Tutorial 3B: Lambda-terms

The lambda-calculus is defined by the grammar:

$$M, N \quad ::= \quad x \quad | \quad \lambda x.\, N \quad | \quad N M$$

This defines the **lambda-terms** (or λ**-terms**), the terms of the lambda-calculus, as being either:

- a **variable** x (from a pre-defined set of variables),
- an **abstraction** $\lambda x.\, N$ with a variable x over a lambda-term N, or
- an **application** NM of one lambda-term N to another M.

Parentheses are added where necessary to make sure terms are unambiguous. Application associates to the left: $N M_1 M_2 \dots M_k$ is $(\dots(N M_1) M_2 \dots) M_k$. Variables come in two flavours, depending on their context:

- a **free** variable is one not associated with any abstraction λx. When building a term inductively, all variables start out free.
- a **bound** variable does belong to an abstraction λx. When building a term $\lambda x.N$, all previously free variables x in N are now bound, by the new abstraction λx. Variables that were already bound, stay bound, and remain with their original binder.

Note that a λ-term is any term of the λ-calculus, not just those of the form $\lambda x.N$ that start with an abstraction.

Exercise 1: For each of the following terms:

1. say if it's a **variable**, **abstraction**, or **application**,
2. encircle the free variables,
3. connect each bound variable to its binder with an arrow, and
4. underline any **redexes**.

For example, the term

$$\lambda x.(\lambda y.x)z$$

is an **abstraction**, and would be analyzed as follows:

$$\lambda x.\ \underline{(\lambda y.\ x)\ z}$$

a) x

b) $\lambda x.x$

c) $(\lambda a.z)\,a$

d) $\lambda a.z\,a$

e) $(\lambda n.n)\,z$

f) $(\lambda x.x)\,(\lambda x.x)$

g) $x\,(\lambda y.y)\,(\lambda z.z)$

h) $\lambda z.(\lambda y.\,(\lambda x.x)\,y)\,z$

i) $\lambda x.(\lambda y.(\lambda z.x\,y)\,y\,z)$

j) $(\lambda t.((\lambda t.(\lambda t.t)\,t)\,t))\,t$