Solution - Homework 4: Lectures 7 & 8

CS 440: Programming Languages and Translators, Spring 2020

Lecture 7: Regular Expressions, part 2

1. (Regular expression matching even number of `a`'s or of `b`'s):

   `b*(ab*ab*)*|a*(ba*ba*)*` The first disjunct looks for some number of pairs of `a`'s, with possible `b`'s before, between, and after the `a`'s. The second disjunct is similar.

2. (Regular expression for comments of the form `/*` .... `*/`)

   The two ends of the regular expression are easy: `[ \t]*/\*` and `\*/[ \t]*`. The `[ \t]*` match zero or more whitespace characters and we use backslashes before the stars to keep them from being interpreted as Kleene stars. For the middle part,

   - Symbols that aren't stars are no problem; we can use `[^*\r]` (anything but a star or carriage return – we're looking at comments that fit on one line).
   - If we see a star, it might lead to the terminating `*/` or it might not.
     - To match the stars just before the ending `*/`, we can use `\**` (zero or or more stars).
     - For stars that don't lead to the ending `*/`, we can have `\*\**[^*/\r]` (one or more stars followed by something not a star, slash, or return).

   So altogether, the expression is

   ```
   [ \t]*/\*([^*\r]|\*\**[^*/\r])*\**\*\/[ \t]*
   [ \t]*      -- whitespace
   /\*         -- Start comment with /*
   (           -- Begin sequence of interior symbols
     [^*\r]         -- Interior symbol (not *, not return)
     |              -- or
     \*\**[^*\r]    -- Interior sequence of star, maybe more stars, and an interior symbol
   )*          -- End sequence of interior symbols
   \**\*\/     -- End comment with some optional stars and then */
   ```

   Pleasantly, `([^*\r]|\*\**[^*/r])*` can be simplified to just `(\**[^*/\r])*` (some optional stars followed by an interior symbol), so the whole expression can also be `[ \t]*/\*(\**[^\*\r])*\**\*\/[ \t]*`.

Programming Assignment (Lecture 7)

3. (Extend the regular expression matcher with (…)* and […])

```
-- We treat exp* as (exp exp*|empty)
match (re_star @(RE_star rexp)) input =
    match (RE_or [ RE_and [rexp, re_star], RE_empty ]) input

match (RE_in_set set) (head_inp : input')
    | head_inp `elem` set = Just input' -- head symbol in set?
match (RE_in_set _) _ = Nothing  -- if not, fail
```

4. (Capture matching string for regular expression)

   See attached files `HW_04_Capture_soln.hs` and `HW_04_Capture_Tests.hs`. To test, you can `:load HW_04_Capture_Tests`; it will import the capture solutions file.

Lecture 8: Finite State Automata

5. (DFAs & NFA for even `a`'s or even `b`'s)

DFA1 for b*(ab*ab*)*.

5a. Start and accepting state `0a`

State 0: We've seen an even number of `a`'s

State 1: We've seen an odd number of `a`'s

State	a	b
0a	1	0
1a	0	1

DFA2 for a*(ba*ba*)*.

5b. Symmetric to DFA1; start & accepting state `0b`.

States `0b`, `1b` (represent even/odd number of `b`'s respectively).

State	a	b
0b	0	1
1b	1	0

5c. (NFA combining parts (a) and (b)) New start state is *S*; accepting states are 0a and 0b.

State	ε	a	b
S	0a, 0b	.	.
0a	.	1	0
1a	.	0	1
0b	.	0	1
1b	.	1	0

6. (Convert NFA to DFA)

Original NFA. Start state *A*, accepting state *G*.

State	ε	x	y	z
A	*B*	*C*	*F*	
B	*D*		*H*	*G*
C	*H*	*C*	*A*	
D		*D, E*	*E*	
E	*D*	*D*		*E*
F		*G*		*G*
G		*C*	*E*	*F*
H	*C*	*H*		*H*

6a. After removing error states *D*, *E*. Start state *A*, accepting state *G*.

State	ε	x	y	z
A	*B*	*C*	*F*	
B			*H*	*G*
C	*H*	*C*	*A*	
F		*G*		*G*
G		*C*		*F*
H	*C*	*H*		*H*

6b. After taking ε-closure. Start state *AB*, accepting state *G*. Notation: *AB* and *CH* stand for states that combine the various states: {*A*, *B*} and {*C*, *H*}. *CH F* means two arrows: one to *CH* and one to {*F*}

State	x	y	z
AB	*CH*	*CH, F*	*G*
CH	*CH*	*AB*	*CH*
F	*G*		*G*
G	*CH*		*F*

Some explanation: $A \to_{\varepsilon} B$ tells us we need a state *AB*. We don't need a state *B* because there aren't any transitions that go to *B*, just to *A* (and hence *B*). But if we had had $C \to_{\mathtt{y}} B$ instead of $C \to_{\mathtt{y}} A$, we would have needed a separate state *B*. Since $C \to_{\varepsilon} H \to_{\varepsilon} C$, we don't need either state alone, just *CH*.

6.c After converting to set-of-states DFA and adding error state.

Start state *AB*, accepting states *G* and *CGH*. (Empty transitions go to an implicit error state.)

State	x	y	z
AB	*CH*	*CFH*	*G*
CH	*CH*	*AB*	*CH*
F	*G*		*G*
G	*CH*		*F*
CFH	*CGH*	*AB*	*CGH*
CGH	*CH*	*AB*	*CFH*

