Concordia University

Dept. of Computer Science and Software Engineering

COMP 335 – Introduction to Theoretical Computer Science

Fall 2018

Assignment 2 Solutions

1. **Solution.** Claim: If L is regular then $chop(L)$ is regular.

   Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA such that $L(M) = L$. Take two independent copies of M, which we will call

   $$M^1 = (Q^1, \Sigma, \delta^1, q_0^1, F^1), \text{ and}$$
   $$M^2 = (Q^2, \Sigma, \delta^2, q_0^2, F^2).$$

   We construct an NFA N such that $L(N) = chop(L(M)) = chop(L)$. We define N as follows $N = (Q^1 \cup Q^2, \Sigma, \delta', q_0^1, F^2)$, where δ' contains all the transitions of δ^1 and δ^2 plus some extra lambda transitions. More specifically, if there is a path from state q to state p in M we add a λ-transition between q^1 and p^2 in N. **NOTE:** this construction is done *for all* states q and p such that p is reachable from q in M. In particular, q is reachable from q (via the path of length 0) in M, thus we would add λ-transitions between q^1 and q^2.

   What is the intuition behind the above construction? You can visualize the two copies of M drawn one under another. You always start processing input strings at the top copy, and all the accept states are in the bottom copy. Lambda transitions only go from top to bottom, so once you jump down, there is no way to jump back up. All other parts of N, except for λ-transitions, are deterministic. In order for N to accept a string w, it has to start at the top copy of M and end up in an accepting state of the bottom copy of M. Thus, it has to process prefix of w, say x, and use one of theλ-transitions to move to the bottom and finish processing the remaining part of w, say z, i.e. $w = xz$. Suppose that after processing x, N ends up in state q^1, and after taking the lambda transition, it ends up in state p^2, then there is some path from q to p in N (by construction). This path corresponds to some string y such that processing xyz in M is equivalent to processing xz in N, that is the automaton ends up in the same state in the bottom copy of M in N as in the M itself. Check your intuition: what would $y = \lambda$ correspond to? Hint: consider the **NOTE** above.

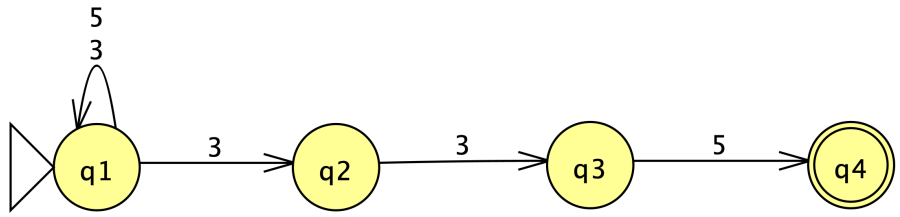
   Formally, we need to show that $L(N) = chop(L(M))$. With our informal description above in mind, this is straightforward.

   If $w \in chop(L(M))$ then we can write $w = xz$ such that xyz is accepted by M for some y. In N, there is an accepting path for w that consists of processing x on the top, ending up in state q^1, then taking λ-transition to state p^2 such that $p = \delta^*(q, y)$, and processing y from p^2, ending up in an accepting state $\delta^*(q_0, xyz)$. Thus, $w \in L(N)$.

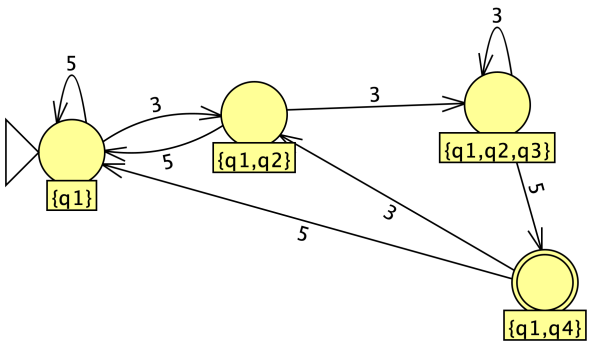
   If $w \in L(N)$, then as argued above, N has to start at the top, use λ-transition exactly once, and end up in an accepting state at the bottom. Let x be the part of w processed

at the top, let y be a string corresponding to the λ-transition, and let z be the part of w processed at the bottom. Then xyz is accepted by M.

2. (a) **Solution.** The following NFA skips over several characters in state q_1 until it non-deterministically guesses that there are only 3 characters left. At that point, it checks that the last three characters are 335 by transitioning to q_4.



(b) **Solution.** If you follow the NFA-to-DFA conversion procedure, you would end up with the following DFA.



3. **Solution.** Since L is regular there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts L, i.e., $L(M) = L$. We will create an NFA $N = (Q, \Sigma, \delta', q_0, F)$ such that $L(N) = fifth(L)$. In particular, the states of our NFA are exactly the same as those of the DFA, and the same holds for the set of accepting states and the start state. The only difference is in the transition function. Consider a pair of states q_1, q_2 and a character $\sigma \in \Sigma$. If there is a walk of length 5 in M where the last transition is labelled with σ, then we place a transition from q_1 to q_2 labelled σ in N. We repeat this process for all pairs of states q_1 and q_2 and for all characters $\sigma \in \Sigma$. This defines the transition function for our NFA.

We claim that there is a walk in our NFA $q_1, q_2, \ldots, q_k$ labelled $x = x_1, x_2, \ldots, x_{k-1}$ if and only if there is a string w such that $fifth(w) = x$ and our DFA M while processing w starting from state q_1 reaches state q_k. This can be proved by induction on k. The base case is when $k = 2$ – the statement holds precisely by our construction. Suppose the statement is true for some $k \geq 2$, we need to show that it holds for $k + 1$. Take a walk in the NFA of the form $q_1, \ldots, q_{k+1}$ labelled $x = x_1, x_2, \ldots, x_k$. By induction applied to the first k states visited by the walk, there is a string w such that $fifth(w) = x_1, x_2, \ldots, x_{k-1}$ and the DFA M goes from q_1 to q_k while processing w. Now, by the definition of δ', there is a walk of length 5 in M from q_k to q_{k+1}

where the last label is x_k. Let y_1, y_2, y_3, y_4, y_5 be the labels encountered during that walk in M, where $y_5 = x_k$. Then $w' = wy_1y_2y_3y_4y_5$ has the property $fifth(w') = x_1, x_2, \ldots, x_{k-1}, y_5 = x_1, x_2, \ldots, x_{k-1}, x_k$ and M goes from q_1 to q_{k+1} while processing w'. This proves one part of the "if and only if" statement in the inductive step. To see the other direction, let w be a string such that $fifth(w) = x$ and our DFA M while processing w starting from state q_1 reaches state q_{k+1}. By induction assumption applied to w without its last 5 characters, we can conclude that there is a walk p in our NFA from q_1 to q_k labelled $x_1 \ldots, x_{k-1}$. The last 5 characters of w take DFA from q_k to q_{k+1} and the last transition is labelled x_k. By the definition of δ' this means that there is a transition from q_k to q_{k+1}in N labelled x_k. Combining this with p means that there is a walk from q_1 to q_{k+1} in N labelled $x_1, \ldots, x_k$. This proves the inductive step.

Now, observe that $w \in L$ is such that it takes DFA from q_0 to some $q \in F$ if and only if $fifth(w)$ takes our NFA from q_0 to $q \in F$. This means that $L(N) = fifth(w)$.

4. (a) **Solution.** This language is regular. This language consists of all strings over $\Sigma = \{a\}$ of length divisible by 2018. Consider the following DFA $M = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{q_0, q_1, \ldots, q_{2017}\}$, $\delta(q_i, a) = q_{i+1}$ for $i \in \{0, \ldots, 2016\}$ and $\delta(q_{2017}, a) = q_0$, and $F = \{q_0\}$. You can visualize this DFA as a string of 2017 transitions that wraps around at the end for the total of 2018 transitions to make a single complete cycle. The accepting state is q_0 so that we begin by accepting an empty string and every multiple of 2018 transitions is also accepted, but nothing else. Thus, it is easy to see that $L(M) = L_1$.

   (b) **Solution.** This language is not regular. Assume for contradiction that L_2 is regular and apply the Pumping Lemma. Let m be the integer given by the pumping lemma and choose $w = a^{2018^m}$. Since $w \in L$ and $|w| \geq m$ we can write $w = xyz$ such that $|xy| \leq m$ and $|y| \geq 1$ and $xy^iz \in L$ for all $i = 0, 1, 2, \ldots$ Let $|y| = k$, then $xy^2z \in L$ where $xy^2z = a^{2018^m+k}$. Since strings in L are of lengths that are powers of 2018 and xy^2z is in L, then its length must be a power of 2018. Is this possible? Write $2018^m + k \leq 2018^m + m < 2018^m + 2018^m < 2017 \cdot 2018^m + 2018^m = 2018^{m+1}$. Also, $2018^m + k > 2018^m$. Therefore, the pumped string is of length that is strictly in between two powers of 2018 and therefore cannot be in L. Contradiction.

5. (a) **Solution.** Assume for contradiction that L is regular and apply the Pumping Lemma. Let m be the integer in the Pumping Lemma. Take $w = a^m b^1 c^m$. Since $w \in L$ and $|w| \geq m$ we can write $w = xyz$ such that $|xy| \leq m$ and $|y| \geq 1$ and any string of the form $xy^iz \in L$ for $i = 0, 1, 2, \ldots$ This implies that $y = a^k$ for some $k \in \{1, \ldots, m\}$. Taking $i = 0$ we get that $xz = a^{m-k}b^1c^m \in L$; however, this contradicts the definition of L, since $(m - k) \cdot 1 \neq m$. Contradiction.

   (b) **Solution.**

   (b.1) *Base case.* $n = 1$: left hand side is 2 and right hand side is 2, since $2 \geq 2$, base case holds.

   Induction assumption. Assume for some $n \geq 1$ we have $2^n \geq 2n$.

   Inductive step. We have $2^{n+1} = 2 \cdot 2^n \geq 2 \cdot 2n = 4n = 2n + 2n \geq 2n + 2 = 2(n + 1)$, where the first inequality is given by induction assumption, and the

second inequality follows from $n \geq 1$.

For the observation, note that $f(m+1) = 2^{f(m)} \geq 2f(m)$, where the inequality follows from the statement just proved by induction.

(b.2)

Base case(s). $n = 0$: left hand side is 1 and right hand side is 0. Base case $n = 0$ holds, since $1 > 0$.

$n = 1$: left hand side is 2 and right hand side is 1. Base case $n = 1$ holds, since $2 > 1$.

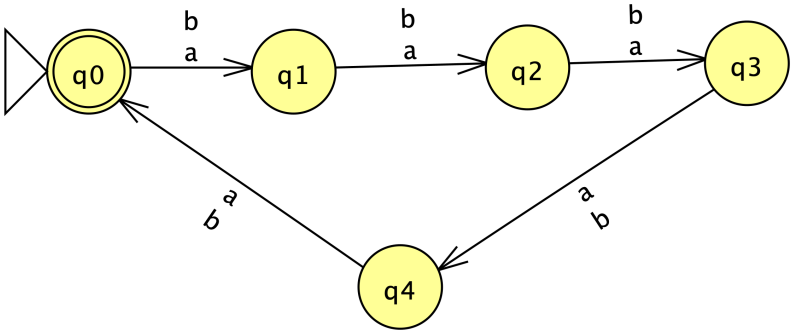
Induction assumption. Assume for some $n \geq 1$ that $f(n) > n$.

Inductive step. Consider $f(n+1) = 2^{f(n)} > 2^n \geq 2n = n + n \geq n + 1$, where the first inequality holds by induction assumption and the fact that function $f(x) = 2^x$ is monotone, whereas the second inequality holds since $n \geq 1$.

(b.3) Assume for contradiction that L is regular and apply the Pumping Lemma. Let m be the integer given by the Pumping Lemma, and choose $w = a^{f(m)}$. Since $w \in L$ and $|w| \geq m$ we can write $w = xyz$ with $|xy| \leq m$ and $|y| \geq 1$ and $xy^iz \in L$ for all $i = 0, 1, 2, \ldots$ Write $y = a^k$ for some $k \in \{1, \ldots, m\}$. Then $xy^2z = a^{f(m)+k} \in L$. We have $f(m)+k > f(m)$ and if we show that $f(m)+k < f(m+1)$ then we would arrive at a contradiction, since xy^2z would not be of the form $a^{f(n)}$, and thus cannot be in L. Observe that $f(m+1) = 2^{f(m)} \geq 2f(m) > f(m)+k$. This concludes the proof. Notice that we used several inequalities here, namely, $f(n) > n$ and that $2^n \geq 2n$, which are given by previous parts.

6. (10 Points)

(a) **Solution.** Note that each string consists of only a's and b's. Thus L_3 can be restated as the languages of all strings of length divisible by 5. It is easy to construct a DFA for this language (this is similar to problem 4a).



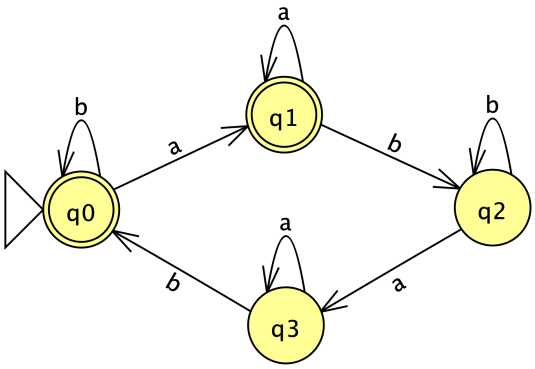
We can use the conversion algorithm to convert it into a right-linear grammar:

$$q_0 \to aq_1|bq_1|\lambda$$
$$q_1 \to aq_2|bq_2$$
$$q_2 \to aq_3|bq_3$$
$$q_3 \to aq_4|bq_4$$
$$q_4 \to aq_0|bq_0$$

Here the variables are q_0, q_1, q_2, q_3, q_4 with q_0 being the start variable, while terminals are a, b.

(b)

Solution. We follow a similar strategy to the previous part: design a DFA and convert it. The following DFA accepts all strings with even number of substrings ab:



It is easy to see that for the above DFA M we have $L(M) = L_4$. Suppose that the automaton finishes processing string x and finds itself in one of the states. Then the following invariants hold:

If automaton is in q_0 then x has an even number of occurrences of ab and the last character was b (unless x is an empty string).

If automaton is in q_1 then x has an even number of occurrences of ab and the last character was a.

If automaton is in q_2 then x has an odd number of occurrences of ab and the last character was b.

If automaton is in q_3 then x has an odd number of occurrences of ab and the last character was a.

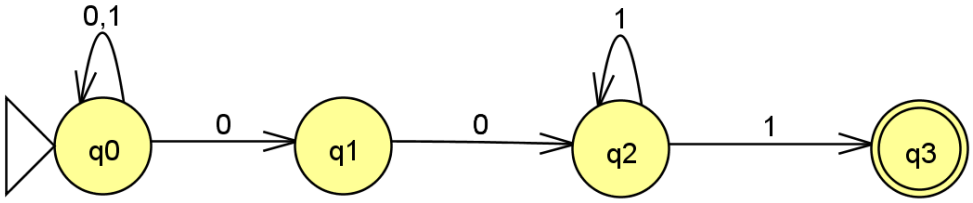
The above claim can be proved by a simple induction on the length of x. From this, the fact that $L(M) = L_4$ follows.

Now, we apply the conversion procedure and obtain the following regular grammar:

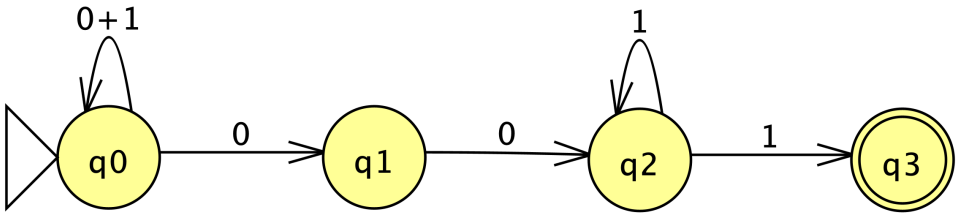
$$q_0 \to aq_1|bq_0|\lambda$$
$$q_1 \to aq_1|bq_2|\lambda$$
$$q_2 \to bq_2|aq_3$$
$$q_3 \to aq_3|bq_0$$

Here, the variables are q_0, q_1, q_2, q_3 and the terminals are a, b.

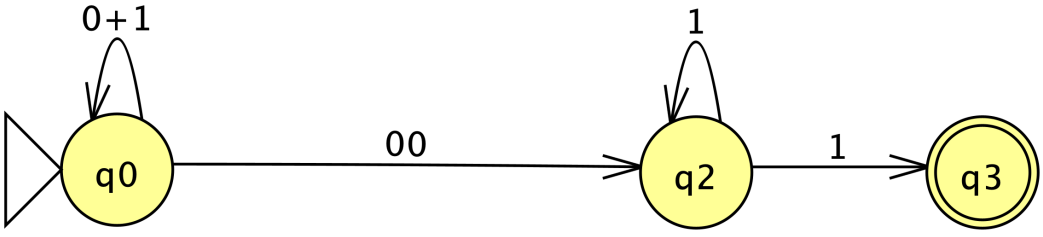
7. (a)



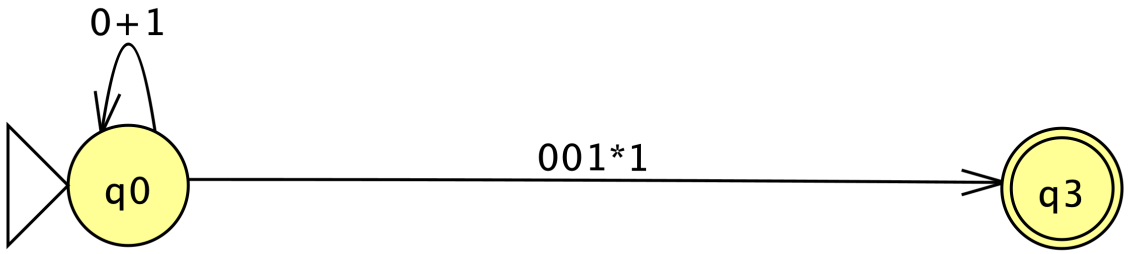
Solution. Start by converting the NFA to GTG:



Then eliminate state $q1$:

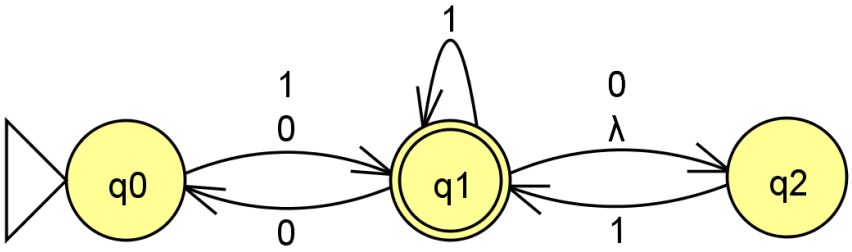


Then eliminate state $q2$:

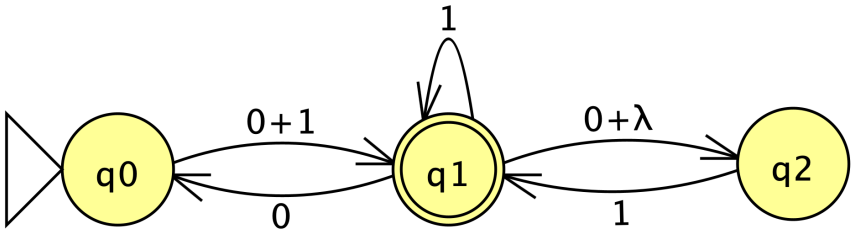


Then read off the answer: $(0+1)^*001^*1$.

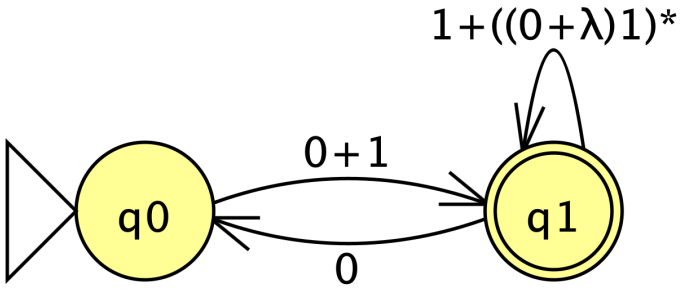
(b)



Solution. Start by converting the NFA into GTG:



Then eliminate state $q2$:



Then read off the answer: $(0+1)(0(0+1)+1+((0+\lambda)1)^*)^*$. This solution would give you full points, but notice that you can simplify the regular expression further: $(0+1)(00+01+1+(0+\lambda)1)^* = (0+1)(00+01+1+01+1)^* = (0+1)(00+01+1)^*$.