15-312 Lecture on Recursive Functions

Defining Recursive Functions

Let's see how the familiar factorial function looks like once we expose the recusion. Here is the definition:

$$\begin{array}{l}\mathbf{fun}\ \mathrm{fact}(n)\ = \\ \quad \mathbf{if}\ n = 0 \\ \quad \mathbf{then}\ 1 \\ \quad \mathbf{else}\ n * \mathrm{fact}(n-1)\end{array}$$

(without patterns, because we haven't studied yet how to handle them.) Then the formal definition looks like this:

$$\begin{array}{l}\mathsf{rec}[\ \mathsf{nat} \rightarrow\ \mathsf{nat}](\mathit{fact}. \\ \quad \lambda n :\ \mathsf{nat}.\, \mathsf{ifz}(n, 1, n * \mathit{fact}(n-1))\end{array}$$

Here, fact is the bound name of the recursion: it stands for the object we are recursing upon. In this case, this is a function from nat to nat. We are using the construct ifz, which discriminates on whether its first argument is zero or not.

For typographic convenience, we will omit the types below (of course this should not be done, but it makes things a lot easier to write). This definition then looks like

$$\mathsf{rec}(\mathit{fact}.\, \lambda n.\, \mathsf{ifz}(n, 1, n * \mathit{fact}(n-1)))$$

Furthermore, we will occasionally use the abbreviation "fact"(quotes included) to refer to this whole expression.

Using Recursive Functions

Let's calculate ("fact"2):

$$
\begin{array}{rcl}
\text{“fact”}\ 2 & \mapsto & ([\text{“fact”}/fact](\lambda n.\, \mathsf{ifz}(n, 1, n * fact(n-1)))\ 2) \\
& = & (\mathsf{ifz}(n, 1, n * \text{“fact”}(n-1))\ 2) \\
& \mapsto & [2/n](\mathsf{ifz}(n, 1, n * \text{“fact”}(n-1))) \\
& = & \mathsf{ifz}(2, 1, 2 * \text{“fact”}(2-1)) \\
& \mapsto & 2 * \text{“fact”}(2-1) \\
& \mapsto & 2 * ([\text{“fact”}/fact](\lambda n.\, \mathsf{ifz}(n, 1, n * fact(n-1))))(2-1) \\
& = & 2 * (\lambda n.\, \mathsf{ifz}(n, 1, n * \text{“fact”}(n-1)))(2-1) \\
& \mapsto & 2 * (\lambda n.\, \mathsf{ifz}(n, 1, n * \text{“fact”}(n-1)))(1) \\
& \mapsto & 2 * [1/n](\mathsf{ifz}(n, 1, n * \text{“fact”}(n-1))) \\
& = & 2 * \mathsf{ifz}(1, 1, 1 * \text{“fact”}(1-1)) \\
& \mapsto & 2 * 1 * \text{“fact”}(1-1) \\
& \mapsto & 2 * 1 * ([\text{“fact”}/fact](\lambda n.\, \mathsf{ifz}(n, 1, n * fact(n-1))))(1-1) \\
& = & 2 * 1 * (\lambda n.\, \mathsf{ifz}(n, 1, n * \text{“fact”}(n-1)))(1-1) \\
& \mapsto & 2 * 1 * (\lambda n.\, \mathsf{ifz}(n, 1, n * \text{“fact”}(n-1)))(0) \\
& \mapsto & 2 * 1 * [0/n](\mathsf{ifz}(n, 1, n * \text{“fact”}(n-1))) \\
& = & 2 * 1 * \mathsf{ifz}(0, 1, 0 * \text{“fact”}(0-1)) \\
& \mapsto & 2 * 1 * 1 \\
& \mapsto & 2 * 1 \\
& \mapsto & 2
\end{array}
$$

For readability, we have separated the three recursive calls and the final arithmetic calculation.