# Transformation

$A$

$f$

$B$

A transformation $f$
maps any input of type $A$
to the output of type $B$.

# Composition



$$\text{chain\_of}(f, g)$$

Transformations with compatible input and output can be composed.

# Composition is a Transformation



Crucially, composition of transformations is again a transformation.

# Composition Combinator

*Add transformation*

*Add transformation*

Composition $\text{chain\_of}(\square, \square)$ is a transformation combinator with two arguments.

# Example: Composition



Maps an employee to the associated department

Maps a department record to its name

Maps employee records to the name of their department

$\text{chain\_of}(\text{department}, \text{name})$

# Counter-example: Plural Component

Dept

employee

$\text{Block}_{0:N}\{\text{Emp}\}$

*Maps a department to the collection of associated employees*

Emp

salary

Int

*Maps an employee to their salary*

*Collection of employee records*

We cannot compose these transformations because their input and output do not quite match.

# Counter-example: Optional Component

Emp

manager

*Maps an employee to their manager, if there is one*

$\text{Block}_{0:1}\{\text{Emp}\}$

Emp

salary

Int

*Collection of 0 or 1 employee records*

**?**

Even so, the input and the output share a common component,
which suggests there should be a way to compose these transformations.

# Idea: Unbundle the Wire

Unbundle the composite wire.

Compose using the object wire.

Dept

employee

$Block_{0:N}\{Emp\}$

Emp

salary

Int

Dept

employee

$Block_{0:N}\{Emp\}$

$Block_{0:N}$    Emp

*"Functor wire"*
*Signals the number of*
*output elements*

*"Object wire"*
*Transmits the output*
*elements*

Dept

employee

$Block_{0:N}\{Emp\}$

Emp

salary

$Block_{0:N}$    Int

Attaching a transformation to the object wire
indicates that the transformation is applied
to all element of the collection.

# Block Type

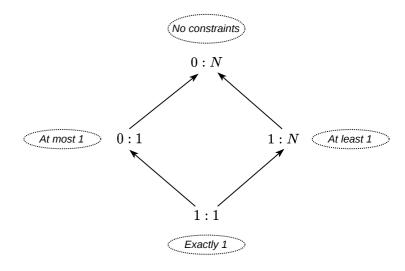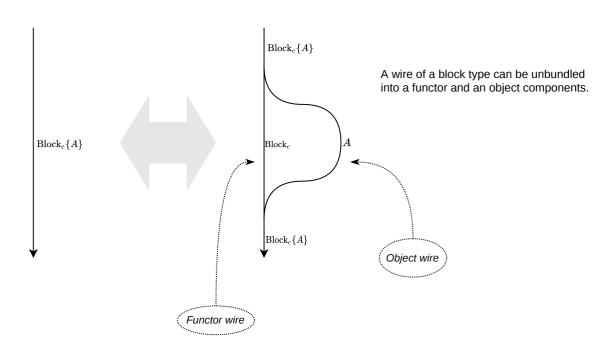A block is a collection
of homogeneous elements.

Cardinality is a constraint
on the number of elements in a block.

*Type of elements*

$\mathrm{Block}_c\{A\}$

*Cardinality of the block*

*No constraints*

$$0 : N$$

*At most 1*    $$0 : 1$$        $$1 : N$$    *At least 1*

$$1 : 1$$

*Exactly 1*

# Unbundling

$\mathrm{Block}_c\{A\}$

$\mathrm{Block}_c\{A\}$

$\mathrm{Block}_c\{A\}$

$\mathrm{Block}_c$

$A$

$\mathrm{Block}_c\{A\}$

A wire of a block type can be unbundled into a functor and an object components.

*Object wire*

*Functor wire*

# Object Transformation

$\text{Block}_c\{A\}$

$\text{Block}_c\{A\}$

$A$

$A$

$f$

$\text{Block}_c$

$f$

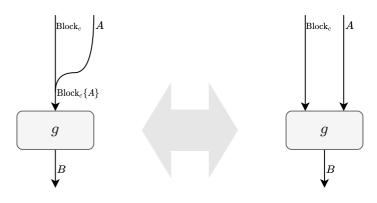$B$

$B$

$\text{Block}_c\{B\}$

$\text{with\_elements}(f)$

Any compatible transformation
can be applied to the object wire,
which indicates that the transformation
is applied to every element of the block.

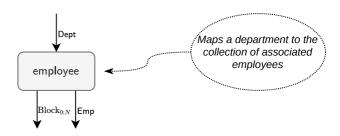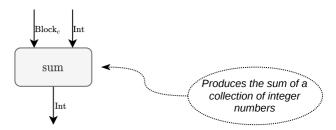# Multiwired Transformations

# Example: Multiwired Transformations
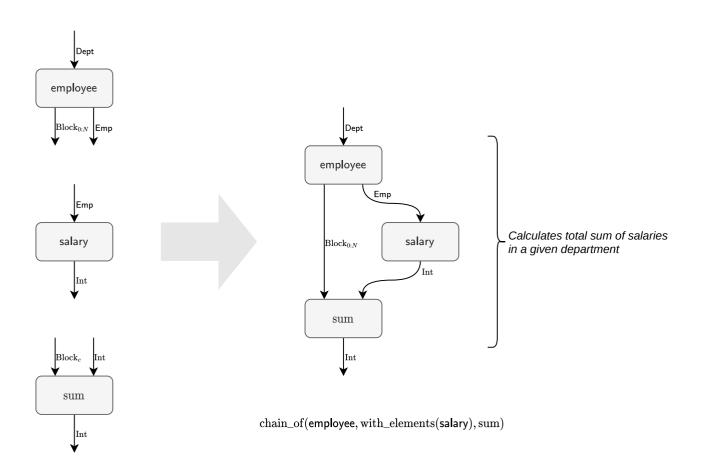
```
         │ Dept
         ▼
    ┌──────────┐          ⋯⋯⋯⋯⋯⋯⋯⋯⋯⋯⋯
    │          │◀⋯⋯⋯⋯⋯   Maps a department to the
    │ employee │          collection of associated
    │          │              employees
    └──────────┘          ⋯⋯⋯⋯⋯⋯⋯⋯⋯⋯⋯
   │            │
   │ Block_{0:N} │ Emp
   ▼            ▼
```
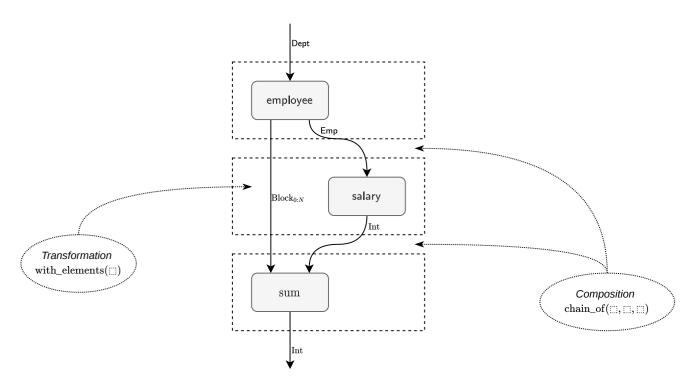
```
   │ Block_c    │ Int
   ▼            ▼
    ┌──────────┐
    │          │◀⋯⋯⋯⋯⋯   ⋯⋯⋯⋯⋯⋯⋯⋯⋯⋯⋯
    │   sum    │          Produces the sum of a
    │          │          collection of integer
    └──────────┘              numbers
         │               ⋯⋯⋯⋯⋯⋯⋯⋯⋯⋯⋯
         │ Int
         ▼
```

# Example: Multiwired Composition



$$\mathrm{chain\_of}(\mathbf{employee}, \mathrm{with\_elements}(\mathbf{salary}), \mathrm{sum})$$

# Example: Multiwired Composition Details



$$\mathrm{chain\_of}(\mathbf{employee}, \mathrm{with\_elements}(\mathbf{salary}), \mathrm{sum})$$
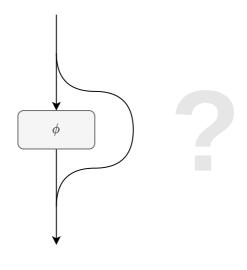
# Challenge: Functor Transformation

So far we have seen different combinations of transformations and wires: transformations that are applied to the object wire, as well as transformations that consume or produce both functor and object wires.

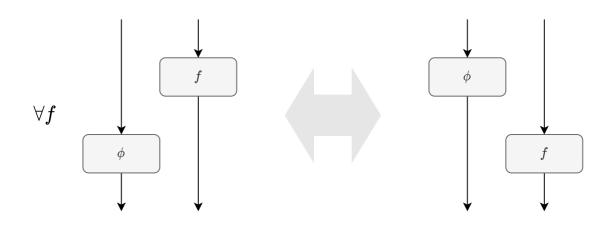What we miss is a notion of a transformation applied to the functor wire.

# Functor Transformation



How to define a functor transformation?

Since a functor transformation does not act on the object wire,
it cannot interfere with any transformations applied to the object wire.

$\forall f$

Such transformations are also called *natural*.

Branch of mathematics that studies natural transformations
is called the *category theory*.