HW3

OK to write a DFA for #1, but make sure the formal definition is in NFA form. (The problem given is not a good one for an NFA.)

DFA is 8 states

3A + B can be DFA or NFA (the combined form will be an NFA)

#2: If #1 is a DFA, all your subsets will be singletons. Make sure you write down the formal construction like null set for all the wasted transitions.

#5: Your output will be either accept or reject. (e.g. accept if first is larger, reject otherwise.)
The alphabet is not specified. You can specify yours.
Suggestion: read pairs of symbols.
Assume the smaller string is padded with zeroes.
NFA's: Examples

$L = \{ w \mid w \text{ ends in } 1001 \}$

For DFA, 16 possible ending combos requires 16 states.

But NFA!

Contains a substring - good for NFA's

$L = \{ w \mid w \text{ contains } 1001 \}$

OK, good for NFA's

$L = \{ w \mid w \text{ contains exactly 2 1's or even } \# \text{ of 0's} \}$

Exactly 2 1's

Even 0's

$L = \{ w \mid \text{ every odd position of } w \text{ is a 1} \}$

Examples: $\epsilon, 1, 01, 101, 111$ [ $\epsilon$ is set b/c it has no odd positions]

Even though all states are accept, it will still reject all the right strings (by dying)
Closure Properties of Regular Languages

Last time: \( A, B \) are reg \( \Rightarrow \) \( A \cup B \) reg

\[ \text{Concat: } A \circ B = \{ w_1 \circ w_2 | w_1 \in A \text{ and } w_2 \in B \} \]

\( A = \{00, 11\} \)

\( A \circ A = \{0000, 0011, 1100, 1111\} \)

Example

Thm: \( A, B \text{ reg } \Rightarrow A \circ B \text{ reg} \)

Proof:

\[ L(N_1) = A \]

\[ L(N_2) = B \]

There cannot remain accept states or we could accept strings in \( A \), not also in \( A \circ B \) (like 00 or 11 from our example)

Formal Proof:

The machine created by the algorithm is correct:

\[ w \in L(N) \iff N \text{ accepts } w \]

\[ \iff \exists \text{ path from } q_0 \text{ to accept state of } N \]

\[ \iff \exists \text{ path from } q_0 \text{ to accept state of } N_1 \]

\[ \text{ and } \exists \text{ path from } q_0 \text{ to accept state of } N_2 \]

\[ \iff \exists x, y \text{ s.t. } w = xy \text{ and } N_1 \text{ accepts } x \]

\[ \text{ and } N_2 \text{ accepts } y \]
**Star Operation**  
A is some lang.

\[ A^* = \{ x_1 x_2 x_3 \ldots x_k \mid x_i \in A \text{ and } k \geq 0 \} \]

**Theorem**  
If \( A \) is reg., then \( A^* \) is reg.

**Proof**  
Let \( L(N) = A \)

**NFA \( N_1 \)**

First idea: Just do \( \epsilon \) transitions from all the accept states back to the start state.

But what about \( \epsilon \), which is always \( \epsilon \in A^* \)?

Can we just always make the start state an accept state?

Fails for some NFA's.

E.g., \( w \) contains a \( 1 \)

So: Add a new accept state at beginning

**NFA \( N \)**