St. Francis Xavier University
Department of Computer Science

CSCI 356/541: Theory of Computing
Assignment 1
Due September 25, 2026 at 11:30am

Assignment Regulations.

- This assignment must be completed individually.
- Please include your full name and email address on your submission.
- You may either handwrite or typeset your submission. If your submission is handwritten, please ensure that the handwriting is neat and legible.

[6 marks] 1. As we saw in lecture, `grep` is a powerful command-line tool for using regular expressions to match patterns in text files. Here, you will use `grep` to find words in a dictionary file that match certain patterns.

Note. If your computer uses MacOS or Linux, you should already have a dictionary file located at `/usr/share/dict/words` or a similar directory. If your computer uses Windows, you should open the Windows Subshell for Linux and download a dictionary file to your current directory by running the following command:
`curl -o [FILENAME] https://raw.githubusercontent.com/eneko/data-repository/master/data/words.txt`

(a) Run the command `grep tion$ /usr/share/dict/words`. What pattern is being matched, and what are three examples of words matching this pattern?

(b) Run the command `grep ^.[aeiou].[rst]$ /usr/share/dict/words`. What pattern is being matched, and what are three examples of words matching this pattern?

(c) Run the command `grep .*[aeiou].*[aeiou].*[aeiou].*[aeiou].* /usr/share/dict/words`. What pattern is being matched, and what are three examples of words matching this pattern?

(d) Write a regex that outputs all words containing the letters `a`, `i`, and `u` in that order, not necessarily consecutively. Give an example of a word having this property.

(e) Write a regex that outputs all words beginning with `con` and also containing `pro` (not necessarily consecutively). Give an example of a word having this property.

(f) Write a regex that outputs all words containing three consecutive pairs of identical letters. Give an example of a word having this property.

(This one is tricky: remember, *regex* can do backtracking, but *regular expressions* cannot.)

[6 marks] 2. (a) For each of the following regular expressions, give a 1–2 sentence description of the language matched by that regular expression.

i. $\boldsymbol{r}_1 = (\epsilon + \mathtt{0})(\mathtt{10})^*(\epsilon + \mathtt{1})$.

ii. $\boldsymbol{r}_2 = (\mathtt{0} + \mathtt{1})^*\emptyset + \mathtt{1}(\mathtt{0} + \mathtt{1})^*$.

(b) For each of the following languages over the alphabet $\Sigma = \{\mathtt{0}, \mathtt{1}\}$, give a regular expression matching exactly that language.

i. $L_1 = \{w \mid$ every run of `1`s in w has even length$\}$. (A *run* is a consecutive sequence of `1`s.)

ii. $L_2 = \{w \mid w$ contains exactly two `1`s and at least one `0`$\}$.

[8 marks] 3. (a) Let $\Sigma = \{\texttt{0}, \texttt{1}\}$. For each of the following languages, give a *deterministic* finite automaton recognizing exactly that language.

i. $L_1 = \{w \mid w \text{ does not contain } \texttt{11}\}$.

ii. $L_2 = \{w \mid \text{the number of } \texttt{0}\text{s in } w \text{ is congruent to 2 mod 4}\}$.

(b) Let $\Sigma = \{\texttt{0}, \texttt{1}\}$. Give a *nondeterministic* finite automaton recognizing the language of binary representations of natural numbers that are divisible by eight. (For example, your nondeterministic finite automaton should accept input words like `0`, `11000`, and `101000`; but reject input words like `100`, `1111`, and `10101`.) You may assume that the input word contains no leading zeroes, with the exception of the specific input word `0`.

4. Let $\Sigma = \{\binom{0}{0}, \binom{0}{1}, \binom{1}{0}, \binom{1}{1}\}$. Consider the language

$$L = \{w = \binom{a_1}{b_1}\binom{a_2}{b_2}\cdots\binom{a_n}{b_n} \mid b_1 b_2 \ldots b_n \text{ is the result of shifting } a_1 a_2 \ldots a_n \text{ rightward by one bit}\}.$$

Here, $n \geq 1$ and for all $1 \leq i \leq n$, $a_i, b_i \in \{\texttt{0}, \texttt{1}\}$.

One can observe that, for example, $\binom{1}{0}\binom{1}{1}\binom{1}{1} \in L$ and $\binom{0}{0}\binom{1}{0}\binom{0}{1} \in L$, but $\binom{1}{1}\binom{0}{1}\binom{0}{0} \notin L$. You may assume that the leftmost bit after applying the right-shift operation is always `0`. (Note that $\epsilon \in L$ as well, since right-shifting the empty word just produces the empty word.)

[5 marks] (a) **Both CSCI 356 and CSCI 541 students:**

Construct a deterministic finite automaton recognizing the language L.

[5 marks] (b) **Only CSCI 541 students:**

Prove that the language L *cannot* be recognized by any deterministic finite automaton with fewer than three states.

[10 marks] 5. **Only CSCI 541 students:**

Given two words $w, v \in \Sigma^*$, the *shuffle* of u and v is any word obtained by interleaving the symbols of u and v while preserving the order of symbols within each word. For example, `chair` is one shuffle of the words `car` and `hi`, but `crhia` is not a shuffle of these words.

For two languages $L_1, L_2 \subseteq \Sigma^*$, define

$$L_1 \text{ Ш } L_2 = \{w \mid w \text{ is a shuffle of some words } u \in L_1 \text{ and } v \in L_2\}.$$

(The symbol Ш is the Cyrillic letter sha, pronounced "sh"—as in "shuffle"!)

Suppose $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic finite automata recognizing L_1 and L_2, respectively. Construct a nondeterministic finite automaton $\mathcal{B}$ recognizing the language L_1 Ш L_2.

[10 marks] 6. **Only CSCI 541 students:**

Investigate another model of finite automaton beyond deterministic and nondeterministic: for example, alternating finite automata or two-way finite automata. Formally define your chosen model, explain how your chosen model differs from the models we saw in lecture, describe connections between your chosen model and the traditional deterministic/nondeterministic finite automaton models, and list some applications of your chosen model.

Find and cite at least three non-internet sources to support your answer.