Homework 1
COSE312, Spring 2017

Hakjoo Oh

Due: 03/31, 24:00

Problem 1 The goal of this assignment is to write a compiler that translates regular expressions to deterministic finite automata (DFAs).

1. Clone the Git repository for programming assignments:

   ```
   git clone https://github.com/kupl/Compiler2017.git
   ```

2. You can find the `hw1` directory and the following files in it:

   - `main.ml`: Driver code with some test cases. You can add your own test cases here.
   - `regex.ml`: The definition of regular expressions.
   - `nfa.ml`: NFA implementation. Read `nfa.mli` to see how to use the NFA module.
   - `dfa.ml`: DFA implementation. Read `dfa.mli` to see how to use the DFA module.
   - `hw1.ml`: Complete and submit this file for homework 1.

3. In `regex.ml`, regular expression is defined as follows:

   ```
   type alphabet = A | B
   type t =
     | Empty
     | Epsilon
     | Alpha of alphabet
     | OR of t * t
     | CONCAT of t * t
     | STAR of t
   ```

   where we assume $\Sigma = \{a, b\}$.

4. In `hw1.ml`, you can find code below:

```
let regex2nfa : Regex.t -> Nfa.t
=fun regex -> raise Not_implemented (* TODO *)

let nfa2dfa : Nfa.t -> Dfa.t
=fun nfa -> raise Not_implemented (* TODO *)

(* Do not modify this function *)
let regex2dfa : Regex.t -> Dfa.t
=fun regex ->
  let nfa = regex2nfa regex in
  let dfa = nfa2dfa nfa in
    dfa

let run_dfa : Dfa.t -> alphabet list -> bool
=fun dfa str -> raise Not_implemented (* TODO *)
```

Your job in this assignment is to implement the functions:

- `regex2nfa`, which converts a regular expression to an equivalent NFA,
- `nfa2dfa`, which converts an NFA to an equivalent DFA, and
- `run_dfa`, which takes a DFA and a string (i.e., a sequence of input symbols) and returns true (i.e., accept) or false (i.e., reject).