Homework 4: Regular Expressions and NFAs

CIS 352: Programming Languages

8 February 2019, Version 3

Administrivia

- **No teams**, this assignment is a solo effort.
- Document in the cover sheet any ideas you use from other students or other sources.
- For Part I, *legible* hand written answers are fine.
- For Part II, copy all the files in `http://www.cis.syr.edu/courses/cis352/code/RegExp2/` and use `Top2.hs` as your starter file.
- Let me know if any of my QuickCheck tests seem dodgy.
- **Turn in Part I by:** dropping the papers in the CIS 352 bin on the 4th floor of SciTech.[1] Include a paper copy of your cover sheet.
- **Turn in Part II via:** Blackboard, include *(i)* your modified versions of `Matches2.hs`, `BuildNFA2.hs`, and `Top2.hs`. from the Reg2 directory, *(ii)* the transcripts of test runs, and *(iii)* your coversheet.

Grading Criteria

- The homework is out of 100 points.
- Each programming problem is ≈ 70% correctness and ≈ 30% testing.
- Omitting your name(s) in the source code looses you 5 points.

Typo corrections in red.

Part I: Problems on Paper

❖ ***Problem 1 (40 points)*** ❖

Use the rules on page 5 of the *Lexical Analysis* slides to give a formal derivation of each of the following. Each part is 4 points except for (i) which is 8 points.

(a) $(\mathbf{a}|(\mathbf{b}|\mathbf{c})) \Downarrow \mathbf{a}$

(b) $(\mathbf{a}|(\mathbf{b}|\mathbf{c})) \Downarrow \mathbf{b}$

(c) $(\mathbf{a}|(\mathbf{b}|\mathbf{c})) \Downarrow \mathbf{c}$

(d) $(\mathbf{a}(\mathbf{bc})) \Downarrow \mathbf{abc}$

(e) $((\mathbf{ab})\mathbf{c}) \Downarrow \mathbf{abc}$

(f) $((\mathbf{ab})|\mathbf{c})^* \Downarrow \epsilon$

(g) $((\mathbf{ab})|\mathbf{c})^* \Downarrow \mathbf{ab}$

(h) $((\mathbf{ab})|\mathbf{c})^* \Downarrow \mathbf{c}$

(i) $((\mathbf{ab})|\mathbf{c})^* \Downarrow \mathbf{cab}$

Fair Warning: Variations of Problems 1, 2, and 3 are likely to show up on quizzes. So you should practice answering such questions "by-hand".

Definition. $\#_c(w)$ = the number of times character c occurs in string w. Example: $\#_{\mathbf{a}}(\mathbf{abaabba}) = 4$ and $\#_{\mathbf{b}}(\mathbf{abaabba}) = 3$.

❖ ***Problem 2 (16 points)*** ❖

(a) Background. Let

$$L_1 = \{\, w \in \{\, \mathbf{a}, \mathbf{b} \,\}^* : (\#_{\mathbf{a}}(w) \bmod 3 = 0\,\}.$$

So, $w \in L_1 \iff$ the number of **a**'s in w is a multiple of 3 (and there can be any number of **b**'s). A regular expression for this language is:

[1] It is next to SciTech 4-226 and the CIS 252 and CIS 675 boxes.

$\mathbf{b}^*(\mathbf{ab}^*\mathbf{ab}^*\mathbf{ab}^*)^*$ and an NFA is $M_1 == (\{0,1,2\}, \mathit{Moves}_1, 0, \{2\})$ where

$$\mathit{Moves}_1 \;=\; \{0 \xrightarrow{\mathbf{b}} 0,\ 0 \xrightarrow{\mathbf{a}} 1,\ 1 \xrightarrow{\mathbf{b}} 1,\ 1 \xrightarrow{\mathbf{a}} 2,\ 2 \xrightarrow{\mathbf{b}} 2,\ 2 \xrightarrow{\mathbf{a}} 0\}$$

or see Figure 1 for the diagram form.

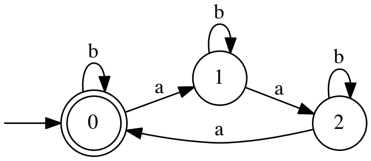


Figure 1: The diagram for M_1

YOUR PROBLEM: ***(4 points)*** Give an M_1-accepting path for **aabbaabaa**. (See pages 21 and 22 of the *Lexical* slides.)

(b) BACKGROUND. Let

$$L_2 = \{\, w \in \{\mathbf{a},\mathbf{b}\}^* \;:\; \#_{\mathbf{a}}(w) \geq 2 \;\text{ or }\; \#_{\mathbf{b}}(w) = 2 \,\}.$$

which has $(((\mathbf{a}|\mathbf{b})^*\mathbf{ab}^*\mathbf{a}(\mathbf{a}|\mathbf{b})^*)|(\mathbf{a}^*\mathbf{ba}^*\mathbf{ba}^*))$ as a regular expression. An NFA is $M_2 = (\{1,\ldots,6\}, \mathit{Moves}_2, 0, \{4,6\})$ where

$$\mathit{Moves}_2 \;=\; \left\{ \begin{array}{lll} 1 \xrightarrow{\mathbf{a}} 1, & 1 \xrightarrow{\mathbf{b}} 5, & 1 \xrightarrow{\epsilon} 2, \\ 2 \xrightarrow{\mathbf{a}} 2, & 2 \xrightarrow{\mathbf{a}} 3, & 2 \xrightarrow{\mathbf{b}} 2, \\ 3 \xrightarrow{\mathbf{a}} 4, & 3 \xrightarrow{\mathbf{b}} 3, & \\ 4 \xrightarrow{\mathbf{a}} 4, & 4 \xrightarrow{\mathbf{b}} 4, & \\ 5 \xrightarrow{\mathbf{a}} 5, & 5 \xrightarrow{\mathbf{b}} 6, & \\ 6 \xrightarrow{\mathbf{a}} 6 & & \end{array} \right\}$$

or see Figure 2 for the diagram form.

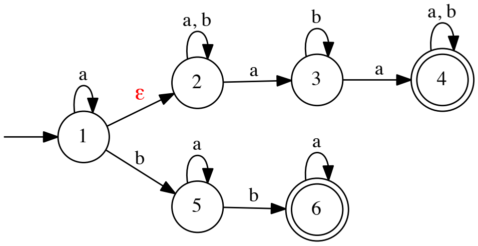


Figure 2: The diagram for M_2

YOUR PROBLEM: ***(12 points)*** Give ***four*** distinct M_2-accepting paths for **abaab**. (See pages 21 and 22 of the *Lexical* slides.)

❖ *Problem 3 (16 points)* ❖

For each of the following languages over $\{\mathbf{a},\mathbf{b}\}$, give both (i) a regular expression and (ii) a NFA that precisely captures it.[2]

(a) Those strings the contain **aaa** as a substring.

(b) Those strings the ***fail*** to contain **aaa** as a substring.

Part II: Programming Problems

You will need the files in `http://www.cis.syr.edu/courses/cis352/code/RegExp2/` and you will end up turning in changed versions of `Matches2.hs` and `BuildNFA2.hs`. This code is a modified version of Simon Thompson's regular expressions and automata library[3].

❖ *Problem 4 (16 points)* ❖

BACKGROUND. On page 13 of Mogensen[4] he defines the shorthands

$$r? \;=_{\text{def}}\; r|\epsilon \qquad\qquad r^+ \;=_{\text{def}}\; r(r^*)$$

A start at modifying Thompson's library to handle these two new forms can be found in:

[2] *Hint:* It is often easier to start with the NFA and then use the NFA to help figure out the regular expression.

[3] Simon Thompson. Regular expressions and automata using Haskell. Technical report, Computing Laboratory, University of Kent at Canterbury, 2000. URL `http://www.haskellcraft.com/craft3e/Reg_exps.html`

[4] Torben Ægidius Mogensen. *Introduction to Compiler Design*. Diku, 2010. URL `http://www.diku.dk/hjemmesider/ansatte/torbenm/Basics/`

`http://www.cis.syr.edu/courses/cis352/code/RegExp2/`

YOUR PROBLEMS.

(a) In `Matches2.hs` the function `matches` does not have cases for `Opt` or `Plus` expressions. Add the missing cases to `matches`.

Testing: Run `(quickCheck prop_equivA)`. Also come up with some convincing tests of your own.

(b) In `BuildNFA2.hs` the function `build` is missing cases for `Opt` or `Plus` expressions. Add the missing cases to `build`.

Testing: Run `(quickCheck prop_equivB)`. Also come up with some convincing tests of your own.

Obvious hint for both parts (a) and (b): The `Opt`-case should be a variation on the `Or`-case and the `Plus`-case should be a variation on the `Star`-case.

❖ *Problem 5 ((12 points) points)* ❖

Do Problem 2.16 in Mogensen and program your answer in Haskell using Simon Thompson's modules. Design and run some tests for your code. (*Hint:* You'll need a "|" in the equation for *nonempty(st)*.)

Testing: Make sure that `tstPos` and `tstNeg` (in `Top2.hs`) both evaluate to `True`. Also come up with some convincing tests of your own.

Reference rule-sets

Rules for a big-step rules for regular expression matching

$$\epsilon\text{:}\ \frac{}{\epsilon \Downarrow \epsilon} \qquad |_1\text{:}\ \frac{r_1 \Downarrow s}{(r_1|r_2) \Downarrow s} \qquad |_2\text{:}\ \frac{r_2 \Downarrow s}{(r_1|r_2) \Downarrow s}$$

$$\textit{Lit:}\ \frac{}{x \Downarrow x} \qquad \textit{Seq:}\ \frac{r_1 \Downarrow s_1 \quad r_2 \Downarrow s_2}{(r_1 r_2) \Downarrow s}\ (s = s_1 s_2)$$

$$*_1\text{:}\ \frac{}{r^* \Downarrow \epsilon} \qquad *_2\text{:}\ \frac{r \Downarrow s_1 \quad r^* \Downarrow s_2}{r^* \Downarrow s}\ (s = s_1 s_2)$$

Example. See page 9 of the *Lexical* slides for sample derivations.

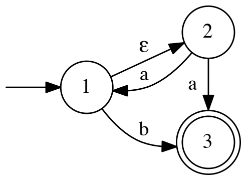
A small-step semantics for an NFA

For $M = (\textit{States}, \textit{Moves}, \textit{start}, \textit{Final})$:

$$\frac{}{M \vdash s \xrightarrow{a} s'}\ ((s, a, s') \in \textit{Moves})$$

$$\frac{}{M \vdash s \xrightarrow{\epsilon} s'}\ ((s, \epsilon, s') \in \textit{Moves})$$

Example. For the NFA with diagram:



an accepting path for input **aab** is:

$$1 \xrightarrow{a} 2 \xrightarrow{\epsilon} 1 \xrightarrow{a} 2 \xrightarrow{\epsilon} 1 \xrightarrow{b} 3$$