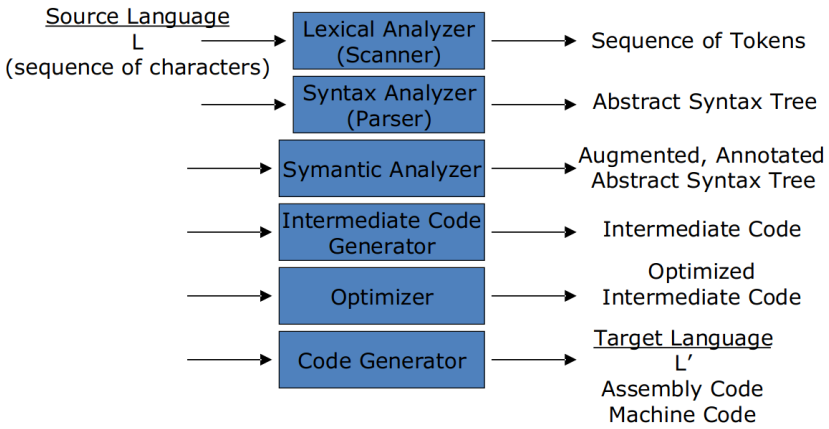
1. Flow chart



Scanner :

to tokens

Parser :

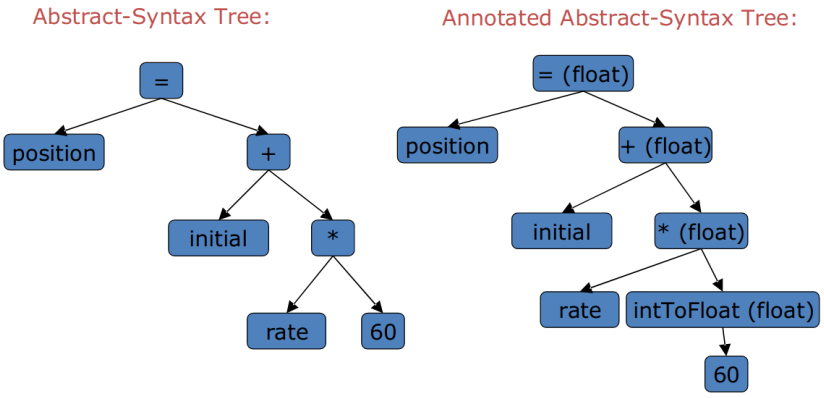
variables that are multiply declared.

might generate code, or build some intermediate representation of the program such as an abstract-syntax tree

Semantic analyzer:

Checks for (more) "static semantic" errors

Annotate and/or change the abstract syntax tree



Intermediate Code Generator:

Translates from abstract-syntax tree to intermediate code

One possibility is 3-address code

each instruction involves at most 3 operands

Example:

temp1 = inttofloat(60)
temp2 = rate * temp1
temp3 = initial + temp2
position = temp3

Optimizer :

Tries to improve code to run faster , be smaller and consume less energy

mainly includes;

Constant Propagation
Constant Propagation
Algebraic Simplification
Copy Propagation
Common Sub-expression Elimination
Dead Code Elimination
Loop Invariant Removal
Strength Reduction

Code Generator:

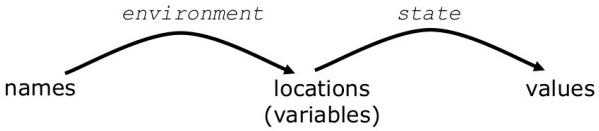
to assembly code

2.Symbol Tables

Keep track of names declared in the program
Separate level for each scope
Used to analyze static symantics:
–Variables should not be declared more than once in a scope
–Variables should not be used before being declared
–Parameter types for methods should match method declaration

3.Programming language basics

Static/Dynamic Distinction
-static : in space(initialized when the program start)
-dynamic: in time(when the code execute this code)
Environments and States



Static Scope and Block Structure
-declaration works in the most sub-block
Explicit Access Control
-class member access control
Dynamic Scope
Parameter Passing
-copy by value
-copy by reference
Aliasing
-refer to the same memory

4.make file

<target> : <dependency list>
<tab><command to satisfy target>
example:

```
JC = ~/usr/local/jdk/javac
JFLAG = -g

Example.class : Example.java IO.class
    $(JC) $JFLAG Example.java

IO.class : IO.java
    $(JC) $JFLAG IO.java

clean:
    rm -rf  *.class
test:
    java -cp . Test.class
```

5. FSA (finite state automata)

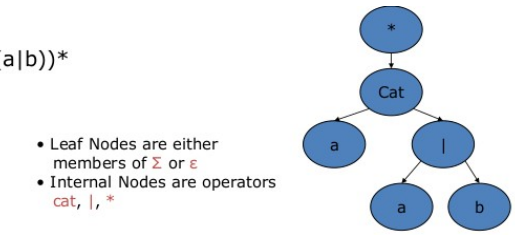
$Q(a finite set of states)$
$\sum(alphabet)$
$\delta(state transition func)$
$q(the start state)$
$F(the set of final states)$

RE (regular expression) priority: * > cat > |
NFA (non-deterministic)
DFA (deterministic)

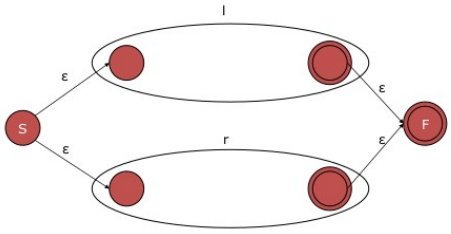
6. RE → NFA

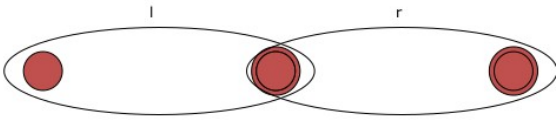
- create a tree from regular expression
- do a post-order Traversal of tree

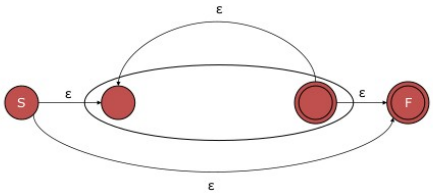
(a(a|b))*



- Leaf Nodes are either members of Σ or ε
- Internal Nodes are operators cat, |, *







7. NFA → DFA

Initially ε-closure(s) is the only state
in DFA and it is unmarked
While (there is unmarked state T in DFA)
    mark T;
    for (each input symbol a) {
        ε-closure(move(T,a));
        if (U not in DFA)
            add U unmarked to DFA
        transition[T,a]=U;

8. JLEX

Symbol	Meaning in RE	Meaning in Char Class
(	Matches with) to group sub-expressions.	Represents itself.
)	Matches with (to group sub-expressions.	Represents itself.
[	Begins a character class.	Represents itself.
]	Is illegal.	Ends a character class.
{	Matches with } to delimit a macro name.	Matches with } to delimit a macro name.
}	Matches with { to delimit a macro name.	Represents itself or matches with { to delimit a macro name.
"	Matches with " to delimit strings (only \ is special within strings).	Matches with " for a string of chars that belong to the char class. Only \" is special within the string.
\	Escapes special characters (n, t, etc). Also used unicode/hex/octal.	Escapes characters that are special inside a character class.
.	Matches any character except newline	Matches itself
|	Or	Matches itself
*	Kleene Closure	Matches itself
+	One or more matches	Matches itself
?	Zero or One matches	Matches itself
^	Matches only at beginning of line	Complements chars in class
$	Matches only at end of line	Matches itself
-	Matches ifself	Range of characters

```
//User Code Section (uninterpreted  java code)

%%

//Directives Section

DIGIT = [0-9]
LETTER = [a-zA-Z]
WHITESPACE = [\040\t\n]

%state SPECIALINTSTATE

//Configure for use with java CUP (Parser generator)
%implements java_cup.runtime.Scanner
%function next_token
%type java_cup.runtime.Symbol

//End of file behavior
%eofval{
System.out.println("All done");
return null;
%eofval}

//Turn on line counting
%line

%%

//Regular Expression rules
```

Macro definitions

State declaration

9. The use of ^

用法一：限定开头

文档上给出了解释是匹配输入的开始，如果多行标示被设置成了true，同时会匹配后面紧跟的字符。比如 /^A/会匹配"An e"中的A，但是不会匹配"ab A"中的A

用法二：（否）取反

当这个字符出现在一个字符集合模式的第一个字符时，他将会有不同的含义。

比如： /[^a-z\s]/会匹配"my 3 sisters"中的"3"　这里的"^"的意思是字符类的否定，上面的正则表达式的意思是匹配不是（a到z和空白字符）的字符。

10.CFG

-RE and FSA can not express all languages.
-**leftmost** and **rightmost** derivation(parse trees)

11.SDT(syntax-directed translation)

-translation is the value of the input
-translation is a string of ids
-rules can have conditions

12.AST(abstract syntax trees) (nodes based)

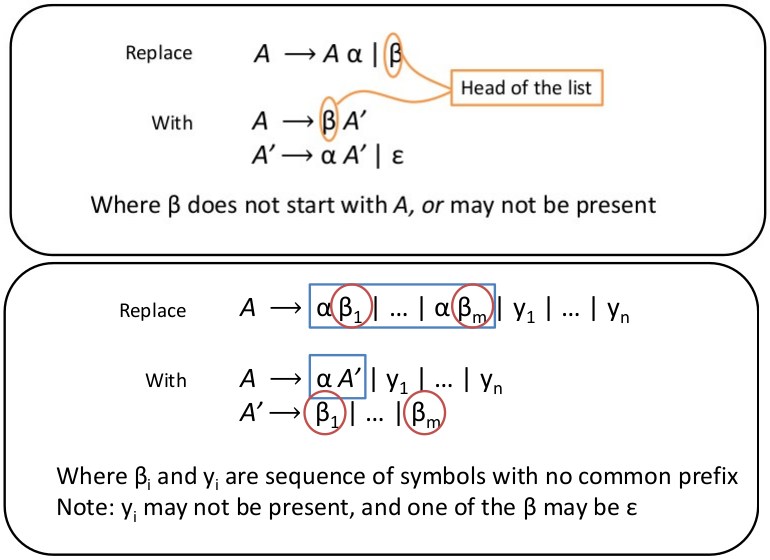
-make a parse tree

-use the parse tree to build an AST

-we construct **a new node object,** which becomes the value of LHS.trans

-populate the **node's fields with the translations of** the RHS nonterminals

13.LL



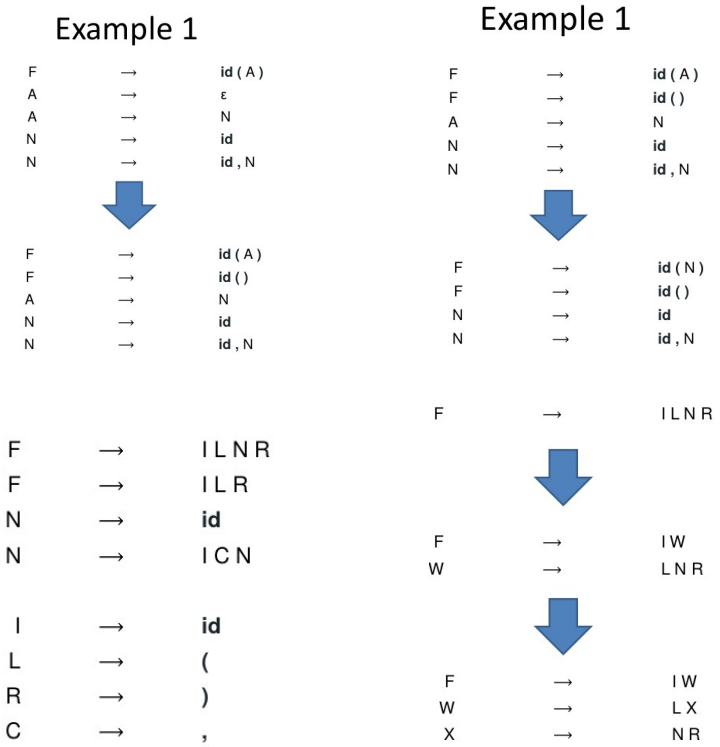
14.CNF(Chomsky Normal Form)

-消除 ϵ

-合并相同状态

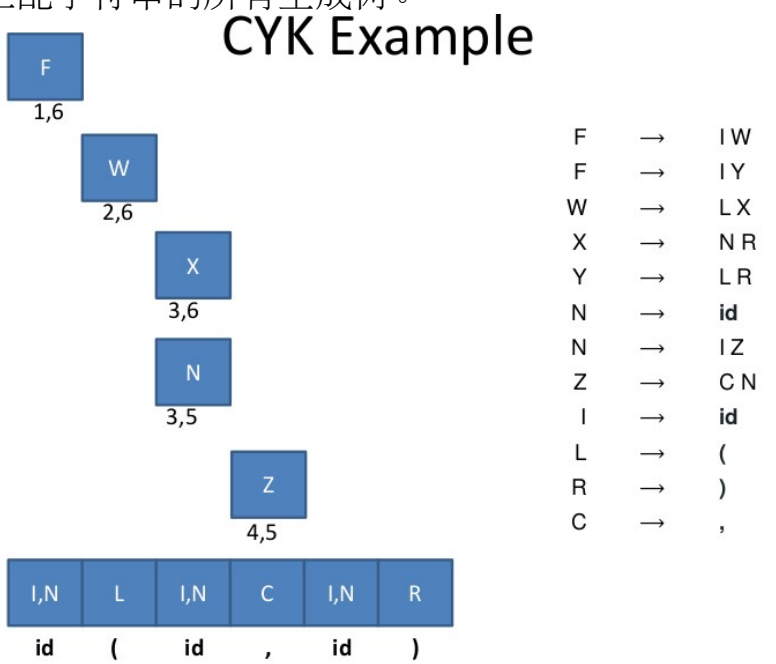
-将终结符设置为非终结符

-将 trans 长度大于 2 的变成长度为 2



15.CYK

-对每一个长度的子串，找到可以与之相匹配的树，从下往上，如果最后能到 S，则找到了匹配字符串的所有生成树。



16.Java cup

-*input* ：jlex’s output

-*output:*

parser.java returns a Symbol whose value field contains the translation of the root non-terminal

sym.java returns each terminal declared in the java cup