CMSC 330: Organization of Programming Languages

Project 4

NFAs & Regular Expressions

- Goal
  - Implement NFAs
    - Calculate epsilon closure & move for NFA
    - Determine whether string accepted by NFA
  - Implement regular expressions
    - Print regular expression in postfix notation
    - Convert regular expression to NFA
  - Implement regular expression parser
    - Convert string to regular expression
    - Convert string to NFA

NFA

- Signature

```
type nfa
type transition = int * char option * int
      (* (s0, Some c, s1)  or (s0, None, s1)     *)

val make_nfa : int -> int list -> transition list -> nfa
val e_closure: nfa -> int list -> int list
val move: nfa -> int list -> char -> int list
val accept : nfa -> string -> bool
```


Regular Expression

- Signature

```
type regexp =
      Empty_String
    | Char of char
    | Union of regexp * regexp
    | Concat of regexp * regexp
    | Star of regexp

val regexp_to_string : regexp -> string
val regexp_to_nfa : regexp -> nfa
```

Simultaneous Declarations

- You can use let rec … and … and … in e to declare multiple functions simultaneously
  - All functions known in bodies of functions at same level
  - Useful for mutual recursion

```
let rec f1 x = print_int x ; f2 (x-1)
    and f2 y = print_int y ; f3 (y-1)
    and f3 z = if z > 0 then f1 z
    in f1 5 (* prints 5 4 3 2 1 0 *)
```


Concatenation Operators

- @
  - Type: 'a list -> 'a list -> 'a list
  - Example: [1;2;3] @ [4;5]
    - Returns the concatenated list [1;2;3;4;5]
- ^
  - Type: string -> string -> string
  - Example: "abc" ^ "de"
    - Returns the concatenated string "abcde"
- Other functions in Pervasives module at
  - http://caml.inria.fr/pub/docs/manual-ocaml/libref/Pervasives.html

String Operations

- String.get
  - Type: string -> int -> char
  - Example: String.get s x
    - Returns char in string s at index x
- String.concat
  - Type: string -> string list -> string
  - Example: String.concat s0 [s1;s2;s3]
    - Returns string s1^s0^s2^s0^s3
- Other functions in String module at
  - http://caml.inria.fr/pub/docs/manual-ocaml/libref/String.html

Tokens (Terminals)

- Token type

```
type token =
  Tok_Char of char
| Tok_Epsilon
| Tok_Union
| Tok_Star
| Tok_LParen
| Tok_RParen
| Tok_END
```

Scanner (Lexer)

- Converts string for regular expression into a list of tokens
- Examples
  - Tokenizer “a” = [(Tok_Char ‘a’); Tok_END]
  - Tokenizer “a*” = [(Tok_Star ‘a’); Tok_END]
  - Tokenizer “a|b” = [(Tok_Char ‘a’) ; Tok_Union ; (Tok_Char ‘b’); Tok_END]

Parser

- Lookahead
  - Head of the token list
- Match c
  - if c = head of token list then return tail, else error
- Functions
  - Take token list as argument
  - Returns tuple
    1. regexp for regular expression
    2. Remaining unmatched portion of token list
- See 12-parser.ml for detailed example