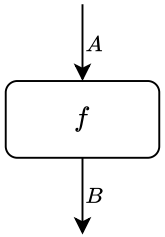
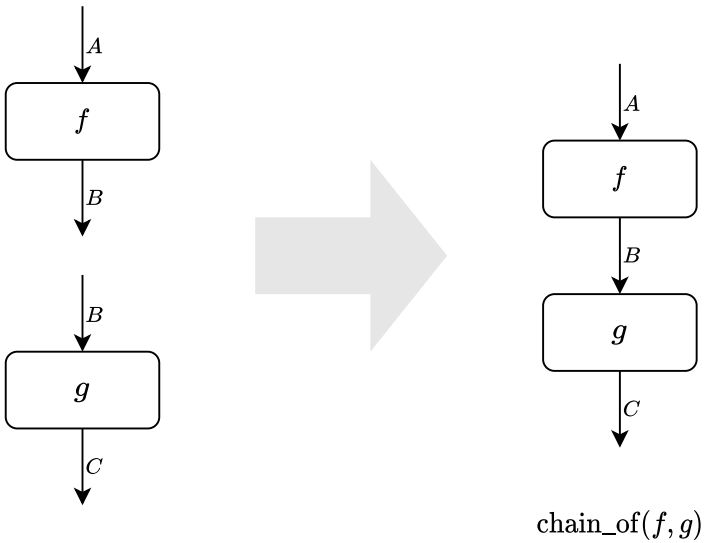
Transformation



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

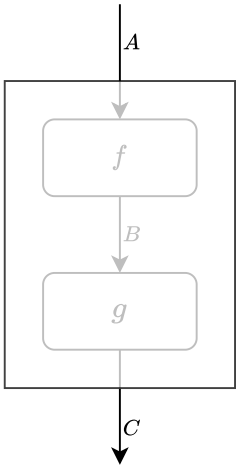
Composition



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

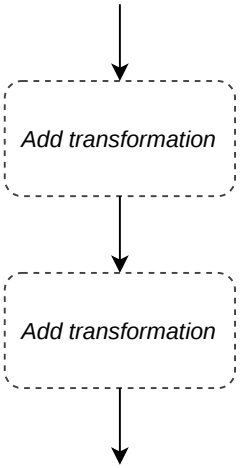
Transformations with compatible input and output can be composed.

Composition is a Transformation



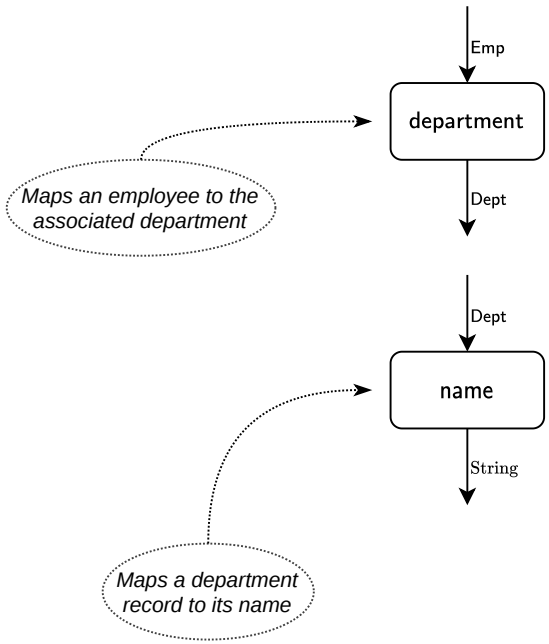
Crucially, composition of transformations is again a transformation.

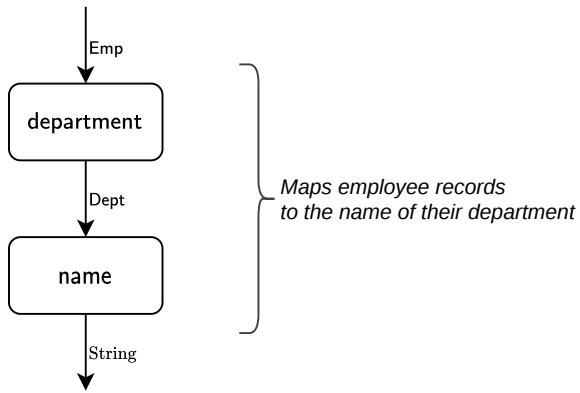
Composition Combinator



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

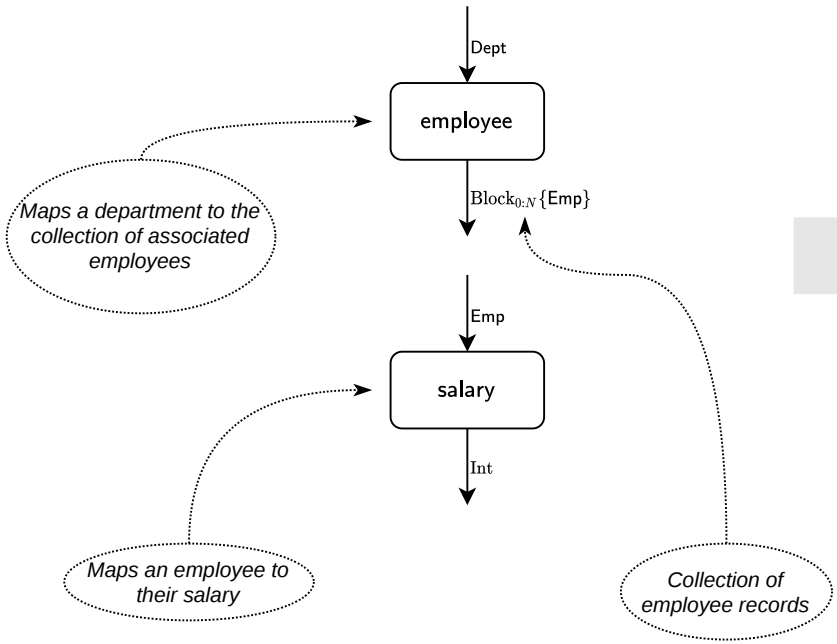
Example: Composition





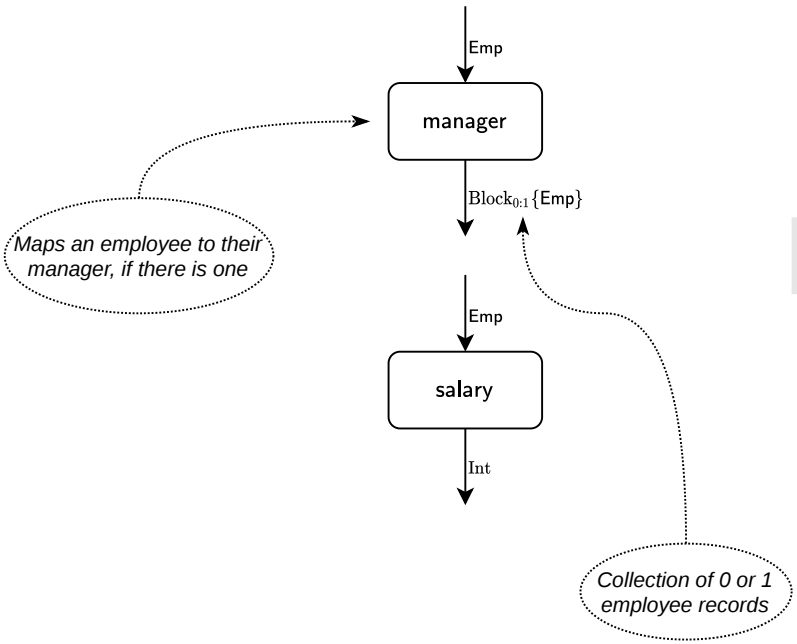
$\mathrm{chain\_of}(\mathsf{department}, \mathsf{name})$

Counter-example: Plural Component



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

Counter-example: Optional Component

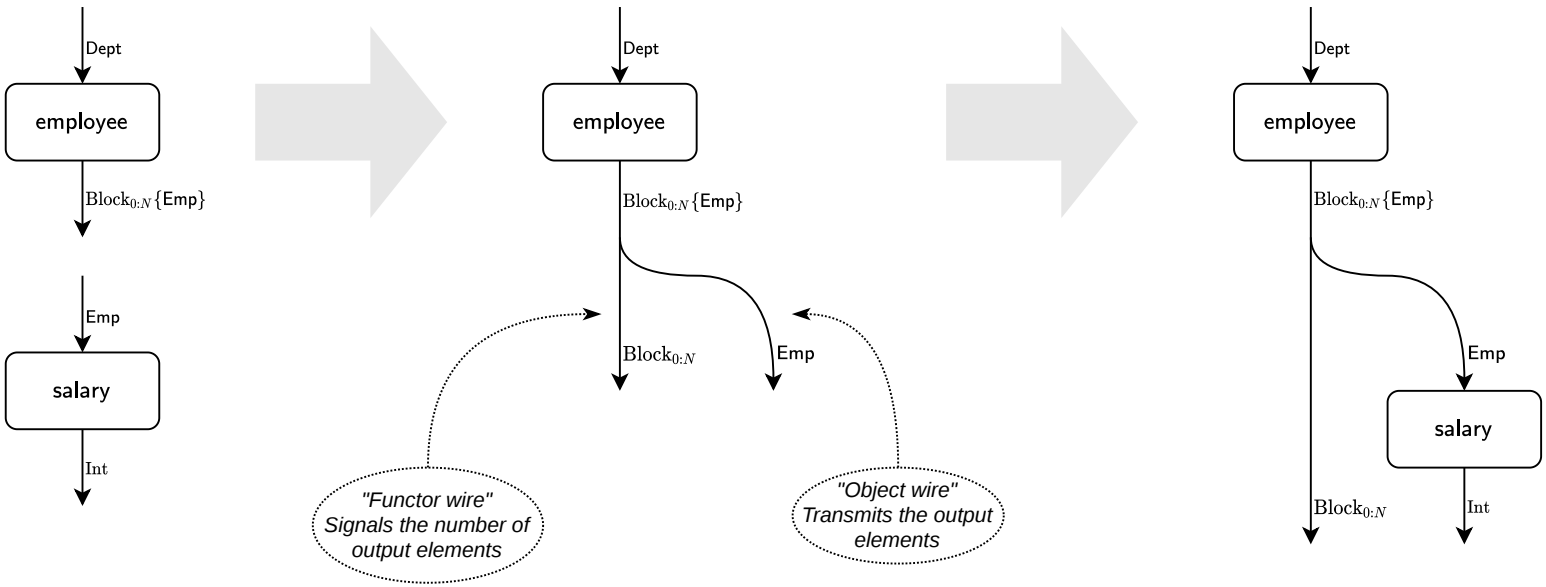


Even so, the input and the output share a common component,
which suggests there should be a way to compose these transformations.

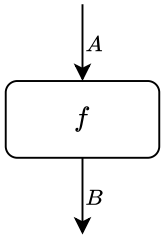
Idea: Unbundle the Wire

Unbundle the composite wire.

Compose using the object wire.

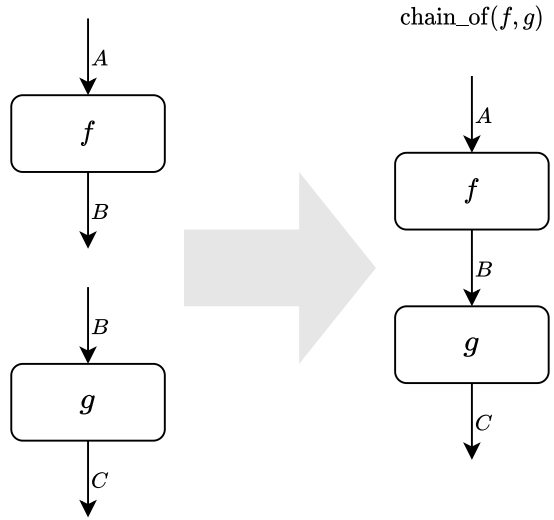


1. Transformation



A transformation f
with input of type A
and output of type B

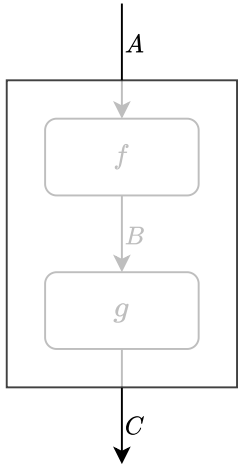
2. Composition



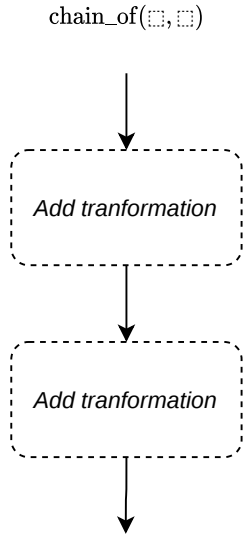
Transformations with
compatible input and output
can be composed

3. Composition is a Transformation

4. Composition Combinator

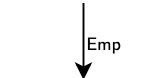


Trivially (but crucially),
a composition of transformations
is again a transformation



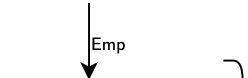
Composition is a transformation combinator
with two placeholders

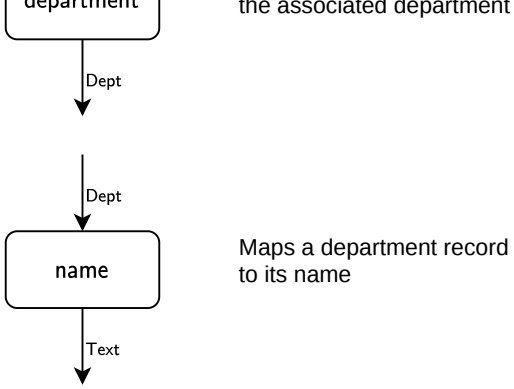
5. Example: Components of a Composition



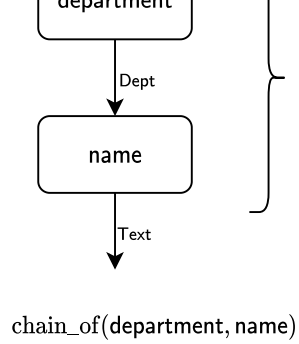
Maps an employee to

6. Example: Composition





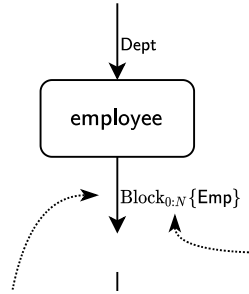
Maps a department record
to its name



Maps an employee record
to the name of their departmer

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

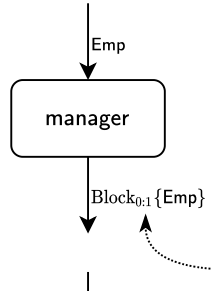
7. Counter-example: Plural Component



Maps a department to the collection
of associated employees

Collection of employee records

8. Counter-example: Optional

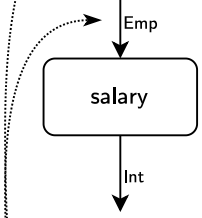


Maps an employee to their manager,
if there is one

Collection of 0 or 1 employee records

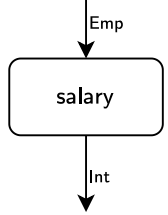
nt

Component



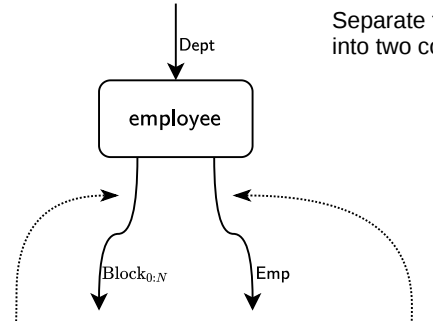
Maps an employee to their salary

Cannot compose because
input and output do not quite match



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

9. Idea: Unbundle the Wire

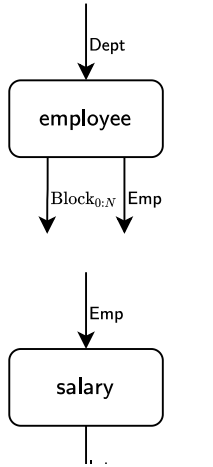


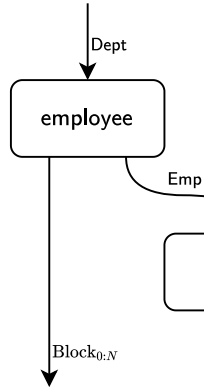
Separate the output wire into two components

Functor wire
Intuitively, it signals the number

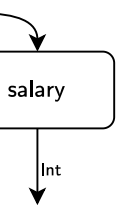
Object wire
Transmits the output elements

10. Idea: Compose Using the





Object Wire



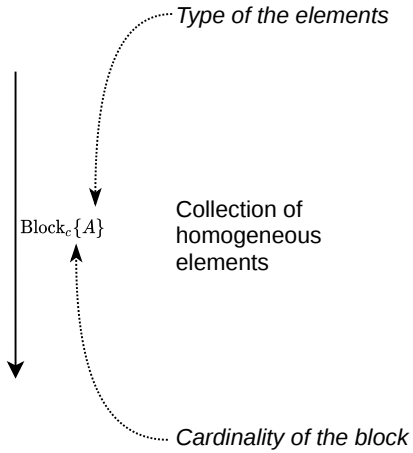
of output elements

transmits the output elements

Int

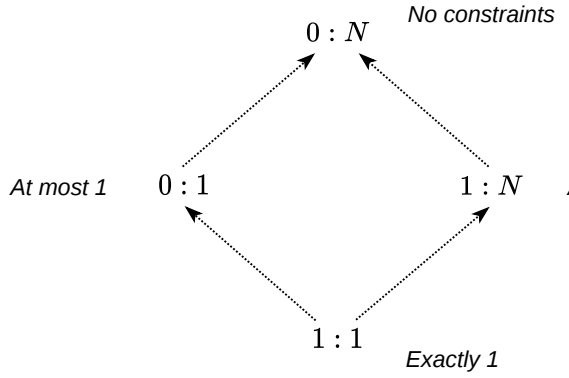
Attaching a transformation to the object wire indicates that the transformation is applied to each element of the collection

11. Block Type



12. Cardinality

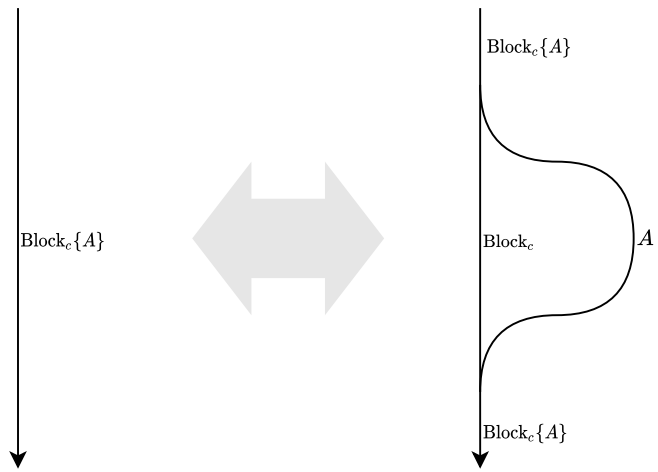
Cardinality is a constraint on the number of elem



ents in a block

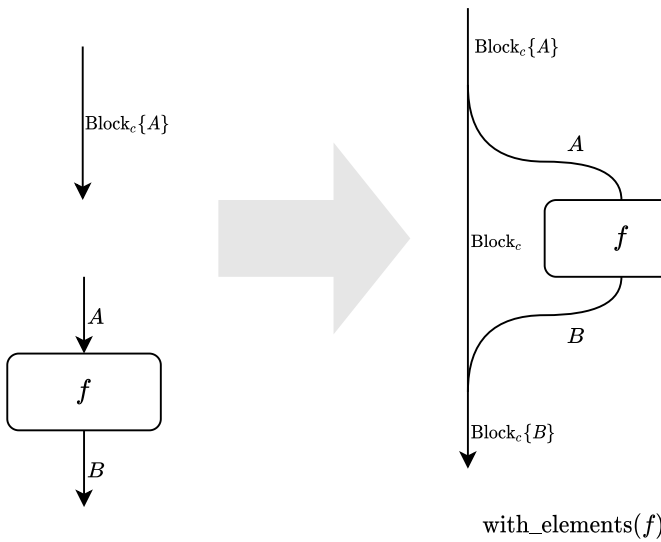
At least 1

13. Unbundling



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

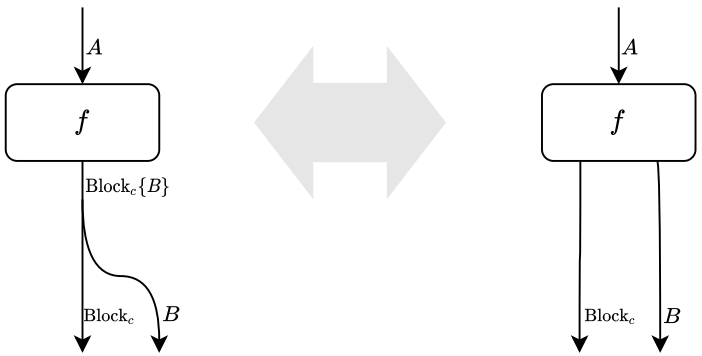
14. Object Transformation

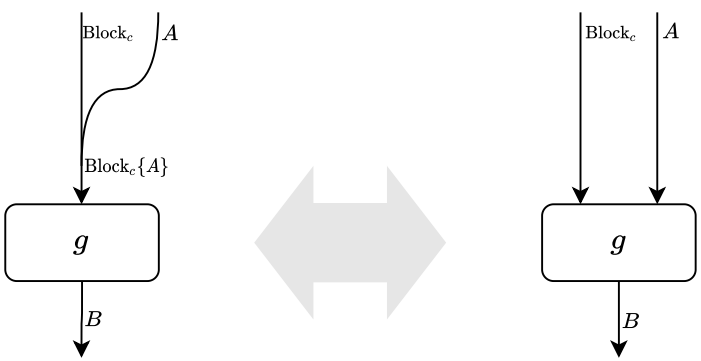


Then any compatible transformation can be applied to the object
which indicates that the transformation is applied to every eleme
of the block

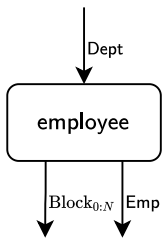
t wire,
ent

13. Multiwired transformations

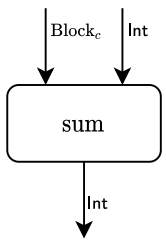




14. Example: Multiwired Trans



Maps a department to the collection of associated employees

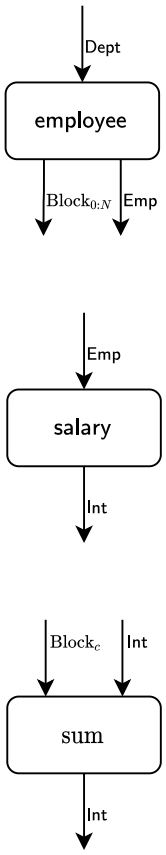


Produces the sum of a collection of intege

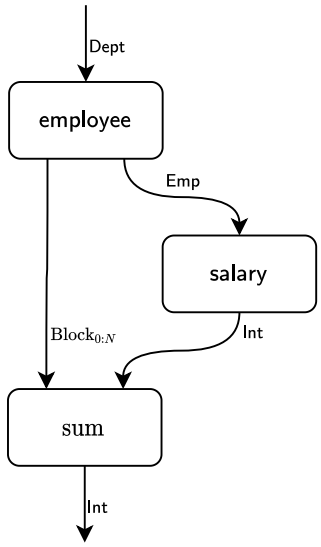
sformations

ers

14. Example: Multiwired Composition

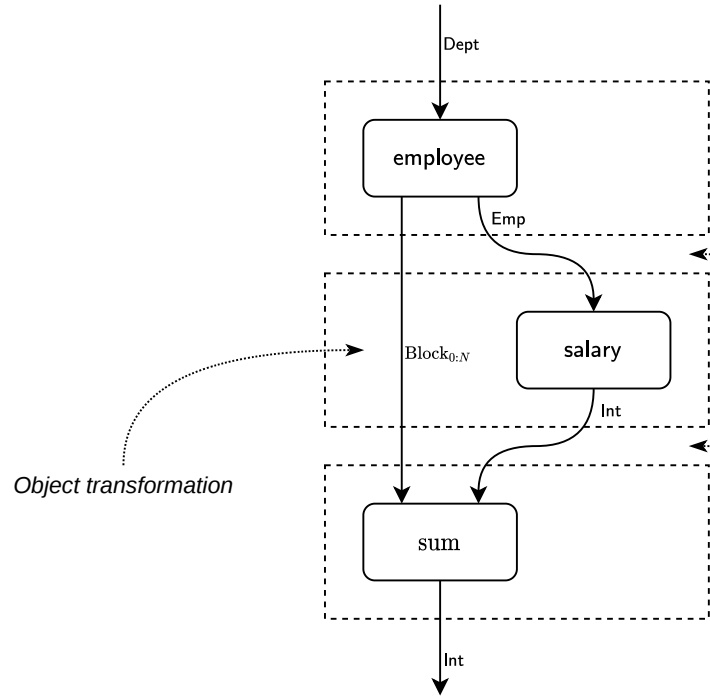


Total sum of salaries in a given department

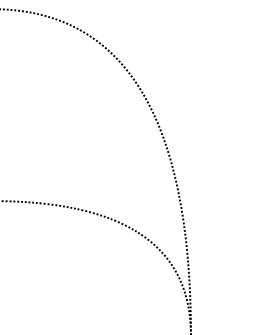


chain_of(employee, with_elements(salary), sum)

15. Example: Details



chain_of(employee, with_elements(salary),



sum)