Homework 5 — assigned Monday 9 April — due Sunday 22 April

All ML code in this homework assignment must use the SML module language, and it must be organized using the SML/NJ Compilation Manager. Place all functor applications, if any, together in a file `link.sml`. Put all tests into a structure `Tests` in the file `tests.sml`. To avoid conflicts, each exercise must be placed in its own subdirectory.

5.1 Lambda-calculus (15pts) [A.1]

(Narrative: provide your answer as a plain-text file or PDF.)

A library of λ-terms

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

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

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

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

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

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

Normal forms of some λ-terms

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

5.1.1

Show that the following λ-terms have a normal form:

1. $(\lambda y.yyy)((\lambda ab.a)\mathsf{I}(\mathsf{SS}))$
2. $(\lambda yz.zy)((\lambda x.xxx)(\lambda x.xxx))(\lambda w.\mathsf{I})$

5.1.2

For each of the following λ-terms either find its normal form or show that it has no normal form:

1. $(\lambda x.xx)(\lambda x.x)$
2. $(\lambda x.xx)(\lambda x.xx)$

3. $\mathbf{Y}$
4. $\mathbf{Y}(\lambda y.y)$

5.1.3

(Turing) Let $A \triangleq \lambda xy.y(xxy)$. Let $\Theta \triangleq AA$. Show that Θ is a fixed-point operator.

5.2 Lambda-calculus Interpreter (35pts) [K.1.1; K.3.1; K.3.2]

(Skeleton files for this exercise have been provided. A summary of the mechanics of the λ-calculus has also been provided.)

Develop an interpreter for the λ-calculus that will automate reductions. This program will follow literally the rules for β-conversion and the rules for substitution. The internal representation of λ-terms is essentially the same as the textual representation, though the data type makes the bracketting structure apparent, and pattern-matching easier.

We must first specify the internal representation for λ-terms. The following type must be used:

```
type var = string
datatype expr = Var of var
              | Abs of var * expr
              | App of expr * expr
```

The following tasks build the interpreter bottom-up.

5.2.1

Implement an environment mapping identifiers to λ-terms, with type `string -> expr`. There should be a mechanism to build new environments out of old ones by introducing a new definition for an identifier.

5.2.2

Implement a function `freeVariables: expr -> var list`.

5.2.3

Implement a function `isFreeVariable: var * expr -> bool`.

5.2.4

Implement a function `substitute: expr * var * expr -> expr`, such that `substitute (e, x, t)` substitutes *t* for *x* in *e*. To generate fresh variable symbols, use the following code (which makes judicious use of imperative features in SML):

```
local
   val counter = ref 0
in
   fun genSym () =
   let
      val x = !counter
      val _ = counter := x+1
   in
      "_" ^ Int.toString x
   end
end
```

5.2.5

Implement a function `isBetaRedex: expr -> bool`.

5.2.6

Implement a function `convertBetaRedex: expr -> expr`.

5.2.7

Implement a function `convert: expr -> expr option` which finds a leftmost outermost β redex, if any, and performs β conversion.

5.2.8

Implement a function `reduce: expr -> expr` that applies β conversion steps in normal order until a normal form is found.

5.2.9

Test your program by reducing various λ-terms, such as: SKK; $\mathsf{K}(\mathsf{SII})$; $\mathsf{S}(\mathsf{S}(\mathsf{KS})(\mathsf{KI}))(\mathsf{KI})$; $\mathsf{SSSSSSS}$.

5.2.10

Implement the factorial function over Church numerals. (Use the Y combinator.) Test your program by having it compute $n!$ for various n. Report how fast the evaluator works for different inputs or input sizes. (Take into account that with a unary representation, different numbers have different sizes.)

5.3 Arithmetic expression translator (10pts) [C; E]

(Narrative: provide your answer as a plain text file or PDF.) Discuss possible improvements of the translation scheme indicated in homework exercise 4.1.

5.4 Abstract machine (40pts) [A.7; C]

An abstract machine specification has been provided separately. In a programming language of your choice, implement an emulator for this machine. Discuss how you chose the implementation language, and how its features helped or hindered your work; document how you approached this problem and include any additional software tools you built along the way; document the specifics of how your emulator is used (narrative: provide your answer as a plain text file or PDF). A few example programs for this machine have been provided separately.

How to turn in

Submission instructions: see course mailing list.

Include the following statement with your submission, signed and dated:
I pledge my honor that in the preparation of this assignment I have complied with the University of New Mexico Board of Regents' Policy Manual.