

Structure-Preserving Functions

by Sven Nilsen, 2017

Of all functions of type $\text{bool} \rightarrow \text{bool}$, there are only two structure-preserving functions:

$\text{not} := \text{if}(\text{false}, \text{true})$

$\text{id} := \text{if}(\text{true}, \text{false})$

$\text{if} := \lambda(a, b) = \lambda(x : \text{bool}) = \text{if } x \{ a \} \text{ else } \{ b \}$

These functions are called structure-preserving because when using in a path, the path set is always non-empty. All structure-preserving functions have inverses.

In order to be a structure preserving function, it must map from one type to another that contains the same or more number of elements.

$f : A \rightarrow B \wedge |A| \leq |B|$

However, this is not sufficient to define a structure-preserving function. A more accurate way is to think of the type B having some implicit sub-type that constrains it to those values returned by f . This constrained type must have the same number of elements as the input type:

$f : A \rightarrow B \wedge |A| = |f(A)|$

Another way of writing is using the existential path:

$f : A \rightarrow B \wedge |A| = |\exists f|$

This is a shorthand version for:

$f : A \rightarrow B \wedge |A| = |\exists f| \text{ true}$

Therefore, one has the following for any function of type $A \rightarrow B$:

$|f(A)| = |\exists f|$

$f(A) \subseteq B$

$f : A \rightarrow B$

To construct all structure-preserving functions of type $A \rightarrow A$, one can use the id function, take the partial function pairs $(x, \text{id}(x))$ and then rearrange the outputs:

$(x_i, \text{id}(x_i)) \quad \neg \exists k: (\neg = i) \{ (x_k, \text{id}(x_k)) \}$

This means there exists $|A|!$ number of structure-preserving functions of type $A \rightarrow A$.

If the existential path of a function is true_1 , then the length of the existential path is equal to the length of the output type. The same is also true in the other direction:

$$(\exists f \Leftrightarrow \text{true}_1) \Leftrightarrow (|\exists f| == |B|)$$

$$f : A \rightarrow B$$

For example:

$$f : \text{bool} \times \text{bool} \rightarrow \text{bool} \times \text{bool} \wedge |\text{bool} \times \text{bool}| == |f(\text{bool} \times \text{bool})|$$

$$|\text{bool} \times \text{bool}| == |\exists f|$$

$$|\text{bool}| \cdot |\text{bool}| == |\exists f|$$

$$2 \cdot 2 == |\exists f|$$

$$4 == |\exists f|$$

$$\exists f \Leftrightarrow \text{true}_1$$

$$\text{Because } |\text{bool} \times \text{bool}| == 4 \text{ and } (\exists f \Leftrightarrow \text{true}_1) \Leftrightarrow (|\exists f| == |\text{bool} \times \text{bool}|).$$

There exists $4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$ number of structure preserving functions of type $\text{bool} \times \text{bool} \rightarrow \text{bool} \times \text{bool}$.

For functions of type $A \rightarrow B$ there is a way to count the number of structure-preserving functions:

$$|f| == |A|! \cdot \text{bin}(|B|, |A|) == |B|! / (|B| - |A|)!$$

$$f : A \rightarrow B \wedge |A| == |f(A)|$$

For example, for functions of type $\text{bool} \rightarrow \text{bool} \times \text{bool}$, there exists 12 structure-preserving functions:

$$|f| == |\text{bool}|! \cdot \text{bin}(|\text{bool} \times \text{bool}|, |\text{bool}|)$$

$$|f| == 2! \cdot \text{bin}(4, 2)$$

$$|f| == 2 \cdot 6$$

$$|f| == 12$$

$$f : \text{bool} \rightarrow \text{bool} \times \text{bool} \wedge |\text{bool}| == |f(\text{bool})|$$

The reason is that there are 2 structure-preserving functions of type $\text{bool} \rightarrow \text{bool}$. For each of these functions one can pick 2 values of type $\text{bool} \times \text{bool}$ to use as output. The values can not be rearranged per function of type $\text{bool} \rightarrow \text{bool}$, because it would be the same as counting the same functions more than once.

From this way of counting structure-preserving functions of different types, there is an algorithm to construct all structure-preserving functions of type $A \rightarrow B$:

1. Construct all structure-reserving functions of type $A \rightarrow A$.
2. Find all ordered maps from B to A .
3. Replace outputs of 1) with 2) using the order of A to look up the value of B .