Homework2 for cs420 error in prob 3 fixed sep 23

J. H. DeBlois

September 23, 2026

Be sure to leave space at the right for your handwritten proof justifications!

```
Homework 2

Out: Mon Sept 21, 12:00pm EDT (noon) Due: Mon Sept 28, 12:00pm EDT (noon)

This assignment continues to explore deterministic finite automata (DFAs) and regular
languages.

H2-1. Make a subdir in your course dir named hw2 - inclass on Mon Sep 21.

H2-2. Fill out the inclass page - handed out inclass on Mon Sep 21

(and submit revisions on CS server as needed).

H2-3. Submit on the CS server to Course Directory subdir hw2:

Homework Problems

1. DFAs and Regular Languages (12 points)

2. Is It A Regular Language? (12 points)

3. DFAs Can Do "Real" Computation (10 points)

4. README (1 point)

Maximum: 35 points
```

1

```
DFAs and Regular Languages

Let the language recognized by DFA T from Homework 1 problem DFA Formal Description be
the following set of strings:

LT = {420, 620}

Prove that the language LT is a regular language.
```

Important: Before attempting the proof, make sure to understand what this problem is asking. This means you must know the precise definitions of the terms used in this problem, and all notation, before you can begin solving this question.

For example, make sure you know the precise formal definition of: string, language, recognize, regular language, etc.

Your proof must be in the form of a Statements and Justifications table, as presented in class. The justifications must be hand-written.

If your answer includes designing a DFA, you may submit a state diagram instead of a formal description of the machine.

You may also refer to DFAs already created in class or previous homework problems, if appropriate. There is no need to re-define a previously-defined definition.

To "prove" that a DFA recognizes a language, the justification (as shown in lecture) must be a three-column table showing:

a) three strings that are both "in the language" and "accepted by the DFA", and

b) three strings that are not "in the language" and not "accepted by the DFA".

The key is that for each example string, the entries in the "in the language?" column and the "accepted by DFA?" column should match. The strings should ideally "test" as many different transition paths through the DFA as possible. (This should also tell you whether the strings you came up with in parts 1 and 2 are "good" examples or not.)

2

Is It A Regular Language?

Here is a language definition:

LAC = {w such that w is a valid area code}

You may assume that strings in the language contain only characters from alphabet {2, 3, 4}.

1. Come up with 3 strings in the language.

2. Come up with 3 strings not in the language.

For this problem, for parts 1 and 2, you are allowed to post examples to Piazza for clarification if you are unsure.

3. Prove that the language is a regular language.

Important: Before attempting the proof, make sure to understand what this problem is asking. This means you must know the precise definitions of the terms used in this

problem, and all notation, before you can begin solving this question.

For example, make sure you know the precise formal definition of: string, language, regular language, etc. Further, you need to know how to read the set (builder) notation above and what it means for something to be in and not in a set.

Your proof must be in the form of a Statements and Justifications table, as presented in class. The justifications must be handwritten.

If your answer includes designing a DFA, you may submit a state diagram instead of a formal description of the machine.

You may also refer to DFAs already created in class or previous homework problems, if appropriate. There is no need to re-define a previously-defined definition.

To "prove" that a DFA recognizes a language, the justification (as shown in lecture) must be a three-column table showing:

a) three strings that are both "in the language" and "accepted by the DFA", and

b) three strings that are not "in the language" and not "accepted by the DFA".

The key is that for each example string, the entries in the "in the language?" column and the "accepted by DFA?" column should match. The strings should ideally "test" as many different transition paths through the DFA as possible. (This should also tell you whether the strings you came up with in parts 1 and 2 are "good" examples or not.)

3

DFAs Can Do "Real" Computation

Here is an example showing that DFAs can perform "real" computation.

Assume that the following is an alphabet of "domino" symbols (one could easily use unicode characters to represent such an alphabet):

$$\Sigma = [\tfrac{0}{0}, \tfrac{0}{1}, \tfrac{1}{0}, \tfrac{1}{1}]$$

Now assume we have a sequences of such dominoes, where the top row of digits are interpreted together as a binary number (leftmost digit is most significant, i.e., "big-endian"). The bottom row of digits are also interpreted in the same way.

For example in the following domino sequence:

$$[\tfrac{0}{1}, \tfrac{1}{0}, \tfrac{1}{1}]$$

the "top row" binary number would be 011 (or 3 in base ten) and the "bottom row" binary number would be 101 (5 in base ten).

Let Lneq be a language consisting of sequences of dominos where:

Lneq = {w such that w is a sequence of dominoes where top row is not equal to bottom row }

1. Come up with 3 strings in the language.

2. Come up with 3 strings not in the language.

3. Prove that Lneq is a regular language.

Your proof must be in the form of a Statements and Justifications table, as presented in class. The justifications must be hand-written.

If your answer includes designing a DFA, you may submit a state diagram instead of a formal description of the machine.

You may also refer to DFAs already created in class or previous homework problems, if appropriate. There is no need to re-define a previously-defined definition.

To "prove" that a DFA recognizes a language, the justification (as shown in lecture) must be a three-column table showing:

a) three strings that are both "in the language" and "accepted by the DFA", and

b) three strings that are not "in the language" and not "accepted by the DFA".

The key is that for each example string, the entries in the "in the language?" column and the "accepted by DFA?" column should match. The strings should ideally "test" as many different transition paths through the DFA as possible. (This should also tell you whether the strings you came up with in parts 1 and 2 are "good" examples or not.)