Kleene Star

- $L(\underbrace{a\,|b^*})$ = $\{a, b, bb, bbb, \cdots, \epsilon\}$
- $L((a|b)^*) =$ $\{a, aa, aaa, \cdots, b, bb, bbb, \cdots, \epsilon, ab, aab, aba, \cdots\}$
- $L\ (R^*) =$ $\{\epsilon\} \cup L(R) \cup \{L(R)L(R)\} \cup \cdots$

- Definition
  - $L^0(R) =$ $\{\epsilon\}$
  - $L^i(R) =$ $L^{i-1}(R)\,L(R)$
  - $L(R^*) =$ $\bigcup_{i\geq 0} L^i(R)$

Lexical Specification

234.5

- Express a numerical constants
  - 1, 2.3, 0.4, .5, 6E7, 8e-9 … 8e+9

$\underline{E7}$ E-7

digit → 0|1|2|…|9

integer → $\underline{\text{digit}}$ digit*

decimal → digit*$\underbrace{(.\ \text{digit} \mid \text{digit}.)}$ digit*

exponent → (e|E) (+|−|ε) integer

real → $(\underbrace{\text{integer exponent}} \mid \underbrace{\text{decimal (exponent} \mid \epsilon)})$

Computational Model

- id → letter (letter | digit)* ;
- How can we check the grammar **automagically?**

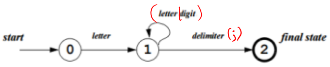
Finite Automata

id → letter (letter|digit)*;

```
state0:
        c = getchar();
        if (isalpha(c)) goto state1;
        error();

state1:
        c = getchar();
        if (isalpha(c) || isdigit(c)) goto state1;
        if (isdelimiter(c)) goto state2;
        error();

state2:
        return();
```



Finite Automata

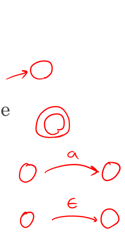
- A finite automata consists of
  - An input alphabet Σ
  - A set of states S
  - A start state
  - A set of accepting states F ⊆ S
  - A set of transitions state $\rightarrow^{\text{input}}$ state

Finite Automata

- Transition
  - $s_1 \ \rightarrow^a \ s_2$
- Is read:
  - In state s_1 on input ‘a’ go to state s_2
- If end of input and in accepting state: accept
- Otherwise: reject

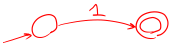
Finite Automata State Graphs

- A state
- The start state
- An accepting state
- A transition
- An epsilon move

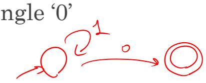


Examples

- A finite automaton that accepts only ‘1’

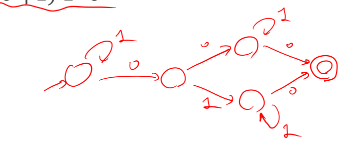


- A finite automaton accepting any number of ‘1’s followed by a single ‘0’



Examples

- Alphabet {0,1}
- $1^*0\ (0\ |\ 1)\ 1^*\ 0$



Big Picture

- Regular Expression
- ➔Algorithm

```
state0:
        c = getchar();
        if (isalpha(c)) goto state1;
        error();
```

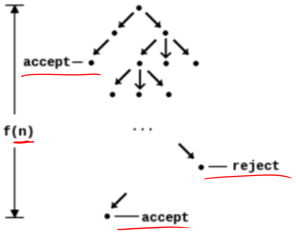
- Regular Expression
- ➔NFA
- ➔DFA
- ➔Algorithm

```
state1:
        c = getchar();
        if (isalpha(c) || isdigit(c)) goto state1;
        if (isdelimiter(c)) goto state2;
        error();
```

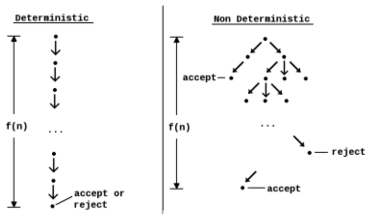
Deterministic and Nondeterministic

- Deterministic Finite Automata (DFA)
  - One transition per input per state
  - No ε-moves
  - Can take only one path through the state graph
- Nondeterministic Finite Automata (NFA)
  - Can have multiple transitions for one input in a given state
  - Can have ε-moves
  - Which of multiple transitions for a single input to take

A Function (Accept or Reject)



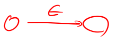
Deterministic and Nondeterministic



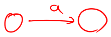
Regular Expressions to NFA

- For each kind of regular expression, define an NFA
- Notation: NFA for regular expression M

- For ε

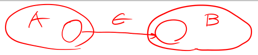


- For input a

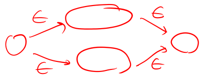


Regular Expressions to NFA

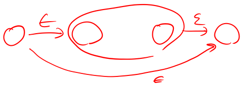
- For AB



- For A | B

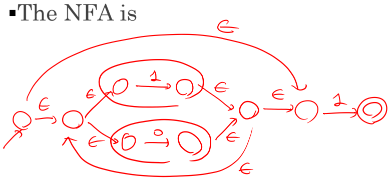


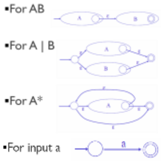
- For A*



Regular Expressions to NFA

- Consider the regular expression
  - (1|0)*1
- The NFA is




- For AB
- For A | B
- For A*
- For input a