Chapter 5

List Comprehensions

Connor Baker

Compiled on December 21, 2019 at 9:04pm

5.1 Basic Concepts

In mathematics, we can construct a set with a generator. If we wanted to create the set of all odd numbers, we could use the set-builder notation $\{2k+1|x \in \mathbb{Z}^+\}$.

Haskell presents a similar opportunity, but it calls it a list comprehension. If we were to want to build a list of the first 10 perfect squares, we could do

```
> [x^2 | x <- [1..10]]
[1,4,9,16,25,36,49,64,81,100]
```

Just like in mathematics, the bar (`|`) is read as such that. The expression `x <- [1..10]` is called a generator, where `x` is the variable, `<-` is read as drawn from, and `[1..10]` is a list of integers from 1 to 10, inclusive.

If the generators are being used to construct a tuple, changing the order of the generator changes the order of the resulting tuples. Additionally, generators can depend upon the variables of previous generators.

When we don't care to store certain elements in a list comprehension, we can use the wildcard pattern (`_`). As an example, one implementation of the length function is

```
length :: [a] -> Int
length xs = sum [1 | _ <- xs]
```

5.2 Guards

If we wanted additional constraints, called guards, we can add them after a generator.

5.3 The zip Function

The `zip` function takes two lists and combines them to create a list of pairs. The function stops when either one of the lists is exhausted.

> Within the definition for positions, the expression `[0..]` produces the list of indices `[0,1,2,3,...]`. This list is notionally infinite, but under lazy evaluation only as many elements of the list as required by the context in which it is used, in this case zipping with the input list xs, will actually be produced. Exploiting lazy evaluation in this manner avoids the need to explicitly produce a list of indices of the same length as the input list.
>
> Excerpt From: Graham Hutton. "Programming in Haskell" (2nd ed.).

5.4 String Comprehensions

`String`s are not primitives in Haskell; they are lists of characters. As such, all the functions that work on lists also work on `String`s.

5.6 Chapter Remarks

> The term comprehension comes from the axiom of comprehension in set theory, which makes precise the idea of constructing a set by selecting all values that satisfy a particular property.
>
> Excerpt From: Graham Hutton. “Programming in Haskell” (2nd ed.).