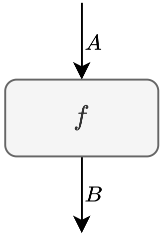
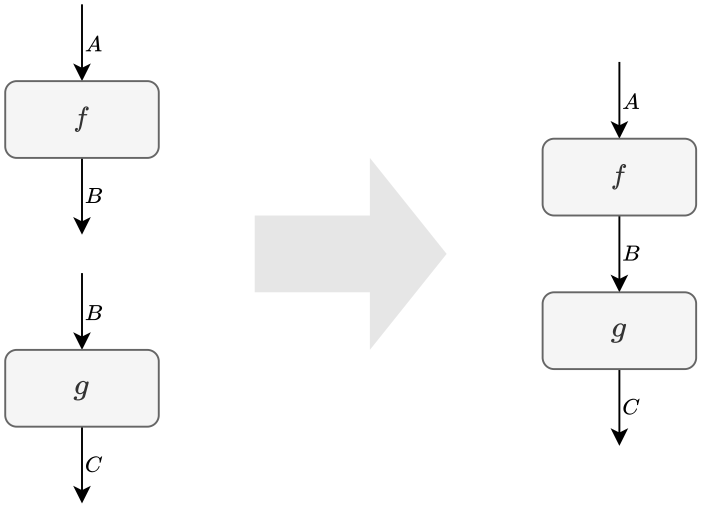
Transformation



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

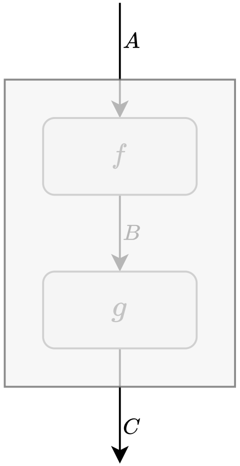
Composition



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

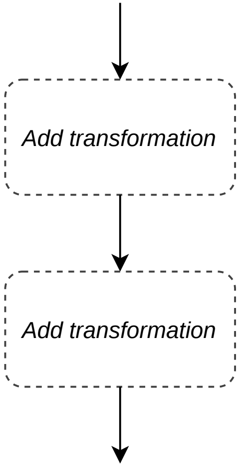
Transformations with compatible input and output can be composed.

Composition is a Transformation



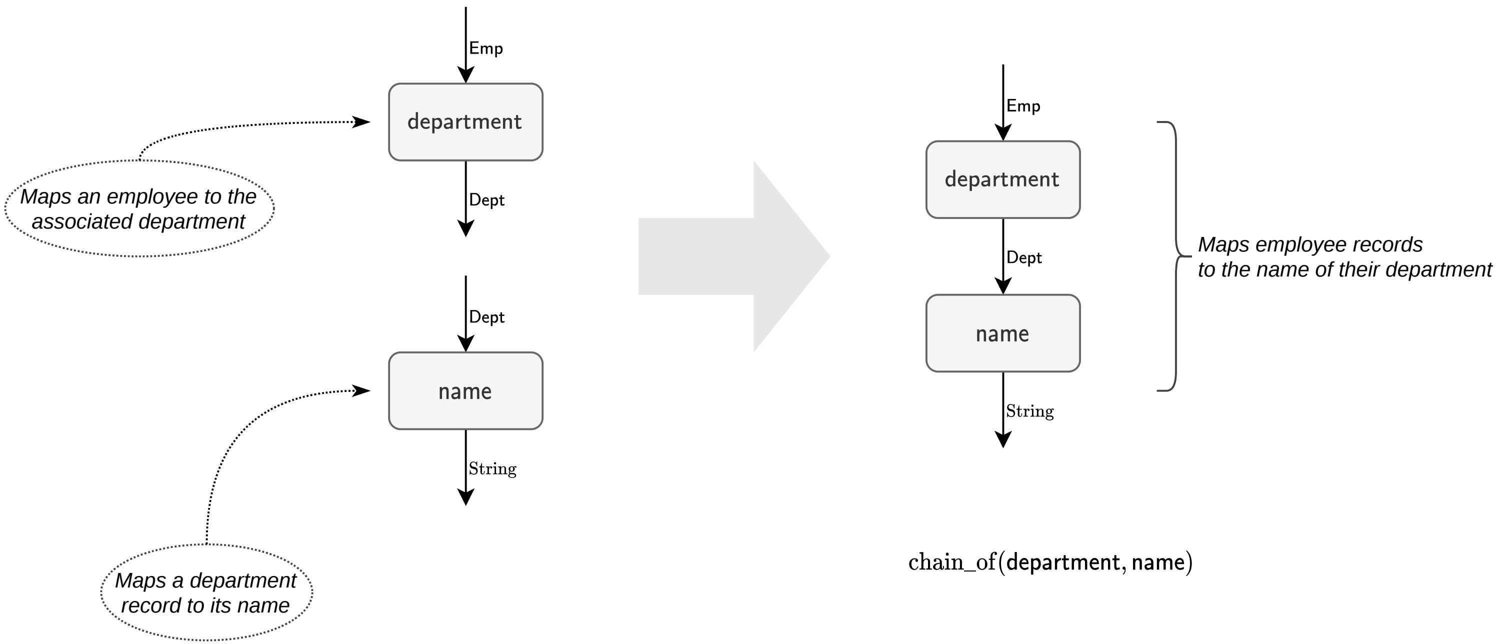
Crucially, composition of transformations
is again a transformation.

Composition Combinator



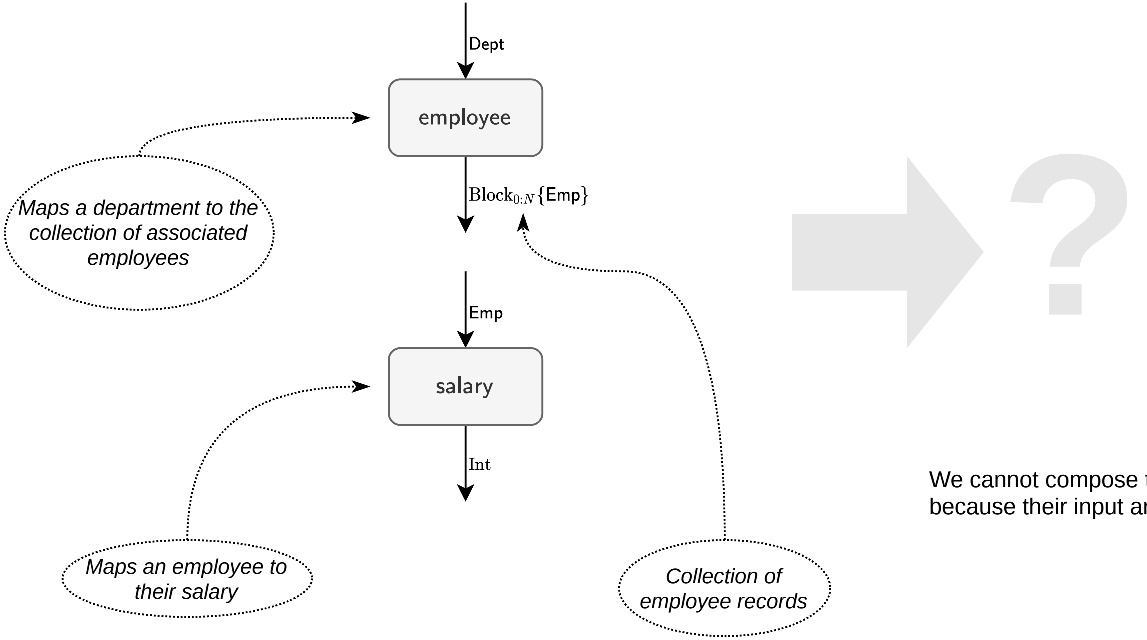
Composition $\mathrm{chain\_of}(\square, \square)$
is a transformation combinator
with two arguments.

Example: Composition



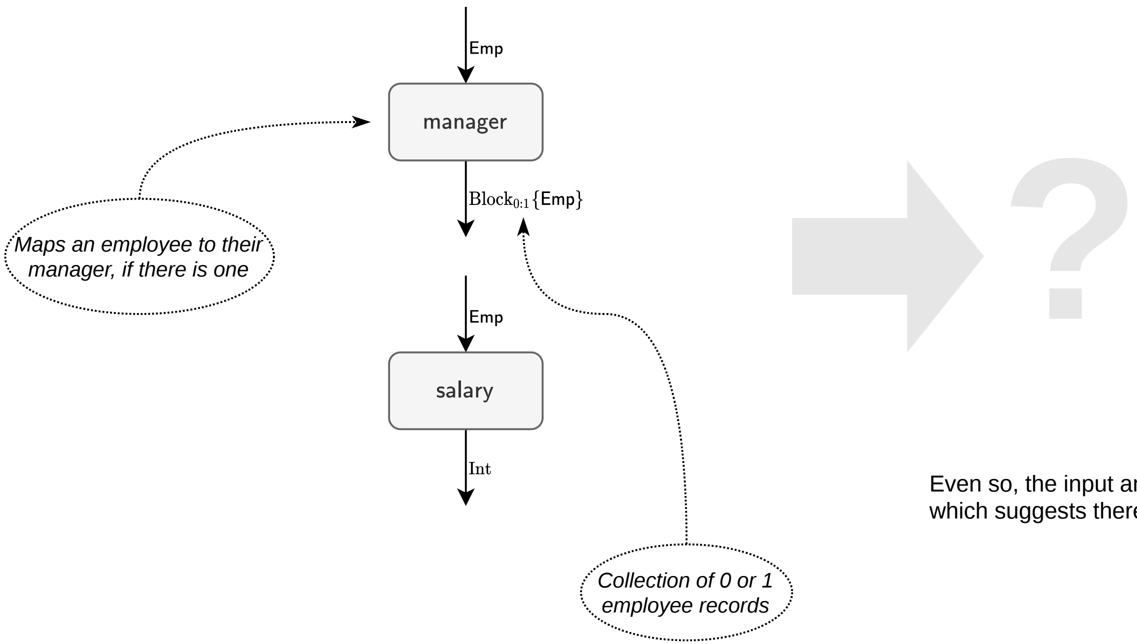
chain_of(**department**, **name**)

Counter-example: Plural Component



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

Counter-example: Optional Component

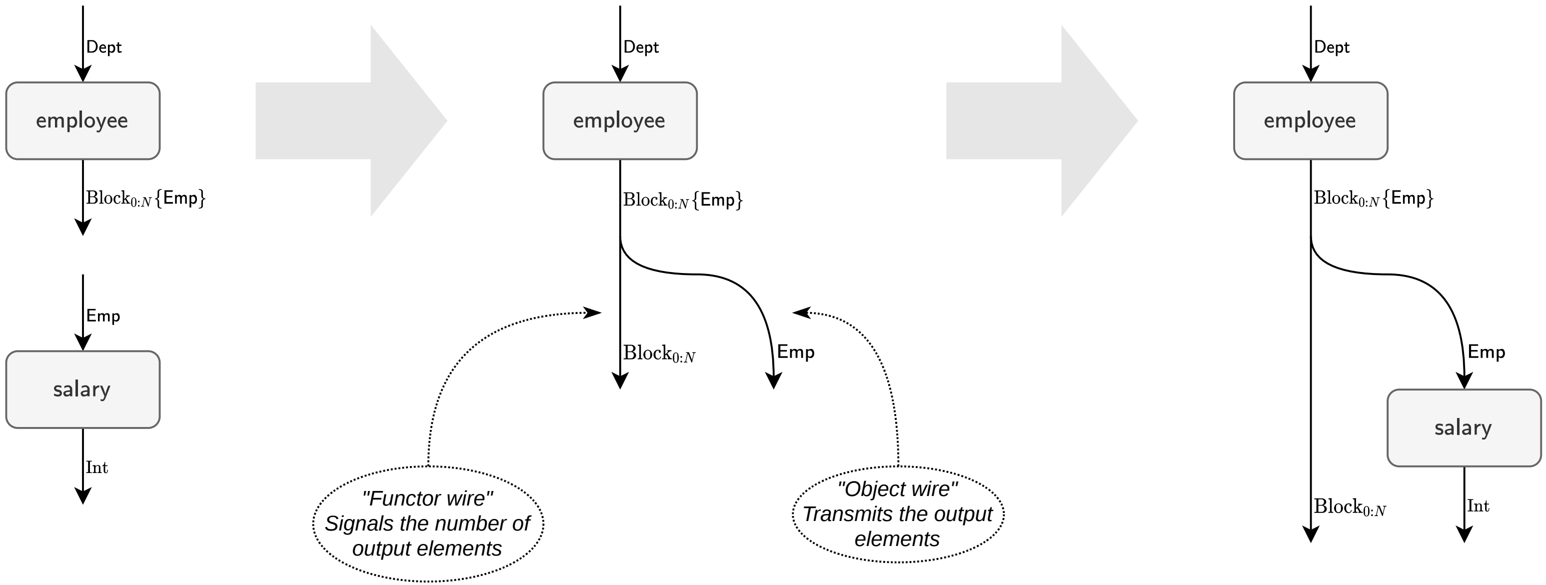


Even so, the input and the output share a common component,
which suggests there should be a way to compose these transformations.

Idea: Unbundle the Wire

Unbundle the composite wire.

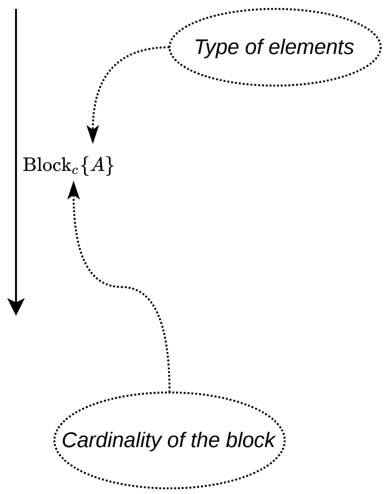
Compose using the object wire.



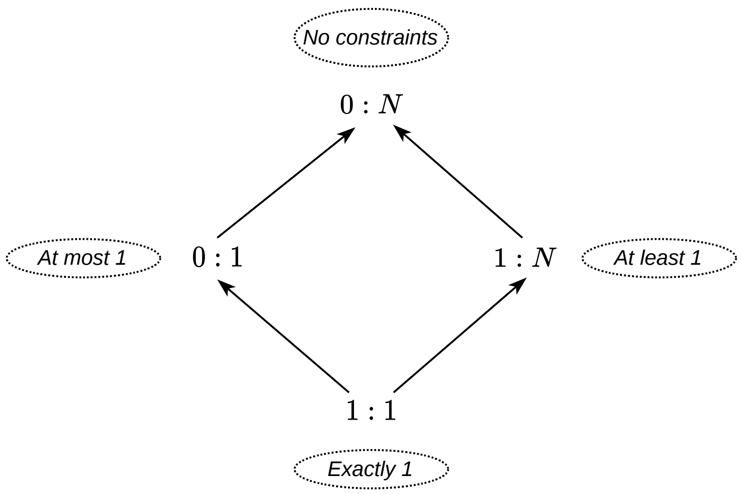
Attaching a transformation to the object wire
indicates that the transformation is applied
to all element of the collection.

Block Type

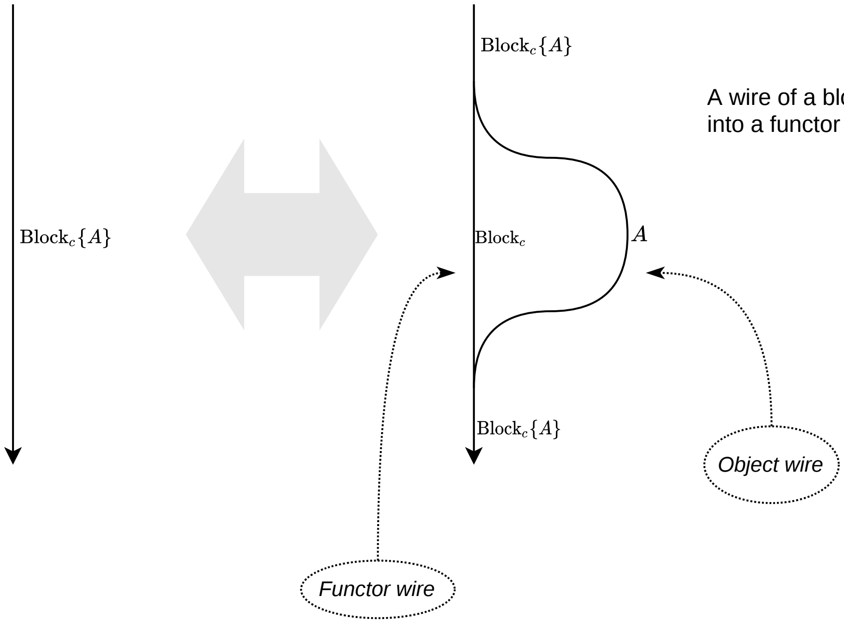
A block is a collection
of homogeneous elements.



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

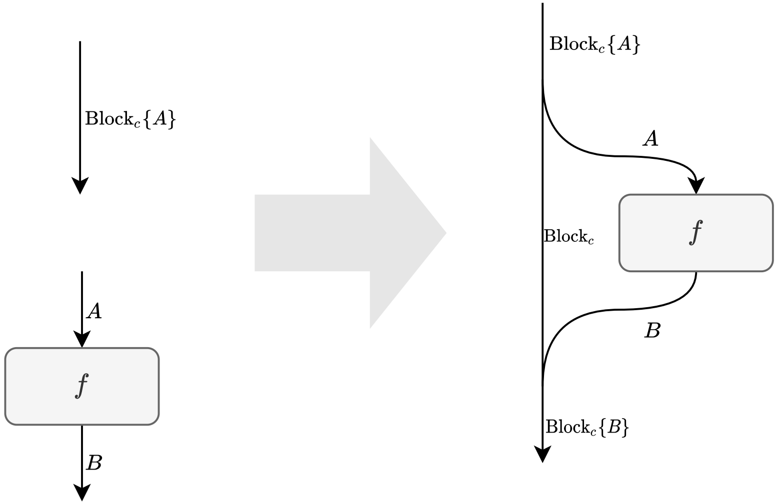


Unbundling



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

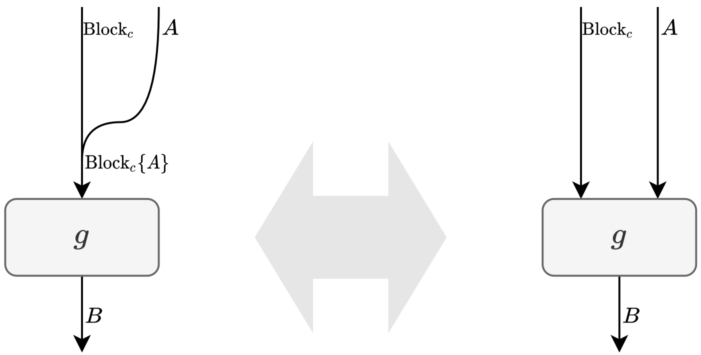
Object Transformation



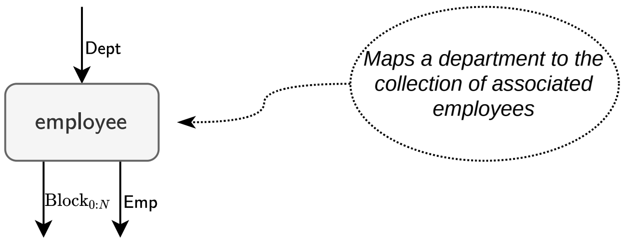
$\text{with\_elements}(f)$

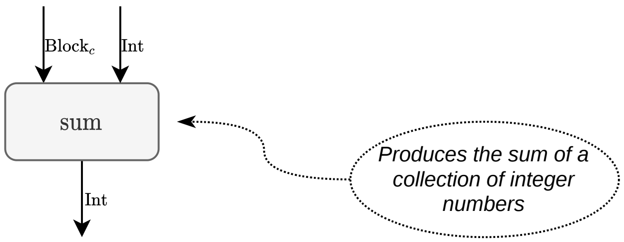
Any compatible transformation
can be applied to the object wire,
which indicates that the transformation
is applied to every element of the block.

Multiwired Transformations

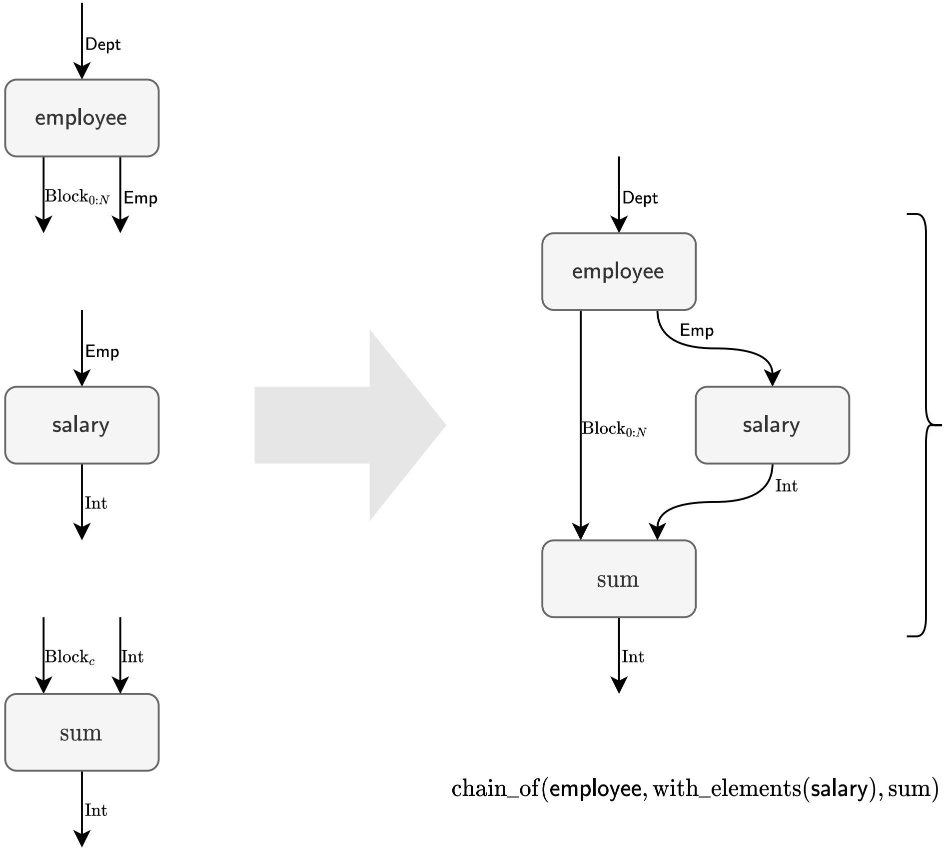


Example: Multiwired Transformations





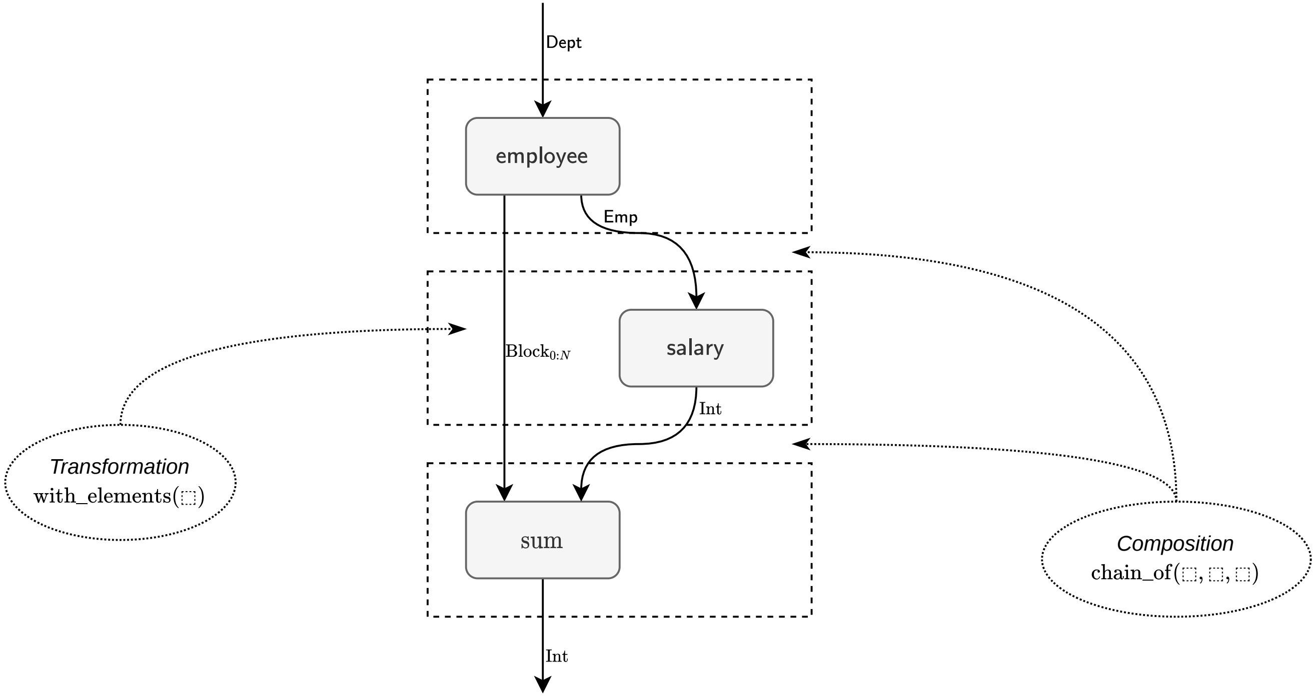
Example: Multiwired Composition



Calculates total sum of salaries in a given department

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

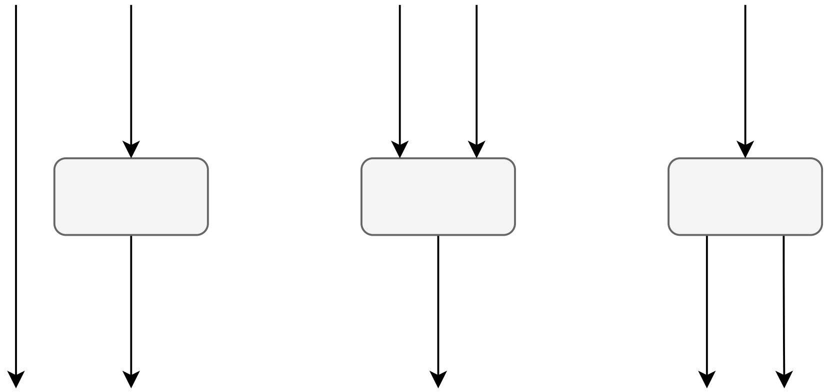
Example: Multiwired Composition Details



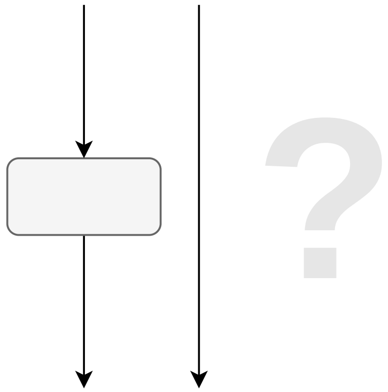
$$\mathrm{chain\_of}(\mathsf{employee}, \mathrm{with\_elements}(\mathsf{salary}), \mathrm{sum})$$

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.

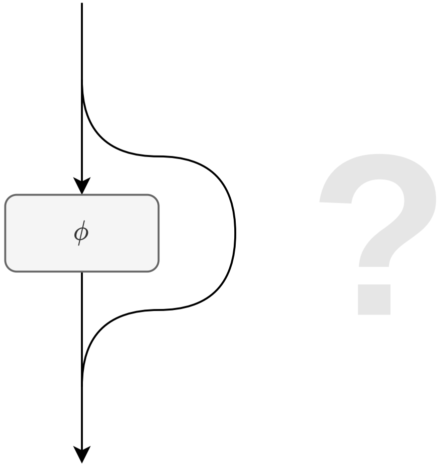


What we miss is a notion of a transformation applied to the functor wire.

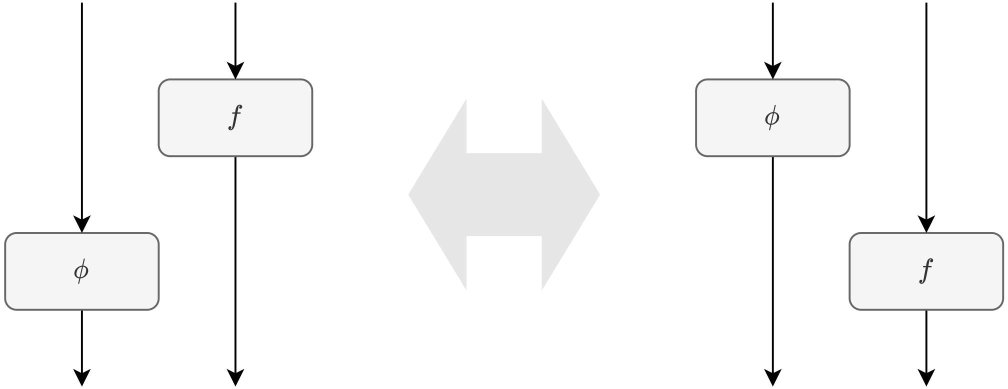


Functor Transformation

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

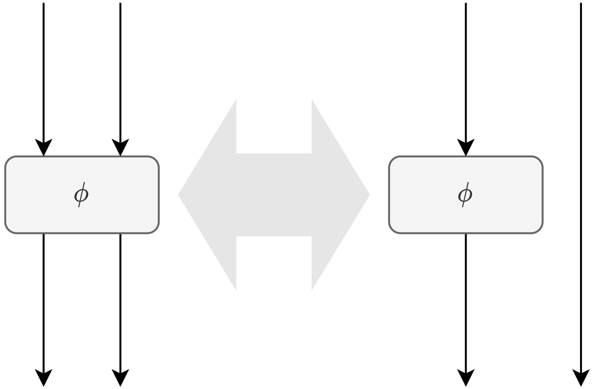


How to define a functor transformation?

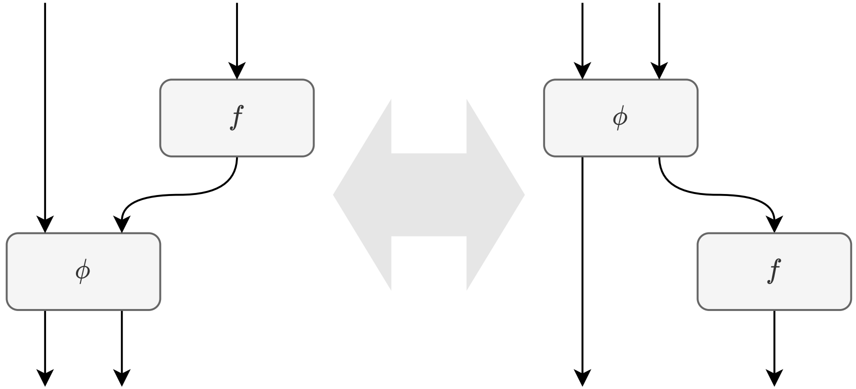


This property can be taken as a definition of a functor transformation.

Definition: Functor Transformation



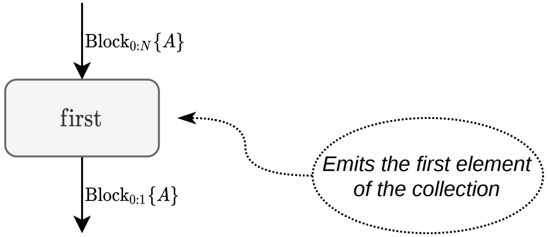
when for any f



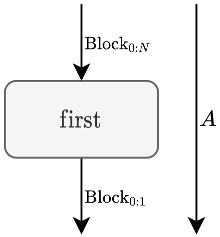
Such transformations are also called *natural*.

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

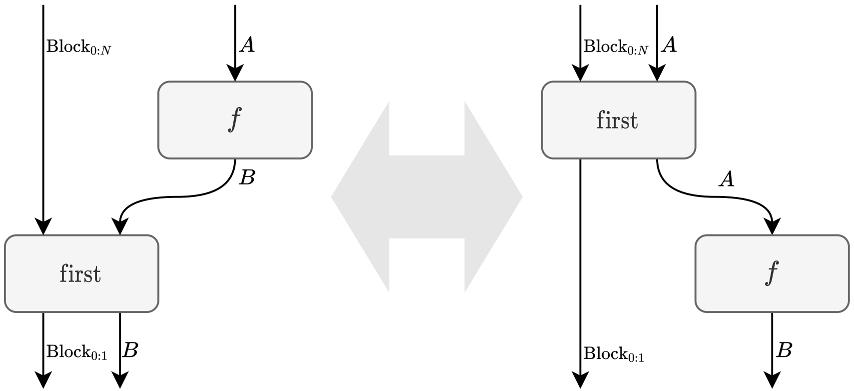
Example: Functor Transformation



This is, in fact, a natural transformation.



Indeed, for any $f : A \to B$,



$$\mathrm{chain\_of}(\mathrm{with\_elements}(f), \mathrm{first}) \equiv \mathrm{chain\_of}(\mathrm{first}, \mathrm{with\_elements}(f))$$