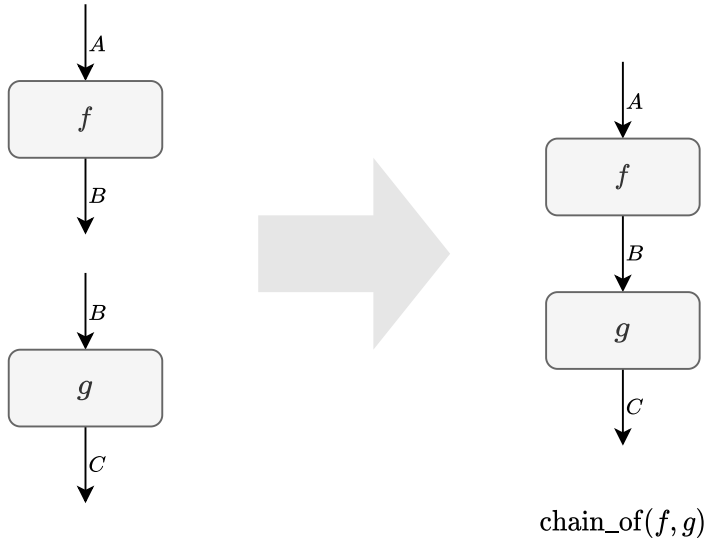


# Transformation



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



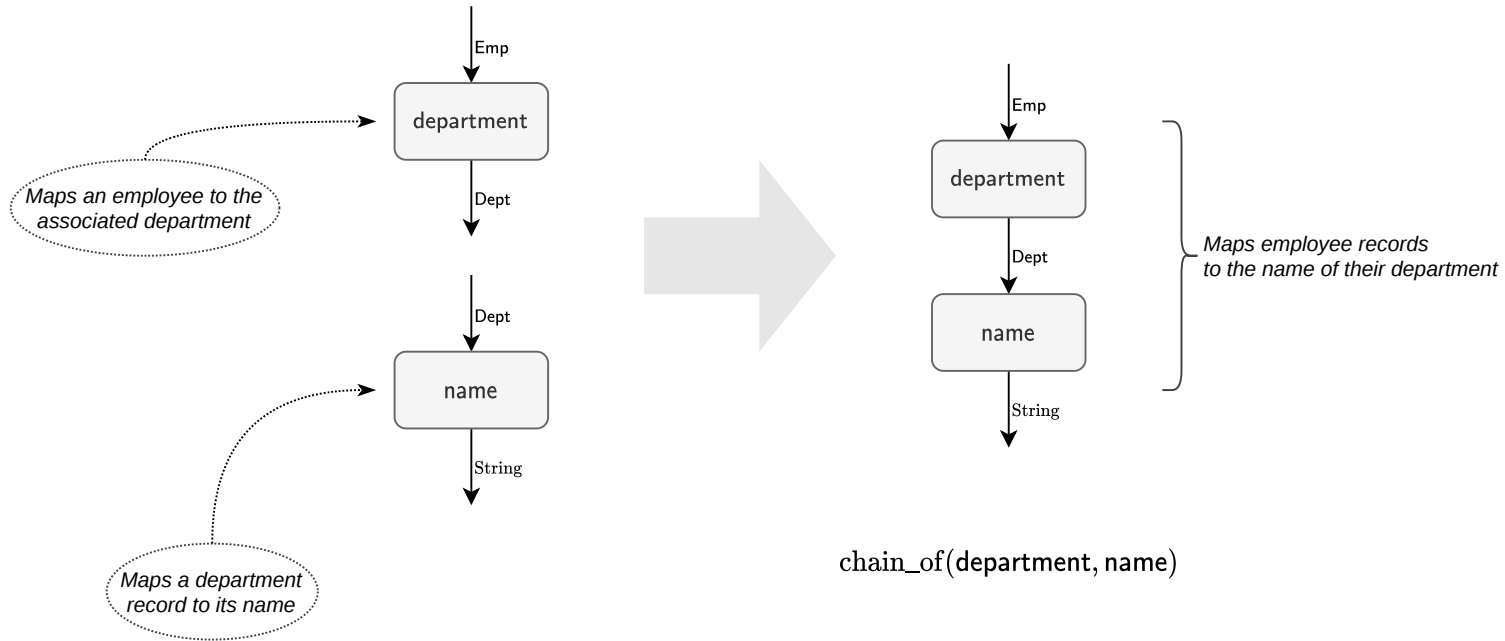
Crucially, composition of transformations is again a transformation.

# Composition Combinator



Composition `chain_of([], [])`  
is a transformation combinator  
with two arguments.

## Example: Composition



## 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



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



# Block Type

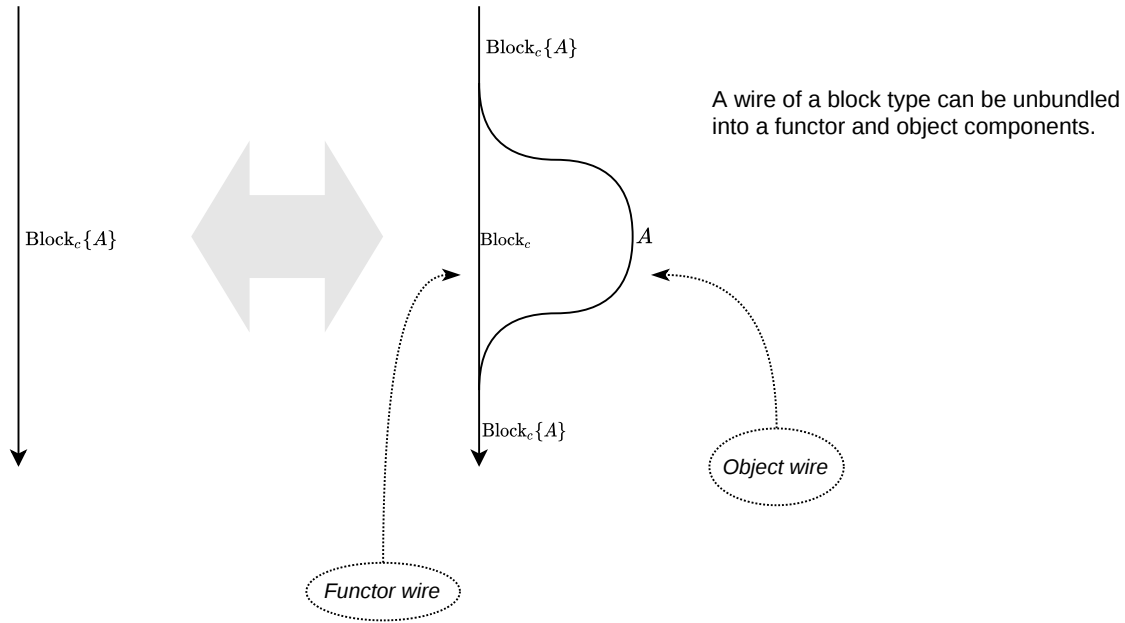
A block is a collection of homogeneous elements.



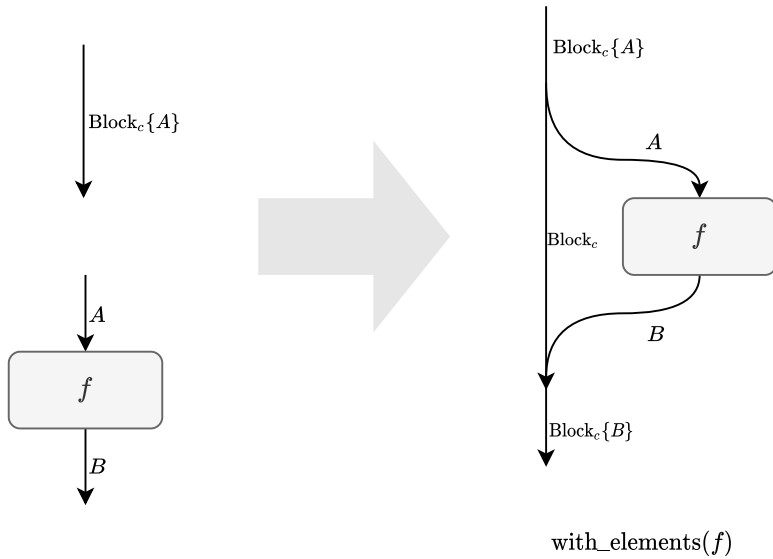
Cardinality is a constraint on the number of elements in a block.



# Unbundling

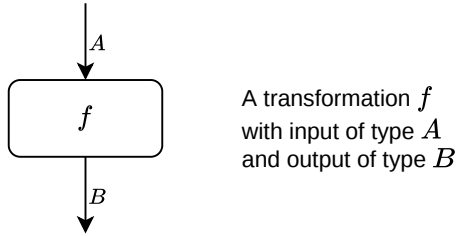


# Object Transformation

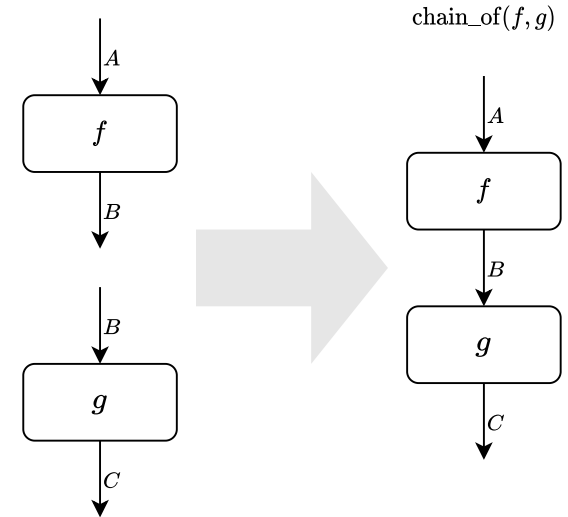


Any compatible transformation can be applied to the object wire, which indicates that the transformation is applied to every element of the block.

## 1. Transformation



## 2. Composition

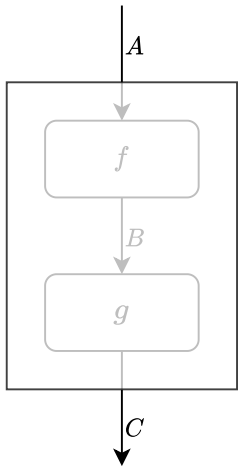


Transformations with compatible input and output can be composed

## 3. Composition is a Transformation

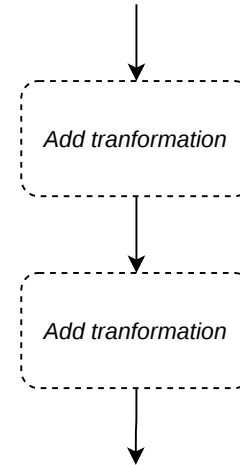
## 4. Composition Combinator





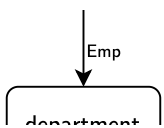
Trivially (but crucially),  
a composition of transformations  
is again a transformation

chain\_of(`[], []`)



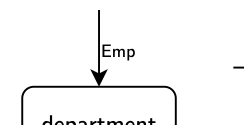
Composition is a transformation combinator  
with two placeholders

## 5. Example: Components of a Composition

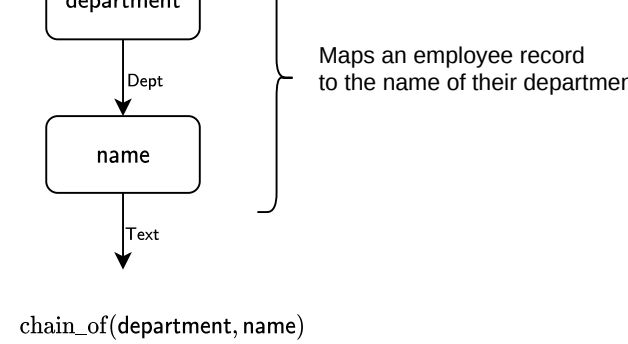
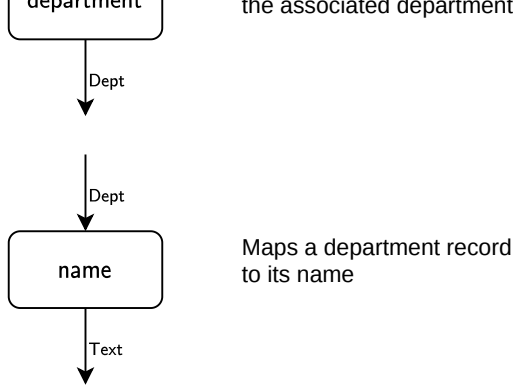


Maps an employee to

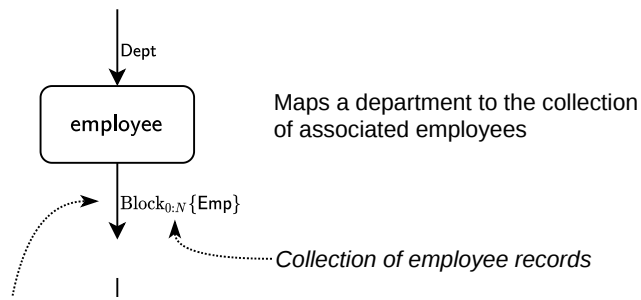
## 6. Example: Composition



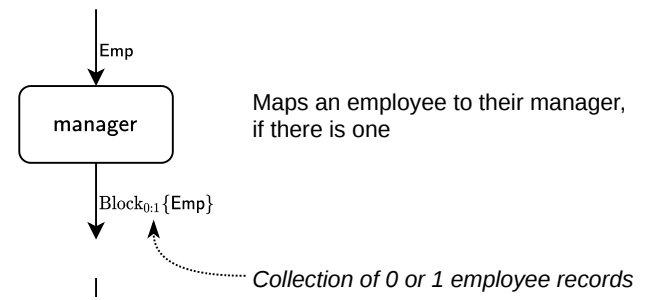




## 7. Counter-example: Plural Component

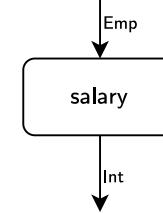
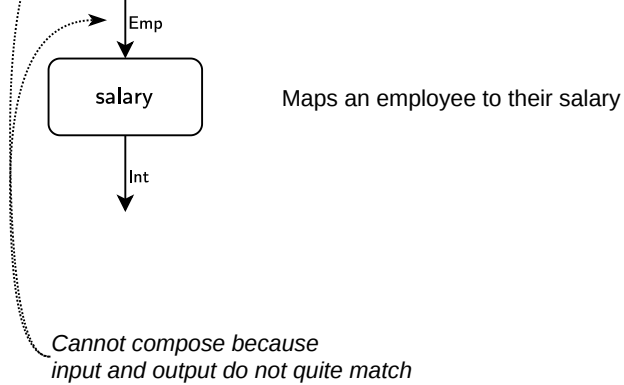


## 8. Counter-example: Optional



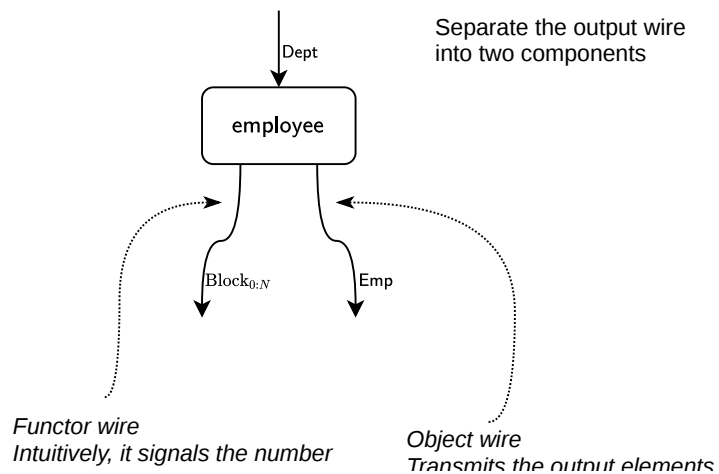


**Component**

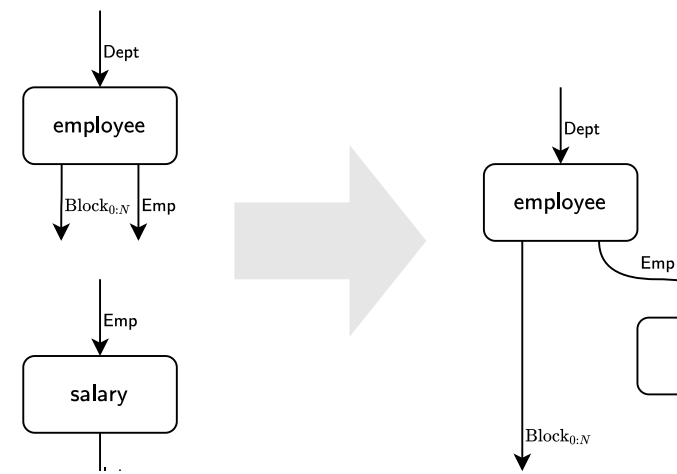


Can we represent composition of these transformations with an intuitive diagrammatic notation?

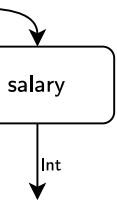
## 9. Idea: Unbundle the Wire



## 10. Idea: Compose Using the



# Object Wire



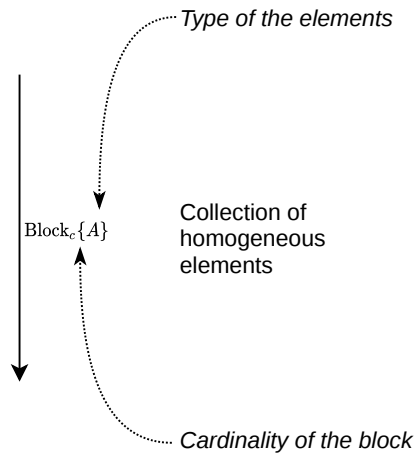
of output elements

transmits the output elements

Int  
↓

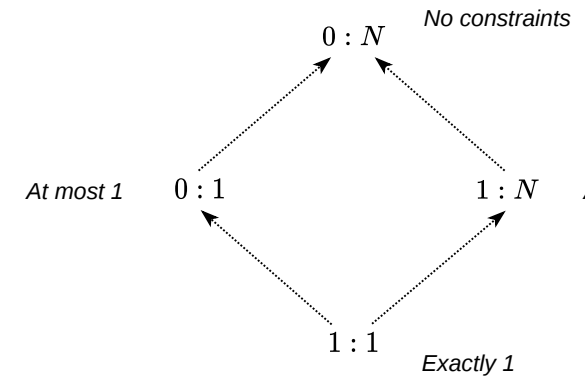
Attaching a transformation to the object wire indicates that the transformation is applied to each element of the collection

## 11. Block Type



## 12. Cardinality

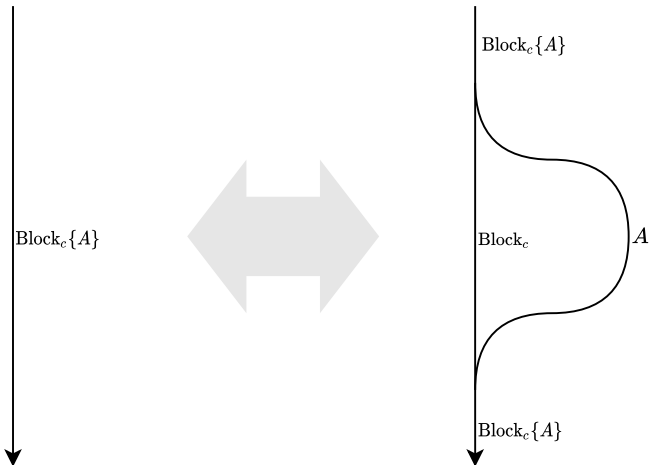
Cardinality is a constraint on the number of elements



ents in a block

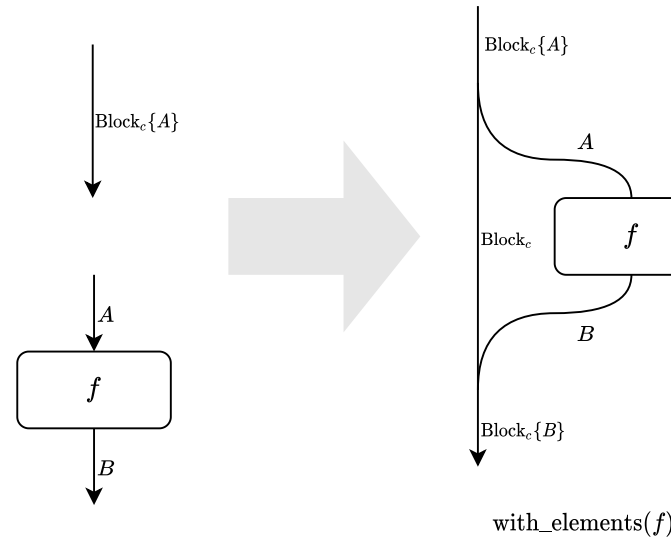
*At least 1*

## 13. Unbundling



We can unbundle a wire of a block type into a functor and object components

## 14. Object Transformation



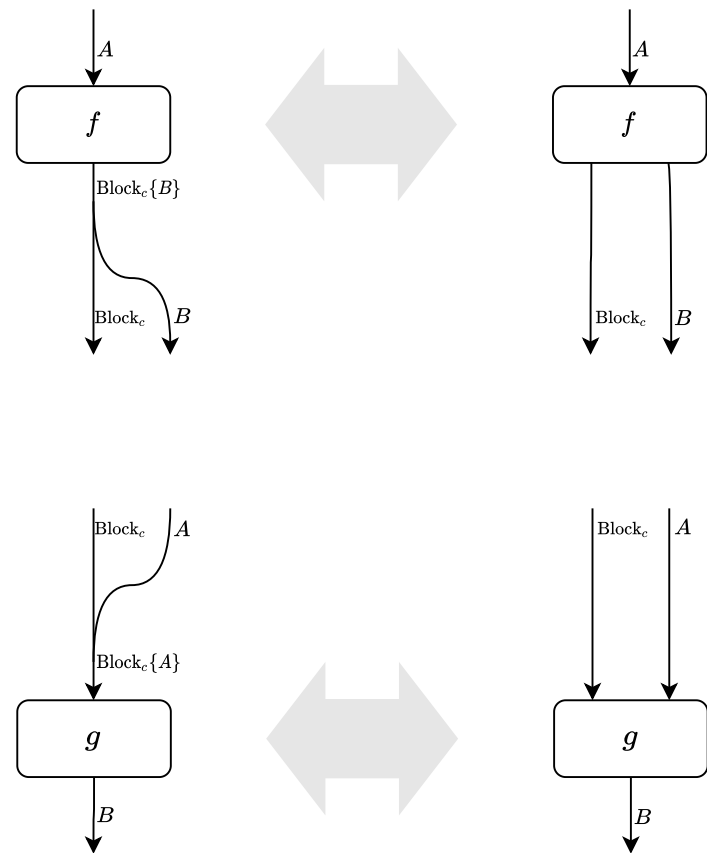
Then any compatible transformation can be applied to the object  $f$ , which indicates that the transformation is applied to every element of the block



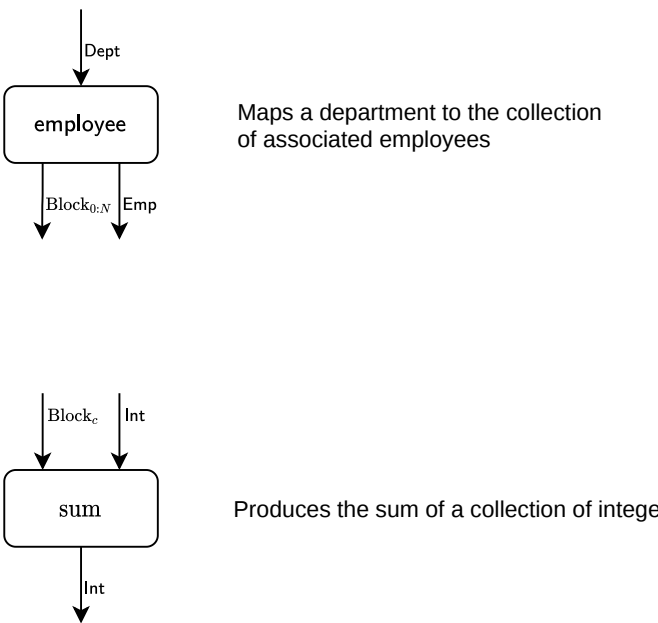
)

t wire,  
ent

13. Multiwired transformations



14. Example: Multiwired Trans

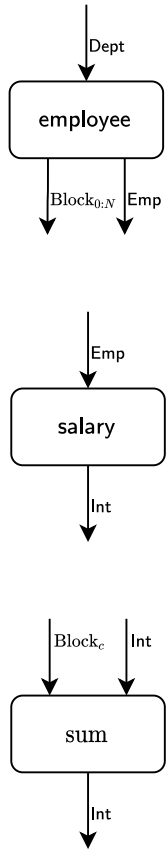




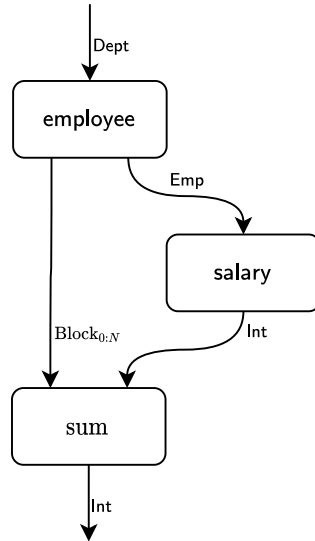
sformations

ers

## 14. Example: Multiwired Composition

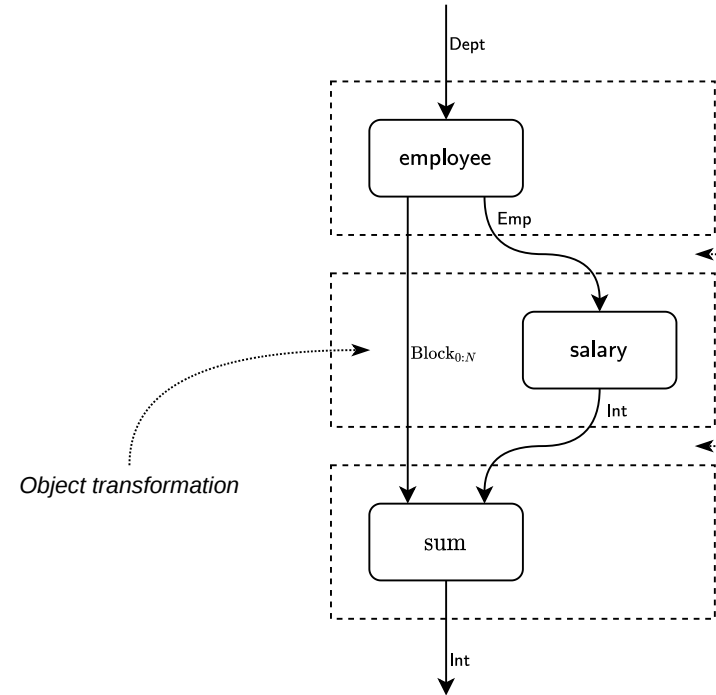


Total sum of salaries in a given department



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

## 15. Example: Details



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



*Composition*

sum)