DFA=NFA=RE Algorithm Summary

Computational models A and B have equivalent power if

1. Every A can be converted into an equivalent B, and
2. Every B can be converted into an equivalent A.

The following show that DFA, NFA and RE have equivalent power.

DFA to NFA

No DFA breaks any NFA design rules, so every DFA is also an NFA. Also, a DFA when treated as an NFA will have a path to an accept state for string *s* if and only if the DFA accepts *s*.

NFA to DFA

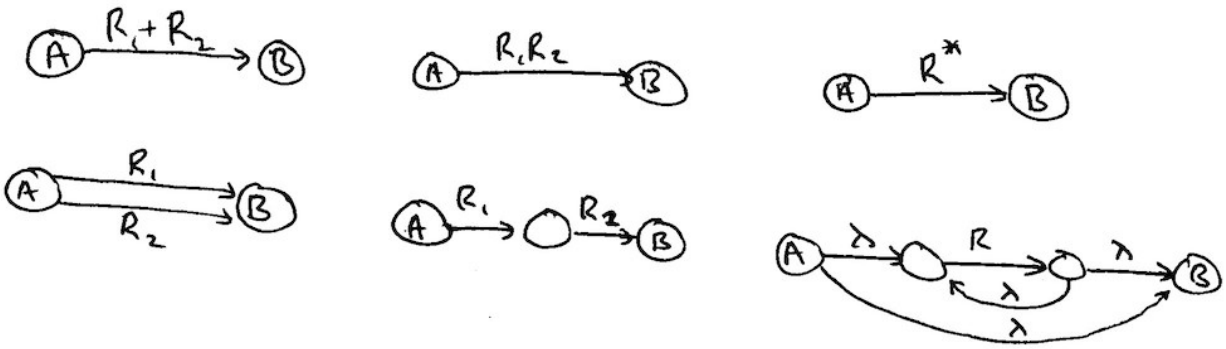
Idea: Each DFA tracks which NFA states the NFA could be in after consuming a sequence of characters.

1. Label DFA start state with where NFA could be without consuming anything.
2. Make DFA legal. Repeat:
    - Identify missing DFA arrow and determine which states NFA could be in if consuming that character.
    - New arrow goes to state labeled with the states.
3. Any DFA state listing NFA accept is accept.

RE to NFA

Idea: Write RE on an FA arrow and repeatedly eliminate RE ops until only atomic REs remain, which is then an NFA.

1. Draw start and accept states with an arrow between them labeled with the RE.
2. Repeat in reverse RE operation order (ie, unions first, concatenations second, stars third, unless parenthesis force a different order): Replace an arrow that contains a regular expression operation from state A to state B with the equivalent structure shown below.



NFA to RE

Idea: Remove states from NFA, replacing them with equivalent arrows labeled by REs.

1. Replace NFA start and accept states with new ones, connecting them to the old ones with lambda transitions.
2. Repeat until only one arrow remains: remove a non-start non-accept state, replacing all of it's in-out arrow combinations with equivalent direct arrows labeled by REs.
3. Whenever two arrows have the same start and destination state, replace them with a single equivalent arrow (ie, replace arrows labeled R_1 and R_2 with an arrow labeled R_1+R_2).