Homework 7
SNU 4190.210 Fall 2014

Chung-Kil Hur

due: 12/03 (Wed) 24:00

Exercise 1 "Turing Machine$_{\text{ml}}$"

Implement the Turing machine in OCaml. Note that you implemented it in Scheme for HW5.

Define a module `TuringMachine` with the following signature.

```
module type TM =
  sig

    type symbol = string
    type move = Right | Left | Stay
    type todo = Erase | Write of symbol
    type state = string
    type rule = state * symbol * todo * move * state
    type ruletable = rule list
    type tape
    type tm

    (* tape part *)
    val init_tape: symbol list -> tape
    val read_tape: tape -> symbol
    val write_tape: tape -> symbol -> tape
    val move_tape_left: tape -> tape
    val move_tape_right: tape -> tape
    val print_tape: tape -> unit

    (* rule table part *)
```

```
    val match_rule: state -> symbol -> ruletable -> todo * move * state

    (* main *)
    val make_tm: symbol list -> state list -> state -> ruletable -> tm
    val run_tm: tm -> tm
    val print_tm: tm -> int -> unit

  end
```
□

Exercise 2 "SKI combinator calculus$_{\text{ml}}$"

Implement the SKI combinator calculus in OCaml. Note that you implemented it in Scheme for HW4.

Define a module `SkiLiquid` with the following signature.

```
module type SKI =
  sig
    type liquid = S
                | K
                | I
                | V of string          (* variable *)
                | M of liquid * liquid (* mix of two liquids *)
    val react: liquid -> liquid
    val pprint: liquid -> unit
  end
```
□

Challenge 3 "Regular Expression Match"

Strings are sequences of 0,1,2 including empty string: ex) 0000, 1102201, 2010110012. A string s matches a regular expression c when s is included in the set of strings represented by c.

Regular expressions c are defined as follows:

$$c \quad \rightarrow \quad 0 \mid 1 \mid 2 \mid c \cdot c \mid c\texttt{+}c \mid c\texttt{?} \mid c\texttt{*}$$

The set of strings $[\![c]\!]$ represented by c are defined as follows, where ϵ represents

the empty string (*i.e.*, the string of length 0):

$$\begin{array}{rcl}
[\![0]\!] & = & \{0\} \\
[\![1]\!] & = & \{1\} \\
[\![2]\!] & = & \{2\} \\
[\![c_1 \cdot c_2]\!] & = & \{s_1 s_2 \mid s_1 \in [\![c_1]\!], s_2 \in [\![c_2]\!]\} \\
[\![c_1\texttt{+}c_2]\!] & = & [\![c_1]\!] \cup [\![c_2]\!] \\
[\![c\texttt{?}]\!] & = & \{\epsilon\} \cup [\![c]\!] \\
[\![c\texttt{*}]\!] & = & \{\epsilon\} \cup [\![c]\!] \cup [\![c \cdot c]\!] \cup [\![c \cdot c \cdot c]\!] \cup \cdots
\end{array}$$

Define a function `smatch` that, given a string s and a regular expression c, determines whether s matches c. For example,

`smatch "11"` $1 \cdot 0\texttt{*} \cdot 1\texttt{?}$

returns `true`; and

`smatch "11"` $(10)\texttt{*} \cdot 1$

returns `false`.

Write a module `Smatch` with the following signature `SMATCH`.

```
module type SMATCH =
  sig
    type c = Zero | One | Two
           | Mult of c * c
           | Sum of c * c
           | Opt of c              (* c? *)
           | Star of c             (* c* *)
    val smatch: string -> c -> bool
  end
```

□