Homework 1 — Simple ML core language programs — assigned Monday 27 January, due Wednesday 5 February

Total number of points available on this homework is 140. Full credit is equivalent to 100 points.

Reading assignment

Read Chapters 1, 2, and 3 of *ML for the Working Programmer*.

1.1 Integers (10pts)

Define a function called *cube*, of type $int \rightarrow int$, which returns the integer which is the cube of the integer it is applied to.

1.2 Types (15pts)

What types are assigned to:

1. **fn** $x => x$
2. **fn** $(\_, (x, \_)) => x$
3. **fn** $(x, y) => (y, x)$

Give one instantiation of each type.

1.3 List types (15pts)

Write a function *swapl* that takes a list of pairs as argument and returns a list of pairs in which the elements of each pair are swapped. Specify its type.

1.4 List types and higher-order functions (15pts)

Write a function *map2* that applies a function to all elements in all element lists in a list of lists. Specify its type. Compare the meaning of *map* in ML and in Scheme.

1.5 Using lists for arithmetic (45pts)

Numerals can be represented as lists of integers. For instance, decimal numerals can be expressed as lists of integers from 0 to 9. In this representation, the integer 12345678901234567890 would be represented as the ML list `[1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0]: int list`.

Write the following functions:

- (15pts) `makeLongInt: int -> int -> int list`, such that `makeLongInt` r n computes the list representation of the integer n in radix r. You can assume that $n \geq 0$, and that $r > 1$.
- (15pts) `evaluateLongInt: int -> int list -> int`, such that `evaluateLongInt` r l computes an integer corresponding to the value of list l, which uses radix r. You can assume that l is a valid list for radix r, and the value of the list is small enough to fit into an ML `int`.
- (15pts) `addLongInts: int -> (int list * int list) -> int list`, such that `addLongInts` r (a,b) computes the sum of the nonnegative integers given by lists a and b; all lists use radix r. Lists a and b can represent arbitrarily large integers.

1.6 Transmission codes (40pts)

An alphabet is a set of symbols; for instance $\Sigma_1 = \{0,1,2,3,4,5,6,7,8,9\}$ is the alphabet of decimal digits. A string is a finite sequence of symbols drawn from a specific alphabet; for instance $x_1 = 2718281828$ is a string over the alphabet Σ_1. (In ML, we can use the ML type `int` for the symbols.) Given two strings of equal length, the Hamming distance between them is the number of positions in which their symbols differ. For instance, $H(2718281828, 1828182827) = 6$. For strings of length N, the Hamming distance is at least 0 and at most N.

With an alphabet of K symbols, there are K^N strings of length N. We'll say that two strings x and y are distinguishable if $H(x,y) \geq D$, where D is a prescribed minimum Hamming distance. Out of the total K^N strings, how many (at most) can we choose such that each two of them are distinguishable?

Try to work out (mathematicaly) how large the biggest possible subset is, with K, N, and D as parameters.

Write an ML function `maxSafeSubset: int list -> int -> int -> int list list` that will produce an exemplar of such a subset of distinguishable strings—the larger, the better. The function should be called like this: `maxSafeSubset alphabet wordLength minHammingDistance`; for instance we might invoke it as: `maxSafeSubset [0, 1] 3 2`, and it should then evaluate to `[[0, 0, 0], [0, 1, 1], [1, 0, 1], [1, 1, 0]]`.

Of particular interest are alphabets of size 2 and 4, and strings of length 10, 16, 18, and 32. You should test your implementation of `maxSafeSubset` on these cases.

How to turn in

Turn in your code by running

˜roshan/handin your-file

on a regular UNM CS machine.

You should use whatever filename is appropriate in place of your-file. You can put multiple files on the command line, or even directories. Directories will have their entire contents handed in, so please be sure to clean out any cruft.