9 11

1. Operations of languages
   1. Intersections L1 ^ L2
   2. Concatenation
   3. Kleene closure
2. Powers of language
   1. Repeated concatenation
   2. Exponential concatenation
3. Kleene star- \*
   1. Kleene closure- 0 or more reps
   2. 4
4. All these correspond to finite automata
5. Regular languages
   1. Fix alphabet S, languages over it is defined
      1. Basis
         1. The sets phi, aeS (regular set)
      2. Recursive
         1. Combine regular sets
      3. Closure
         1. Only finite long combinations
   2. This also generates reg expressions
6. Implementing finite state automata
   1. Lots of choices- switch statements
   2. Regex describes a language
   3. Scanner- finds largest prefix that could be in language and sends to parser
      1. Sometimes have to look ahead
      2. 4tran example- do100i=1.50
         1. Do I set I to 50? Am I in a do loop?
         2. Gotta look ahead to be sure
7. Unix extensions
   1. Regex ubiquitous in unix
      1. Scanf
8. Finite automata
   1. Map from state to state
9. Nfa
   1. Can spontaneously transition
   2. 2 ways to think about it
      1. Infinite possible universes where all machines exist, all operating simultaneously, if one accepts, the other accept
      2. Precognition
   3. Really just help with simplifying complex dfa
10. Equivalence
    1. 2 automata accept exactly the same strings
11. Nfa->dfa
    1. Subset construction
    2. Find all possible subsets
    3. Simulating
       1. S0: m ->s0/s1
       2. S0L not m ->s0
    4. Combine states
12. FA limitations
    1. Can only count up to max amount of states
    2. Cant recognize binary strings with equal # 1’s and 0’s
    3. Cant recognize parenthesis balance
    4. CAN do it with stacks
       1. Push down autamata
13. Fa from RE
    1. Create mini FA’s for each basic RE operand
    2. Simple minded find NFA
    3. Convert via subset construction to DFA