CSL226: Programming Languages

Assignment: DFA for Numbers

The abstract syntax tree of regular expressions over a type `'a` may be defined as follows:

```
datatype 'a RE = Nil | Eps | Atom of 'a | Dot of 'a RE * 'a RE |
                 Or of 'a RE * 'a RE | Star of 'a RE
```

where the various constructors and their interpretations are obvious. Let the alphabet be exactly the characters in the string "`0123456789+-.E`". Let $NUM = INT \cup FIXED \cup FLOAT$ denote the set of all number tokens, where

1. INT is the set of all signed and unsigned integers
2. $FIXED$ is the set of all fixed-point real numbers
3. $FLOAT$ is the set of all floating point real numbers

To facilitate classification and recognition of the three kinds of patterns that are in NUM define another datatype

```
datatype class = INT | FIXED | FLOAT
```

What you have to do.

1. A **fully bracketed regular expression** is a string which represents a regular expression unambiguously as a string. Hence every non-atomic sub-regular-expression is enclosed in a pair of matching parentheses. In particular, the atomic regular expressions `Nil`, `Eps` and `Atom ''E''` are represented as the strings "Nil", "Eps" and "E" respectively. A non-atomic regular expression such as `(Dot (Star (Or (Atom ``0'', Atom ``1'')), Atom ``2''))` is represented by the string `` ``((0|1)*.2)'' ``.

   Write the following functions in SML.

   `toString: ('a -> string) -> 'a RE -> string.` This function outputs a fully bracketed string representation of the regular expression.

   `fromString: (string -> 'a option) -> string -> 'a RE option.` This function takes as input a string (without any whitespaces) and if it represents a regular expression in fully bracketed form, it constructs the regular expression r and outputs the value `SOME` r. Otherwise it outputs `NONE`.

2. Implement a function `dfa: 'a RE -> (int * 'a * int) list` which constructs a deterministic finite automaton to recognize the set of valid strings in the language NUM. In particular, the following exceptions have to be declared and raised. It includes error states when invalid symbols are encountered or when the input string to the function `accept` does not conform to the pattern expected for INT, $FIXED$ or $FLOAT$.

3. Implement a function `accept: string -> class option` which for any given input string of characters outputs one of the following from the type `class option`.

   - `SOME INT`
   - `SOME FIXED`
   - `SOME FLOAT`
   - `NONE`