Course 2

Algorithm Scanning v2

```
While (not(eof)) do
    detect(token);
    if token is reserved word OR operator OR separator
         then genFIP(token, 0)
         else
        if token is identifier OR constant
             then index = pos(token, ST);
                  genFIP(token, index)
             else  message "Lexical error"
        endif
     endif
endwhile
```

Remarks:

- genPIF = adds a pair (token, position) to PIF

- Pos(`token`,ST) – searches *`token`* in symbol table S*T*; if found then return position; if not found insert in SR and return position
- Order of classification (reserved word, then identifier)
- If-then-else imbricate => detect error if a token cannot be classified
- Scanning also eliminates spaces (all) and comments

Symbol Table

Definition *= contains all information collected during compiling regarding the* *<u>symbolic names</u>* *from the source program*

identifiers, constants, etc.

Variants:

- Unique symbol table – contains all symbolic names
- distinct symbol tables: IT (identifiers table) + CT (constants table)

ST organization

Remark: search and insert

1. Unsorted table – in order of detection in source code O(n)
2. Sorted table: alphabetic (numeric) O(lg n)
3. Binary search tree (balanced) O(lg n)
4. Hash table O(1)

Hash table

- K = set of keys (symbolic names)
- A = set of positions ($|A| = m$; m –prime number)

$h : K \rightarrow A$

$$h(k) = (\mathrm{val}(k) \bmod m) + 1$$

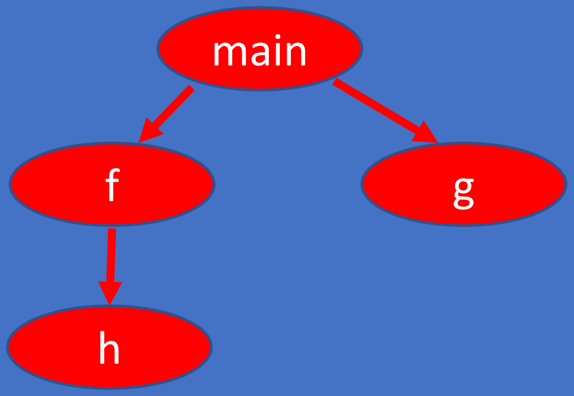
- Conflicts: $k_1 \neq k_2$, $h(k_1) = h(k_2)$

Methods to solve conflicts?
Trick: **Open** (linked list)/
closed (next free pos) bucket

Visibility domain (scope)

- Each scope – separate ST
- Structure -> inclusion tree

Hierachical structure of STs:



Example:

```
Int main(){
… int a;

void f()
  {float a;
     … int h() {…a}


 …
void g()
  {
     …a
  } }
}
```

Formal Languages

- basic notions-

Examples of languages

- natural (ex. English, Romanian)
- programming (ex. C,C++, Java, Python)
- formal

A formal language is a set
Ex.:
$L = \{a^n b^n | \ n>0\}$ $L = \{ab, aabb, aaabbb, \ldots\}$
$L' = \{01^n | n>=0\}$ $L' = \{0, 01, 011, \ldots\}$
$L'' = \{(01)^n | n>=0\}$ $L'' = \{\text{nothing}, 01, 0101, \ldots\}$

Example

a boy has a dog

$S \rightarrow PV$
$P \rightarrow$ *a* N
$N \rightarrow$ *boy* or $N \rightarrow$ *dog*
($N \rightarrow$ *boy*|*dog*)
$V \rightarrow QC$
$Q \rightarrow$ *has*
$C \rightarrow BN$

$B \rightarrow$ *a*

- $A \rightarrow \alpha$ = **rule**
- S,P,V,N,Q,C,B = **nonterminal symbols**
- *a, boy,dog,has* = **terminal symbols**

Remarks

1. Sentence = word, sequence (contains only terminal symbols) ; denoted w.
2. $S \Rightarrow PV \Rightarrow a\ NV \Rightarrow a\ NQC \Rightarrow a\ N\ has\ C$ - sentential form

   In general : $w = a_1 a_2 \ldots a_n$

3. The rule guarantees syntactical correctness, but <u>not</u> the semantical correctness (*A dog has a boy*)

Grammar

- ***Definition***: A (formal) **grammar** is a 4-tuple: $G=(N,\Sigma,P,S)$ with the following meanings:
  - N – set of nonterminal symbols and $|N| < \infty$
  - Σ - set of terminal symbols (alphabet) and $|\Sigma|<\infty$
  - P – finite set of productions (rules), with the propriety:

    $P\subseteq(N\cup\Sigma)^* N(N\cup\Sigma)^* X(N\cup\Sigma)^*$
  - $S\in N$ – start symbol /axiom

Remarks :

1. $(\alpha,\beta)\in P$ is a production denoted $\alpha\rightarrow\beta$
2. $N \cap \Sigma = \emptyset$

A^* = transitive and reflexive closure = $\{a,aa,aaa,\ldots\} \cup \{a^0\}$

$A = \{a\}$

$A+ = \{a,aa,aaa,\ldots\}$

$X^0 =\varepsilon$

Binary relations defined on $(N \cup \Sigma)^*$

- ***Direct derivation***
  $\alpha \Rightarrow \beta$, $\alpha,\beta \in (N \cup \Sigma)^*$ ***if*** α=x1xy1 , β=x1yy1 ***and*** x→y∈P
  (x is transformed in y)
- ***k derivation***

$\alpha \overset{k}{\Rightarrow} \beta$,$\alpha,\beta \in (N \cup \Sigma)^*$

sequence of k direct derivations $\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_{k-1} \Rightarrow \beta$, $\alpha,\alpha_1,\alpha_2,\ldots\alpha_{k-1},\beta \in (N \cup \Sigma)^*$

- ***+ derivation***

$\alpha \overset{+}{\Rightarrow} \beta$ ***if*** $\exists$ k>0 ***such that*** $\alpha \overset{k}{\Rightarrow} \beta$ (there exists at least one direct derivation)

- **** derivation***

$\alpha \overset{*}{\Rightarrow} \beta$ ***if*** $\exists$ k≥0 ***such that*** $\alpha \overset{k}{\Rightarrow} \beta$ namely, $\alpha \overset{*}{\Rightarrow} \beta \Leftrightarrow \alpha \overset{+}{\Rightarrow} \beta$ ***OR*** $\alpha \overset{0}{\Rightarrow} \beta$ $(\alpha=\beta)$

Definition: ***Language generated*** by a grammar $G=(N,\Sigma,P,S)$ is:

$$L(G)=\{w\in\Sigma^* \mid S \stackrel{*}{\Rightarrow} w\}$$

L1 = {a,b,aa}
L2 = {c,d,cd}
L1L2 = {ac,ad,acd,bc,bd,bcd,aac,aad,aacd}

Remarks:

1. $S \stackrel{*}{\Rightarrow} \alpha, \alpha\in(N\cup\Sigma)^*$ = sentential form
   $S \stackrel{*}{\Rightarrow} w, w\in\Sigma^*$ = word / sequence
2. Operations defined for languages (sets) :
   $L1\cup L2$, $L1\cap L2$, $L1-L2$, $\overline{L}$ (complement) , $L^+=\bigcup_{k>0} L^k$, $L^*=\bigcup_{k\geq 0} L^k$
   Concatenation: $L=L_1L_2 = \{w_1w_2 \mid w_1\in L_1 , w_2\in L_2\}$
3. $|w|=0$ (empty word - denoted ε)

Definition: Two grammar G_1 and G_2 are equivalent if they generate the same language

$$L(G_1)=L(G_2)$$

Chomsky hierarchy(based on form $\alpha \to \beta \in P$)

- type 0 : no restriction
- type 1 : context dependent grammar ($x_1Ay_1 \to x_1\gamma y_1$)
- type 2 : context free grammar ($A \to \alpha \in P$,where $A \in N$ and $\alpha \in (N \cup \Sigma)^*$)
- type 3 : regular grammar ($A \to aB|a \in P$)

Remark :

type 3 $\subseteq$ type 2 $\subseteq$ type 1 $\subseteq$ type 0

Notations

- A,B,C,... – nonterminal symbols
- $S \in N$ – start symbol
- a,b,c,... $\in \Sigma$ – terminal symbol
- $\alpha,\beta,\gamma \in (N \cup \Sigma)^*$ - sentential forms
- ε – empty word
- $x,y,z,w \in \Sigma^*$ - words
- $X,Y,U,... \in (N \cup \Sigma)$ – grammar symbols (nonterminal or terminal)