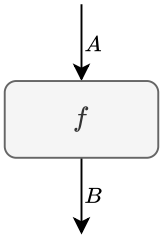
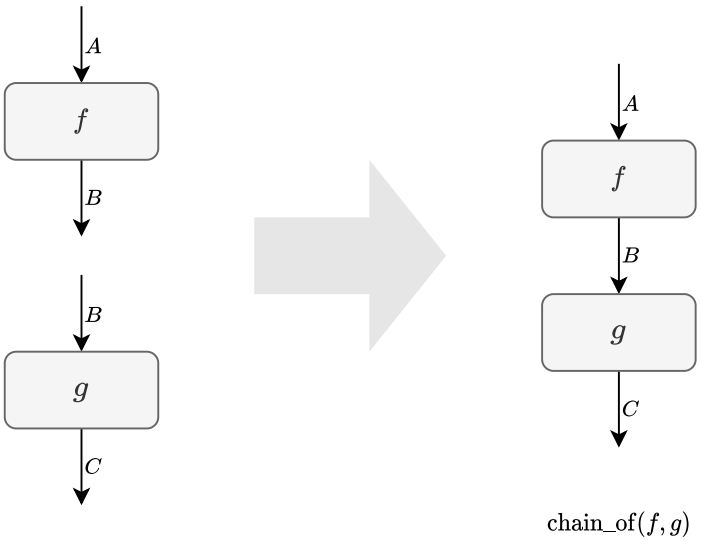
Transformation



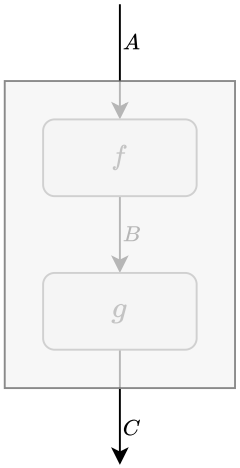
A transformation f
maps any input of type A
to the output of type B.

Composition



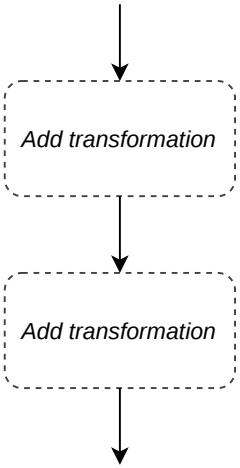
Transformations with compatible input and output can be composed.

Composition is a Transformation



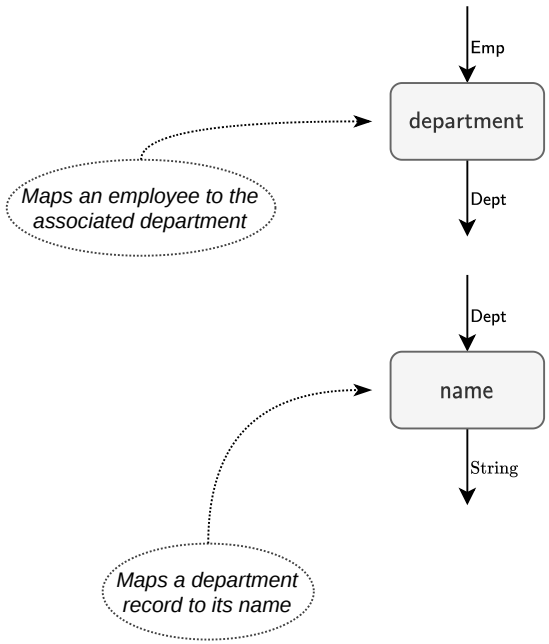
Crucially, composition of transformations is again a transformation.

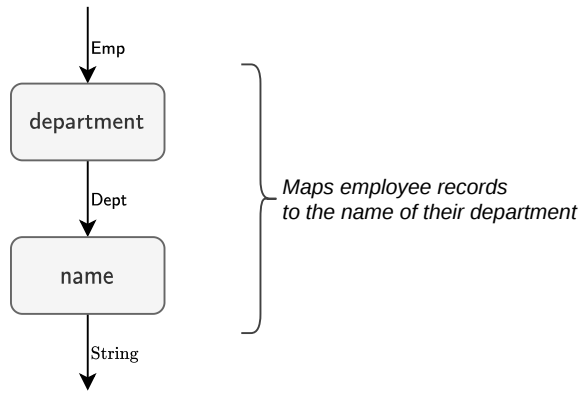
Composition Combinator



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

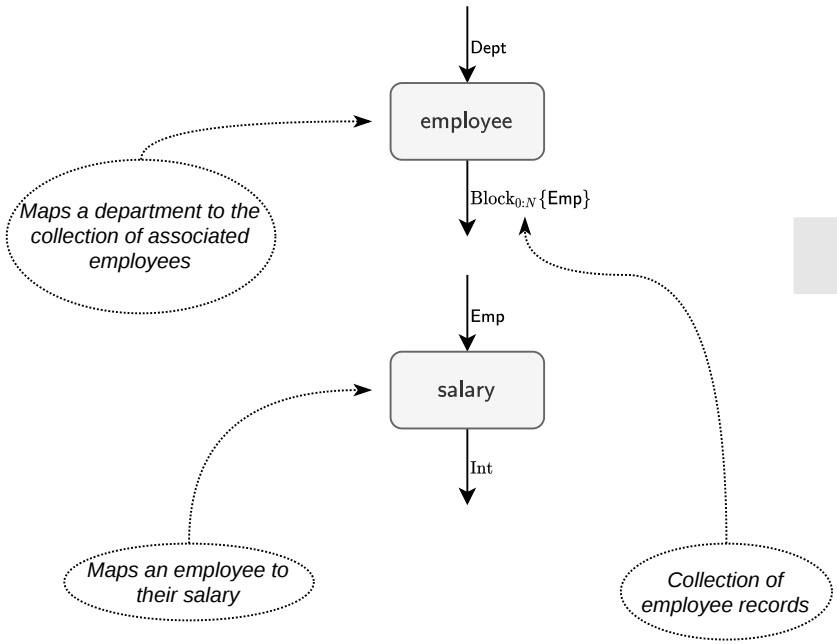
Example: Composition





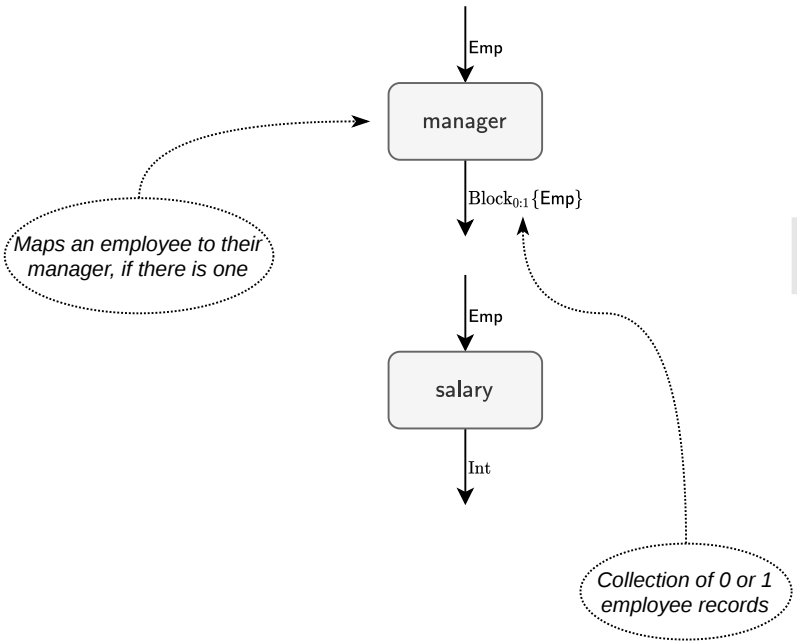
$\mathrm{chain\_of}(\mathsf{department}, \mathsf{name})$

Counter-example: Plural Component



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

Counter-example: Optional Component

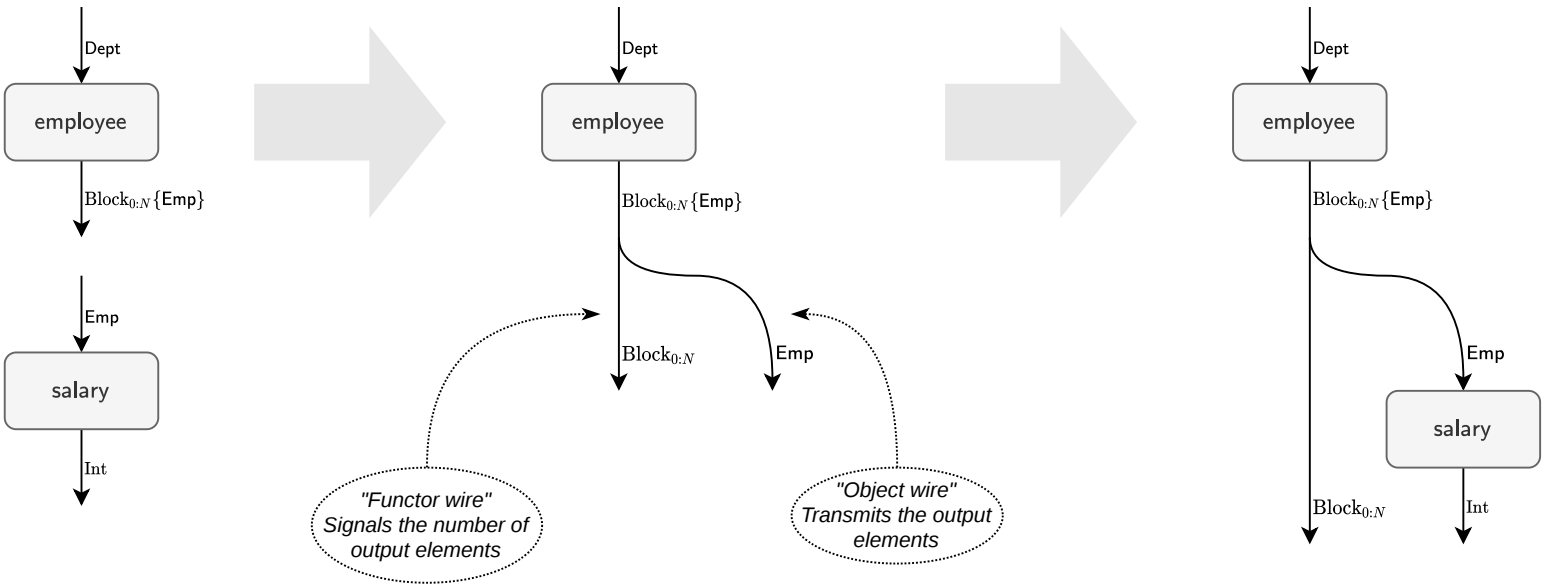


Even so, the input and the output share a common component,
which suggests there should be a way to compose these transformations.

Idea: Unbundle the Wire

Unbundle the composite wire.

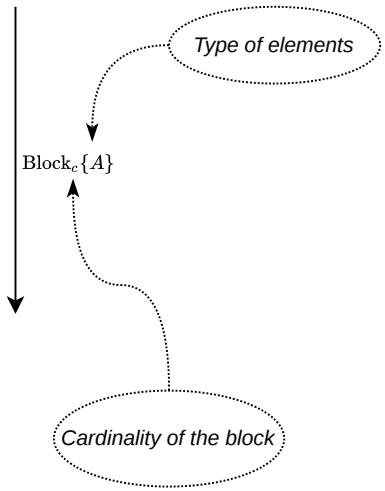
Compose using the object wire.



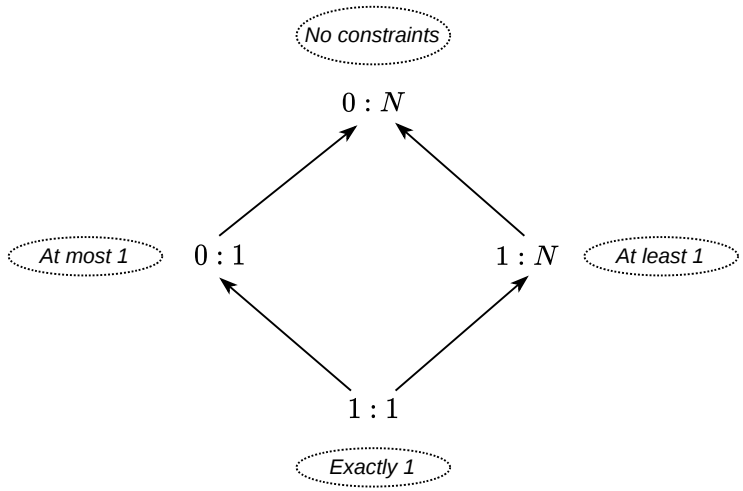
Attaching a transformation to the object wire indicates that the transformation is applied to all element of the collection.

Block Type

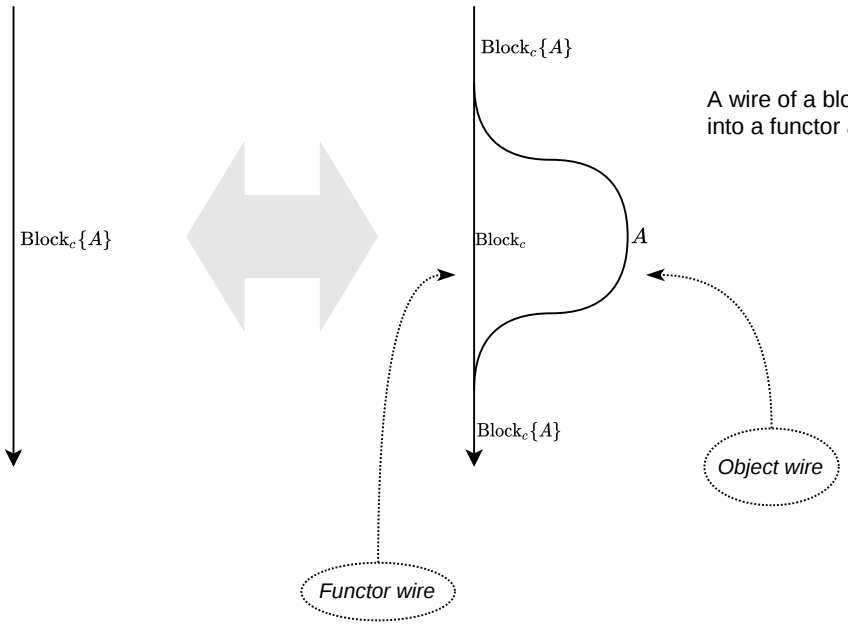
A block is a collection
of homogeneous elements.



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

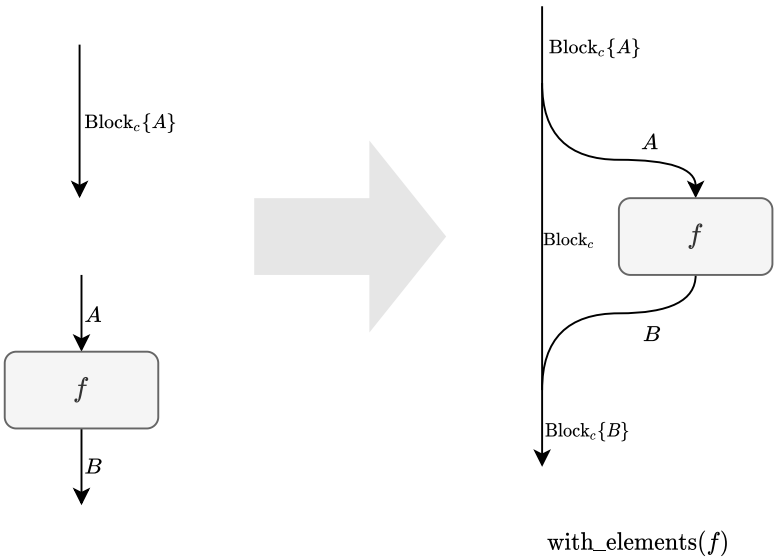


Unbundling



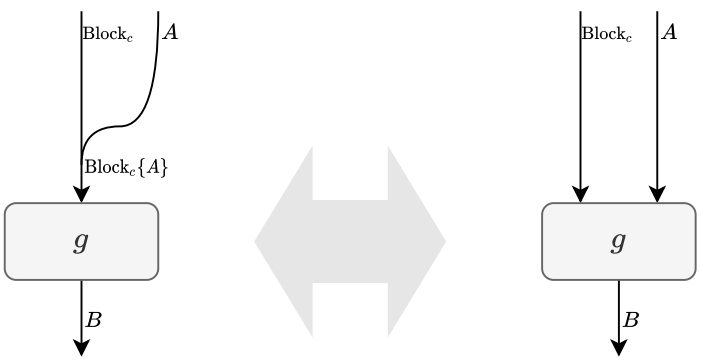
A wire of a block type can be unbundled into a functor and an object components.

Object Transformation

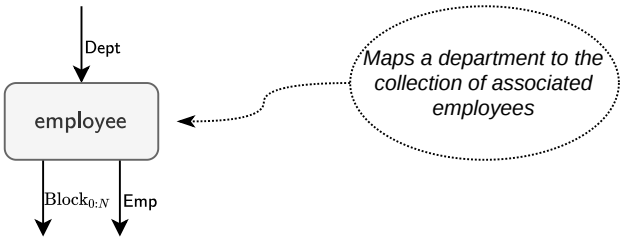


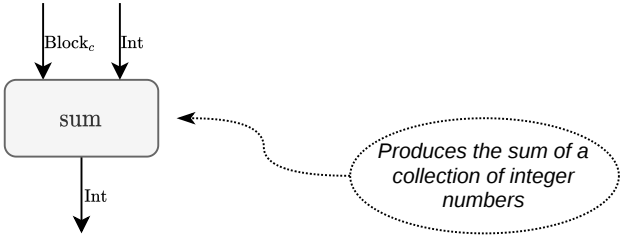
Any compatible transformation
can be applied to the object wire,
which indicates that the transformation
is applied to every element of the block.

Multiwired Transformations

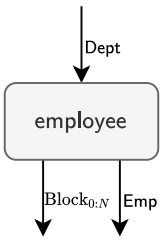


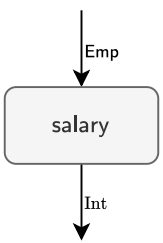
Example: Multiwired Transformations

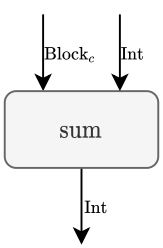


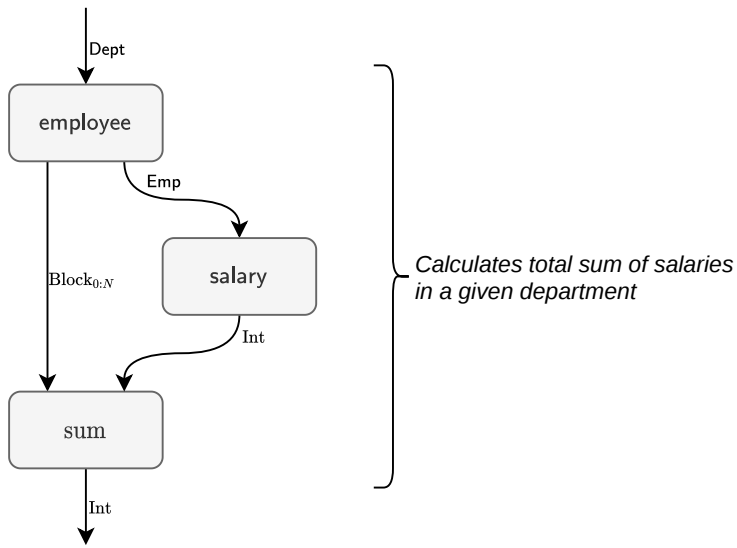


Example: Multiwired Composition



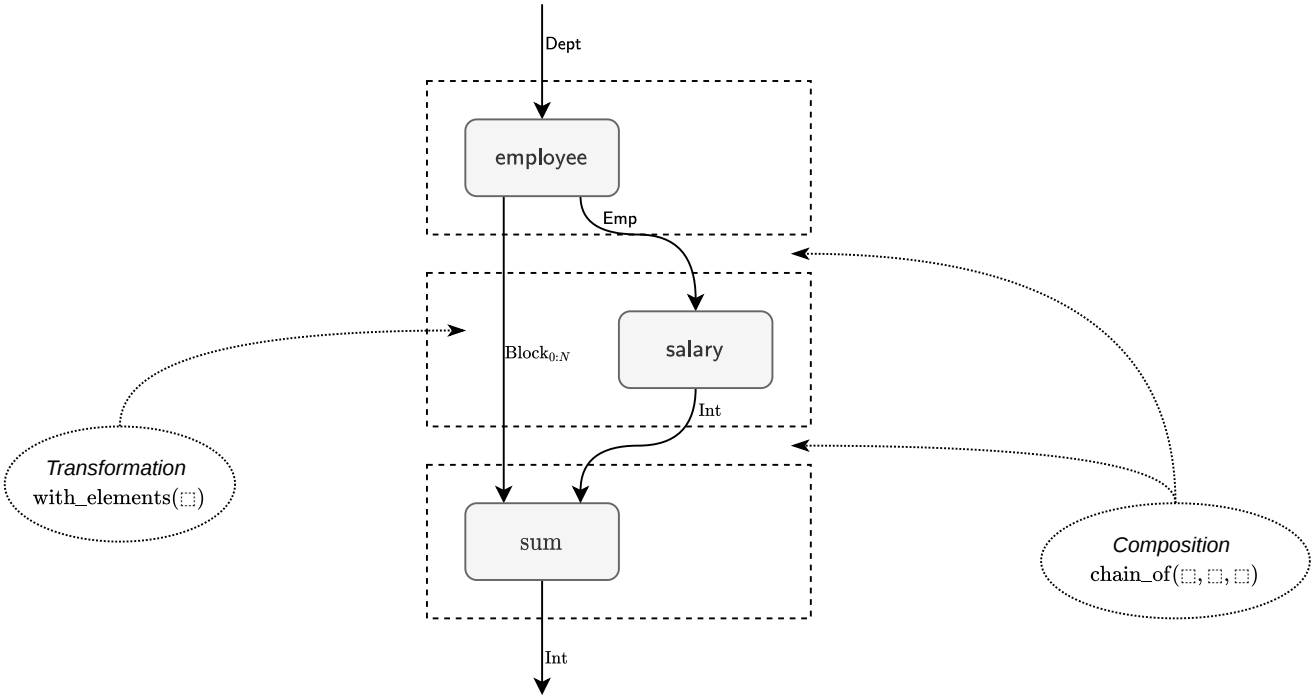






chain_of(**employee**, with_elements(**salary**), sum)

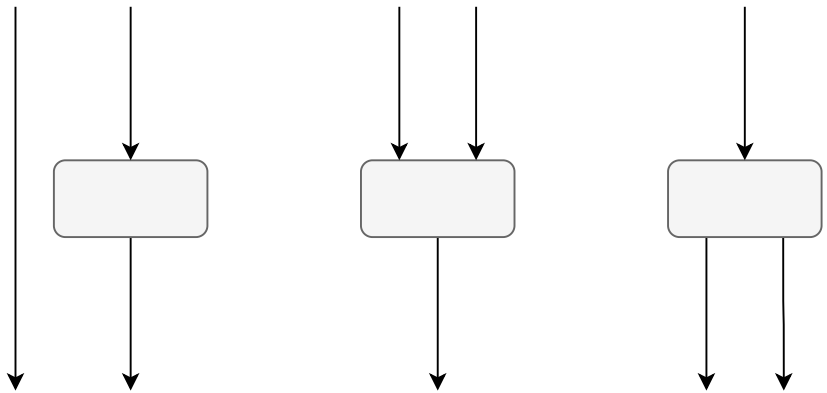
Example: Multiwired Composition Details



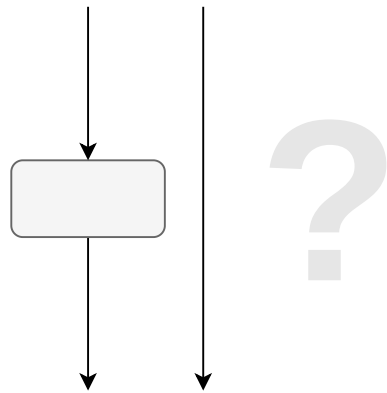
chain_of(**employee**, with_elements(**salary**), sum)

Challenge: Functor Transformation

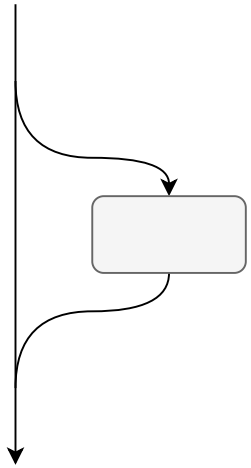
So far we have seen different combinations of transformations and wires: transformations that are applied to the object wire, as well as transformations that consume or produce both functor and object wires.



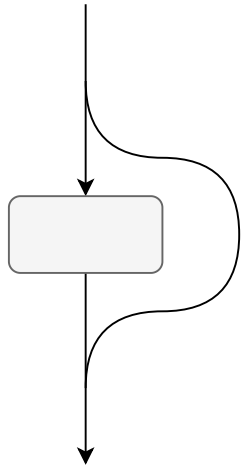
What we miss is a notion of a transformation applied to the functor wire.



Challenge: Functor Transformation



When we unbundle a wire
into functor and object components,
we can apply a transformation
to the object wire.



But can we possibly define a notion
of a transformation applied
to the functor wire?