CS 332: *Theory of Computation* | Professor Sofya Raskhodnikova
Boston University | September 13, 2019

# Homework 2 – Due Friday, September 20, 2019 before noon

**Reminder** Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to the course staff if asked. You must also identify your collaborators. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.

**Exercises** Please practice on exercises and solved problems in Chapter 1 and on the exercise below. The material they cover may appear on exams.

1. (**Conversion procedures**) Use asymptotic (big-$O$) notation to answer the following questions. Provide brief explanations.

   (a) Let $N$ be an NFA that has $n$ states. If we convert $N$ to an equivalent DFA $M$ using the procedure we described, how many states would $M$ have?

   (b) Let $R$ be a regular expression that has $n$ symbols (each constant/operation counts as one symbol). If we convert $R$ to an equivalent NFA $N$ using the procedure described in class, how many states would $N$ have in the worst case?

   (c) Let $M$ be a DFA that has $n$ states. If we convert $M$ to an equivalent regular expression $R$ using the procedure we descried, how many symbols would $R$ have in the worst case?

   In an *extended* regular expression, we may use the complement operation ($\neg$) in addition to the three regular operations ($\cup, \circ, \star$). For example,

   $$\neg(\Sigma^* 001 \Sigma^*) \cup \neg(\Sigma^* 100 \Sigma^*)$$

   is an extended regular expression that describes the collection of all strings that either do not contain the substring 001 or do not contain the substring 100.

   (d) Describe how to modify the conversion procedure from regular expressions to NFAs so that it becomes a conversion procedure from extended regular expressions to NFAs.

   (e) Let $R$ be an *extended* regular expression that has $n$ symbols (each constant/operation counts as one symbol). If we convert $R$ to an equivalent NFA $N$ using the procedure you described above, how many states would $N$ have in the worst case?

**Problems** *This homework contains 1 automatically graded problem. Your solution to all other problems should be submitted on Gradescope.* There are 3 mandatory problems, worth 10 points each.

1. (**DFAs, NFAs, regular expressions and converting between them**) Your solution to this problem will be automatically graded by `automatatutor.com`. (We will rescale the points assigned by Automata Tutor, so that each part is worth 2 points.)

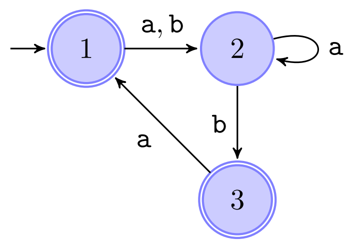
   (a) (**Description to NFA**) Let $\Sigma$ contain all letters and punctuation marks used in English. Give an NFA with 4 states recognizing the language $\{w \in \Sigma^* \mid w \text{ ends in } \texttt{LOL} \text{ or in } \texttt{LL}\}$.

(b) (**NFA to DFA**) Convert your NFA from part (a) to an equivalent DFA. Give only the portion of the DFA that is reachable from the start state.

(c) (**Rex to NFA**) Use the procedure described in class (also in Sipser, Lemma 1.55) to convert $\varepsilon\emptyset^*(\mathtt{ab} \cup \mathtt{bc})$ to an equivalent NFA.

(d) (**Description to rex**) Sipser, 1.22(b).

(e) (**NFA to rex**) Use the procedure described in Lemma 1.60 to convert the following finite automaton to a regular expression.




2. **(Number of states)** (**10 points**) Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. For each $k \geq 1$, let $C_k$ be the language consisting of all strings that contain an $\mathtt{a}$ exactly $k$ places from the right-hand end. Thus $C_k = \Sigma^*\mathtt{a}\Sigma^{k-1}$.

   (a) Describe an NFA with $k+1$ states that recognizes $C_k$ in terms of both a state diagram and a formal description.

   (b) Prove that for each $k$, no DFA can recognize $C_k$ with fewer than $2^k$ states.

   Hint: If a DFA enters different states after reading two different input strings $xz$ and $yz$ with the same suffix $z$ then the DFA must enter different states after reading input strings $x$ and $y$. (Explain why.) Find $2^k$ strings on which every DFA recognizing $C_k$ must enter different states. (Start by finding two such strings. Review Slide 9 for Lecture 3.)

3. (**Non-regular languages**) Use the pumping lemma to prove that the following languages are not regular.

   (a) (**3 points**) $L_1 = \{www \mid w \in \{0, 1\}^*\}$.

   (b) (**3 points**) $L_2 = \{y = \mathtt{10} \times x \mid x$ and $y$ are binary integers with no leading $\mathtt{0}$s, and $y$ is two times $x\}$. (The alphabet for this languages is $\{\mathtt{0}, \mathtt{1}, \times, =\}$.) For example, $\mathtt{1010} = \mathtt{10} \times \mathtt{101}$ is in $L_2$, but $\mathtt{1010} = \mathtt{10} \times \mathtt{1}$ is not.

   (c) (**4 points**) Let $\Sigma_2 = \left\{\begin{bmatrix}\mathtt{0}\\\mathtt{0}\end{bmatrix}, \begin{bmatrix}\mathtt{0}\\\mathtt{1}\end{bmatrix}, \begin{bmatrix}\mathtt{1}\\\mathtt{0}\end{bmatrix}, \begin{bmatrix}\mathtt{1}\\\mathtt{1}\end{bmatrix}\right\}$. Consider each row to be a binary number and let $L_3 = \left\{w \in \Sigma_2^* \mid \text{ the bottom row of } w \text{ is the square of the top row of } w\right\}$. For example, $\begin{bmatrix}\mathtt{0}\\\mathtt{1}\end{bmatrix}\begin{bmatrix}\mathtt{0}\\\mathtt{0}\end{bmatrix}\begin{bmatrix}\mathtt{1}\\\mathtt{0}\end{bmatrix}\begin{bmatrix}\mathtt{0}\\\mathtt{0}\end{bmatrix}\begin{bmatrix}\mathtt{0}\\\mathtt{0}\end{bmatrix} \in L_3$, but $\begin{bmatrix}\mathtt{0}\\\mathtt{1}\end{bmatrix}\begin{bmatrix}\mathtt{0}\\\mathtt{0}\end{bmatrix}\begin{bmatrix}\mathtt{0}\\\mathtt{0}\end{bmatrix}\begin{bmatrix}\mathtt{1}\\\mathtt{0}\end{bmatrix}\begin{bmatrix}\mathtt{0}\\\mathtt{0}\end{bmatrix}\begin{bmatrix}\mathtt{0}\\\mathtt{0}\end{bmatrix} \notin L_3$.