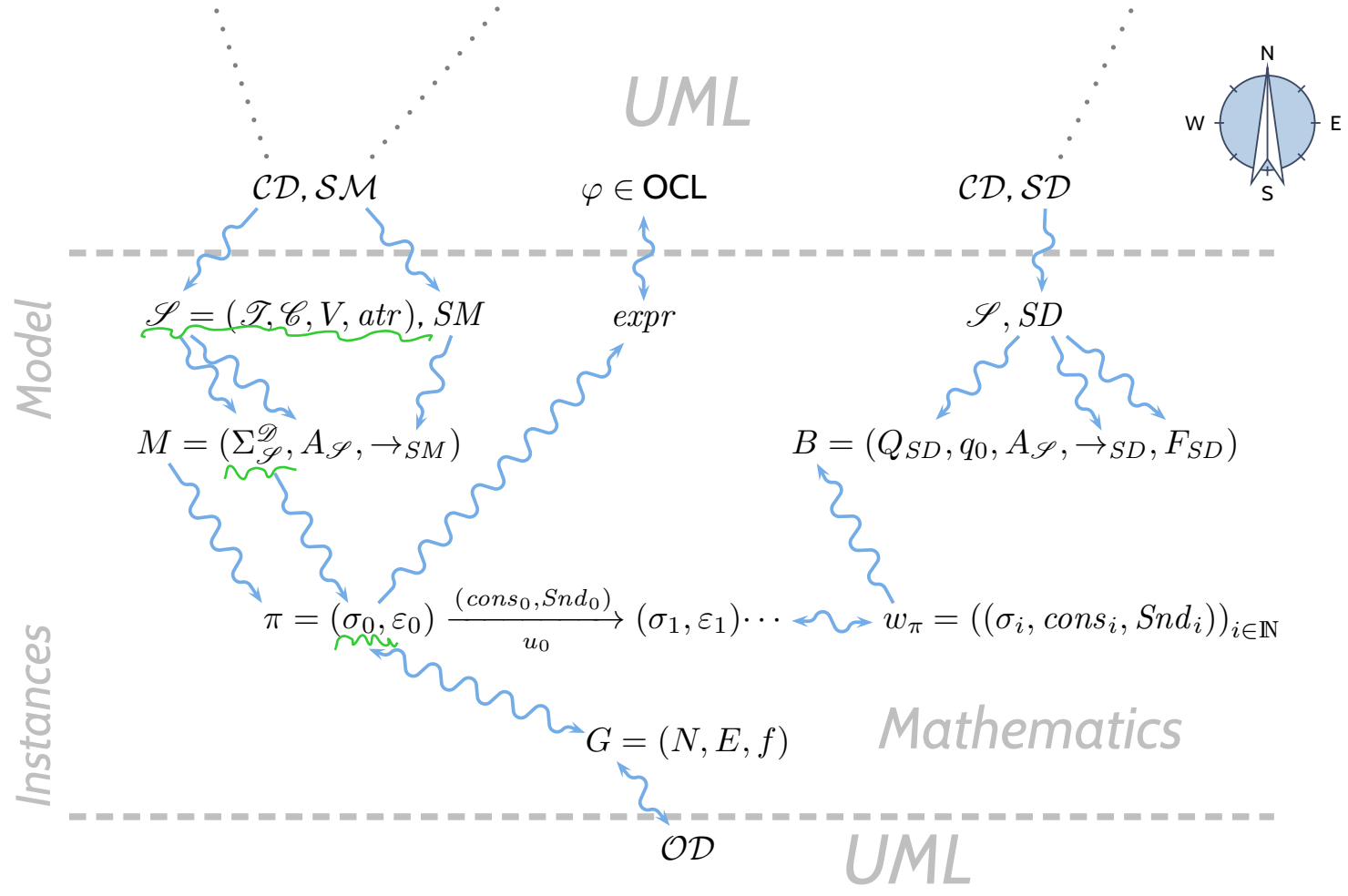
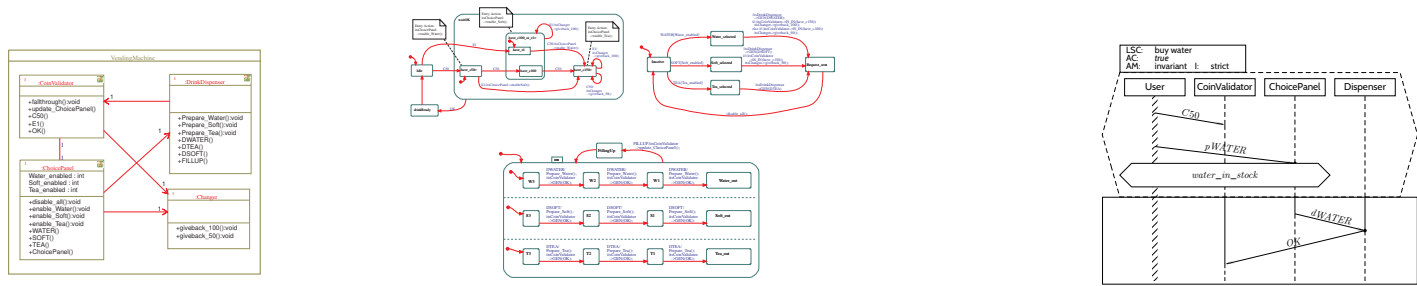
2016-10-20

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

Jochen Hoenicke

Albert-Ludwigs-Universität Freiburg, Germany

Course Map



- **Basic Object System Signature**

- (basic) types, classes,
- typed attributes,
- attribute mapping.

- **Basic Object System Structure**

- objects / object identities,
- domains of basic and derived types.

- **System State**

- concrete and symbolic,
- dangling references,

- **A Complete Example**

Semantical Foundation

Basic Object System Signature

Definition. A (Basic) Object System **Signature** is a quadruple

$$\mathcal{S} = (\mathcal{T}, \mathcal{C}, V, atr) \quad C_1, C_2$$

where

- \mathcal{T} is a set of (basic) **types**,
- \mathcal{C} is a finite set of **classes**,
- V is a finite set of **typed attributes**, i.e., each $v \in V$ has a type
 - $\tau \in \mathcal{T}$, or
 - $C_{0,1}$ or C_* , where $C \in \mathcal{C}$(written $v : \tau$ or $v : C_{0,1}$ or $v : C_*$),
- $atr : \mathcal{C} \rightarrow 2^V$ maps each class to its set of attributes.

Class name
total functions
Power-Set of V

Note: Inspired by OCL 2.0 standard [OMG \(2006\)](#), Annex A.

Basic Object System Signature Example

$\mathcal{S} = (\mathcal{T}, \mathcal{C}, V, atr)$ where

- (basic) **types** \mathcal{T} and **classes** \mathcal{C} (both finite),
- **typed attributes** V , τ from \mathcal{T} , or $C_{0,1}$ or C_* , for some $C \in \mathcal{C}$,
- $atr : \mathcal{C} \rightarrow 2^V$ mapping classes to attributes.

Example:

$\mathcal{S}_0 = (\{Int\}, \{C, D\}, \{x : Int, p : C_{0,1}, n : C_*\}, \{C \mapsto \{p, n\}, D \mapsto \{x\}\})$

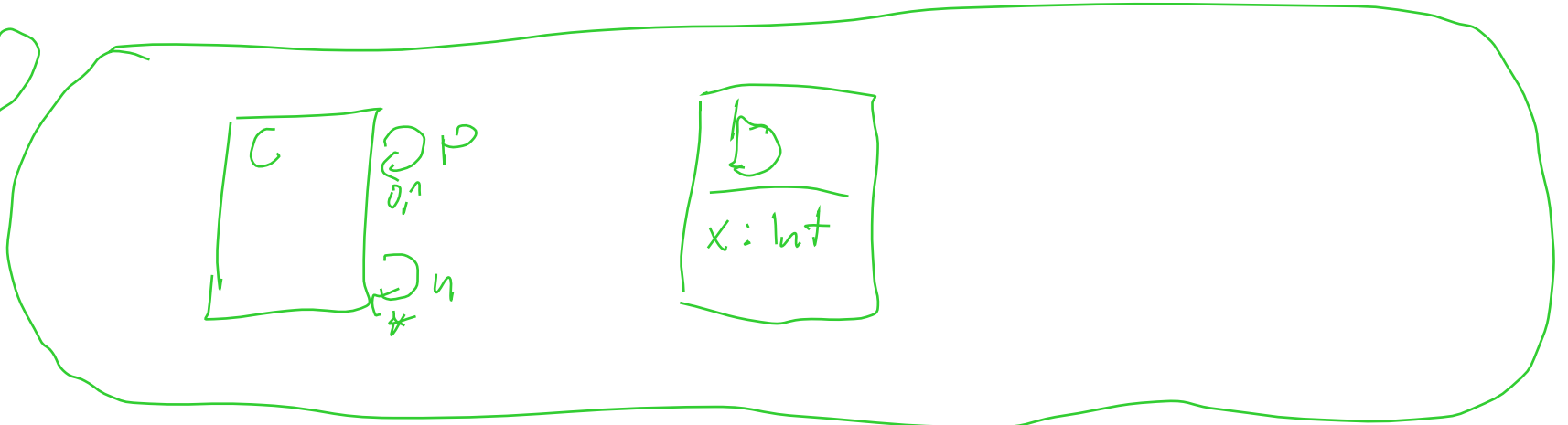
\mathcal{T} : set of basic type

Attributes V

atr

set of classe \mathcal{C}

$atr(C) = \{p, n\}$
 $atr(D) = \{x\}$



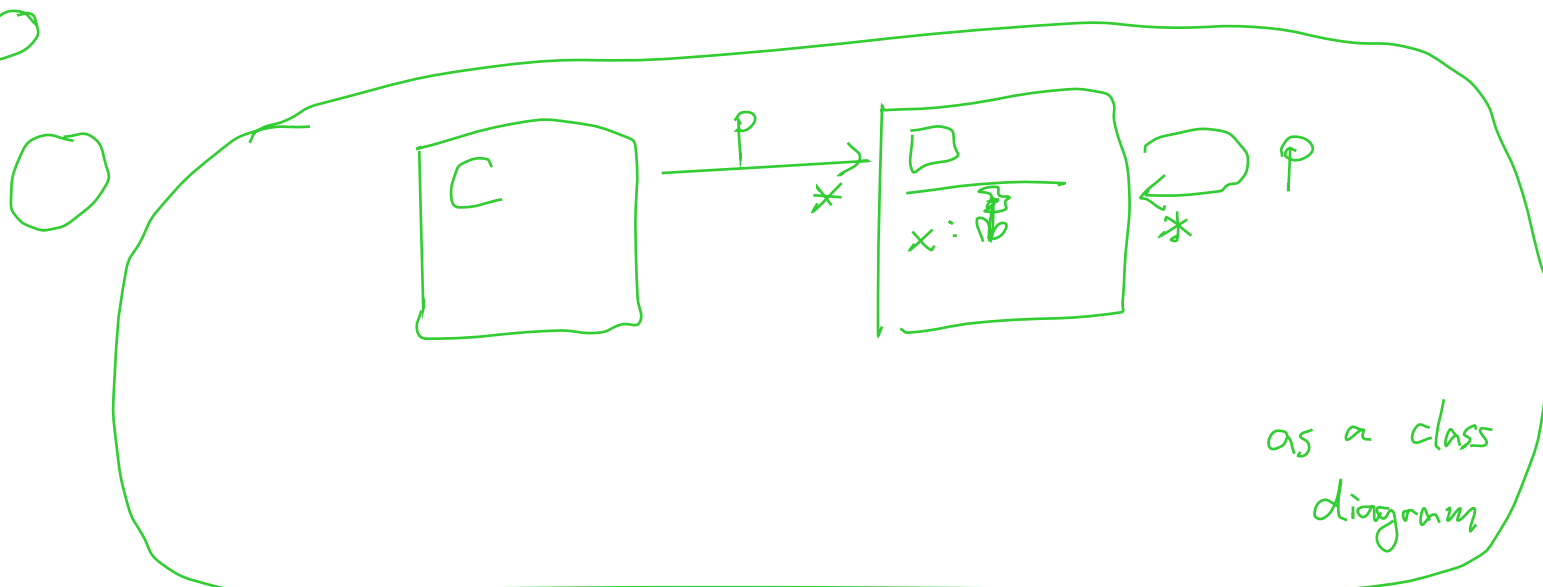
Basic Object System Signature Another Example

$\mathcal{S} = (\mathcal{T}, \mathcal{C}, V, atr)$ where

- (basic) **types** \mathcal{T} and **classes** \mathcal{C} (both finite),
- **typed attributes** V , τ from \mathcal{T} , or $C_{0,1}$ or C_* , for some $C \in \mathcal{C}$,
- $atr : \mathcal{C} \rightarrow 2^V$ mapping classes to attributes.

Example:

$$\mathcal{S}_1 = (\{\mathbb{B}\}, \{MyType\}, \{C, D\}, \{\cancel{x: \text{Int}}, x: \mathbb{B}, p: D_*, q: D_{0,1}\}, \\ \{C \mapsto \{P\}, D \mapsto \{x, P\}\})$$



as a class diagram

Basic Object System Structure

Definition. A Basic Object System **Structure** of $\mathcal{S} = (\mathcal{T}, \mathcal{C}, V, atr)$ is a domain function \mathcal{D} which assigns to each type a domain, i.e.

- $\tau \in \mathcal{T}$ is mapped to $\mathcal{D}(\tau)$, $\mathcal{D}(\mathcal{T})$ is a set
- $C \in \mathcal{C}$ is mapped to an infinite set $\mathcal{D}(C)$ of (object) identities.
Note: Object identities only have the "=" operation.
- Sets of object identities for different classes are disjoint, i.e.

$$\forall C, D \in \mathcal{C} : C \neq D \rightarrow \mathcal{D}(C) \cap \mathcal{D}(D) = \emptyset.$$

- C_* and $C_{0,1}$ for $C \in \mathcal{C}$ are mapped to $2^{\mathcal{D}(C)}$.

We use $\mathcal{D}(\mathcal{C})$ to denote $\bigcup_{C \in \mathcal{C}} \mathcal{D}(C)$; analogously $\mathcal{D}(\mathcal{C}_*)$.

$$\begin{aligned} \mathcal{D}(C_*) &:= 2^{\mathcal{D}(C)} \\ \mathcal{D}(C_{0,1}) &:= 2^{\mathcal{D}(C)} \end{aligned}$$

$$\mathcal{D}(\mathcal{C}_*) = \bigcup_{C \in \mathcal{C}} \mathcal{D}(C_*)$$

Note: We identify objects and object identities, because both uniquely determine each other (cf. OCL 2.0 standard).

Basic Object System Structure Example

Wanted: a structure for signature

$$\mathcal{S}_0 = (\{Int\}, \{C, D\}, \{x : Int, p : C_{0,1}, n : C_*\}, \{C \mapsto \{p, n\}, D \mapsto \{x\}\})$$

\mathcal{D} needs to map:

- $\tau \in \mathcal{T}$ to **some** $\mathcal{D}(\tau)$,
- $C \in \mathcal{C}$ to **some** set of identities $\mathcal{D}(C)$ (infinite, disjoint for different classes),
- C_* and $C_{0,1}$ for $C \in \mathcal{C}$: always mapped to $\mathcal{D}(C_*) = \mathcal{D}(C_{0,1}) = 2^{\mathcal{D}(C)}$.

$$\mathcal{D}(Int) = \mathbb{Z}$$

$$\mathcal{D}(C) = \mathbb{N}^+ \times \{C\} \cong \{1_C, 2_C, 3_C, 4_C, \dots\}$$

$$\mathcal{D}(D) = \mathbb{N}^+ \times \{D\} \cong \{1_D, 2_D, 3_D, \dots\}$$

$$\mathcal{D}(C_{0,1}) = \mathcal{D}(C_*) = 2^{\mathcal{D}(C)}$$

$$\mathcal{D}(D_{0,1}) = \mathcal{D}(D_*) = 2^{\mathcal{D}(D)}$$

$$\begin{aligned} \mathcal{D}(Int) &= \{-2, -1, 0, 1, 2\} \\ \mathcal{D}(C) &= \{a, aa, aaa, \dots\} \\ \mathcal{D}(D) &= \{b, bb, bbb, \dots\} \end{aligned}$$

System State

Definition. Let \mathcal{D} be a structure of $\mathcal{S} = (\mathcal{I}, \mathcal{C}, V, atr)$.

A **system state** of \mathcal{S} wrt. \mathcal{D} is a **type-consistent** mapping

$$\sigma : \mathcal{D}(\mathcal{C}) \rightarrow (V \rightarrow (\mathcal{D}(\mathcal{I}) \cup \mathcal{D}(\mathcal{C}_*)))$$

That is, for each $u \in \mathcal{D}(C)$, $C \in \mathcal{C}$, if $u \in \text{dom}(\sigma)$ and all subset of object identifies

- $\text{dom}(\sigma(u)) = atr(C)$
 - $(\sigma(u))(v) \in \mathcal{D}(\tau)$ if $v : \tau, \tau \in \mathcal{I}$
 - $\sigma(u)(v) \in \mathcal{D}(D_*)$ if $v : D_{0,1}$ or $v : D_*$ with $D \in \mathcal{C}$
- } type-consistent

We call $u \in \mathcal{D}(\mathcal{C})$ **alive** in σ if and only if $u \in \text{dom}(\sigma)$.

We use $\Sigma_{\mathcal{D}}$ to denote the set of all system states of \mathcal{S} wrt. \mathcal{D} .

System State Example

$$\mathcal{S}_0 = (\{Int\}, \{C, D\}, \{x : Int, p : C_{0,1}, n : C_*\}, \{C \mapsto \{p, n\}, D \mapsto \{x\}\})$$

$$\mathcal{D}(Int) = \mathbb{Z}, \quad \mathcal{D}(C) = \{1_C, 2_C, 3_C, \dots\}, \quad \mathcal{D}(D) = \{1_D, 2_D, 3_D, \dots\}$$

Wanted: $\sigma : \mathcal{D}(\mathcal{C}) \rightarrow (V \rightarrow (\mathcal{D}(\mathcal{I}) \cup \mathcal{D}(\mathcal{C}_*)))$ such that (i) $\text{dom}(\sigma(u)) = \text{atr}(C)$, and
 (ii) $\sigma(u)(v) \in \mathcal{D}(\tau)$ if $v : \tau, \tau \in \mathcal{I}$, (iii) $\sigma(u)(v) \in \mathcal{D}(C_*)$ if $v : D_*$ with $D \in \mathcal{C}$.

$\sigma_0 = \emptyset$ ("empty function")

alive in σ_0 : none

$$\sigma_1 = \left\{ 1_C \mapsto \left\{ \begin{array}{l} p \mapsto \emptyset \\ n \mapsto \{1_C, 5_C\} \end{array} \right\}, 5_C \mapsto \left\{ \begin{array}{l} p \mapsto \{1_C\} \\ n \mapsto \emptyset \end{array} \right\}, 3_D \mapsto \{x \mapsto 3\} \right\}$$

$$\sigma_2 = \left\{ 1_D \mapsto \{x \mapsto 27\}, 2_D \mapsto \{x \mapsto 27\}, 17_D \mapsto \{x \mapsto 0\} \right\}$$

System State Example

$$\mathcal{S}_0 = (\{\text{Int}\}, \{C, D\}, \{x : \text{Int}, p : C_{0,1}, n : C_*\}, \{C \mapsto \{p, n\}, D \mapsto \{x\}\})$$
$$\mathcal{D}(\text{Int}) = \mathbb{Z}, \quad \mathcal{D}(C) = \{1_C, 2_C, 3_C, \dots\}, \quad \mathcal{D}(D) = \{1_D, 2_D, 3_D, \dots\}$$

Wanted: $\sigma : \mathcal{D}(\mathcal{C}) \rightarrow (V \rightarrow (\mathcal{D}(\mathcal{T}) \cup \mathcal{D}(\mathcal{C}_*)))$ such that (i) $\text{dom}(\sigma(u)) = \text{atr}(C)$, and
(ii) $\sigma(u)(v) \in \mathcal{D}(\tau)$ if $v : \tau, \tau \in \mathcal{T}$, (iii) $\sigma(u)(v) \in \mathcal{D}(C_*)$ if $v : D_*$ with $D \in \mathcal{C}$.

Two options:

- **Concrete, explicit** identities:

$$\sigma = \{1_C \mapsto \{p \mapsto \emptyset, n \mapsto \{5_C\}\}, 5_C \mapsto \{p \mapsto \emptyset, n \mapsto \emptyset\}, 1_D \mapsto \{x \mapsto 23\}\}.$$

$1_C \mapsto \emptyset$

- **Alternative: symbolic** system state.

$$\sigma = \{c_1 \mapsto \{p \mapsto \emptyset, n \mapsto \{c_2\}\}, c_2 \mapsto \{p \mapsto \emptyset, n \mapsto \emptyset\}, d \mapsto \{x \mapsto 23\}\}$$

assuming $c_1, c_2 \in \mathcal{D}(C), d \in \mathcal{D}(D), c_1 \neq c_2$.

System State: Spot the 10 (?) Mistakes

$$\mathcal{S}_0 = (\{\text{Int}\}, \{C, D\}, \{x : \text{Int}, p : C_{0,1}, n : C_*\}, \{C \mapsto \{p, n\}, D \mapsto \{x\}\})$$

$$\mathcal{D}(\text{Int}) = \mathbb{Z}, \quad \mathcal{D}(C) = \{1_C, 2_C, 3_C, \dots\}, \quad \mathcal{D}(D) = \{1_D, 2_D, 3_D, \dots\}$$

Wanted: $\sigma : \mathcal{D}(\mathcal{C}) \rightarrow (V \rightarrow (\mathcal{D}(\mathcal{T}) \cup \mathcal{D}(\mathcal{C}_*)))$ such that (i) $\text{dom}(\sigma(u)) = \text{atr}(C)$, and
 (ii) $\sigma(u)(v) \in \mathcal{D}(\tau)$ if $v : \tau, \tau \in \mathcal{T}$, (iii) $\sigma(u)(v) \in \mathcal{D}(C_*)$ if $v : D_*$ with $D \in \mathcal{C}$.

- $\sigma = \{1_C \mapsto \{p \mapsto \emptyset, n \mapsto \{5_C\}\}, 5_C \mapsto \{p \mapsto \emptyset, n \mapsto 1_C\}, 1_D \mapsto \{x \mapsto 2.3\}\}$.
(iii) $\uparrow \notin \mathcal{D}(C_*)$ (ii) $\uparrow \notin \mathcal{D}(\text{Int})$
- $\sigma = \{1_C \mapsto \{p \mapsto \emptyset, n \mapsto \{5_C\}\}, 5_C \mapsto \{p \mapsto 1_C, n \mapsto \emptyset\}, 1_D \mapsto \{x \mapsto 23\}\}$.
(iii) $\uparrow \notin \mathcal{D}(C_*)$
- $\sigma = \{1_C \mapsto \{p \mapsto \emptyset, n \mapsto \{1_D\}\}, 5_C \mapsto \{p \mapsto \emptyset, n \mapsto \emptyset\}, 1_D \mapsto \{x \mapsto 22\}\}$.
(iii) $\uparrow \notin \mathcal{D}(C_*)$
- $\sigma = \{1_C \mapsto \{p \mapsto \emptyset, n \mapsto \{5_C\}\}, 5_C \mapsto \{n \mapsto \emptyset\}, 1_D \mapsto \{x \mapsto 1, p \mapsto \{1_C\}\}\}$.
(ii) $\notin \text{dom}(\sigma(5_C))$ →
- $\sigma = \{1_C \mapsto \{p \mapsto \emptyset, n \mapsto \{5_C\}\}, 5_C \mapsto \{p \mapsto \emptyset, n \mapsto \{9_C\}\}\}$
(iii): $\{9_C\} \in \mathcal{D}(C_*)$

Dangling References

Definition. Let $\sigma \in \Sigma_{\mathcal{D}}$ be a system state.

We say attribute $v \in V_{0,1,*}$, i.e. $v : C_{0,1}$ or $v : C_*$, in object $u \in \text{dom}(\sigma)$ has a **dangling reference** if and only if the attribute's value comprises an object which is not alive in σ , i.e. if

$$\sigma(u)(v) \notin \text{dom}(\sigma).$$

We call σ **closed** if and only if no attribute has a dangling reference in any object alive in σ .

Example:

- $\sigma = \{1_C \mapsto \{p \mapsto \emptyset, n \mapsto \{5_C\}\}\}$

A Complete Example: Vending Machine Structure

$$\mathcal{J} = (\{ \text{Int}, \text{Bool} \}, \\ \{ \text{VM}, \text{CP}, \text{DD} \}, \\ \{ \text{cp} : \text{CP}_*, \text{dd} : \text{DD}_{0,1}, \text{win} : \text{Int}, \text{wen} : \text{Bool} \}, \\ \{ \text{VM} \mapsto \{ \text{cp}, \text{dd} \} \\ \text{CP} \mapsto \{ \text{wen} \} \\ \text{DD} \mapsto \{ \text{win} \} \})$$

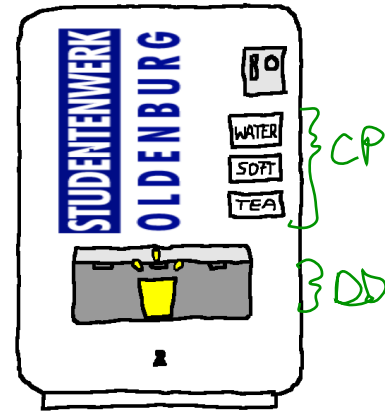
$$\mathcal{D}(\text{Bool}) = \{ \text{true}, \text{false} \} \quad \mathcal{D}(\text{Int}) = \mathbb{Z}$$

$$\mathcal{D}(\text{VM}) = \{ 1_{\text{VM}}, 2_{\text{VM}}, \dots \}$$

$$\mathcal{D}(\text{DD}) = \{ \dots \}$$

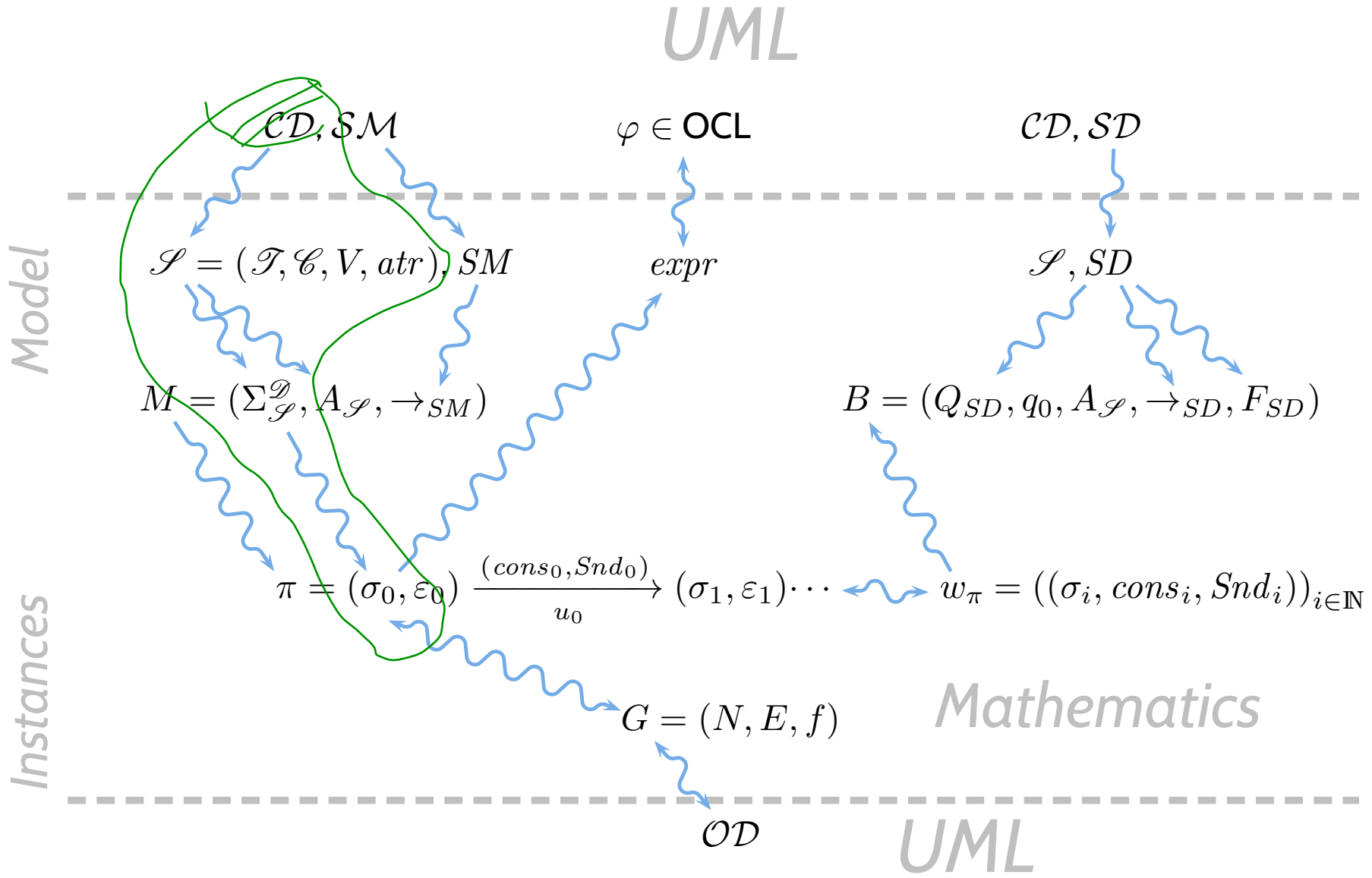
$$\sigma = \{ 1_{\text{VM}} \mapsto \{ \text{dd} \mapsto \{ 1_{\text{DD}} \}, \text{cp} \mapsto \{ 42_{\text{CP}}, 3_{\text{CP}} \} \}$$

$$1_{\text{DD}} = \{ \text{win} \mapsto 10 \}, \quad 42_{\text{CP}} = \{ \text{wen} \mapsto \text{true} \}, \quad 3_{\text{CP}} = \{ \text{wen} \mapsto \text{false} \} \}$$



You Are Here.

Course Map



Tell Them What You've Told Them. . .

- We can directly use **object system signatures** to model the structure of systems.
 - We don't **need** diagrams, they will be more pleasant to read.
- We introduce
 - **basic types** and **classes**,
 - basic type and derived type **attributes**, and
 - assign to each class a set of attributes.
- **Object system structures** provide domains for basic and derived types.
- An **object system signature** \mathcal{S} and an **object system structure** \mathcal{D} uniquely define the set $\Sigma_{\mathcal{S}}^{\mathcal{D}}$ of **system states**.
- **Outlook:**
 - **Object system signatures** will be used to capture the **abstract syntax** of class diagrams.
 - **OCL expressions** will be evaluated on **system states**.
 - **State machines** will define sequences of **system configurations** (consisting of a **system state** and an **ether**).

References

References

OMG (2006). Object Constraint Language, version 2.0. Technical Report formal/06-05-01.

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.