Lecture #3 Regular Expressions / Regex.
DFA & NFA.

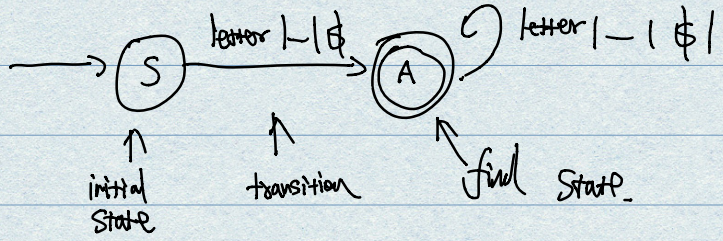
DFA that recognize JS identifiers

JS Id: sequence of 1+ letters or underscore or dollar sign.

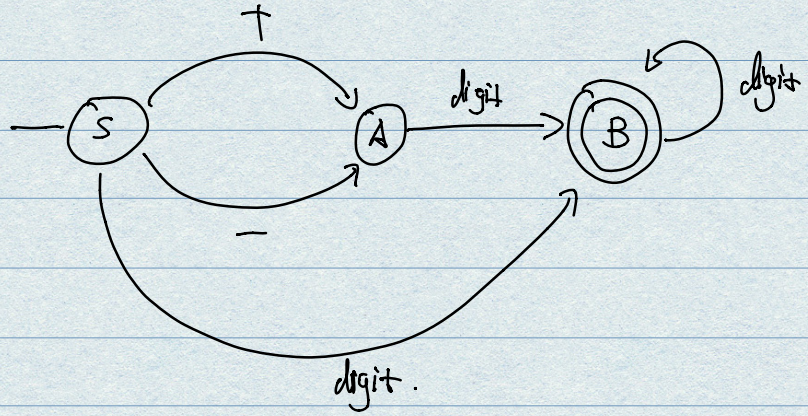
→ starting with a letter or underscore or dollar sign.

→ when a string is accepted:

fully consumed / automaton is in final state.



⇒ Another DFA.



- (1.34) → rejected.
- −4 → accepted
- 123 → accepted.

Implementation of DFA

- A DFA can only take one path through the state graph.

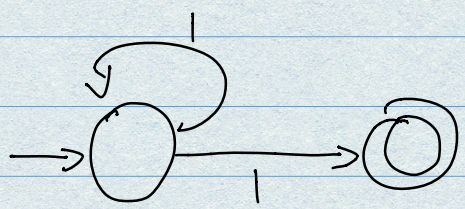
- Table based implementation:

```
nextState := transitions [ currentState , nextChar ].
```

NFA

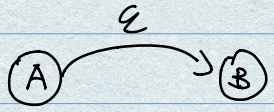
- Nondeterminism:

  when multiple choices exist, automaton "magically" guesses which transition to take so that the string can be accepted (if it is possible to accept the string).



input "11" can be in either state.

Epsilon Moves.



Move from state A to B without consuming an input character.

Execution of NFAs

→ NFA can choose

- whether to make ε- moves
- which of multiple transitions for a single character.

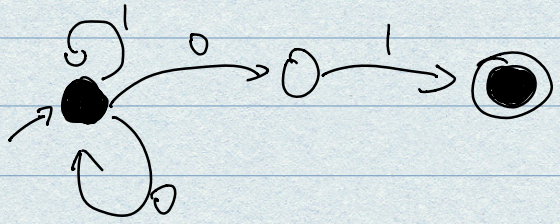
→ Emulate NFA $O(N \cdot S)$ time (S => # of States)

or convert to DFA : $O(N)$ matching time

Σ^S in size.

Acceptance of NFAs.

An NFA can get into multiple states.



Input : 1 0 1

Rule: NFA accepts if it <u>can</u> get into a final state.

Q: when in multiple states at once, how do we decide whether to accept.

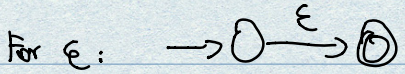
A: at least one of these states must be final.

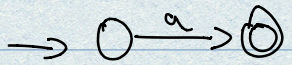
Compiling Regular Expressions to NFAs

=> For each kind of rexp, define an NFA.

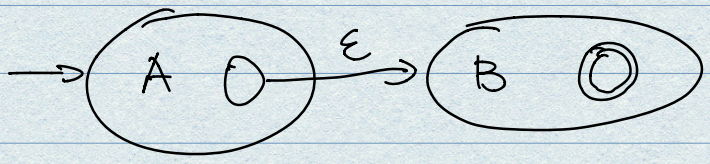
- Construct NFA for its constituents
- Notation: NFA for reg M.

=> Two base cases:

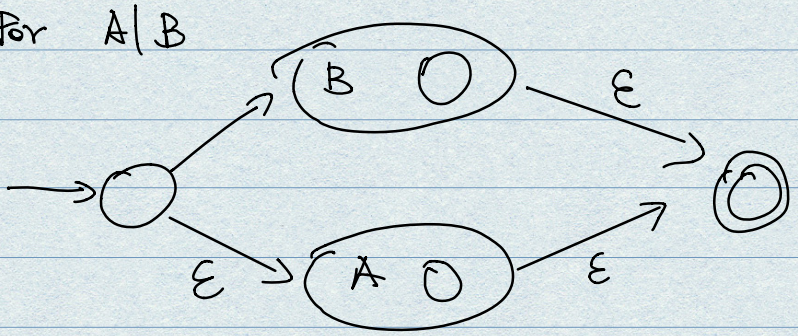
For ε: 

For literal character a: 

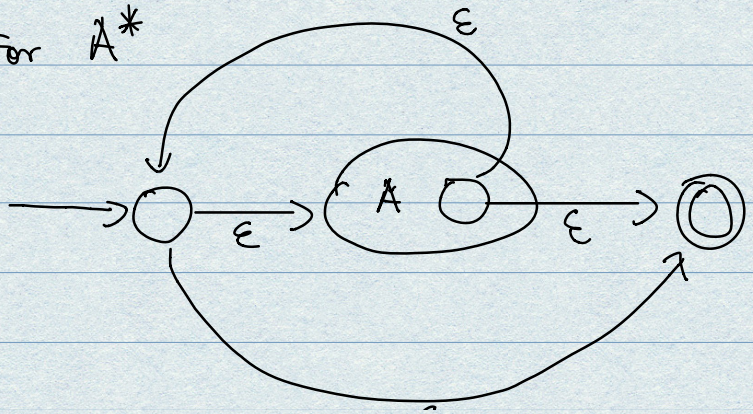
=> For A B A ∘ B.



=> For A|B

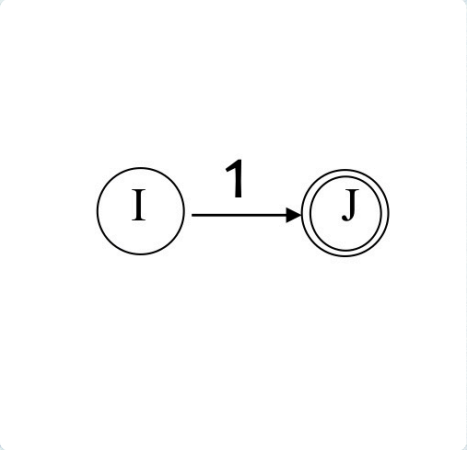


=> For A*

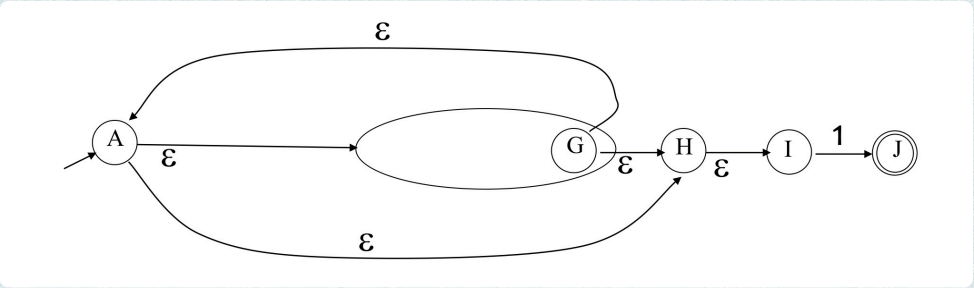


ε

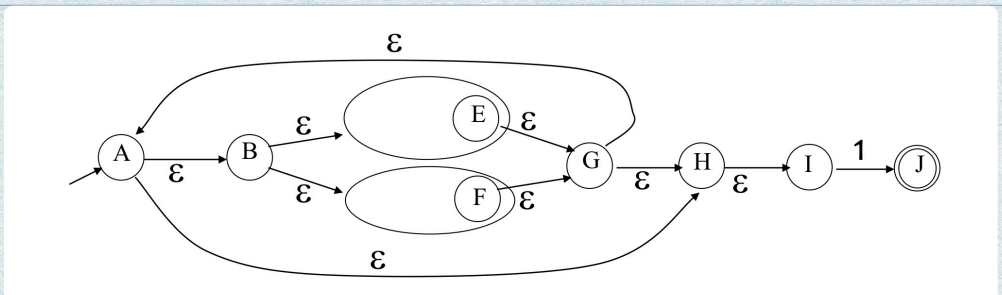
Ex] NFA of (1|0)*1



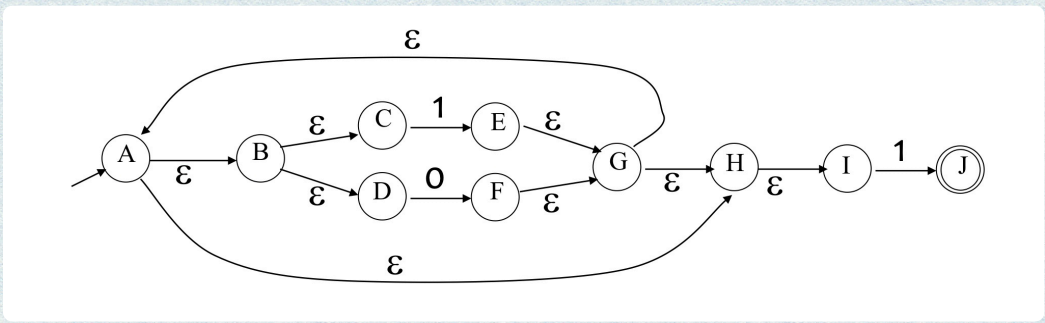
Consider (...1)



Consider (..)*1

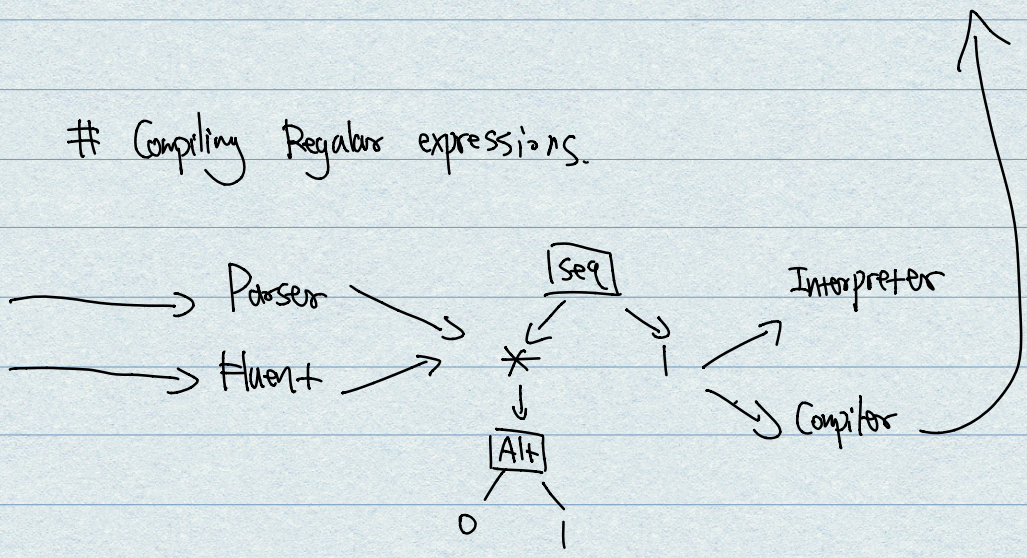


Consider (...|...)*1



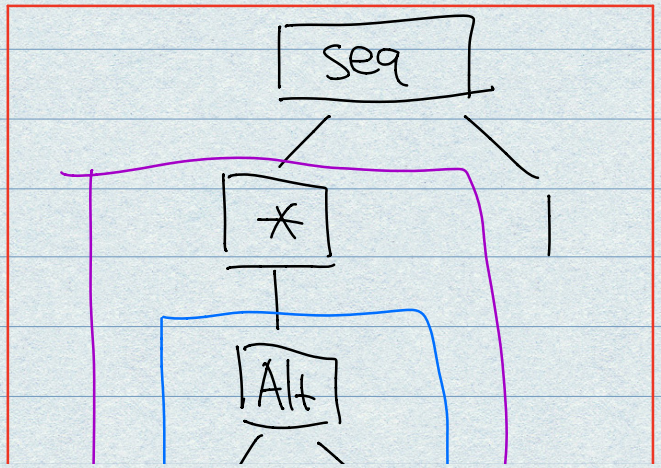
Consider $(1|0)^*1$

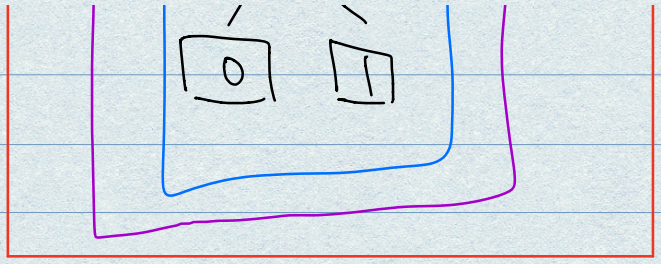
Compiling Regular expressions.

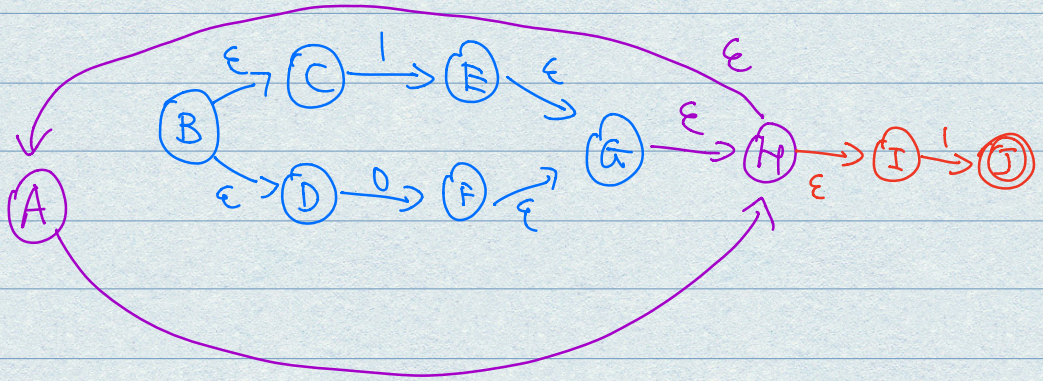


Recursively (Bottom Up) rewrite AST into Automaton.

$(1|0)^*1$




0
1


A
B
C
D
E
F
G
H
I
J
ε
1
0
ε
ε
ε
ε
ε
1