Compiler Construction

Lexical Analysis

Chapter -3

Regular Expressions

1. **All Strings that start with “tab” or end with “bat”**
   tab{A,…,Z,a,...,z}*|{A,…,Z,a,....,z}*bat

2. **All Strings in Which Digits 1,2,3 exist in ascending numerical order:**
   {A,…,Z}*1 {A,…,Z}*2 {A,…,Z}*3 {A,…,Z}*

3. **All strings of lowercase letters in which the letters are in ascending lexicographic order.**

   a*b*c*………z*

Regular Expression

All strings of lowercase letters that contain the five vowels in order.

want -> other* a (other|a)* e (other|e)* i (other|i)* o (other|o)* u (other|u)*

other -> [bcdfghjklmnpqrstvwxyz]

All strings of a's and b's that do not contain the substring abb.

b*(a^+b?)*

All strings of a's and b's that do not contain the subsequence abb.

b* | b*a^+ | b*a^+ba*

Regular Expressions

1. **All strings of lowercase letters that contain the five vowels in order**

   C → b|c|d|f……..|z
   C*a(C|a)*e(C|e)* i (C|i)* o (C|o)* u(C|u)*

2. **All strings of a’s and b’s with an odd number of a’s**

   b* (ab*ab*)*ab*

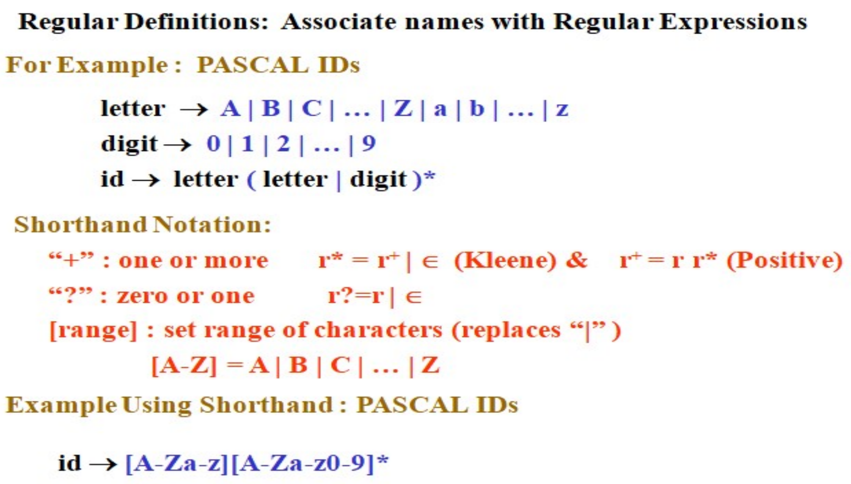
3. **All stirings of 0’s and 1’s in which any two 0’s in α are separated by three 1’s**

   1*(0111)*01* + 1*

Find Regular Languages

1. a(a|b)*a
2. ((ε|a)b*)*
3. (a|b)*a(a|b)(a|b)
4. a*ba*ba*ba*
5. (aa|bb)* ((ab|ba)(aa|bb)*(ab|ba)(aa|bb)*)*

Regular Definitions



Regular Definitions: Associate names with Regular Expressions

For Example : PASCAL IDs

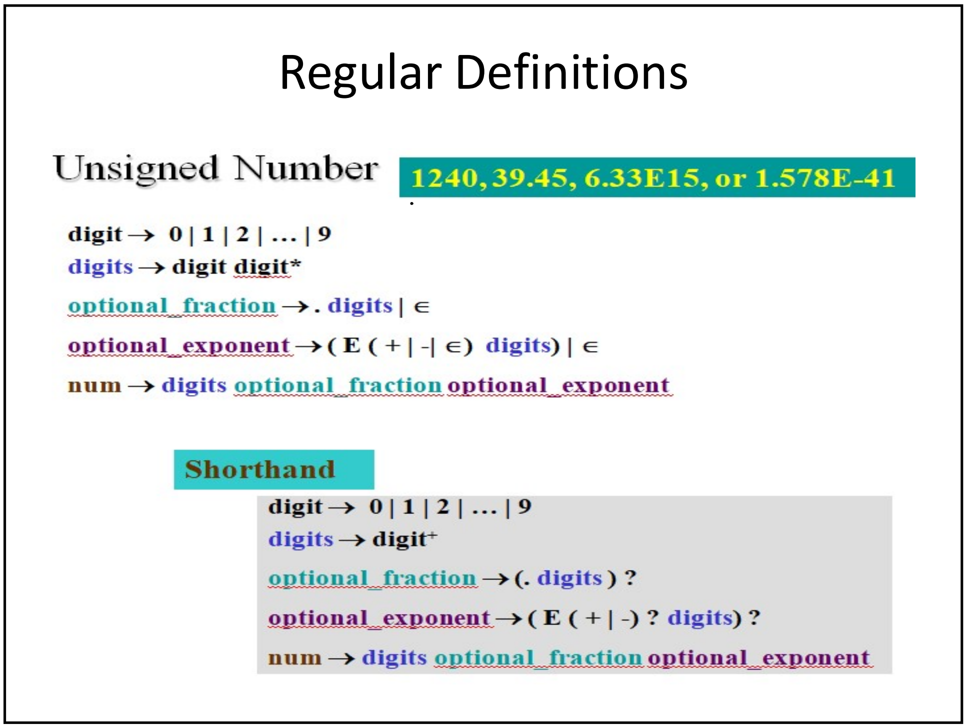
letter → A | B | C | … | Z | a | b | … | z
digit → 0 | 1 | 2 | … | 9
id → letter (letter | digit)*

Shorthand Notation:

"+" : one or more $r^* = r^+ | \in$ **(Kleene) &** $r^+ = r\, r^*$ **(Positive)**
"?" : zero or one $r? = r | \in$
[range] : set range of characters (replaces "|")
[A-Z] = A | B | C | … | Z

Example Using Shorthand : PASCAL IDs

id → [A-Za-z][A-Za-z0-9]*



Regular Definitions

Unsigned Number **1240, 39.45, 6.33E15, or 1.578E-41**

digit → 0 | 1 | 2 | … | 9

digits → digit digit*

optional_fraction → . digits | ∈

optional_exponent → (E (+ | -| ∈) digits) | ∈

num → digits optional_fraction optional_exponent

Shorthand

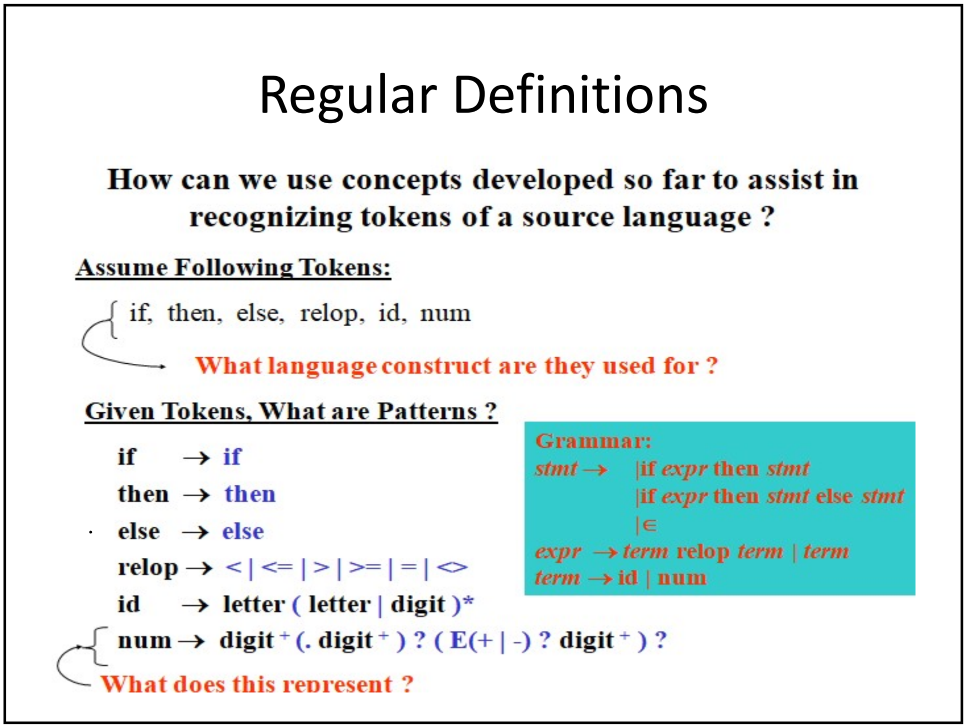
digit → 0 | 1 | 2 | … | 9

digits → digit$^+$

optional_fraction → (. digits) ?

optional_exponent → (E (+ | -) ? digits) ?

num → digits optional_fraction optional_exponent



Regular Definitions

How can we use concepts developed so far to assist in recognizing tokens of a source language ?

Assume Following Tokens:

if, then, else, relop, id, num

What language construct are they used for ?

Given Tokens, What are Patterns ?

if → if

then → then

else → else

relop → < | <= | > | >= | = | <>

id → letter (letter | digit)*

num → digit$^+$ (. digit$^+$) ? (E(+ | -) ? digit$^+$) ?

What does this represent ?

Grammar:

stmt → |if *expr* then *stmt*

|if *expr* then *stmt* else *stmt*

|∈

expr → *term* relop *term* | *term*

term → id | num

Other tasks done by Lexical Analyzer

Scan away *blanks*, new lines, tabs

Can we Define Tokens For These?

blank	→	**blank**
tab	→	**tab**
newline	→	**newline**
delim	→	**blank | tab | newline**
ws	→	**delim** $^{+}$

Ans: No token is returned to parser

Pattern, Token, Attribute-Value

Regular Expression	Token	Attribute-Value
ws	-	-
if	**if**	-
then	**then**	-
else	**else**	-
id	**id**	**pointer to table entry**
num	**num**	**Exact value**
<	**relop**	LT
<=	**relop**	LE
=	**relop**	EQ
<>	**relop**	NE
>	**relop**	GT
>=	**relop**	GE

Note: Each token has a unique token identifier to define category of lexemes

(DFA)

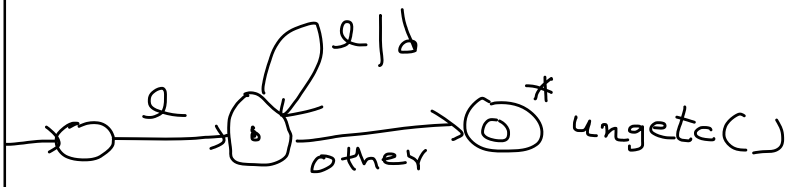
Transition Diagrams

For Example : PASCAL IDs

letter → A | B | C | ... | Z | a | b | ... | z

digit → 0 | 1 | 2 | ... | 9

id → letter (letter | digit)*



Transition Diagrams

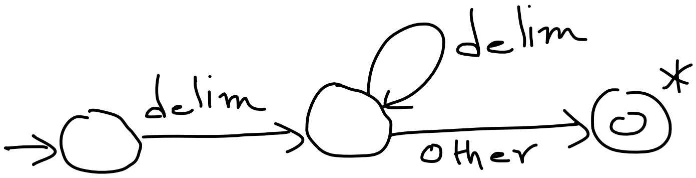
blank → blank

tab → tab

newline → newline

delim → blank | tab | newline

ws → delim $^+$

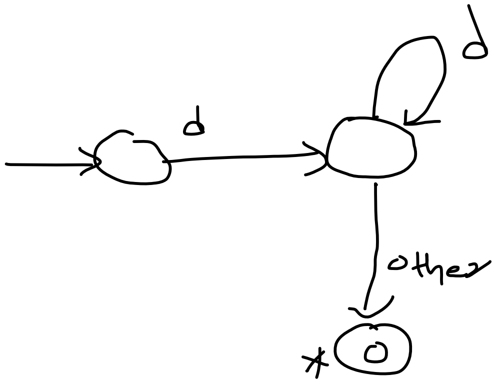


Transition Diagrams

digit → 0 | 1 | 2 | … | 9

digits → digit digit*

Num → digits



Transition Diagrams

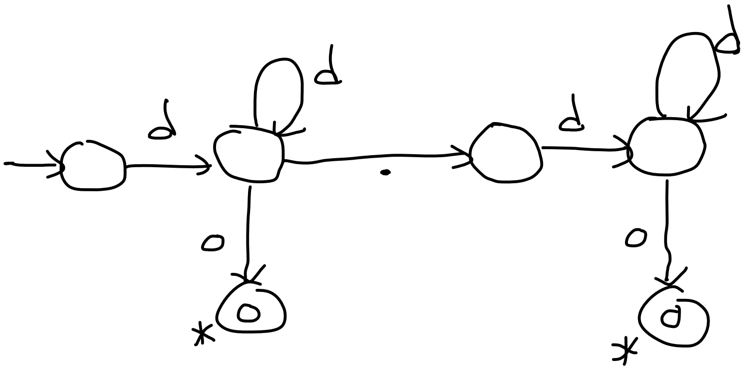
digit → 0 | 1 | 2 | … | 9

digits → digit digit*

optional_fraction → . digits | ∈

Num → digits optional_fraction

12.45



Transition Diagrams

Unsigned Number

1240, 39.45, 6.33E15, or 1.578E-41

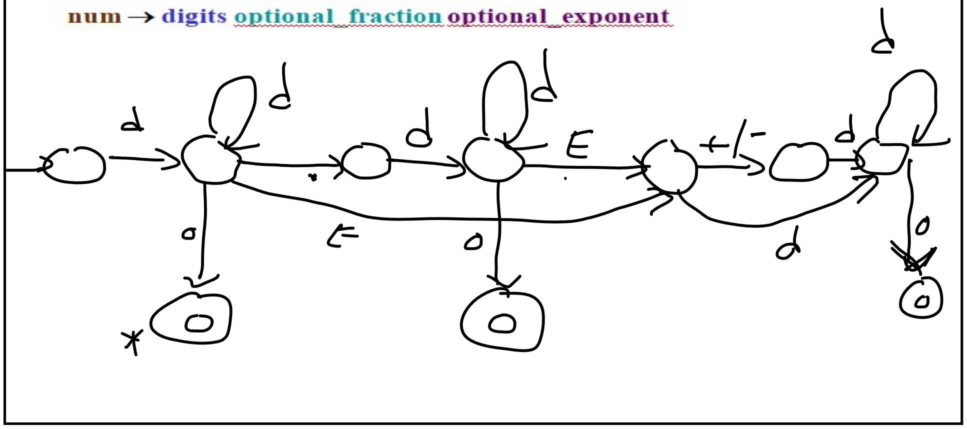
digit → 0 | 1 | 2 | ... | 9

digits → digit digit*

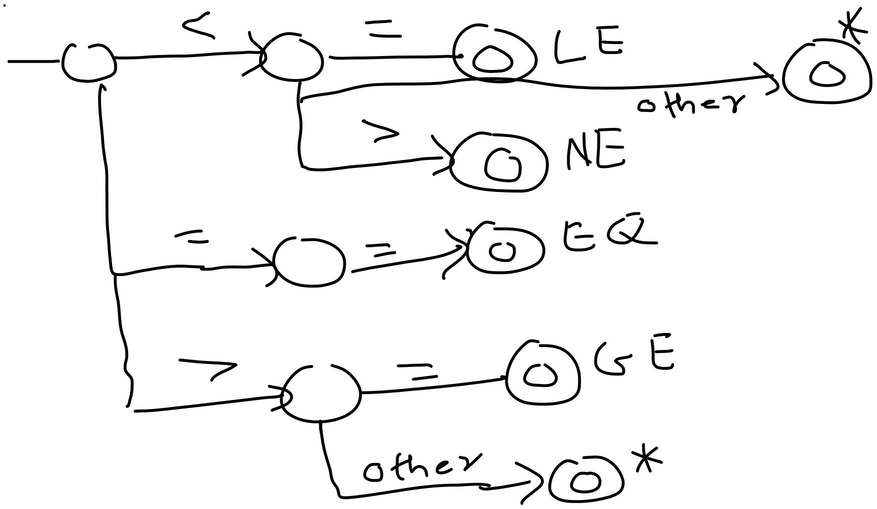
optional_fraction → . digits | ∈

optional_exponent → (E (+ | -| ∈) digits) | ∈

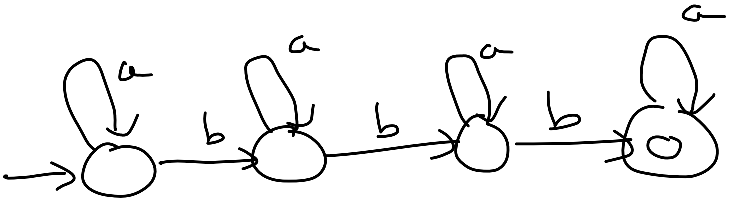
num → digits optional_fraction optional_exponent



Transition Diagrams (Relational Operators)

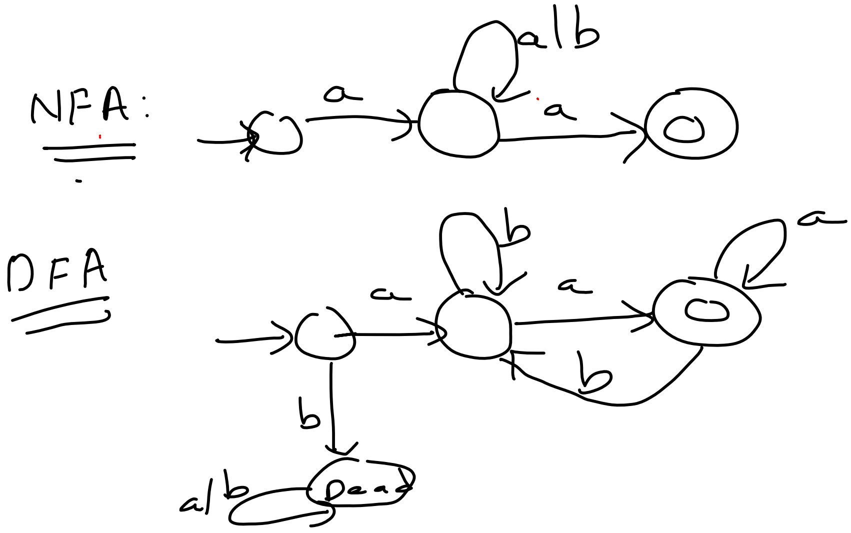


Transition Diagram
a*ba*ba*ba*



DFA

Transistion Diagram
a(a|b)*a



Transition Diagram (DFA)

(a|b)*a(a|b)(a|b)

Try Yourself

Transition Diagram (DFA)

(aa|bb)* ((ab|ba)(aa|bb)*(ab|ba)(aa|bb)*)*

Try Yourself