CSc 30400 Introduction to Theory of Computer Science

2nd Homework Set - Solutions

Due Date: 3/4

1 Practice Questions

P 1. $L(r_1) = \{01, 001, 0001, 00001, \ldots\}$
$L(r_2) = \{11, 01\}$
$L(r_3) = \{1, 101, 111\}$
$L(r_4) = \{01, 101, 1101, 11101, 111101, \ldots\}$

P 2. $r_1 = 10 + 110 + 111$
$r_2 = 110(0+1)^*$
$r_3 = 0^*(0^*10^*10^*)^*$
$r_4 = (0+1)^*01$

2 Easy Questions

E 1. $r_1 = (0+1)(0+1)0$
$r_2 = 010(1+0)^*$
$r_3 = (0+1)^*011(0+1)^*$
$r_4 = (0+1)^*111$
$r_5 = 1^+0(1+0)*$

E 2. The transition graphs are shown in figures 1 - 4.

E 3. (a) No. 0 can be produced by the first but not by the second.
(b) No. 0 can be produced by the first but not by the second.

(c) No. 0 can be produced by the second but not by the first.

(d) Yes. In fact, both regular expressions represent Σ^*. Any symbol 0 or 1 can be created either by 0+1 or by 0^*1^*. Thus, both expressions stared represent Σ^*. Another way to see the equivalence is by considering FA. Check the NFA_ε and the DFA shown in figures 5 and 6 respectively. They represent the two regular expressions and they are equivalent (I can transform the NFA_ε of figure 5 to the DFA of figure 6).

(e) Yes. The fact that any string that is produced by the second can also be produced by the first is obvious (since the first is, as we said, Σ^*). Now it suffices to show that any string in Σ^* can also be produced by the second expression. Indeed, for any string $s \in \Sigma^*$ then $s = s\varepsilon$, which can be produced by the second expression (the s from $(0+1)^*$ and the ε from the stared parenthesis).

E 4.
- **Integers:** $0 + (1+2+3)(0+1+2+3)^*$.
- **Identifiers:** $(a+b+c+d+e)(a+b+c+d+e+0+1+2+3)^*$.
- **Whitespaces:** $␣^+$.
- **Comments:** $/\#(z+/)^*\#(\#+z(z+/)^*\#)^*/$, where z is a space or an alphanumeric character, $z = ␣+a+b+c+d+e+0+1+2+3$. The DFA for the comments is shown in figure 7.

3 Hard Questions

H 1. The transformation is not complete. Consider the case where q_1 is the start state. By removing the ε-move we disconnect q_1 from q_2 and its neighbors.

The ideal appropriate change would be to make q_2 a second start state and being able to initiate the computation from either of q_1, q_2. However this is not possible since an NFA should have a unique start state by definition. We overcome this difficulty in the following way:

After removing the ε-move, we have to choose which of the two gets to be the unique start state. We don't have to change the start state unless q_1 is non-accepting and q_2 is accepting. In that case, the ε string is accepted by the NFA_ε, thus we should choose q_2 (an accepting state) to be the new start state of the NFA in order to maintain this fact.

We then add transitions from the new start state directly to the neighbors of the other.

- If we leave q_1 as the start state, for every transition $\delta(q_2, a) = q''$ we add the transition $\delta(q_1, a) = q''$.
- If we make q_2 the new start state, then for every transition $\delta(q_1, a) = q''$ we add the transition $\delta(q_2, a) = q''$.

H 2.

- L_1 is not regular. We prove it using the pumping lemma.

  For any $n > 0$ being the possible pumping length consider the string $s = 0^n1^n$. $s \in L_1$ and $|s| \geq n$. Now, in every possible splitting of s in x, y, z with $|xy| \leq n$ and $|y| \geq 1$, observe that $y = 0^m$ with $1 \leq m \leq n$. The string $xy^2z = 0^{m+n}1^n \notin L_1$. Thus the pumping lemma doesn't hold which means that L_1 is not regular.

- L_2 is regular. Actually, it turns out that L_2 is the same as the language $L = \{w :$ alternations between 0 and 1 in w occur an even number of times$\}$. Notice that an alternation between 0 and 1 occurs with the appearance of either a 01 or of an 10, thus L can be restated as the language of strings that contain an even number of appearances of 01 and 10 in total.

  For simplicity, call a_s and b_s the number of appearances of 01 and 10 in string s respectively. Thus $L_2 = \{s : a_s = b_s\}$ whereas $L = \{s : a_s + b_s = 2k$ for some $k \geq 0\}$.

  - $L_2 \subseteq L$ If s is in L_2 then $a_s = b_s$, thus $a_s + b_s = 2a_s$ which is an even number. Thus s is in L.
  - $L \subseteq L_2$ The first important observation we have to make here is that, in any string s with at least one appearance of 01 (10), in order for a second 01 (10) to occur, an 10 should first occur in between. This means that a_s and b_s should differ by at most 1, which in other words means that either $a_s - b_s = \pm 1$ or $a_s = b_s$.

    Consider some string s in L. Then $a_s + b_s = 2k$ for some $k \geq 0$. If $a_s - b_s = \pm 1$ then $2a_s = 2k \pm 1$ which cannot be the case (an even number cannot be equal with an odd number). Thus it should be the case that $a_s = b_s$ which proves that s is in L_2.

  That having been said, it suffices to find a DFA for L which would also be a DFA for L_2. The intuition in creating such an automaton is that you need to simulate the procedure described by the DFA of figure 8. We consider two cases, one where the string starts with 0 and one where it starts with 1. The complete DFA is shown in figure 9.

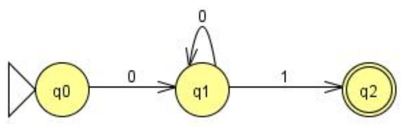

Figure 1: An NFA for r_1 of question E 2.

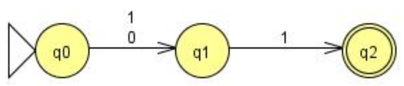

Figure 2: An NFA for r_2 of question E 2.

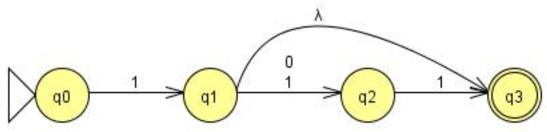

Figure 3: An NFA for r_3 of question E 2.

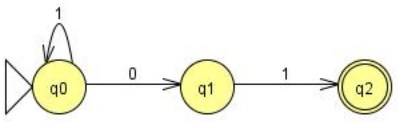

Figure 4: An NFA for r_4 of question E 2.

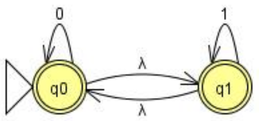

Figure 5: An NFA$_\varepsilon$ for $(0^*1^*)^*$ (question E 3).

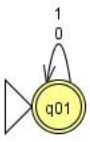

Figure 6: A DFA for $(0+1)^*$ (question E 3).

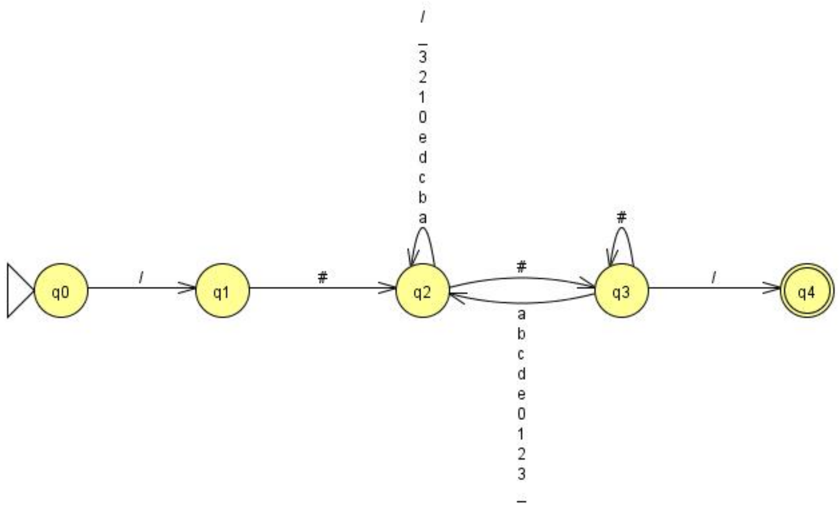

Figure 7: The DFA for the comments (question E 4).

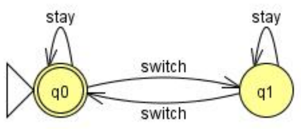

Figure 8: The intuition behind the construction of a DFA for L_2 (question H 2).

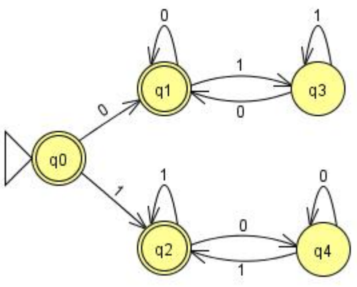

Figure 9: A DFA for L_2 (question H 2).