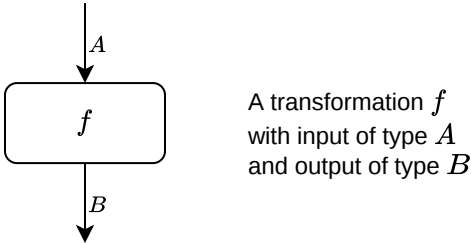
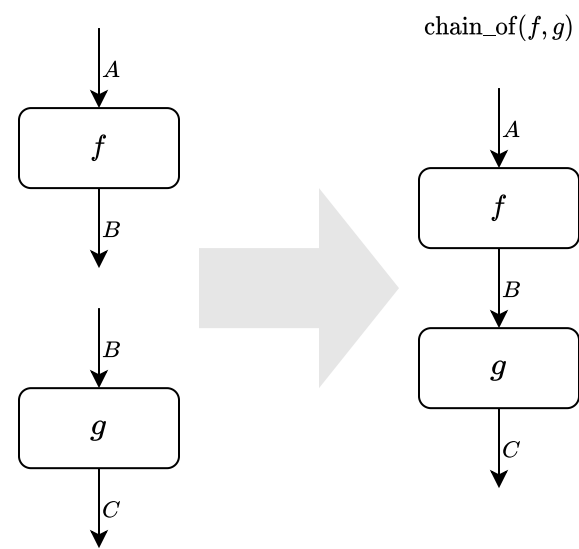


1. Transformation

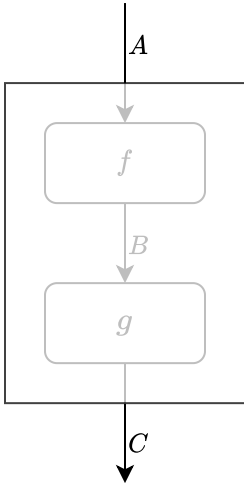


2. Composition



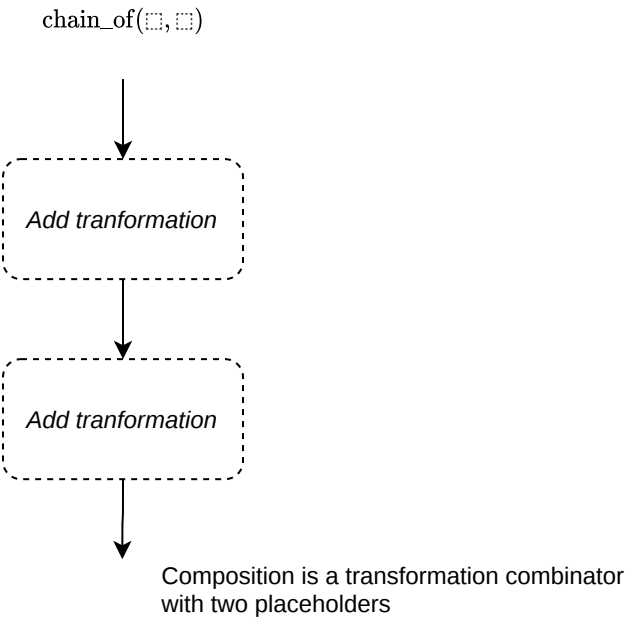
Transformations with compatible input and output can be composed

3. Composition is a Transformation

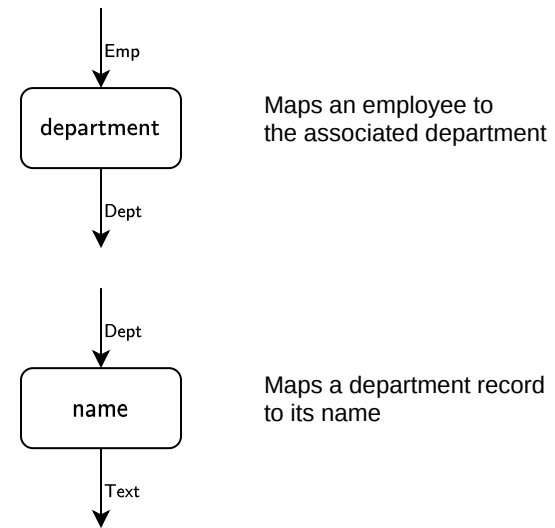


Trivially (but crucially),
a composition of transformations
is again a transformation

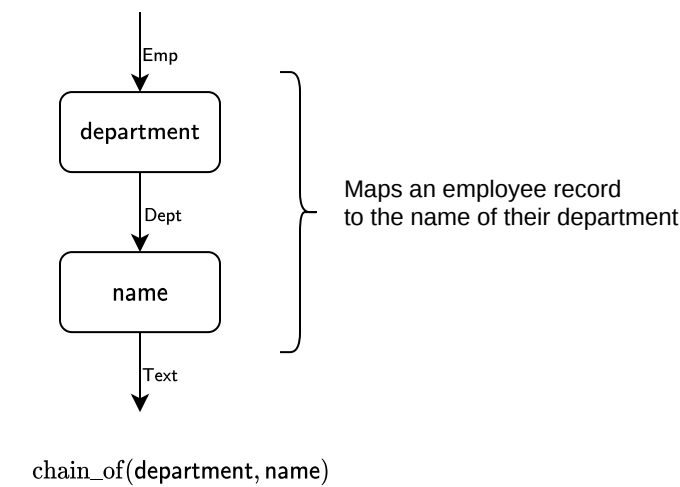
4. Composition Combinator



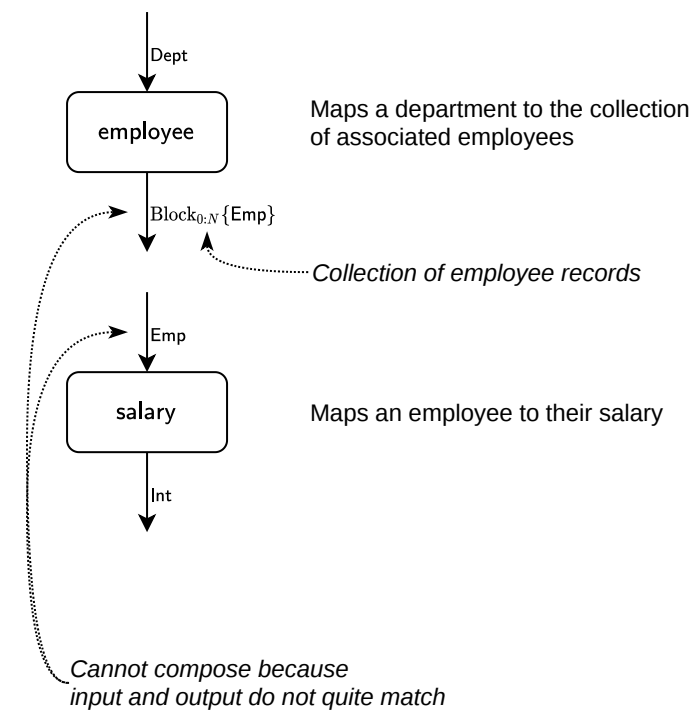
5. Example: Components of a Composition



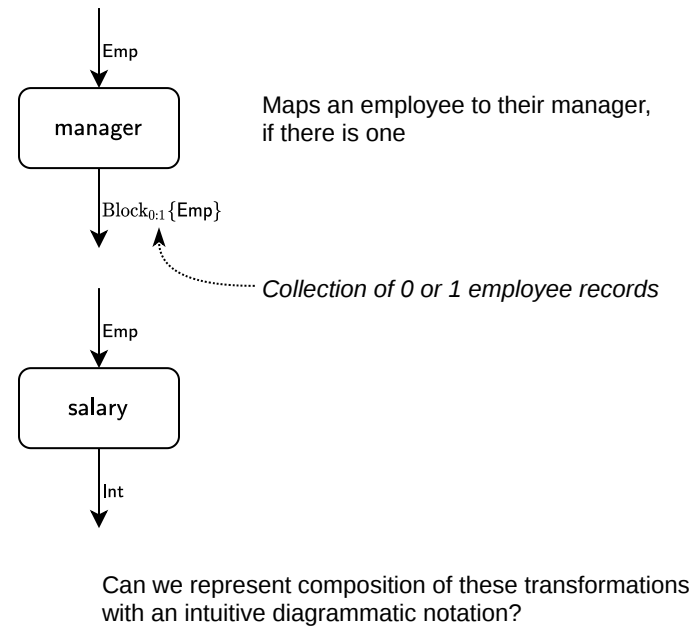
6. Example: Composition



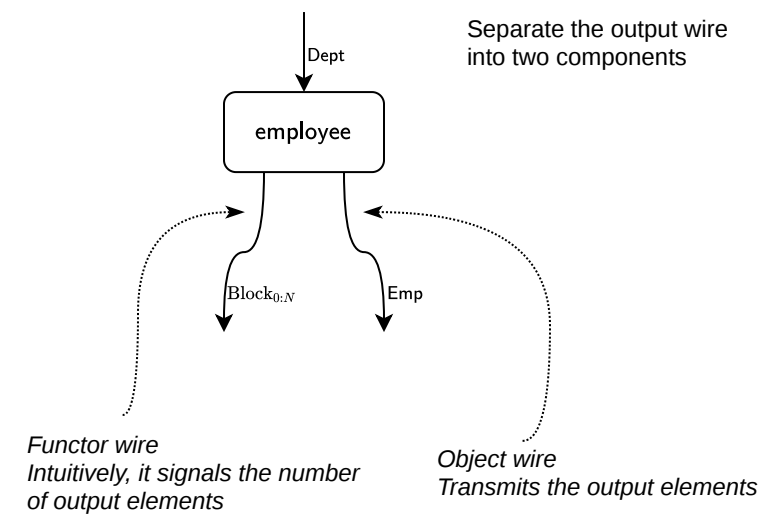
7. Counter-example: Plural Component



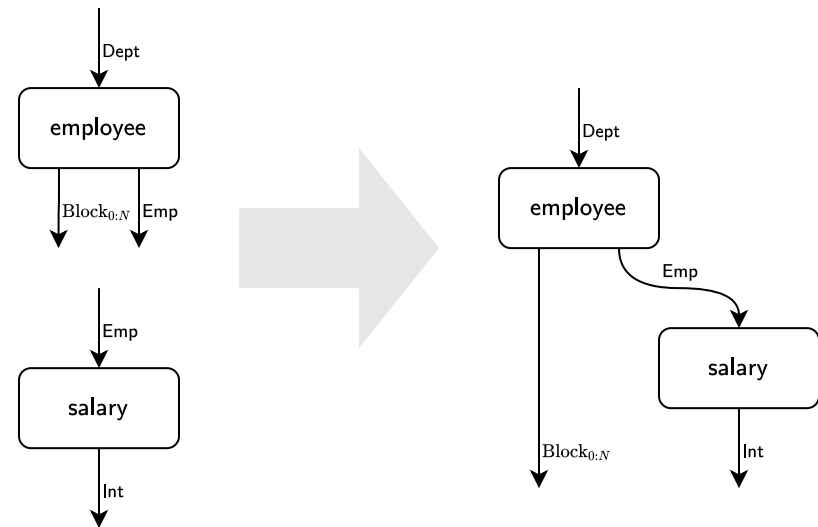
8. Counter-example: Optional Component



9. Idea: Unbundle the Wire

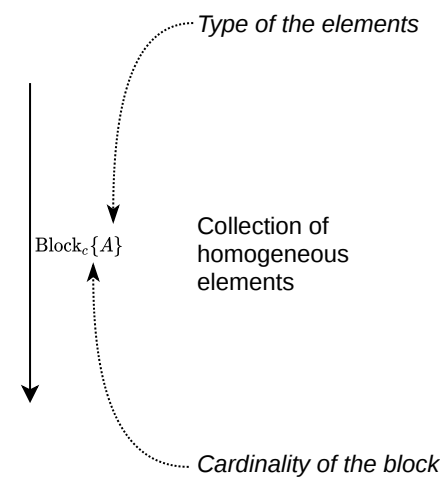


10. Idea: Compose Using the Object Wire



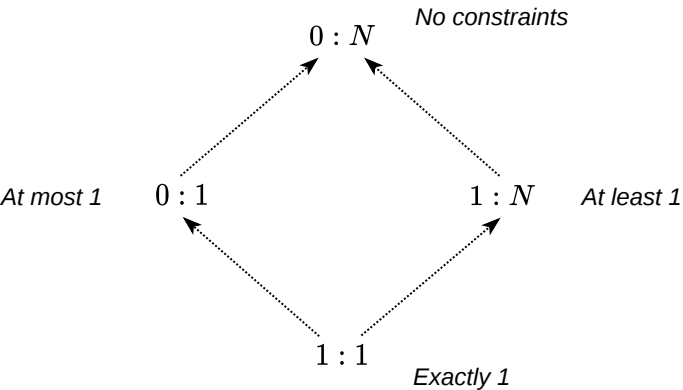
Attaching a transformation to the object wire indicates that the transformation is applied to each element of the collection

11. Block Type

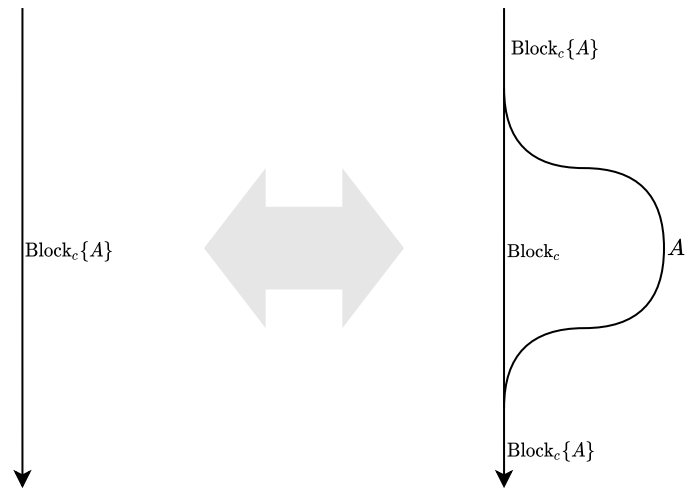


12. Cardinality

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

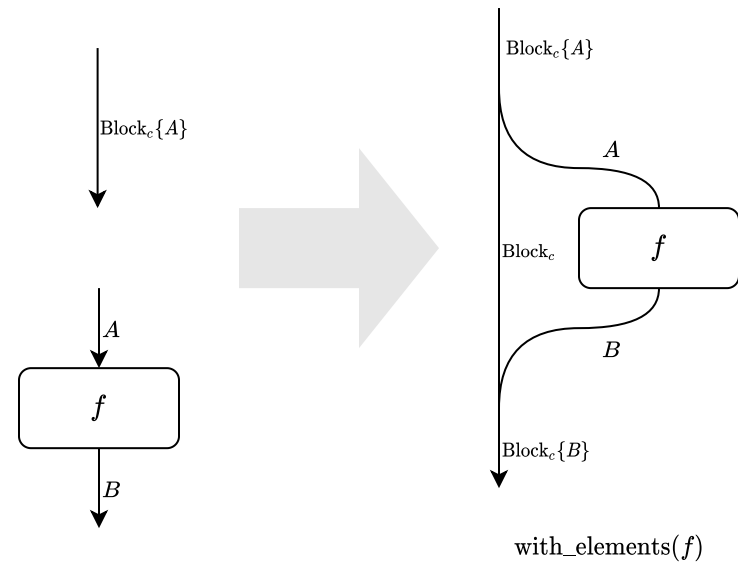


13. Unbundling



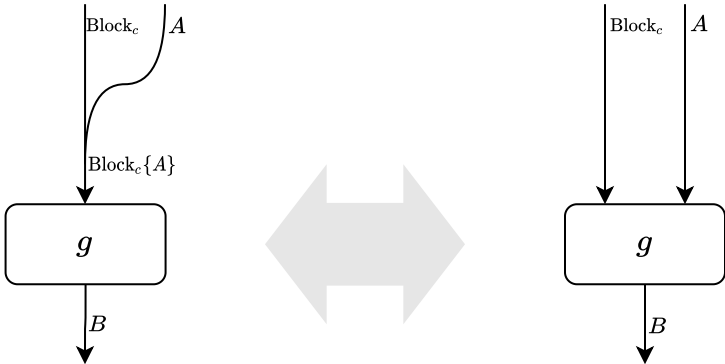
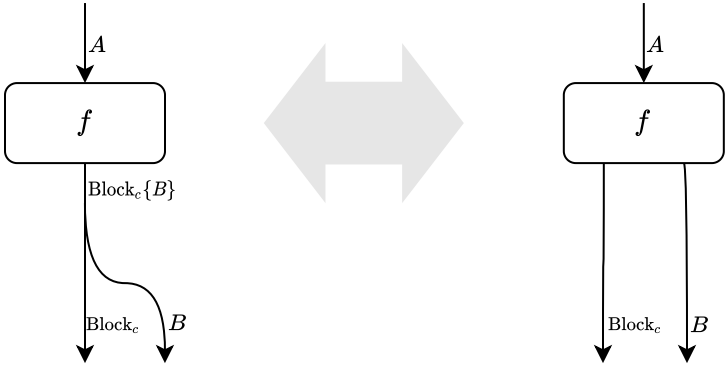
We can unbundle a wire of a block type into a functor and object components

14. Object Transformation

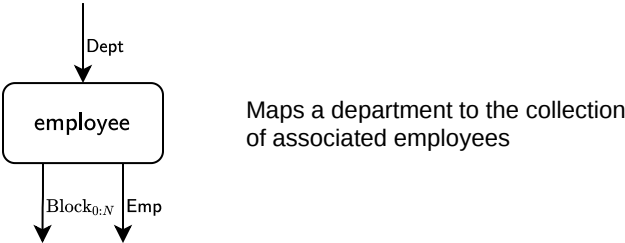


Then any compatible transformation can be applied to the object wire, which indicates that the transformation is applied to every element of the block

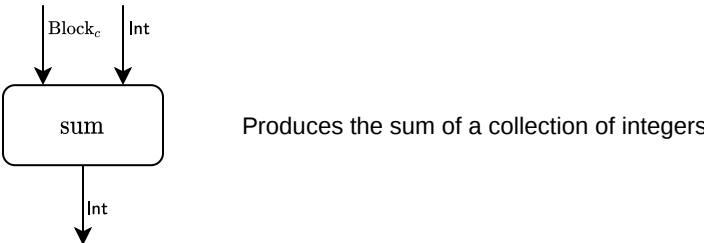
13. Multiwired transformations



14. Example: Multiwired Transformations

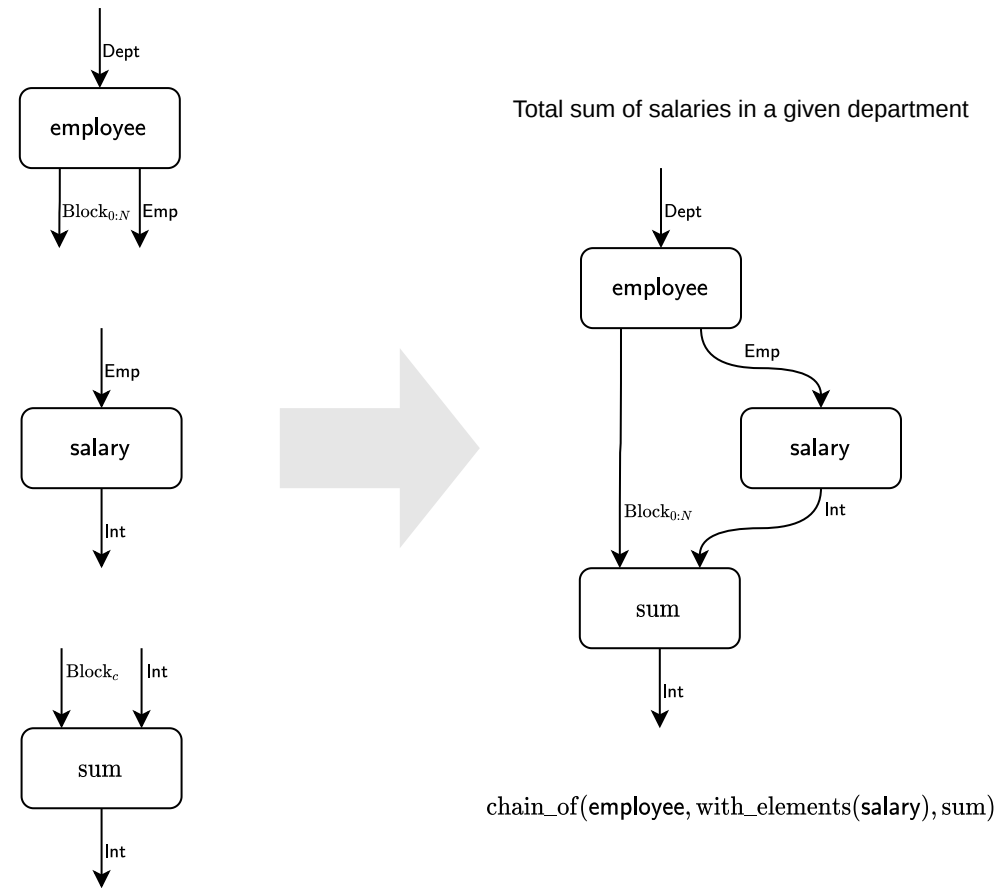


Maps a department to the collection of associated employees

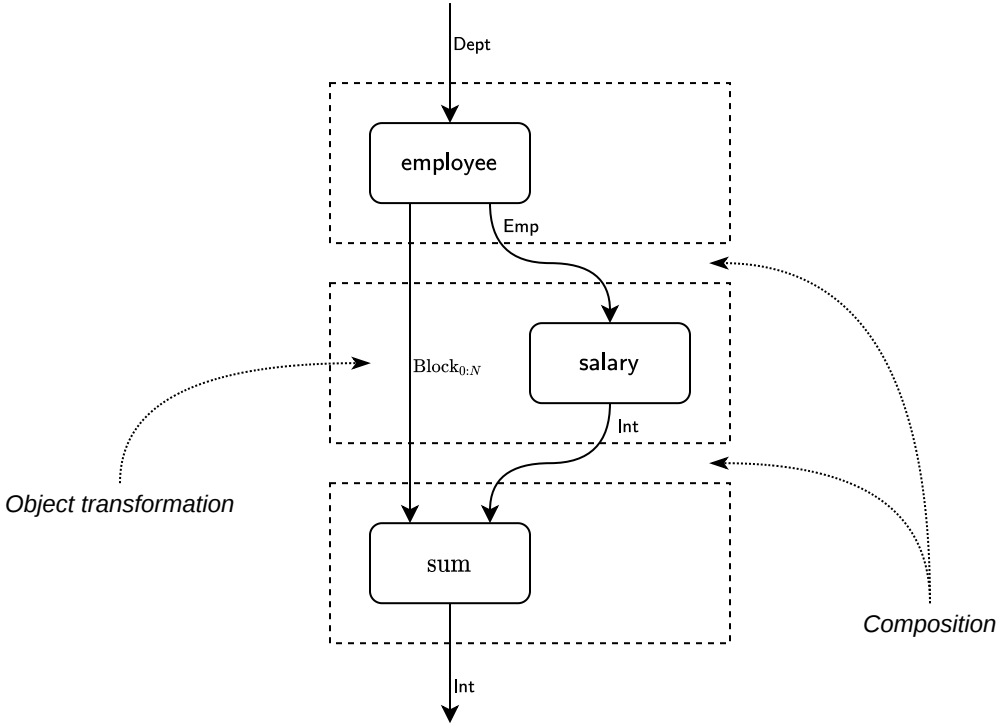


Produces the sum of a collection of integers

14. Example: Multiwired Composition

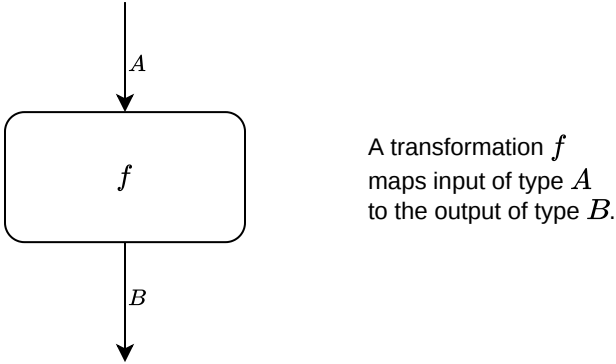


15. Example: Details

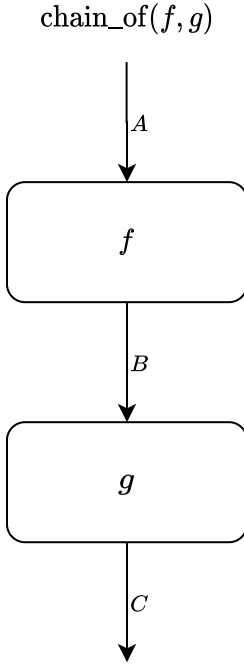
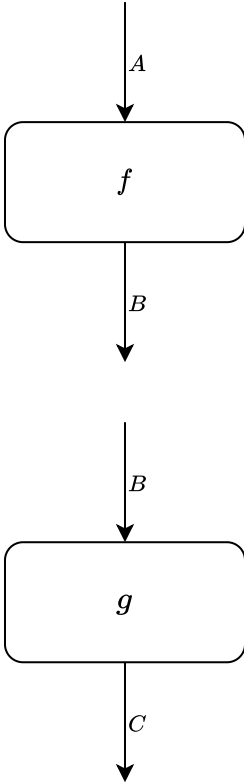


chain_of(employee, with_elements(salary), sum)

Transformation

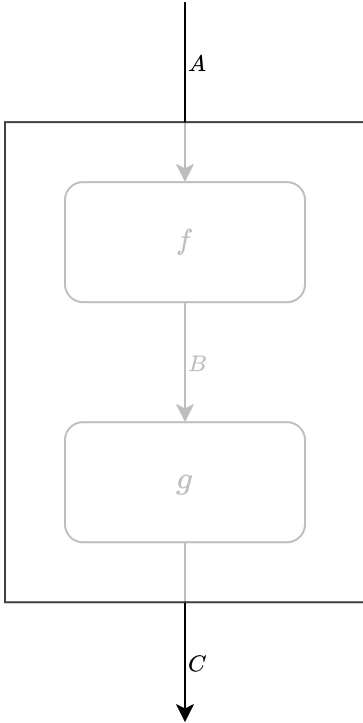


Composition



Transformations with compatible input and output
can be composed.

Composition is a Transformation



Crucially, a composition of transformations is again a transformation.