5.2 Component-based Development

Schemas as OOD Frameworks

In next-generation component-based software development (CBD) . . .

. . . **Object-Oriented Design** (**OOD**) frameworks will be an important kind of **components**

OOD frameworks are . . .

. . . specification frameworks with **state transitions**

So (correct) schemas can be used in CBD

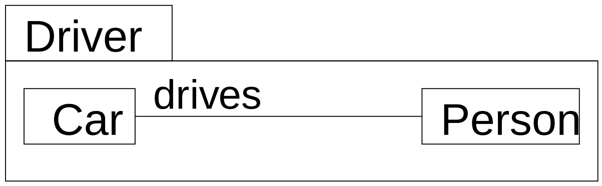
. . . to develop (correctly reusable) programs in OOD frameworks

. . . i.e. **methods** for (**groups** of) **objects**.

OOD Frameworks

- An OOD framework is a group of (interacting) objects

  For example, in the CBD methodology *Catalysis*, a driver may be represented as the OOD framework:



- OOD frameworks are better units of *reuse* ...

  ... than objects (used in existing OOD/CBD methods)

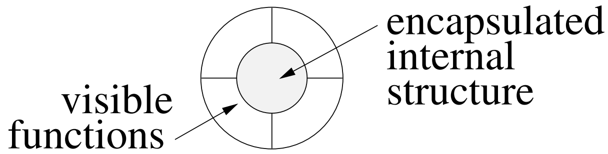
  because in practical systems, objects tend to have multiple roles in different contexts ...

  ... and OOD frameworks can capture this.

Objects with Single Roles

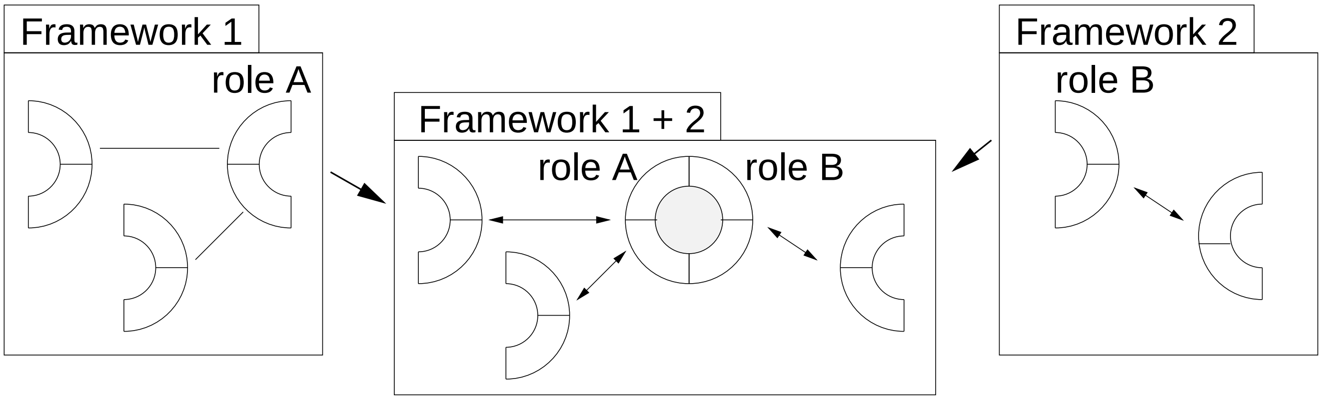
Existing OOD/CBD methods

- use objects as units of reuse
- view an object as a closed entitiy with one role:

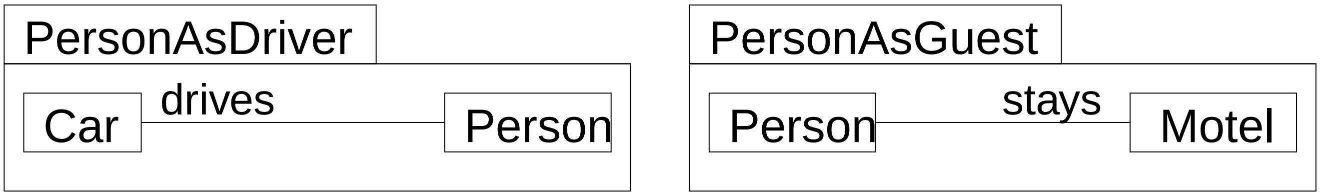


Objects with Multiple Roles in Different OOD frameworks

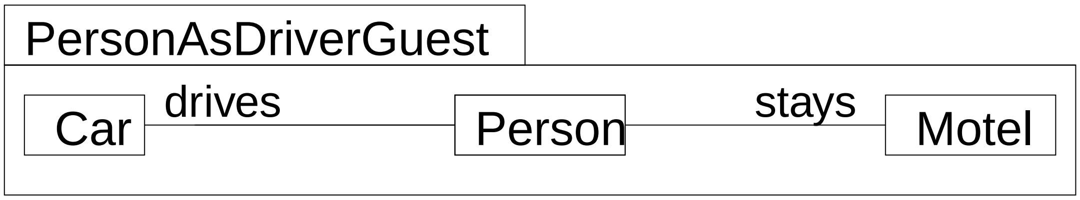
OOD frameworks allow objects with multiple roles:



For example, a person can play the roles of a *driver* and a *guest* at a motel simultaneously:



If we compose these two frameworks, we get:



OBJ-Frameworks

- An OBJ-framework is a *class* (of objects).

Example

The Car class can be defined as the following *open* framework:

$$
\begin{array}{ll}
\multicolumn{2}{l}{\textbf{OBJ-Framework}\ \mathcal{CAR}(.km, .option);} \\
\text{IMPORT:} & \mathcal{INT}, year96; \\
\text{DECLS:} & .km \ : \ [\,] \to Int \\
 & .option \ : \ [year96.opts] \\
\text{CONSTRS:} & .km \geq 0
\end{array}
$$

where:

$\mathcal{INT}$ is a predefined ADT of integers

$year96$ is an object that contains the sort $year96.opts$ of the possible options for a car in the year 96

the constraint $.km \geq 0$ is an axiom for the open symbol $.km$.

- Objects are (*closed*) *instances* of OBJ-frameworks.

  E.g. an object of class $\mathcal{CAR}$ can be created by:

$$
\begin{array}{llll}
\multicolumn{4}{l}{\text{NEW}\ spider : \mathcal{CAR};} \\
\text{CLOSE:} & spider.km & \text{BY} & spider.km = 25000; \\
 & spider.option & \text{BY} & spider.option(x) \leftrightarrow \ x = Airbag \vee \\
 & & & \qquad x = AirCond.
\end{array}
$$

- The axioms used to close an OBJ-framework into an object represent the **state** of the object.

OOD-Frameworks

An OOD framework is a composite OBJ-framework.

It can be viewed as a *system* of objects, in which objects can be created (and deleted) and updated dynamically.

Example

The PersonAsDriver OOD framework can be formalised as the following framework:

OOD-Framework $\mathcal{DRIVER}[\mathcal{PERSON}, \mathcal{CAR}]$;

$$
\begin{array}{lll}
\text{DECLS:} & .drives : [obj]; & \\
\text{CONSTRS:} & `X.drives(c) \rightarrow & PERSON(`X) \wedge `X.age \geq 18 \wedge \\
& & CAR(c); \\
& (\exists c : obj)(.drives(c)); &
\end{array}
$$

where:

obj is a reserved sort symbol that contains the set of names of all existing objects in the system

$`X$ is a meta-symbol that stands for any object name

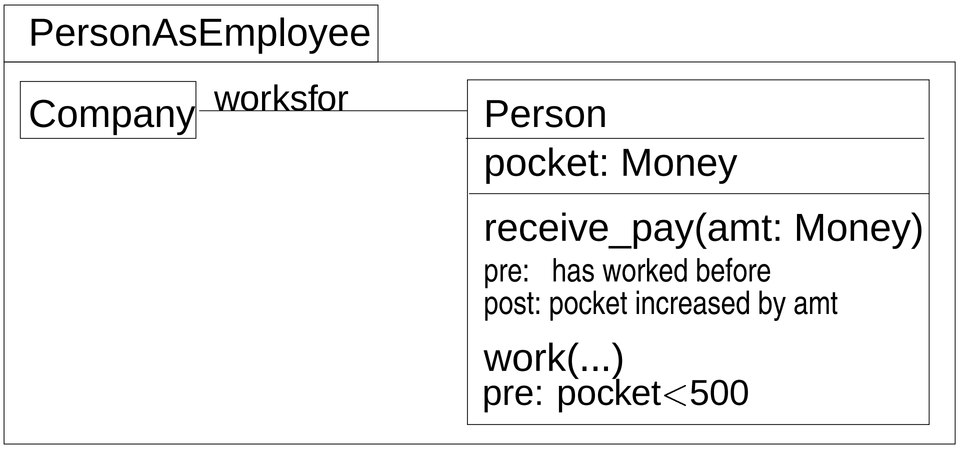
$\mathcal{PERSON}$ is an OBJ-framework.

A driver object is thus a composite object …

… of person and car objects.

OOD Frameworks with State Transition Axioms

Consider the following *Catalysis* OOD-framework, where a person plays the role of a company employee:



OBJ-Framework $\mathcal{PERSON}$;

IMPORT: $\mathcal{MONEY}$

DECLS:
$$\begin{array}{rcl} .pocket & : & [\,] \to Money; \\ .receive\_pay & : & [Money]; \\ .work & : & [\ldots]; \end{array}$$

AXIOMS: $\ldots$

ST-AXIOMS:

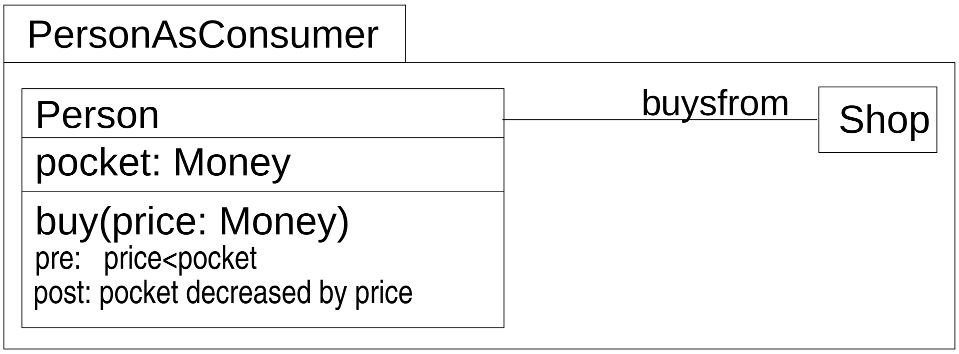
$$\forall_a \rhd .receive\_pay(a) \Rightarrow P \odot .work(\ldots);$$
$$\forall_{a,n} \odot .receive\_pay(a) \Rightarrow Y(.pocket = n) \Rightarrow .pocket = n + a;$$
$$\rhd .work(\ldots) \Rightarrow .pocket < \pounds 500.$$

The ST-AXIOMS (the *state transition axioms*)

- specify the (dynamic) behaviour model (**event structure**)
- do not affect the (static) isoinitial model.

Composing OOD Frameworks with State Transition Axioms

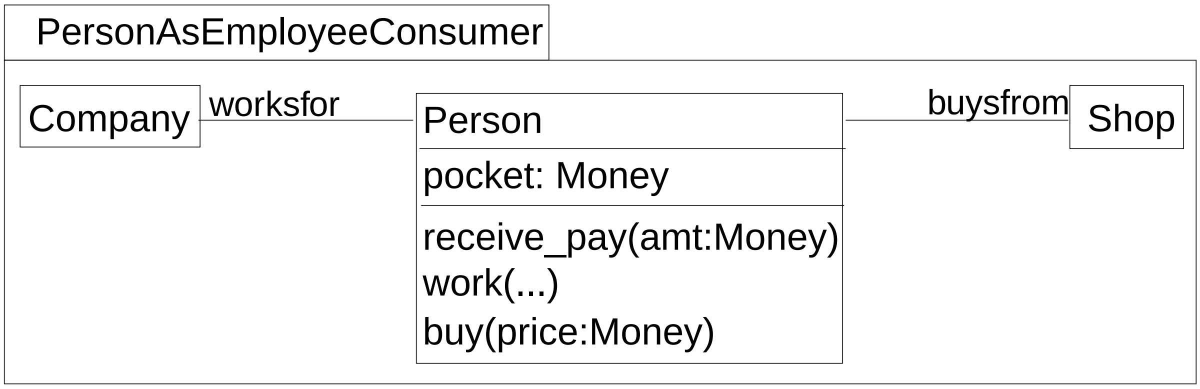
We can combine the PersonAsEmployees OOD-framework with the following, in which a person plays the role of a consumer:



with ST-AXIOMS:

$$\forall_p \rhd .buy(p) \Rightarrow .pocket > p$$
$$\forall_{p,n} \odot .buy(p) \Rightarrow Y(.pocket = n) \Rightarrow .pocket = n - p$$

into:



with the union of the ST-AXIOMS of the two frameworks

(and a ‘composition’ of their event structures)