Function Composition

$[Person, Char, B, Record, ValidatedRecord, ProcessedRecord, Output]$

Example 1 : Basic Forward Composition

Forward composition applies f first, then g:

$$
\begin{array}{|l}
f : \mathbb{N} \longrightarrow \mathbb{N} \\
g : \mathbb{N} \longrightarrow \mathbb{N} \\
h : \mathbb{N} \longrightarrow \mathbb{N} \\
\hline
\forall\, n : \mathbb{N} \bullet f(n) = n + 1 \\
\forall\, n : \mathbb{N} \bullet g(n) = n * 2 \\
h = f \circ g
\end{array}
$$

h(n) = g(f(n)) = g(n + 1) = (n + 1) * 2 = 2n + 2

Example 2 : Backward Composition

Backward composition applies the second function first (using comp):

$$
\begin{array}{|l}
f2 : \mathbb{N} \longrightarrow \mathbb{N} \\
g2 : \mathbb{N} \longrightarrow \mathbb{N} \\
h2 : \mathbb{N} \longrightarrow \mathbb{N} \\
\hline
\forall\, n : \mathbb{N} \bullet f2(n) = n + 1 \\
\forall\, n : \mathbb{N} \bullet g2(n) = n * 2 \\
h2 = f2 \fatsemi g2
\end{array}
$$

h2(n) = f2(g2(n)) = f2(n * 2) = (n * 2) + 1 = 2n + 1

Example 3 : Comparison - Forward vs Backward

Forward composition f ∘ g:

Input → f → g → Output

(f ∘ g)(x) = g(f(x))

Backward composition f o g:

Input → g → f → Output

(f o g)(x) = f(g(x))

Notice the $reversal : \circ applies left - to - right$, o applies right-to-left.

Example 4 : Three - Function Composition

$$
\begin{array}{|l}
addOne : \mathbb{N} \longrightarrow \mathbb{N} \\
double : \mathbb{N} \longrightarrow \mathbb{N} \\
square : \mathbb{N} \longrightarrow \mathbb{N} \\
pipeline : \mathbb{N} \longrightarrow \mathbb{N} \\
\hline
\forall\, n : \mathbb{N} \bullet addOne(n) = n + 1 \\
\forall\, n : \mathbb{N} \bullet double(n) = 2 * n \\
\forall\, n : \mathbb{N} \bullet square(n) = n * n \\
pipeline = addOne \circ double \circ square
\end{array}
$$

pipeline(n) = square(double(addOne(n))) = square(double(n + 1)) = square(2(n + 1)) = (2(n + 1))ˆ{}2

Example 5 : Identity Composition Laws

$$
\begin{array}{|l}
identity : \mathbb{N} \longrightarrow \mathbb{N} \\
anyFunc : \mathbb{N} \longrightarrow \mathbb{N} \\
leftId : \mathbb{N} \longrightarrow \mathbb{N} \\
rightId : \mathbb{N} \longrightarrow \mathbb{N} \\
\hline
\forall\, n : \mathbb{N} \bullet identity(n) = n \\
\forall\, n : \mathbb{N} \bullet anyFunc(n) = n * n \\
leftId = identity \circ anyFunc \\
rightId = anyFunc \circ identity
\end{array}
$$

$leftId = anyFunc$ (identity is left identity for ∘)

$rightId = anyFunc$ (identity is right identity for ∘)

Example 6 : Associativity

Composition is associative:

(f ∘ g) ∘ $h = f$ ∘ (g ∘ h)

You can group compositions in any order—the result is the same.

$$
\begin{array}{|l}
f3 : \mathbb{N} \longrightarrow \mathbb{N} \\
g3 : \mathbb{N} \longrightarrow \mathbb{N} \\
h3 : \mathbb{N} \longrightarrow \mathbb{N} \\
comp1 : \mathbb{N} \longrightarrow \mathbb{N} \\
comp2 : \mathbb{N} \longrightarrow \mathbb{N} \\
\hline
\forall\, n : \mathbb{N} \bullet f3(n) = n + 1 \\
\forall\, n : \mathbb{N} \bullet g3(n) = n * 2 \\
\forall\, n : \mathbb{N} \bullet h3(n) = n * n \\
comp1 = (f3 \circ g3) \circ h3 \\
comp2 = f3 \circ (g3 \circ h3)
\end{array}
$$

$comp1 = comp2$ for all inputs.

Example 7 : Composition with Partial Functions

$$
\begin{array}{|l}
safeSqrt : \mathbb{N} \pfun \mathbb{N} \\
addTen : \mathbb{N} \fun \mathbb{N} \\
composed : \mathbb{N} \pfun \mathbb{N} \\
\hline
\forall\, n : \mathbb{N} \bullet n * n = n \Rightarrow safeSqrt(n) = n \\
\forall\, n : \mathbb{N} \bullet addTen(n) = n + 10 \\
composed = safeSqrt \circ addTen
\end{array}
$$

composed is only defined where safeSqrt is defined, then adds 10 to the result.

Example 8 : Relation Composition

Composition works for relations too:

$$
\begin{array}{|l}
parent : Person \rel Person \\
grandparent : Person \rel Person \\
\hline
grandparent = parent \circ parent
\end{array}
$$

grandparent relates a person to their grandparents via two parent links.

Example 9 : Function Pipeline Example

$$
\begin{array}{|l}
toLowerCase : \seq\, Char \fun \seq\, Char \\
trim : \seq\, Char \fun \seq\, Char \\
reverse : \seq\, Char \fun \seq\, Char \\
process : \seq\, Char \fun \seq\, Char \\
\hline
process = toLowerCase \circ trim \circ reverse
\end{array}
$$

Process a string by converting to lowercase, trimming whitespace, then reversing.

Example 10 : Mathematical Functions

$$
\begin{array}{|l}
squareFunc : \mathbb{N} \fun \mathbb{N} \\
tripleFunc : \mathbb{N} \fun \mathbb{N} \\
combined : \mathbb{N} \fun \mathbb{N} \\
\hline
\forall\, n : \mathbb{N} \bullet squareFunc(n) = n * n \\
\forall\, n : \mathbb{N} \bullet tripleFunc(n) = 3 * n \\
combined = squareFunc \circ tripleFunc
\end{array}
$$

Combined applies squareFunc first, then triples the result.

Example 11 : Inverse via Composition

$$
\begin{array}{|l}
encrypt : \seq\, Char \fun \seq\, Char \\
decrypt : \seq\, Char \fun \seq\, Char \\
identity\_check : \seq\, Char \fun \seq\, Char \\
\hline
identity\_check = encrypt \circ decrypt
\end{array}
$$

If encrypt and decrypt are inverses, then identity_check should be the identity function.

Example 12 : Partial Composition

When composing f : $A \longrightarrow B$ and $g : C \longrightarrow$ D, we need ran f subseteq dom g.

$$
\begin{array}{|l}
f4 : \mathbb{N} \pfun \mathbb{N} \\
g4 : \mathbb{N} \pfun \mathbb{N} \\
h4 : \mathbb{N} \pfun \mathbb{N} \\
\hline
f4 = \{1 \mapsto 10, 2 \mapsto 20\} \\
g4 = \{10 \mapsto 100, 20 \mapsto 200\} \\
h4 = f4 \circ g4
\end{array}
$$

h4 = $\{1 \mapsto 100, 2 \mapsto 200\}$. Composition is defined where ranges align.

Example 13 : Monoid Structure

Functions from $A \longrightarrow A$ with composition form a monoid:

1. Composition is associative: (f ∘ g) ∘ $h = f \circ$ (g ∘ h)

2. Identity *element* : $id(x) =$ x, id ∘ $f = f = f \circ$ id

3. Closure: composing $A \longrightarrow A$ functions gives another $A \longrightarrow A$ function

Example 14 : Practical Example - Data Processing Pipeline

In practice, you might compose functions to create data processing *pipelines* : *validate* ∘ *process* ∘ *format*.*Eachstagetransformsdatathrough* a series of operations. The notation f ∘ g ∘ h means "apply f, then g, then h" which creates a pipeline from input to output.

Example 15 : Composition Operator Precedence

Composition operators bind tighter than logical operators:

f ∘ $g = h$ means (f ∘ g) = h, not f ∘ $(g = h)$

Use parentheses to clarify: (f ∘ g) or f ∘ (g ∘ h)

Example 16 : Self - Composition

$$
\begin{array}{|l}
doubleIt : \mathbb{N} \longrightarrow \mathbb{N} \\
quadrupleIt : \mathbb{N} \longrightarrow \mathbb{N} \\
\hline
\forall\, n : \mathbb{N} \bullet doubleIt(n) = 2 * n \\
quadrupleIt = doubleIt \circ doubleIt
\end{array}
$$

quadrupleIt(n) = doubleIt(doubleIt(n)) = 2 * (2 * n) = 4n

Example 17 : Iterated Composition

$$
\begin{array}{|l}
increment : \mathbb{N} \longrightarrow \mathbb{N} \\
addFive : \mathbb{N} \longrightarrow \mathbb{N} \\
\hline
\forall\, n : \mathbb{N} \bullet increment(n) = n + 1 \\
addFive = increment \circ increment \circ increment \circ increment \circ increment
\end{array}
$$

Composing increment 5 times gives addFive.

Example 18 : Functional Programming Style

Composition enables a functional programming style:

Instead $of : result = \mathrm{h(g(f(x)))}$

$Use : pipeline = \mathrm{f} \circ \mathrm{g} \circ \mathrm{h};\ result = pipeline(\mathrm{x})$

This separates pipeline definition from application, improving modularity.

Example 19 : Best Practices

When using composition:

1. Choose forward (∘) or backward (o) consistently in your codebase

2. Document the data flow direction

3. Break complex pipelines into named intermediate steps

4. Type-check that domains and ranges align

5. Use composition to create reusable pipelines

6. Test individual functions before composing them

7. $Remember : \circ readsleft - to - right$ (natural for pipelines), o reads right-to-left (traditional math notation)