Problem Set 8

Due: Friday, June 2, by 11:59pm

Instructions

Solutions submission. You must submit your solution via Gradescope and on grin.cs.washington.edu as noted. In particular:

- **There are no late days for this homework. HW8 Solutions will be posted under Pages in Canvas on Saturday, June 3rd.**
- Submit your solution to tasks 1 and 2 online to grin.cs as described in the tasks themselves.
- Submit a single PDF file in Gradescope with a copy of your state diagrams from task 2 showing how you have labeled the states of your DFAs plus your solutions to all the regular tasks 3-7.
- The extra credit is submitted separately in Gradescope

Task 1 – Design Intervention (Online) [18 pts]

For each language below, create a state machine to recognize it. Read carefully to see whether you are asked to construct an NFA or a DFA.

For this problem, you do *not* need to document your states.

a) Create an *NFA* that recognizes all binary strings with at least two 0s **or** at least two 1s.

b) Create an *NFA* that recognizes all binary strings with at least four 0s **and** end with 010.

Hint: This can be done without the product construction.

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

Submit and check your answers to this question here:

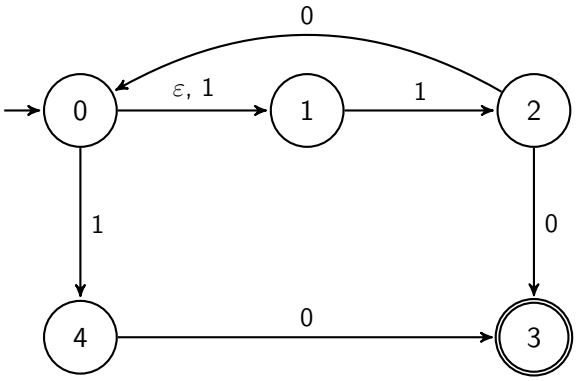
https://grin.cs.washington.edu

Think carefully about your answer to make sure it is correct before submitting. You have only 3 chances to submit a correct answer.

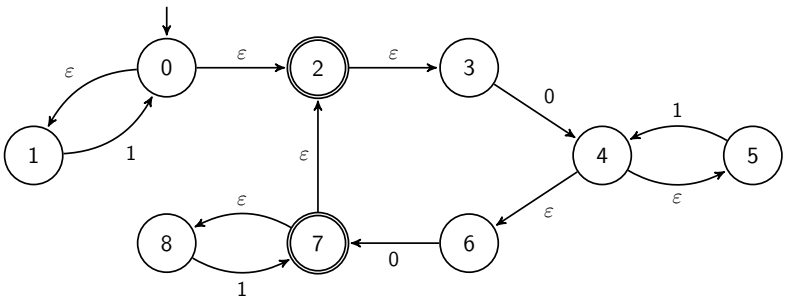
Task 2 – Devious Machinations (Online) [20 pts]

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.

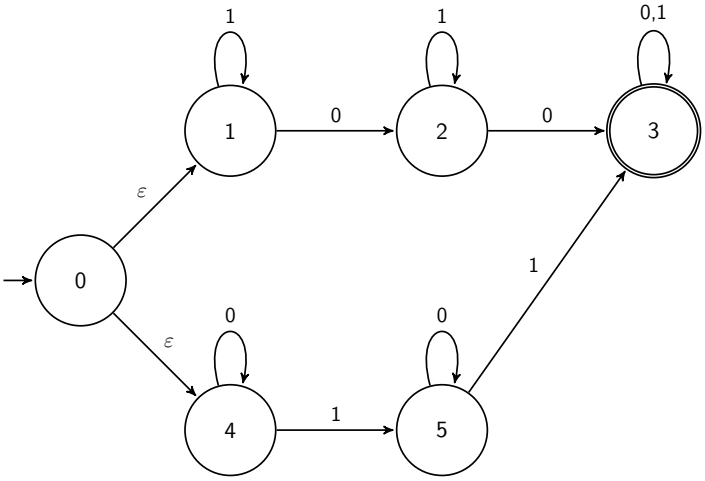
a) The NFA below:



b) The NFA below, which is a slightly simplified version of the one that is produced by the construction described in class on the regular expression $1^*(01^*01^*)^*$:



c) The NFA below, which is another way to match the language from HW7 Problem 5e.



Submit and check your answers to this question here:

https://grin.cs.washington.edu

Think carefully about your answer to make sure it is correct before submitting. You have only 3 chances to submit a correct answer.
You must *also* submit the a screenshot/sketch of your submitted DFA in Gradescope showing that you have correctly labelled the DFA states with the sets of NFA state names as part of the powerset construction so that we can check it. (You do not need to document this diagram any further.)

Task 3 – Expression Is the Better Part of Valor [6 pts]

Use the algorithm from lecture to convert the following regular expression into an NFA that accepts the same language. You may skip adding ε-transitions for concatenation if they are *obviously* unnecessary, but otherwise, you should **precisely** follow the construction from lecture.

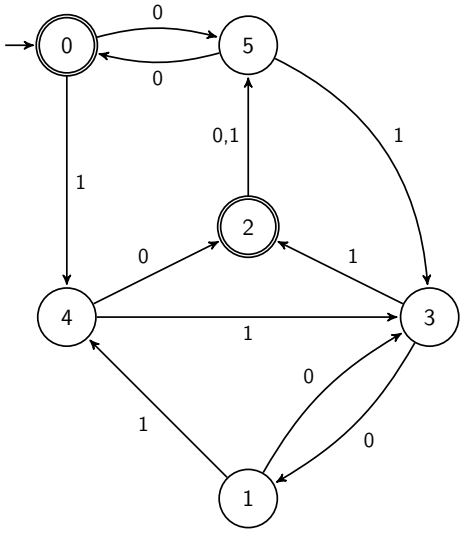
$$1(0 \cup 111)^* \cup 000$$

Task 4 – A Whole New Small Game [16 pts]

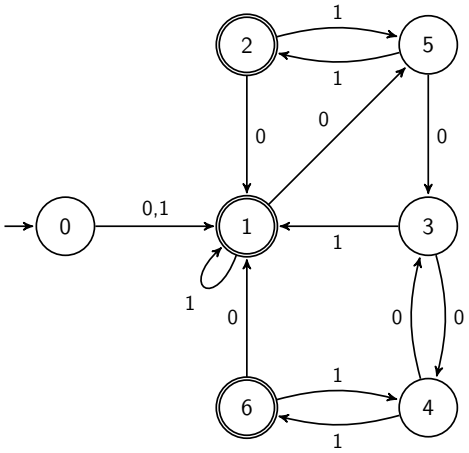
Use the algorithm from lecture to minimize the each of the following DFAs.

For each step of the algorithm, write down the groups of states, which group was split in that step and the reason for splitting that group. At the end, write down the minimized DFA, with each state named by the set of states of the original machine that it represents (e.g., "B, C" if it represents B and C).

a)



b)



Task 5 – Just Irregular Guy [20 pts]

Use the method described in lecture to prove that each of the following languages is **not regular**.

a) All binary strings in the set $\{0^m 1^n 0^{2n+m} : m, n \geqslant 0\}$.

b) All binary strings of the form $x\#y$, with $x, y \in \{0,1\}^*$ and x a subsequence of y^R. (Here y^R means the reverse of y. Also, a string w is a subsequence of another string z if you can delete some characters from z to arrive at w.)

Task 6 – Diagonalization [10 pts]

Let B be the set of all infinite binary sequences that are 1 in even positions, i.e., any string in B is of the form

$$\_1\_1\_1\_1\ldots$$

where we can have 0 or a 1 instead of each "_". Show that B is uncountable using a proof by diagonalization.

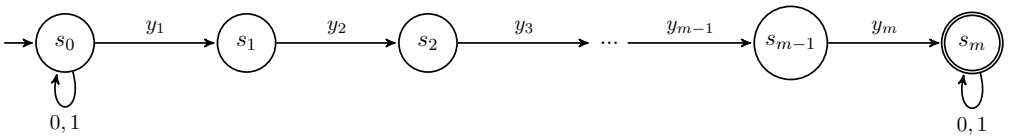
Task 7 – Countability [10 pts]

Let $\mathbb{Q}^+ = \{x \in \mathbb{Q} \ : \ x > 0\}$ be the set of positive rational numbers. In lecture, we showed that $\mathbb{Q}^+$ is countable.

Prove that $\mathbb{Q}^+ \times \mathbb{Q}^+$, the set of all pairs of positive rational numbers, is also countable.

Task 8 – Extra Credit: Strings to Mind

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 is $O(n)$, or equivalently, when m is $O(\sqrt{n})$, which is typically the case for m and n in practice.