CSE 311: Foundations of Computing I Winter 2026

Problem Set 5

Due: Friday, February 27th by 11:00pm

Instructions

Write up carefully argued solutions to the following problems. Each solution should be clear enough that it can explain (to someone who does not already understand the answer) why it works.

Collaboration policy. You are required to submit your own solutions. You are allowed to discuss the homework with other students. However, the **write-up** must clearly be your own, and moreover, you must be able to explain your solution at any time. We reserve ourselves the right to ask you to explain your work at any time in the course of this class.

Solutions submission. Submit your solution via Gradescope. In particular:

- Each numbered task should be solved on its own page (or pages). Do not write your name on the individual pages. (Gradescope will handle that.)
- When you upload your pages, make sure each one is **properly rotated**. If not, you can use the Gradescope controls to turn them to the proper orientation.
- Follow the Gradescope prompt to **link tasks to pages**.
- You are not required to typeset your solution, but your submission must be **legible**. It is your responsibility to make sure solutions are readable — we will *not* grade unreadable write-ups.
- Extra practice problems are included at the bottom of the assignment. These will not be graded, so don't submit solutions to them.

Task 1 – i l∗vε rεg∪10r εxprε2210n2 [12 pts]

For each of the following, construct regular expressions that match the given set of strings.

a) *Binary strings with an odd number of 1s.*

b) *Binary strings which do not have any 3 or more consecutive repetitions of a digit* (e.g., the following would *not* belong to this language: 111, 1000110, 0011000011).

c) *Years Rick Astley was alive* (1966 to present, 2026). That is, all numbers from 1966 to 2026, inclusive. Your expression must not be "brute force" (e.g., $1966 \cup 1967 \cup \ldots \cup 2026$): you should **never** have more than 10 things unioned together at once (e.g., $1 \cup 2 \cup \ldots \cup 20$). Unfortunately, you're **gonna** have to write out long chains of unions to represent the digits 0-9; GRIN will not accept shorthand such as "$0 \cup 1 \cup \ldots \cup 9$". **Give** extra thought to breaking down the structure of this language so that **you** can avoid brute-forcing. This regular expression is a little tedious, but don't give **up**!

Submit and check your answers to all parts here:

http://grin.cs.washington.edu

Think carefully about your answer to make sure it is correct before submitting. While you have practically unlimited chances (999 allowed attempts) to submit a correct answer in this homework, you will only have one chance on the quiz.

Task 2 – Glitz and Grammar [12 pts]

For each of the following, construct a context-free grammar that generates the given set of strings.

a) Binary strings matching the regular expression "$10(01 \cup 1)^*11 \cup 0110$".

Hint: You can use the technique described in lecture to convert this RE to a CFG.

b) All strings of the form $\#y\#x$, with $x, y \in \{0,1\}^*$ and y is x^R, but with each of the characters optionally doubled.

(Here x^R means the reverse of x.)

c) All binary strings in the set $\{1^m 0^n : m, n \in \mathbb{N} \text{ and } m > 2n + 1\}$.

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. While you have practically unlimited chances (999 allowed attempts) to submit a correct answer in this homework, you will only have one chance on the quiz.

Task 3 – Relatin' Canes [12 pts]

One would think that, by this point in the quarter, the TA's would stop getting into situations involving their fast food orders, but, alas, it has happened once more. During the TA's trip to Relatin' Canes, an incredibly popular chicken establishment (and, apparently, gathering spot for discrete math nerds), the TA's couldn't figure out what to order, where to sit, and who's going! They have encoded the whole situation into several relations over sets; we need your help to figure out the properties of these relations and solve this once and for all!

For each of the relations below, determine whether or not it has each of the properties of reflexivity, symmetry, antisymmetry, and/or transitivity. If a relation has a property, simply say so without any further explanation. If a relation does not have a property, state a counterexample, but do not explain your counterexample further.

a) Define $R \subseteq \mathbb{Z} \times \mathbb{Z}$ by $(a, b) \in R$ iff $a + b$ is odd.

b) Define $S \subseteq \mathbb{N} \times \mathbb{N}$ by $(a, b) \in S$ iff $a \cdot b \geqslant 0$.

c) Let $A = \{k \in \mathbb{Z} : k < 0\}$ be the set of negative integers. Define $T \subseteq A \times A$ by $(a, b) \in T$ iff $a \times b > 0$

d) Let $B = \mathcal{P}(\mathbb{N})$. Define $U \subseteq B \times B$ by $(X, Y) \in U$ iff $X \cup [6] \subseteq Y \cap [6]$. (Remember that $[n] = \{1, \ldots, n\}$.)

Hint: What can be in X? What can't be?

Task 4 – Better Get Proving [12 pts]

Let R be a relation on a set A. Recall from lecture the definition of the *composition* of two relations:

$$R \circ R := \{(a, b) \mid \exists \mathbf{c}\,((a, \mathbf{c}) \in R \wedge (\mathbf{c}, b) \in R)\}$$

and "R is *transitive*" iff

$$\forall a \in A\,\forall b \in A\,\forall c \in A\,(((a, b) \in R \wedge (b, c) \in R) \to (a, c) \in R)$$

and "R is *antisymmetric*" iff

$$\forall a \in A\,\forall b \in A\,(((a, b) \in R \wedge (b, a) \in R) \to a = b)$$

The definition of antisymmetric is different from the one in the lecture, but it is equivalent. Using this definition can make the proof slightly easier.

Now, consider the following claim:

> Given that R is transitive and antisymmetric, it follows that $R \circ R$ is antisymmetric.

Write an **English proof** that the claim holds.

Note: Even though we want you to write your proof directly in English, it must still look like the translation of a formal proof. In particular, you must include all steps that would be required of a formal proof, excepting only those that we have explicitly said are okay to skip in English proofs (e.g., Elim $\exists$).

Task 5 – McDFAnalds [15 pts]

The TA's are quickly growing tired of fast food – mainly, because they keep getting into these absurd situations involving change. Luckily for them, this week, things are different... yes, they've actually managed to plan our order this time! However, they're experiencing one slight hiccup: Their language has become... irregular... and they can no longer figure out how to order! They have encoded their language debacle into several DFA-type problems, and they need your help to order their meals. Moreover, the TA's feel bad – every week, they promise fast food and never deliver. As such, they promise that this week the rewards will be immense; they just can't say what the rewards are for reasons out of their control (or something). Either way, please help! It's almost their turn to order...

For each of the following, create a *DFA* that recognizes exactly the language given. For all states in your DFA, write "documentation" for them by describing, in English, the set of strings that end in that state. **(No need to turn in the state documentation for this homework, but don't skip it — you can and will be asked to write it on a quiz/exam with FSM problems.)**

a) Binary strings that start with 110 and have an odd number of 0's

b) Binary strings where every 0 is followed by an odd number of 1s.

c) Binary strings with exactly three 1's that also end with exactly two 0's.

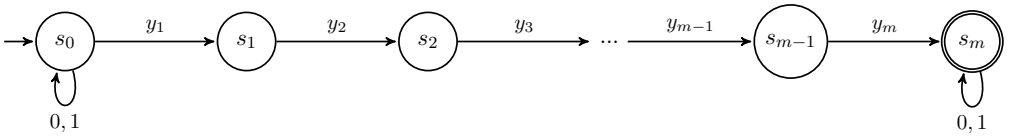
Submit and check your DFAs here:

http://grin.cs.washington.edu

Think carefully about your answer to make sure it is correct before submitting. While you have practically unlimited chances (999 allowed attempts) to submit a correct answer in this homework, you will only have one chance on the quiz.

Task 6 – Extra Credit: Strings to Mind [0 pts]

Suppose we want to determine whether a string x of length n contains a string $y = y_1 y_2 \dots y_m$ with $m \ll n$. To do so, we construct the following NFA:



(where the ... includes states $s_3, \dots, s_{m-2}$). We can see that this NFA matches x iff x contains the string y.

We could check whether this NFA matches x using the parallel exploration approach, but doing so would take $O(mn)$ time, no better than the obvious brute-force approach for checking if x contains y. Alternatively, we can convert the NFA to a DFA and then run the DFA on the string x. *A priori*, the number of states in the resulting DFA could be as large as 2^m, giving an $\Omega(2^m + n)$ time algorithm, which is unacceptably slow. However, below, you will show that this approach can be made to run in $O(m^2 + n)$ time.

(a) Consider any subset of states, S, found while converting the NFA above into a DFA. Prove that, for each $1 \leqslant j < m$, knowing $s_j \in S$ *functionally determines* whether $s_i \in S$ or not for each $1 \leqslant i < j$.

(b) Explain why this means that the number of subsets produced in the construction is at most $2m$.

(c) Explain why the subset construction thus runs in only $O(m^2)$ time (assuming the alphabet size is $O(1)$).

(d) How many states would this reduce to if we then applied the state minimization algorithm?

(e) Explain why part (c) leads to a bound of $O(m^2 + n)$ for the full algorithm (without state minimization).

(f) Briefly explain how this approach can be modified to count (or, better yet, find) *all* the substrings matching y in the string x with the same overall time bound.

Note that any string matching algorithm takes $\Omega(m + n) = \Omega(n)$ time in the worst case since it must read the entire input. Thus, the above algorithm is optimal whenever $m^2 = O(n)$, or equivalently, $m = O(\sqrt{n})$, which is the case for normal inputs circumstances.

Task 7 – Optional Practice Problems (Ungraded) [0 pts]

The problems below are **optional practice problems** that are **not required and will not be graded**. They are provided to help you practice; you do not need to submit solutions to these problems. The difficulty of these problems has not been vetted as thoroughly, so don't fret if they seem challenging.

a) Since regular expressions, DFAs, and NFAs all describe exactly the *regular languages*, any language expressible by one can be expressed by all three. Moreover, since every regular language is also context-free, a CFG can describe any regular language too (**though not vice versa**). With this in mind, revisit every regular language from the graded homework tasks and the quiz section handout: for each one, build whichever representations you did *not* originally construct (e.g., if the problem asked for a regex, also write a DFA/NFA and CFG).

b) For each of the following regular languages, construct a regular expression, an FSM, and a CFG.

(a) Binary strings containing an even number of `0`'s.

(b) Binary strings that do *not* contain the substring `00`.

(c) Strings over $\{a, b\}$ in which every `a` is immediately followed by a `b`.

(d) Strings over $\{0, 1, 2\}$ whose digit sum is divisible by 3.

c) Construct CFGs for the following languages (which are *not* regular).

(a) $\{a^n b^m : n \leqslant m \leqslant 2n,\ n \geqslant 0\}$.

(b) $\{w \in \{0,1\}^* : w$ has strictly more `1`'s than `0`'s$\}$.

d) Let R be a relation on an arbitrary set A.

(a) Prove: if R is reflexive and transitive, then $R \circ R = R$. (How do you prove set equality?)

(b) Prove: if R is symmetric, then $R \circ R$ is symmetric.

(c) Prove: if R_1 and R_2 are both equivalence relations on A, then $R_1 \cap R_2$ is also an equivalence relation on A.