HW 2: Extra problems

Instructors: Har-Peled

CS/ECE 374A: Intro. Algorithms & Models of Computation, Fall 2024 Version: **1.0**

Solved problem

1 ***C comments*** are the set of strings over alphabet $\Sigma = \{*, /, A, \square, \langle\langle \texttt{Enter} \rangle\rangle\}$ that form a proper comment in the C program language and its descendants, like C++ and Java. Here $\langle\langle\texttt{Enter}\rangle\rangle$ represents the newline character, $\square$ represents any other whitespace character (like the space and tab characters), and A represents any non-whitespace character other than $*$ or $/$.[1] There are two types of C comments:

- Line comments: Strings of the form $// \cdots \langle\langle\texttt{Enter}\rangle\rangle$.
- Block comments: Strings of the form $/* \cdots */$.

Following the C99 standard, we explicitly disallow ***nesting*** comments of the same type. A line comment starts with $//$ and ends at the first $\langle\langle\texttt{Enter}\rangle\rangle$ after the opening $//$. A block comment starts with $/*$ and ends at the first $*/$ completely after the opening $/*$; in particular, every block comment has at least two $*$s. For example, each of the following strings is a valid C comment:

- $/***/$
- $//\square//\square\langle\langle\texttt{Enter}\rangle\rangle$
- $/*///\square*\square\langle\langle\texttt{Enter}\rangle\rangle**/$
- $/*\square//\square\langle\langle\texttt{Enter}\rangle\rangle\square*/$

On the other hand, *none* of the following strings is a valid C comments:

- $/*/$
- $//\square//\square\langle\langle\texttt{Enter}\rangle\rangle\square\langle\langle\texttt{Enter}\rangle\rangle$
- $/*\square/*\square*/\square*/$

1.A. Describe a DFA that accepts the set of all C comments.

1.B. Describe a DFA that accepts the set of all strings composed entirely of blanks ($\square$), newlines ($\langle\langle\texttt{Enter}\rangle\rangle$), and C comments.

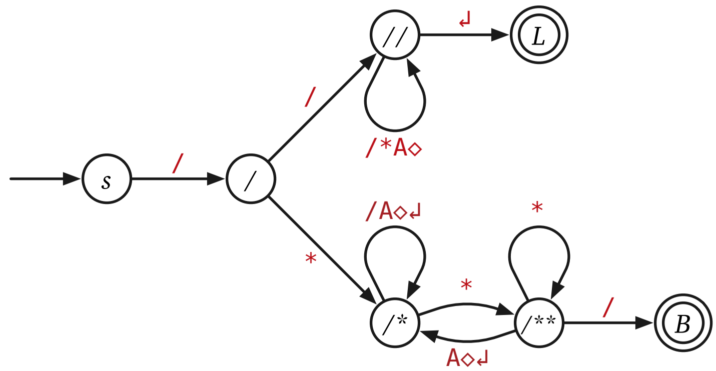
You must explain *in English* how your DFAs work. Drawings or formal descriptions without English explanations will receive no credit, even if they are correct.

[1]The actual C commenting syntax is considerably more complex than described here, because of character and string literals.

- The opening $/*$ or $//$ of a comment must not be inside a string literal ($"\cdots"$) or a (multi-)character literal ($'\cdots'$).
- The opening double-quote of a string literal must not be inside a character literal ($'"'$) or a comment.
- The closing double-quote of a string literal must not be escaped ($\backslash"$)
- The opening single-quote of a character literal must not be inside a string literal ($"\cdots'\cdots"$) or a comment.
- The closing single-quote of a character literal must not be escaped ($\backslash'$)

Solution:

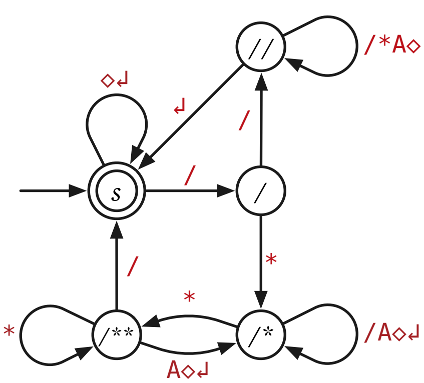
1.A. The following eight-state DFA recognizes the language of C comments. All missing transitions lead to a hidden reject state.



The states are labeled mnemonically as follows:

- s - We have not read anything.
- / - We just read the initial /.
- // - We are reading a line comment.
- L - We have read a complete line comment.
- /* - We are reading a block comment, and we did not just read a ∗ after the opening /∗.
- /** - We are reading a block comment, and we just read a ∗ after the opening /∗.
- B - We have read a complete block comment.

1.B. By merging the accepting states of the previous DFA with the start state and adding white-space transitions at the start state, we obtain the following six-state DFA. Again, all missing transitions lead to a hidden reject state.



- A backslash escapes the next symbol if and only if it is not itself escaped (\\) or inside a comment.

For example, the string "/∗\\"∗/"/∗"/∗\"/∗"∗/ is a valid string literal (representing the 5-character string /∗\"\∗/, which is itself a valid block comment!) followed immediately by a valid block comment. ***For this homework question, just pretend that the characters ', ", and \ don't exist.***

Commenting in C++ is even more complicated, thanks to the addition of *raw* string literals. Don't ask.

Some C and C++ compilers do support nested block comments, in violation of the language specification. A few other languages, like OCaml, explicitly allow nesting block comments.

The states are labeled mnemonically as follows:

- s - We are between comments.
- / - We just read the initial / of a comment.
- // - We are reading a line comment.
- /* - We are reading a block comment, and we did not just read a * after the opening /*.
- /** - We are reading a block comment, and we just read a * after the opening /*.

Rubric: 10 points = 5 for each part, using the standard DFA design rubric (scaled)

Rubric:[DFA design] For problems worth 10 points:

- 2 points for an unambiguous description of a DFA, including the states set Q, the start state s, the accepting states A, and the transition function δ.
  - **For drawings:** Use an arrow from nowhere to indicate s, and doubled circles to indicate accepting states A. If $A = \varnothing$, say so explicitly. If your drawing omits a reject state, say so explicitly. **Draw neatly!** If we can't read your solution, we can't give you credit for it,.
  - **For text descriptions:** You can describe the transition function either using a 2d array, using mathematical notation, or using an algorithm.
  - **For product constructions:** You must give a complete description of the states and transition functions of the DFAs you are combining (as either drawings or text), together with the accepting states of the product DFA.
- **Homework only:** 4 points for *briefly* and correctly explaining the purpose of each state *in English*. This is how you justify that your DFA is correct.
  - For product constructions, explaining the states in the factor DFAs is enough.
  - **Deadly Sin:** ("Declare your variables.") No credit for the problem if the English description is missing, *even if the DFA is correct.*
- 4 points for correctness. (8 points on exams, with all penalties doubled)
  - -1 for a single mistake: a single misdirected transition, a single missing or extra accept state, rejecting exactly one string that should be accepted, or accepting exactly one string that should be accepted.
  - -2 for incorrectly accepting/rejecting more than one but a finite number of strings.
  - -4 for incorrectly accepting/rejecting an infinite number of strings.
- DFA drawings with too many states may be penalized. DFA drawings with *significantly* too many states may get no credit at all.
- Half credit for describing an NFA when the problem asks for a DFA.

3 Questions

2 (100 PTS.) Regularize this [Spring, 2019].

For each of the following languages over the alphabet $\{0, 1\}$, give a regular expression that describes that language, and briefly argue why your expression is correct.

2.A. (20 PTS.) All strings that contain the subsequence 101.

2.B. (20 PTS.) All strings that do not contain the subsequence 111.

2.C. (20 PTS.) All strings that start in 11 and contain 110 as a substring.

2.D. (20 PTS.) All strings that do not contain the substring 100.

2.E. (20 PTS.) All strings in which every nonempty maximal substring of consecutive 0s is of length 1. For instance 1001 is not in the language while 10111 is.

3 (100 PTS.) Then, shalt thou find two runs of three [Spring, 2019].

Let L be the set of all strings in $\{0, 1\}^*$ that contain the substrings 000 and 111.

3.A. (60 PTS.) Describe a DFA that over the alphabet $\Sigma = \{0, 1\}$ that accepts the language L. Argue that your machine accepts every string in L and nothing else, by explaining what each state in your DFA *means*.

You may either draw the DFA or describe it formally, but the states Q, the start state s, the accepting states A, and the transition function δ must be clearly specified.

3.B. (40 PTS.) Give a regular expression for L, and briefly argue why the expression is correct.

4 (100 PTS.) Construct This [Spring, 2019]

Let L_1 and L_2 be regular languages over Σ accepted by DFAs $M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$, respectively.

4.A. (30 PTS.)

Describe a DFA $M = (Q, \Sigma, \delta, s, A)$ in terms of M_1 and M_2 that accepts $L = L_1 \cup \overline{L_2} \cup \{\epsilon\}$. Formally specify the components Q, δ, s, and A for M in terms of the components of M_1 and M_2.

4.B. (30 PTS.)

Let $H_1 \subseteq Q_1$ be the set of states q such that there exists a string $w \in \Sigma^*$ where $\delta_1^*(q, w) \in A_1$. Consider the DFA $M' = (Q_1, \Sigma, \delta_1, s_1, H_1)$. What is the language $L(M')$? Formally prove your answer!

4.C. (40 PTS.) Suppose that for every $q \in A_2$ and $a \in \Sigma$, we have $\delta_2(q, a) = q$. Prove that $\epsilon \in L_2$ if and only if $L_2 = \Sigma^*$.

5 (100 PTS.) Spring 2020 Q2.1

For each of the following languages over the alphabet $\{0, 1\}$, give a regular expression that describes that language, and briefly argue why your expression is correct.

5.A. All strings that contain 10110110 or 1101 as a substring.

5.B. All strings that begin with 110 and do **not** end with 0110.

5.C. All strings x such that the number of 0's in x is divisible by 3 and x contains 1101 as a substring.

5.D. All strings x such that between any two 1's in x, the number of 0's is divisible by 3. (For example, 0100010000001100 is in the language, but 010001000000101 is not.)

6 (100 PTS.) Spring 2020 Q2.2

Describe a DFA that accepts each of the following languages over the alphabet $\{0, 1\}$. Describe briefly what each state in your DFA *means*.

6.A. All strings that contain 101100 as a substring.

6.B. All strings x such that the number of 0's in x is divisible by 3 and x does **not** end in 110.

[Hint: use the product construction.]

6.C. All strings x such that between any two 1's in x, the number of 0's is divisible by 3. (For example, 0100010000001100 is in the language, but 010001000000101 is not.)

7 (100 PTS.) Spring 2020 Q2.3

Describe a DFA that accepts each of the following languages. Describe briefly what each state in your DFA *means*. Do not attempt to draw your DFA (the number of states could be huge!). Instead, give a formal description of the states Q, the start state s, the accepting states A, and the transition function δ. Describe briefly what each state in your DFA *means*.

7.A. All strings in $\{0, 1, 2\}^*$ such that the number of 0's is divisible by 11, or the number of 1's is divisible by 13, or the number of 2's is divisible by 17.

7.B. The language L from Problem 1.2, i.e., of all strings in $\{0, 1\}^*$ that contain a balanced substring with length at least 6. (Recall that a string is *balanced* if it has the same number of 0's and 1's.)

[Hint: you may use the result from Problem 1.2.]

8 (100 PTS.) Regular expressions [Fall 20].

For each of the following languages over the alphabet $\{0, 1\}$, give a regular expression that describes that language, and briefly argue why your expression is correct.

8.A. (10 PTS.) All strings that end in 1011.

8.B. (10 PTS.) All strings except 11.

8.C. (10 PTS.) All strings that contain 101 or 010 as a substring.

8.D. (10 PTS.) All strings that contain 111 and 000 as a subsequence (the resulting expression is long – describe how you got your expression, instead of writing it out explicitly).

8.E. (10 PTS.) The language containing all strings that do not contain 111 as a substring.

8.F. (10 PTS.) All strings that do *not* contain 000 as a subsequence.

8.G. (10 PTS.) Strings in which every occurrence of the substring 00 appears before every occurrence of the substring 11.

8.H. (10 PTS.) Strings that do not contain the subsequence 010.

8.I. (10 PTS.) Strings that do not contain the subsequence 0101010.

8.J. (10 PTS.) Strings that do not contain the subsequence 10.

8.K. (Not for credit, do not submit a solution.) Strings that do not contain the subsequence 111000.

9 (100 PTS.) DFA I [Fall 20].

Let $\Sigma = \{0, 1\}$. Let L be the set of all strings in Σ^* that contain an even number of 0s and an even number of 1s.

9.A. (50 PTS.) Describe a DFA over Σ that accepts the language L. Argue that your machine accepts every string in L and nothing else, by explaining what each state in your DFA *means*. (Hint: Zero is even)

You may either draw the DFA or describe it formally, but the states Q, the start state s, the accepting states A, and the transition function δ must be clearly specified, in either case.

9.B. (50 PTS.) (Harder.) Give a regular expression for L, and briefly argue why the expression is correct. (Hint: First solve the much easier case where the strings do not contain any consecutive 0s or 1s.)

10 (100 PTS.) DFA II [Fall 20].

Let L_1, L_2, and L_3 be regular languages over Σ accepted by DFAs $M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$, $M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$, and $M_3 = (Q_3, \Sigma, \delta_3, s_3, A_3)$, respectively.

10.A. (20 PTS.) Describe formally the product construction of the DFA M that accepts the language $L_1 \cap L_2 \cap L_3$.

10.B. (30 PTS.) In the DFA M constructed in (**10.A.**), a state is a triple (q_1, q_2, q_3). Let δ the transition function of M, and let δ^* be the standard extension of δ to strings. Prove by induction that for any string $w \in \Sigma^*$, we have that

$$\delta^*\big((q_1, q_2, q_3), w\big) = \big(\delta_1^*(q_1, w), \delta_2^*(q_2, w), \delta_3^*(q_3, w)\big).$$

10.C. (20 PTS.) Describe a DFA $M = (Q, \Sigma, \delta, s, A)$ in terms of M_1, M_2, and M_3 that accepts $L = \left\{ w \;\middle|\; w \text{ is in exactly two of } \{L_1, L_2, L_3\} \right\}$. Formally specify the components Q, δ, s, and A for M in terms of the components of M_1, M_2, and M_3. Argue that your construction is correct.

10.D. (30 PTS.) You are given a DFA $M = (Q, \Sigma, \delta, s, A)$, for $\Sigma = \{0, 1\}$. Describe in detail how to build a DFA that accepts the language

$$L = \left\{ w \in \Sigma^* \;\middle|\; w \notin L(M), \overline{w} \in L(M) \text{ and } 1^{|w|} \in L(M) \right\}.$$

How many states does your DFA has as a function of $n = |Q|$? Argue that the DFA you constructed indeed accepts the specified language.

Here, for $w = w_1 w_2 \ldots w_m \in \Sigma^*$, the *complement string* $\overline{w}$ is $\overline{w_1}\,\overline{w_2}\,\overline{w_3} \ldots \overline{w_m}$, where $\overline{0} = 1$, and $\overline{1} = 0$.

11 (100 PTS.) 374 Balanced [Fall 22].

A string s over $\Sigma = \{0, 1\}$ is ***balanced*** if (i) $\#_0(s) = \#_1(s)$, and for any prefix p of s we have that $\#_0(p) \geq \#_1(p)$. Here, for any character $c \in \Sigma$, and any string $w \in \Sigma^*$, the quantity $\#_c(w)$ is the number of times the character c appears in w. Thus, the strings

0101010101, 00101101001011, 00011101, and 010011,

are balanced, while

10, 001, 001110, 0001110111111, and 01001110,

are not balanced. A string w is 374 ***balanced*** if w is balanced, and for any prefix p of w, we have that $0 \leq \#_0(p) - \#_1(p) \leq 374$.

For both languages specified below, describe *formally* a DFA that accepts them. In addition, explain informally and precisely the idea beyond your DFA and how it works.

11.A. (50 PTS.) Let L_1 be the language of all 374 balanced strings.

11.B. (50 PTS.) Let L_2 be the language of all binary strings w, such that:

(i) w is 374 balanced,
(ii) $|w|$ is divisible by 16, and
(iii) w contains 0000 as a substring.

(The language of all balanced strings is not regular, so this question is interesting because the more restricted languages L_1 and L_2 are regular.)

12 (100 PTS.) Freedom of regular expressions [Fall 22].

For each of the following languages over the alphabet $\{0, 1\}$, give a regular expression that describes that language, and briefly argue why your expression is correct.

12.A. (30 PTS.) The language containing all strings that do not contain 000 as a substring.

12.B. (70 PTS.) All strings that do *not* contain 0110 as a subsequence.

(Hint: (A) Break the input string into runs – a ***run*** of a string w is a maximal substring s all made of the same character. (B) You might want to solve an easier version of this question first, where 010 is a forbidden subsequence.)