Problem Set 7

Due: Friday, May 31st by 11:00pm

Instructions

Solutions submission. You must submit your solution via Gradescope. In particular:

- Submit a *single* PDF file containing your solutions to Task 3, 5, 6 (and optionally 7). Follow the prompt on Gradescope to link tasks to your pages.
- The instructions for submitting Tasks 1, 2, and 4 appear below those individual problems.

Task 1 – Few and Far Machine [25 pts]

For each of the following, create a *DFA* that recognizes exactly the language given.

a) Binary strings that start with 1 and have odd length.

b) Binary strings where every occurrence of a 1 is immediately followed by a 00.

c) Binary strings with an odd number of 1s.

d) Binary strings with at least three 1s.

e) Binary strings with at least three 1s **or** at most two 0s.

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.

Task 2 – Design Intervention [15 pts]

For each of the following, create an *NFA* that recognizes exactly the language described.

a) Binary strings with at least three 1s **or** at most three 0s

Hint: Since this is an NFA, your answer can be much smaller than for Task 1(e).

b) Binary strings that start with 11 **or** contain at least three 0s.

c) Binary strings that start with 11 **and** contain at least three 0s.

Note that this is "and" rather than "or" like above!

> 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.

Task 3 – Expression Is the Better Part of Valor [10 pts]

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 **precisely** follow the construction from lecture.

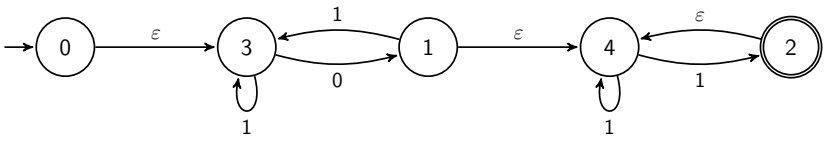
a) $0(1 \cup 00)^* \cup 111$

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

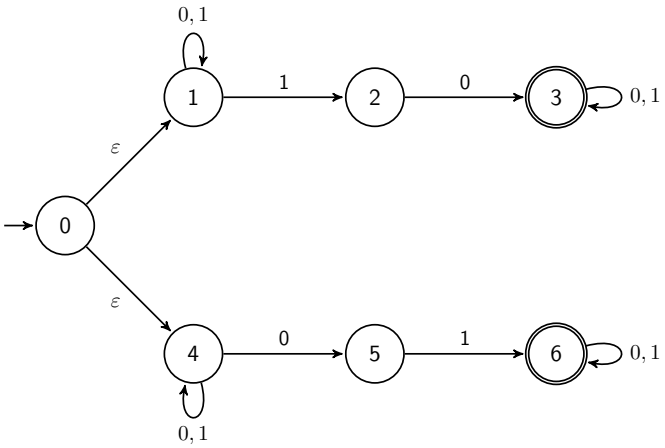
Task 4 – Machin-o Acids [10 pts]

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, 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

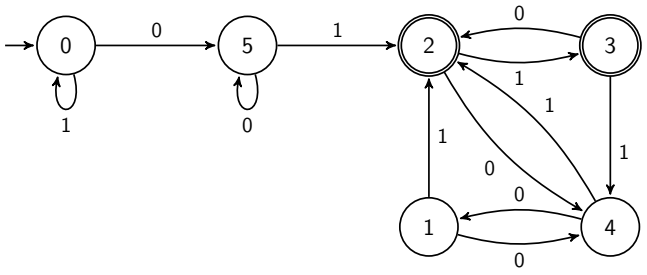
Think carefully about your answer to make sure it is correct before submitting. You have only **5 chances** to submit a correct answer.

Task 5 – A Whole New Small Game [16 pts]

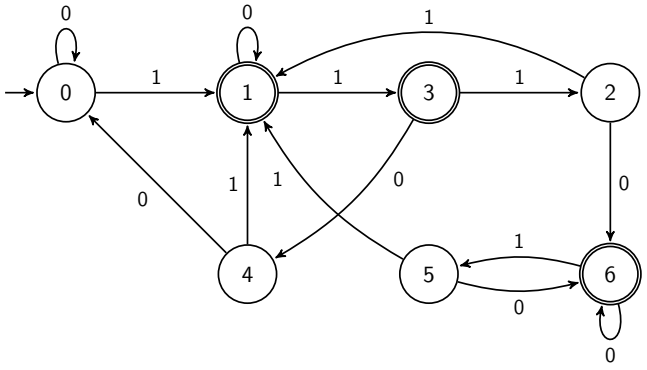
Use the algorithm from lecture to minimize the each of the following DFAs.

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., "$\{B, C\}$").

a) The following machine:



b) The following machine:



Task 6 – Just Irregular Guy [20 pts]

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

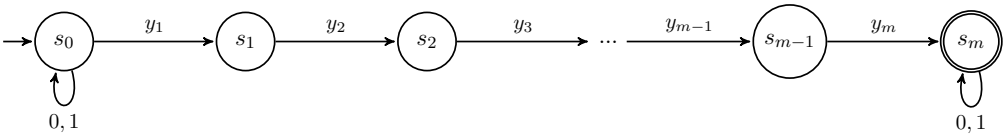
a) All binary strings in the set $\{0^m 1^n : m, n \in \mathbb{N} \text{ and } m < n\}$.

b) All strings of the form $x\#y$, with $x, y \in \{0,1\}^*$ and y a subsequence of x^R.

(Here x^R means the reverse of x. Also, a string w is a subsequence of another string z if you can delete some characters from z to arrive at w.)

Task 7 – Extra Credit: Strings to Mind [0 pts]

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



(where the ... includes states $s_3, \ldots, s_{m-2}$). 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.