CSE 425 / L-17 / 28.10.2024 /

⊛ if-else ambiguity:

<if-stmt> ⟶ if (<logic-expression>) <stamt> |

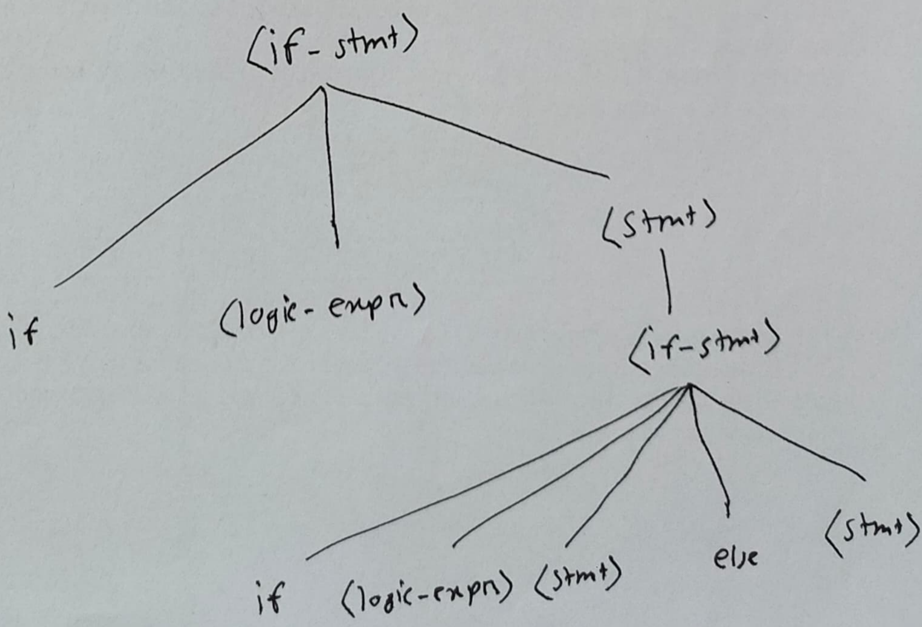
if (< logic-expression>) <stmt> else <stmt>

<stmt> ⟶ <if-stmt> | ... ...

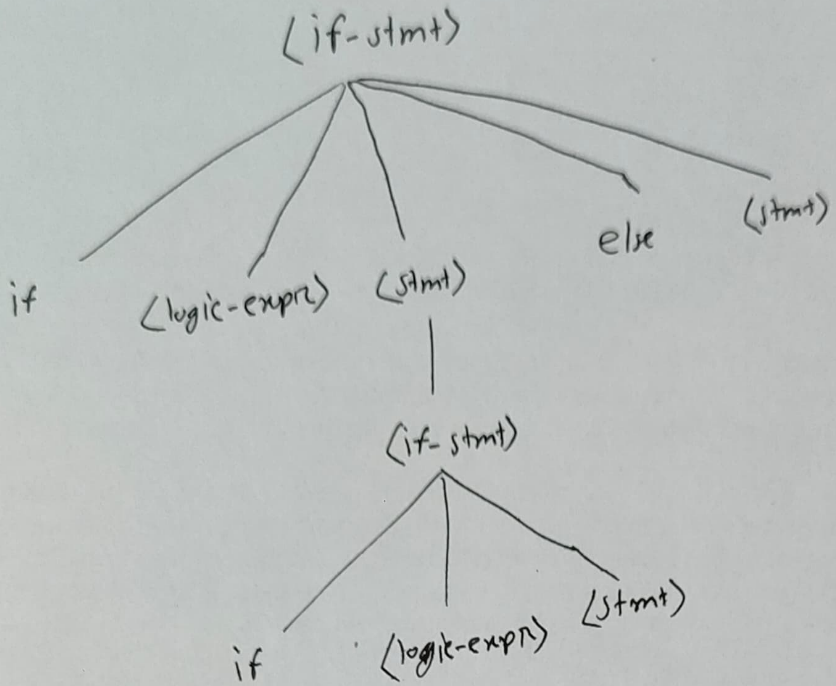
⇒ language constructs

```
if ( done == TRUE)
if ( denom == 0)
      qutient = 0;
else
      qutient = num/denom ;
```

⊛ Parse tree:



⊛ Another parse tree:



⇒ Therefore, ambiguity exist.

⊛

```
if (sum == 0)
    if (count == 0)      ] then
        result = 0;
    else  result = 1;    ] else
```

then

two possibilities

- known as Dangling if-else.

⊛ Lexical Analyzer:

- Regular language (Regular grammar)
  - vocab
    - we define all the allowed "words" in a source code.

↳ collection of allowed alphabets. Σ (sigma) $= \Sigma\{a, b\}$

Background:

Finite-state Machine

- Finite Automata

with output

without output

- specialize machine, tune for language recognition.
- machine have finite states.

→ String starts traversing the machine states and are accepted if it reaches the final state.

Definition:

A string x is recognized by the machine $M = (S, I, f, S_0, F)$ if it takes the initial states S_0 of the machine to one of Final states (F).

→ S : set of states

→ I : set of allowed alphabet

→ f : Transition function

$f: S \times I \rightarrow S$

$S = \{S_0, S_1\}$

$I = \{a, b\}$

$S \times \Sigma = \{(S_0, a), (S_0, b), (S_1, a), (S_1, b)\}$

Midterm upto this

Next class

L-18/30.10.2024

Midterm Exam - No content

L-19 / 04.11.2024 /

Lexical Analyzer

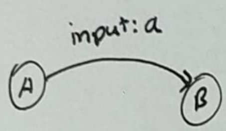
- Finite State Machine - with no output

$M = (S, \Sigma, f, S_0, F)$

S → set of state

Σ → set of Alphabets

f → transition function

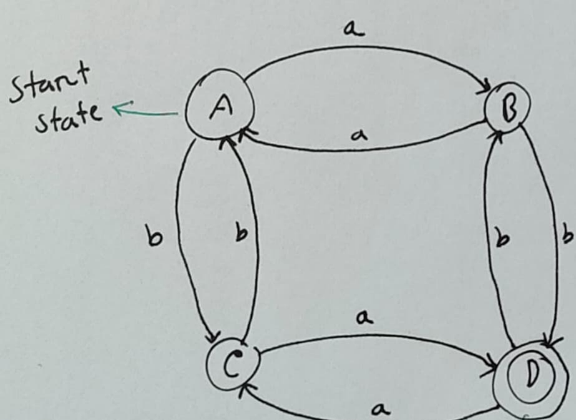


$\Rightarrow (A, a) \Rightarrow B$

transition function:

$f : (S \times \Sigma) \rightarrow S$

$\Rightarrow$



$S = \{A, B, C, D\}$

$F = \{D\}$

$S_0 = \{A\}$

$\Sigma = \{a, b\}$

$f : S \times \Sigma \rightarrow S$

Final State

number of final state can be as many as required.

$a = ?$ → invalid

$ab = ?$ → valid

abbab ⇒ stuck in C state - invalid.

State transition table:

	Next State	
State	a	b
A	B	C
B	A	D
C	D	A
D	C	B

for multiple path
A, B

kleene Closure:

- helps us to produce the vocabulary.
  - set of all possible patterns.
    - allowed alphabets
    - I or Σ

- concatenation:

$$A = \{0, 11\}$$

$$B = \{1, 10, 110\}$$

$$AB = \{01, 010, 0110, 111, 1110, 11110\}$$

$$BA = \{10, 111, 100, 1011, 1100, 11011\}$$

$$A^{0} = \{\in\}$$

$$\underbrace{A^{n+1} = A^{n} \cdot A}_{\text{concatenation.}} \quad ; \text{ for } n = 0, 1, 2, 3, \ldots$$

* Given,

$A = \{1, 00\}$

$A^3 = ?$

$A^0 = \{\epsilon\}$

$A^1 = \{1, 00\}$

$A^2 = A^1 A$

$= \{1, 00\} \{1, 00\}$

$= \{11, 100, 001, 0000\}$

$A^3 = A^2 \cdot A$

$= \{11, 100, 001, 0000\} \{1, 00\}$

$= \{111, 1100, 1001, 10000, 0011, 00100, 00001, 000000\}$

* <u>Definition:</u>

kleene Closure

- Suppose A is the subset of V^*

then, kleene closure of A, denoted as A^*, is the set consisting of all concatenation of arbitary many strings from A. So,

$$A^* = \bigcup_{k=0}^{\infty} A^k$$

$A = \{0\}$

$A^* = \{0^n \mid n = 0, 1, 2, 3, \ldots\}$

produce any copies of 0.

Given,

$$B = \{0, 1\}$$

$$B^* = \bigcup_{k=0}^{\infty} B^k \text{ is the kleene Closure}$$

So, calculate B^* for $k = 2$

$$B^* = \bigcup_{k=0}^{2} B^k = B^0 \cup B^1 \cup B^2$$

$$= \{\epsilon\} \cup \{0, 1\} \cup \{00, 01, 10, 11\}$$

$k = 3$?

$$B^3 = B^2 \cdot B$$

$$= \{00, 01, 10, 11\} \{0, 1\}$$

$$= \{000, 001, 010, 011, 100, 101, 110, 111\}$$

$$B^* = B^0 \cup B^1 \cup B^2 \cup B^3$$

$$= \{\epsilon\} \cup \{0, 1\} \cup \{00, 01, 10, 11\} \cup \{000, 001, 010, 011, 100, 101, 110, 111\}$$

Ans.

L-20 / 06.11.2024 /

Lexical Analyzer:

- Regular language

  - Regular expression
  - Regular grammar

  } Vocabulary
  - kleene Closure

Finite state machine

obtain state transition table

Hand Code ⇒ C/C++

$M = (S, \Sigma, f, S_0, F)$

Implementation:

- use tools:
  - Flex
  - lex

Regular Sets:

- can be formed using
  - concatenation
  - union
  - kleene Closure

- kneth rosen book
- Python Package
  - regex.py

- we can define regular expression
  - ϕ is regular expression
  - AB, (A+B) and A* are regular expression

ACI, BCI

Draw a finite state machine that accept the given string pattern?

ab^* ⇒ single "a" followed by any number of "b".

- ab^0, ab, abb, $abbb$ ...

$(ab)^*$ ⇒ $(ab)^0$, ab, $abab$ ....

$(b+ba)$ ⇒ b or ba

$b(b+a)^*$ ⇒ any string that starts with "b"

- $\left.\begin{matrix} baa \\ bab \\ bba \\ bbb \end{matrix}\right\}$ any pattern possible but starts with b.

$(b^* a)^*$ ⇒ ba

baba

CS143: Lecture 3 Lexical Analysis

Grammar:

→ Regular

Right Regular

$S \to a \mid aP \mid \epsilon$

$P \to$ ...

Left Regular

$S \to a \mid Pa \mid \epsilon$

$P \to$ ...

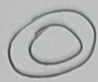
$\gamma \to \alpha$

→ can be terminal or non-terminal with a mandatory presence of non-terminal.

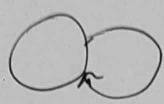
→ string

Notation:

State : 

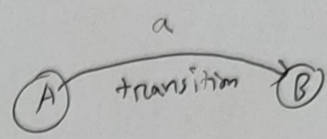
Final State : 

$\underbrace{\text{Final State}}_{\text{Accepting State}}$

Trap State : 

no farther transition.

transition:



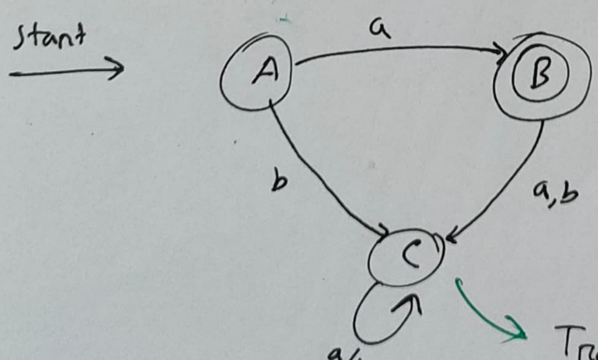
$(\underbrace{A}_{\text{initial}}, \underbrace{a}_{\text{input}}) \to \underbrace{B}_{\text{destination}}$

Finite State machin:

- Deterministic Finite Automata (DFA)
- Non-deterministic Finite Automata (NFA)

M: accepts only "a"

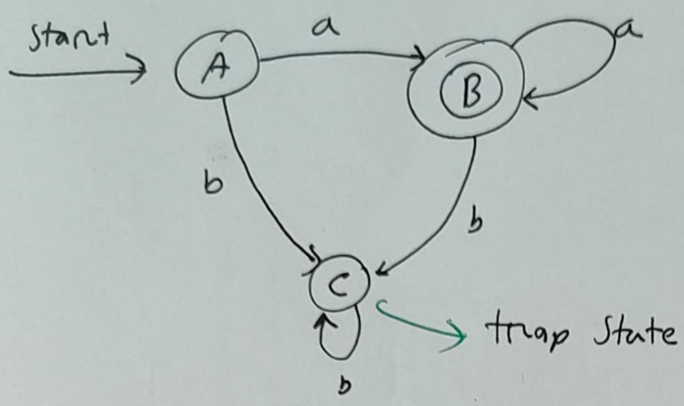
⇒



Trap State

⊛ M: accepts any number of "a"

⇒

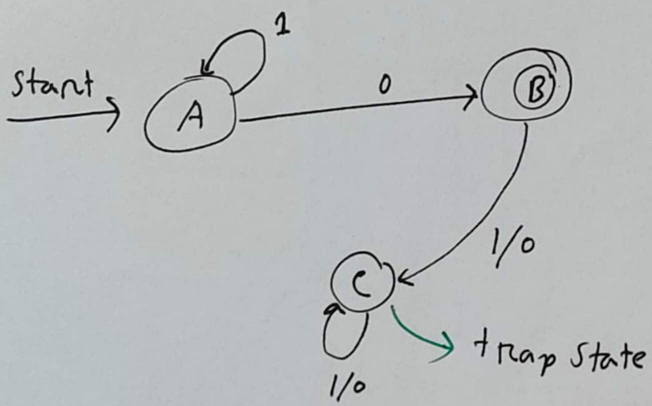


⊛ Any number of 1's followed by a single "0"

$\Sigma = \{0, 1\}$

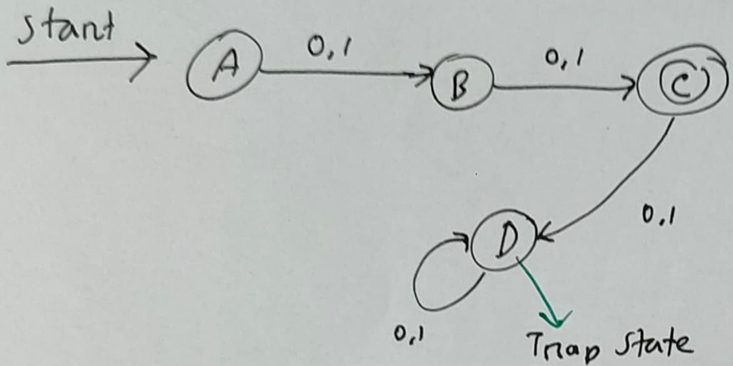
⇒ Sample: 10
110
1110
11110

⇒



* Construct a machine that accepts string of length 2 for $\Sigma = \{0,1\}$

⇒



$\Sigma^{n+1} = \Sigma^n \cdot \Sigma$

$\Sigma^2 = \Sigma \cdot \Sigma$

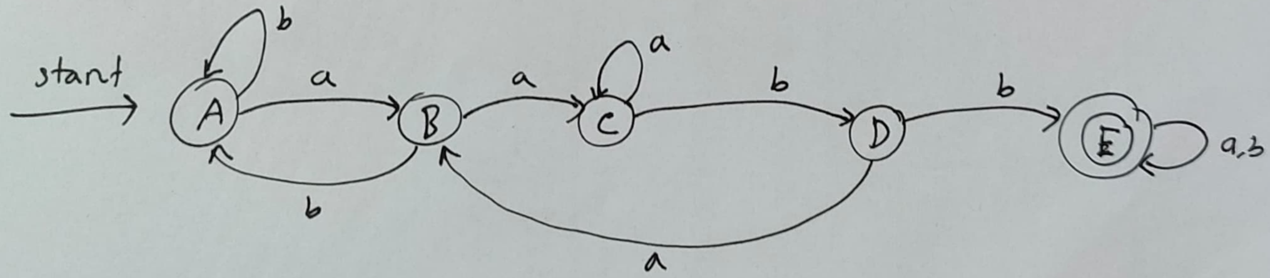
$= \{0,1\}\{0,1\}$

$= \{00, 01, 10, 11\}$

* Design a DFA that accepts any string with aabb in it.

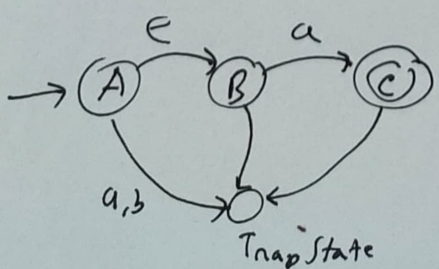
$\Sigma = \{a,b\}$

⇒



* NFA : Non-deterministic Automata.

(i) e transition



(ii) Single input leads to multiple destination