11-711: Algorithms for NLP

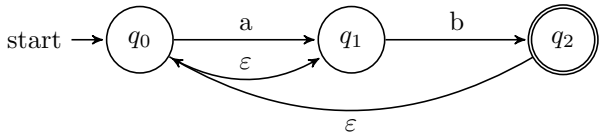
Homework Assignment #1: Formal Language Theory

Solutions

Out: September 15, 2015
Due: September 29, 2015

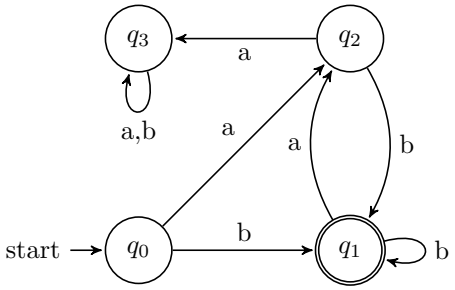
Problem 1 [15 points]

For the following NFA containing ε-moves,



give an equivalent DFA.

Solution

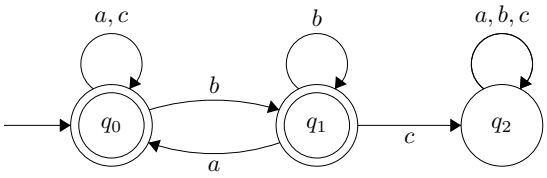


Problem 2 [15 points]

Note: The following problems require you to design finite state machines. We ask that you do not use any automated tools to solve these problems as you will not have access to such tools for similar problems on the exams.

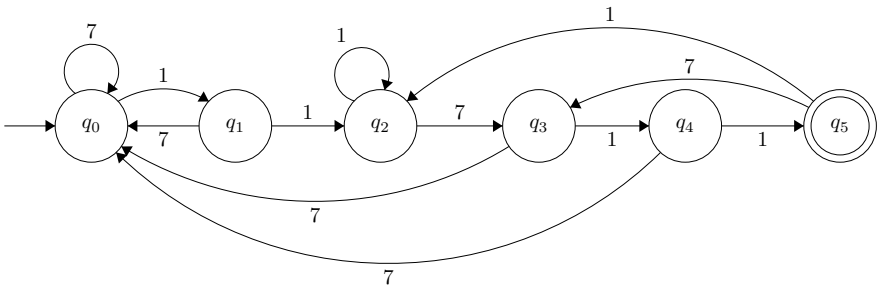
Give *deterministic* FSAs accepting the following languages. Please make sure your DFSAs are fully defined: clearly mark start and final states and do *not* use shorthand notation for arcs.

1. **[5 points]** The set of strings over $\{a, b, c\}$ in which the substring bc never occurs.



Computations in state q_1 have seen a prefix ending in b. Computations in the sink state q_2 have seen a prefix containing bc.

2. **[5 points]** The set of strings over $\{1, 7\}$ ending in 11711.

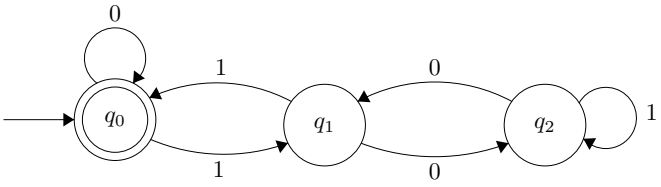


Computations in states q_1, q_2, q_3, q_4, and q_5 have seen a prefix ending in 1, 11, 117, 1171, and 11711, respectively.

3. **[5 points]** The set of strings over $\{0, 1\}$ which are divisible by three when interpreted as a binary number (ignoring leading zeroes).

   For example $00000_b = 0$, which is divisible by 3, so 00000 should be accepted. $001001_b = 9$, and thus should be accepted also. $101_b = 5$, is not divisible by 3, and thus should be rejected.

   Note: ϵ should be interpreted as 0, and thus should be accepted.



Computations in state q_i have seen a prefix x where $x_b \equiv i \bmod 3$. Note that appending 0 to the right of a string multiplies it by two: $(x0)_b = 2 \times x_b$. Appending 1 to a string multiplies by two and adds one: $(x1)_b = 2 \times x_b + 1$. The transitions are defined by:

$$\delta(q_i, 0) = q_{(2i \bmod 3)}$$

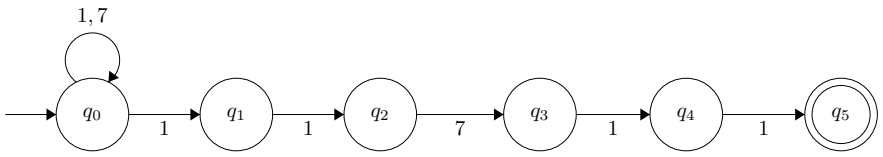
$$\delta(q_i, 1) = q_{(2i+1 \bmod 3)}$$

.

Problem 3 [10 points]

Give a *non-deterministic* FSA (possibly with ϵ moves) accepting the following language. Please clearly mark start and final states.

1. The set of strings over $\{1, 7\}$ ending in 11711. How does your NDFSA differ from the DFSA you constructed in Problem 2.2?



The NDFSA is simpler than the equivalent DFSA of Problem 2 because it can non-deterministically "guess" when the fifth symbol from the right end of the input string has been read.

Problem 4 [15 points]

Give regular expressions for each of the following languages:

1. **[7 points]** The set of strings over $\{a, b, c\}$ in which the substring bc never occurs.

   $(a + c + bb^*a)^*b^*$,

   which can be simplified to

   $(c + b^*a)^*b^*$

   Every string of b's must either end the string or be followed by an a.

2. **[8 points]** The language described in Problem 2.3: The set of strings over $\{0, 1\}$ which are divisible by three when interpreted as a binary number (ignoring leading zeroes).

   For example $00000_b = 0$, which is divisible by 3, so 00000 should be accepted. $001001_b = 9$, and thus should be accepted also. $101_b = 5$, is not divisible by 3, and thus should be rejected.

   Note: ϵ should be interpreted as 0, and thus should be accepted.

   $(0 + 1(01^*0)^*1)^*$

   Examine the FSA in Problem 2.3 and consider all of the sequences of moves beginning in q_0 that lead back to q_0. One possibility is to take the immediate loop from q_0 to q_0 emitting a 0. Another possible loop is to go to q_1 via the 1 arc, then back from q_1 to q_0 along the other 1 arc. While we're in q_1, however, we may take zero or more detours through q_2, via the path 01^*0.

   Thus, we may either follow the q_0 to q_0 loop labeled 0, or go to q_1 and back, possibly with detours. We may follow either of these two loops as many times as we want, in any order, intermingling them if we wish. This gives us a regex of the form $(\text{Loop}_1+\text{Loop}_2)^*$. Plugging in 0 for Loop_1 and $1(01^*0)1$ for Loop_2 gives the final regular expression.

Problem 5 [20 points]

We may define **generalized regular expressions** (GREs) as follows:

1. $\emptyset$ is a GRE denoting the empty language;
2. ε is a GRE denoting the language $\{\varepsilon\}$;
3. for each $\sigma \in \Sigma$, σ is a GRE denoting the language $\{\sigma\}$;
4. if α and β are GREs, denoting the languages A and B, respectively, then
   - $(\alpha \mid \beta)$ is a GRE denoting $A \cup B$;
   - $(\alpha\beta)$ is a GRE denoting $A.B$;
   - α^* is a GRE denoting A^*;
   - **[new]** $(\alpha \wedge \beta)$ is a GRE denoting $A \cap B$; and
   - **[new]** $\neg\alpha$ is a GRE denoting $\overline{A}$.

Prove that the languages denoted by GREs are regular.

Note: you may use refer to the definitions and proofs about REs we provided in class and only deal with the two new clauses in the generalized definition.

Solution

- **[new]** $\neg\alpha$ is a GRE denoting $\overline{A}$.
  We know every RE has an equivalent DFA and vice versa, so we can use a proof by construction. For RE α, there is some DFA $D = (Q, \Sigma, \delta, q_0, F)$ for the same language A.
  We can then construct $D' = (Q, \Sigma, \delta, q_0, Q - F)$
  Claim: $L(D') = \overline{A}$
  For any $x \in \overline{A}$, $x \in \Sigma^* - A$, so $\hat{\delta}(q_0, x) \notin F$ and therefore $\hat{\delta}(q_0, x) \in Q - F$ which means $x \in L(D')$, so $L(D') = \overline{A}$ and is thus the DFA equivalent of $\neg\alpha$, so $\neg\alpha$ is regular.
- **[new]** $(\alpha \wedge \beta)$ is a GRE denoting $A \cap B$
  By set theory, we know $\alpha \wedge \beta = \overline{\overline{\alpha}|\overline{\beta}}$. Since GREs are closed under union and complementation, they are closed under intersection.

Problem 6 [25 points]

For each of the following statements, answer whether the claim is **true** or **false**, and give a *short* (two- to three-sentence) explanation if true or counterexample if false.

1. **[5 points]** Let $L_1 \subset L_2$. If L_1 is not regular, then L_2 must also be not regular.

   False. As a counterexample, let $L_1 = \{0^n 1^n \mid n > 0\}$ and $L_2 = \Sigma^*$. L_1 is non-regular and L_2 is regular, yet $L_1 \subset L_2$.

2. **[5 points]** $L = L_1 \cap L_2$. If L_1 and L are regular languages, then L_2 must also be a regular language. **False.** As a counterexample, let $L_1 = 01$, which is regular, and $L_2 = \{0^n 1^n | n > 0\}$, which is not regular. $L = 01$ is regular.

3. **[5 points]** $L = L_1 \cup L_2$. If L_1 and L are regular languages, then L_2 must also be a regular language.

   False. As a counterexample, let $L_1 = \Sigma^*$, which is regular, and $L_2 = \{0^n 1^n \mid n > 0\}$, which is not regular. Yet $L = L_1 \cup L_2 = \Sigma^*$ is regular.

4. **[5 points]** $L = \bigcap_{i=1}^{\infty} L_i$. If all of the L_i are regular languages, then L is also a regular language.

   False. As a counterexample, let $L_i = \{0,1\}^* \setminus \{0^i 1^i\}$.

   Each language L_i is the complement of a set that consists of a single word, and is thus regular. However, $L = \{0,1\}^* \setminus \{0^i 1^i \mid i \geq 1\}$ is the complement of $\{0^i 1^i \mid i \geq 1\}$ which is not regular, thus L is not regular.

5. **[5 points]** Let α, β and γ be regular expressions.

   If $L(\beta + \alpha\gamma) \subseteq L(\gamma)$, then $L(\alpha^*\beta) \subseteq L(\gamma)$.

   True.

   We show by induction that $\forall n\, L(\alpha^n \beta) \subseteq L(\gamma)$.

   Base case ($n = 0$): $L(\alpha^0 \beta) = L(\beta) \subseteq L(\beta + \alpha\gamma) \subseteq L(\gamma)$.

   Induction step: Assume $L(\alpha^n \beta) \subseteq L(\gamma)$. Then

$$L(\alpha^{n+1}\beta) = L(\alpha)L(\alpha^n\beta)$$

$$\subseteq L(\alpha)L(\gamma) \text{ (by the inductive hypothesis)}$$

$$= L(\alpha\gamma) \subseteq L(\beta + \alpha\gamma) \subseteq L(\gamma).$$