CSCI_4350_1_25_2019.md

RE = reg expr

FA = finite automaton

RE -> FA

1. Build a nondeterministic FA (NFA) from the RE
2. Convert NFA to a determinisitic FA (DFA)
3. Minimize the DFA so it contains as few states as possible

NFA

An NFA is an FA where there may be mulitple paths for any given string

- A transition labelled with an \epsilon can be followed without consuming input
    - "What do you mean by comsuming input?

```
start s_0 -> s_1 -b-> s_2
```

```
start s_0 --\epsilon--> s_1 -b-> s_2
<!-- this work without advancing forward -->
```

    - There may be more than one transition from a given state that is labelled with the same character
    - There may be no transition from a given state this labelled with a particular character
    - A string is accepted by an NFA iff there is some path that consumes the string and ends up in a comsuming state
        - grep, does this. turn into NFA and then do its job

DFA

A DFA is an FA where there is exactly one path for any given string.

- For each state `s` and each character there is exactly one transition from `s` and labelled with a `c`
- \epsilon-transitions are NOT allowed
- Every DFA is also an NFA (with above restrictions)
- For every NFA, a DFA can be constructed that accepts/recognizes the same language

```
RE -> NFA

RE          |  NFA
```

```
----        |---------
\epsilon    -> s_0 --\epsilon--> s_1
a           -> s_0 -- a --> s_1
a b         -> s_0 -- a --> s_1        -> s_2 -- b --> s_3
            <!-- combine with epsilon -->
            -> s_0 -- a --> s_1 -- \epsilon --> s_2 -- b --> s_3
a | b       -> s_0 -- a --> s_1        -> s_2 -- b --> s_3
a | b       -
```