Addendum to Project 1: Programming in ML — main project due Wednesday 28 March, tasks described here due Monday 2 April

A library of λ-terms

$\mathbf{I} \triangleq \lambda x.x$ $\mathbf{K} \triangleq \lambda xy.x$ $\mathbf{S} \triangleq \lambda fgx.(fx)(gx)$ $\mathbf{B} \triangleq \lambda fgx.f(gx)$ $\mathbf{C} \triangleq \lambda fgx.fxg$

$\omega \triangleq \lambda x.xx$ $\Omega \triangleq \omega\omega$ $\mathbf{Y} \triangleq \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$

$\mathbf{true} \triangleq \lambda xy.x$ $\mathbf{false} \triangleq \lambda xy.y$ $\mathbf{not} \triangleq \lambda t.t\ \mathbf{false}\ \mathbf{true}$ $\mathbf{cond} \triangleq \lambda ee_1e_2.ee_1e_2$

$\mathbf{pair} \triangleq \lambda e_1e_2f.fe_1e_2$ $\mathbf{fst} \triangleq \lambda p.p\ \mathbf{true}$ $\mathbf{snd} \triangleq \lambda p.p\ \mathbf{false}$

$\mathbf{0} \triangleq \lambda fx.x$ $\mathbf{1} \triangleq \lambda fx.fx$ $\mathbf{2} \triangleq \lambda fx.f(fx)$ $\mathbf{succ} \triangleq \lambda nfx.nf(fx)$ $\mathbf{add} \triangleq \lambda mnfx.mf(nfx)$
$\mathbf{iszero} \triangleq \lambda n.n(\lambda x.\mathbf{false})\mathbf{true}$ $\mathbf{prefn} \triangleq \lambda fp.\mathbf{pair}\ \mathbf{false}(\mathbf{cond}(\mathbf{fst}\ p)(\mathbf{snd}\ p)(f(\mathbf{snd}p)))$
$\mathbf{pred} \triangleq \lambda nfx.\mathbf{snd}(n(\mathbf{prefn}\ f)(\mathbf{pair}\ \mathbf{true}\ x))$

$\mathbf{cons} \triangleq \lambda hts.sht$ $\mathbf{hd} \triangleq \lambda L.L\ \mathbf{true}$ $\mathbf{tl} \triangleq \lambda L.L\ \mathbf{false}$ $\mathbf{nil} \triangleq \lambda x.\mathbf{true}$ $\mathbf{isempty} \triangleq \lambda L.L(\lambda ht.\mathbf{false})$

Normal forms of some λ-terms

$\mathbf{SKK} \twoheadrightarrow \lambda x.x$ $\mathbf{K}(\mathbf{SII}) \twoheadrightarrow \lambda ab.bb$ $\mathbf{S}(\mathbf{S}(\mathbf{KS})(\mathbf{KI}))(\mathbf{KI}) \twoheadrightarrow \lambda ab.bb$ $\mathbf{SSSSSSS} \twoheadrightarrow \lambda ab.(ab(ab(ab\lambda c.ac(bc))))$

Additional tasks

1. (30 pts. extra credit)

   Implement δ-conversion rules for arithmetic, and input/output syntax for numbers. Extend the parser to recognize decimal integers and the common arithmetic operators +, -, etc. The operators do not need to be infix. You may rely on the underlying SML type *int*.

2. (30 pts. extra credit)

   Implement δ-conversion rules for a primitive list type with constructors **cons** and **nil** that can be used to build lists of λ-terms. Extend the parser to recognize **cons** and **nil**, and extend the pretty-printer to print them.

3. (30 pts. extra credit)

   Extend the parser to recognize and the pretty-printer to print infix arithmetic operators, the infix **cons** (written **::**), and the list notation **[,]**.

Any of the extensions you do should be written modularly, so that they can be included or excluded at will. This will allow you, for instance, to compare the efficiency of arithmetic using Church's λ-terms and arithmetic using built-in primitives.

Grading

Total number of points available on this project is now 230. Full credit remains equivalent to 100 points.