Homework 7: REGEX and CIRCLES!! (aka DFAs/NFAs)

Due date: Friday, August 9nd at 11:59 PM
If you work with others (and you should!), remember to follow the collaboration policy outlined in the syllabus.
In general, you are graded on your work's clarity and accuracy. Your solution should be clear enough that someone in the class who had not seen the problem before would understand it.
We sometimes describe approximately how long our explanations are. These are intended to help you understand approximately how much detail we expect. You can have longer explanations, but explanations significantly longer than necessary may receive deductions.

1. Long Live the String [42 points]

In lecture, we defined the set of binary strings of even length recursively as follows:

Basis: $\varepsilon \in S$.
Recursive Step: If $x \in S$ and $a, b \in \{0, 1\}$, then $xab \in S$.

We could instead define the set of binary strings of even length more directly like this:

$$T \equiv \{x \in \{0,1\}^* : 2 \mid \mathsf{len}(x)\}$$

In the following parts, we will prove that the two definitions are equivalent, i.e., that $S = T$.

(a) Use structural induction to prove that $\forall x \in S\,(x \in T)$.

(b) Use strong induction to prove that $\forall n \in \mathbb{N}\,(\forall x \in \{0,1\}^*(\mathsf{len}(x) = 2n \to x \in S))$.

You will need to use the fact that, if $\mathsf{len}(x) = 0$, then $x = \varepsilon$, which we will call Lemma 1, and the fact that, if $\mathsf{len}(x) > 0$, then $x = ya$ for some $a \in \{0, 1\}$ and $y \in \{0, 1\}^*$, which is Lemma 2.

(c) Explain, in at most three sentences, why parts (a-b) tell us that $S = T$.

2. The great expression [10 points]

Use the algorithm from lecture to convert each of the following regular expressions into NFAs that accept the same language. You may skip adding ε-transitions for concatenation if they are *obviously* unnecessary, but otherwise, you should follow the construction from lecture.

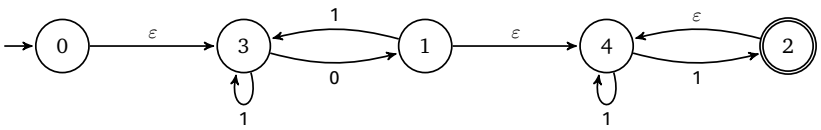
(a) $0(1 \cup 00)^* \cup 111$

(b) $(0(1 \cup 00)^*11)^*$

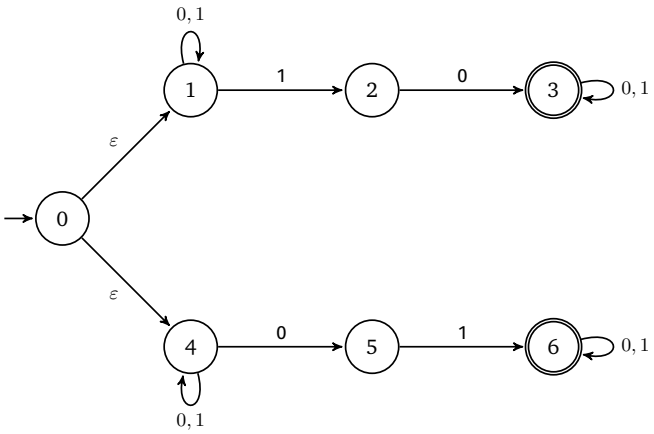
3. Converts [10 points]

Use the algorithm from lecture to convert each of the following NFAs to DFAs. Label each DFA state with the set of NFA states it represents in the powerset construction.

(a) The NFA below, which accepts *some* strings ending with 011^*:



(b) The NFA below, which accepts strings containing "10" or "01":



Submit and check your answers to this question here:
http://grin.cs.washington.edu
Think carefully about your answer to make sure it is correct before submitting.
You have only **5 chances** to submit a correct answer.

4. Build DFAs (Online) [15 points]

For each of the following languages, construct a DFA that accepts exactly the given set of strings. You should submit (and check!) your answers online at https://grin.cs.washington.edu/

Think carefully before entering your DFA; you have a limited number of guesses. Because these are auto-graded, we will not award partial credit.

(a) Create a DFA that accepts binary strings with at least four 1s

(b) Create a DFA that accepts binary strings where every occurrence of a 0 is immediately followed by a 11.

(c) Construct a DFA that recognizes the set of binary strings **not** containing the substring 1011.

(d) Create a DFA that accepts binary strings that **either** have every occurrence of a 0 immediately followed by a 1 or contain at least two 1s **but not both**.

(e) Construct a DFA that accepts binary strings with at most two 1's **and** an even number of 0's

5. Build NFAs (Online) [15 points]

For each of the following languages, construct an **NFA** that accepts exactly the given set of strings. You should submit (and check!) your answers online at https://grin.cs.washington.edu/

Think carefully before entering your NFA; you have a limited number of guesses. Because these are auto-graded, we will not award partial credit.

(a) Create an NFA that recognizes binary strings that end in 101.

(b) Create an NFA that recognizes binary strings with at least three 1s **and** end with 000.

Hint: This can be done without the product construction.

(c) Construct an NFA for binary strings with at least two 0s **or** at least two 1s.

(d) Construct an NFA which recognizes the same language as the regular expression (1 01 001) (0 00).

Hint: While you can use the algorithm from lecture to convert this REGEX to a NFA, we highly DO NOT recommend doing so (it will get very messy). Are the extra "powers" of an NFA really necessary here?

6. Feedback [1 point]

Please keep track of how much time you spend on this homework and answer the following questions. This can help us calibrate future assignments and future iterations of the course, and can help you identify which areas are most challenging for you.

- How many hours did you spend working on this assignment (excluding any extra credit questions, if applicable)? Report your estimate to the nearest hour.
- Which problem did you spend the most time on?
- Any other feedback for us?