

Course 2

Scanning (cont.)

Algorithm Scanning v1

While (not.eof) **do**

 detect(token);

 classify(token);

 codify(token);

End_while

Classify

- Classes of tokens:
 - Identifiers
 - Constants
 - Reserved words (keywords)
 - Separators
 - Operators
- If a token can NOT be classified => LEXICAL ERROR

Codify

- May be codification table

OR

code for identifiers and constants

- Identifier, constant => Symbol Table (ST)
- PIF = Program Internal Form = array of pairs
- pairs (token, position in ST)



identifier, constant

Algorithm Scanning v2


```
While (not.eof) do  
    detect(token);  
    if token is reserved word OR operator OR separator  
        then genPIF(token, 0)  
    else  
        if token is identifier OR constant  
            then index = pos(token, ST);  
                genPIF(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 *ST*; 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

Symbol Table

Definition = contains all information collected during compiling regarding the symbolic names 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)

$$h : K \rightarrow A$$

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

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

Toy hash function to use at
lab:
Sum of ASCII codes of chars

Visibility domain (scope)

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

Example:

```
Int main(){  
  ... int a;  
  
  void f()  
  {float a;  
    ...  
  }  
  
  ...  
  void g()  
  {char 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.:

$L = \{a^n b^n \mid n > 0\}$ $L = \{ab, aabb, aaabbb, \dots\}$

$L' = \{01^n \mid n \geq 0\}$ $L' = \{0, 01, 011, \dots\}$

Example

a boy has a dog

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

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

Remarks

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

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

3. The rule guarantees syntactical correctness, but not 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 =
 $\{a, aa, aaa, \dots\} \{a^0\}$

$A = \{a\}$
 $A^+ = \{a, aa, aaa, \dots\}$

$X^0 = \varepsilon$

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

- **Direct derivation**

$\alpha \Rightarrow \beta$, $\alpha, \beta \in (N \cup \Sigma)^*$ **if** $\alpha = x1xy1$, $\beta = x1yy1$ **and** $x \rightarrow y \in P$
(x is transformed in y)

- **k derivation**

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

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

- **+ derivation**

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

- *** derivation**

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

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

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

Remarks:

1. $S \xRightarrow{*} \alpha, \alpha \in (N \cup \Sigma)^* =$ sentential form
 $S \xRightarrow{*} w, w \in \Sigma^* =$ word / sequence

2. Operations defined for languages (sets) :

$$L_1 \cup L_2, L_1 \cap L_2, L_1 - L_2, \bar{L} \text{ (complement)}, L^+ = \bigcup_{k>0} L^k, L^* = \bigcup_{k \geq 0} L^k$$

$$\text{Concatenation: } L = L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$$

3. $|w|=0$ (empty word - denoted ε)

$$L_1 = \{a, b, aa\}$$

$$L_2 = \{c, d, cd\}$$

$$L_1 L_2 = \{ac, ad, acd, bc, bd, bcd, aac, aad, aacd\}$$

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 $\alpha \rightarrow \beta \in P$)

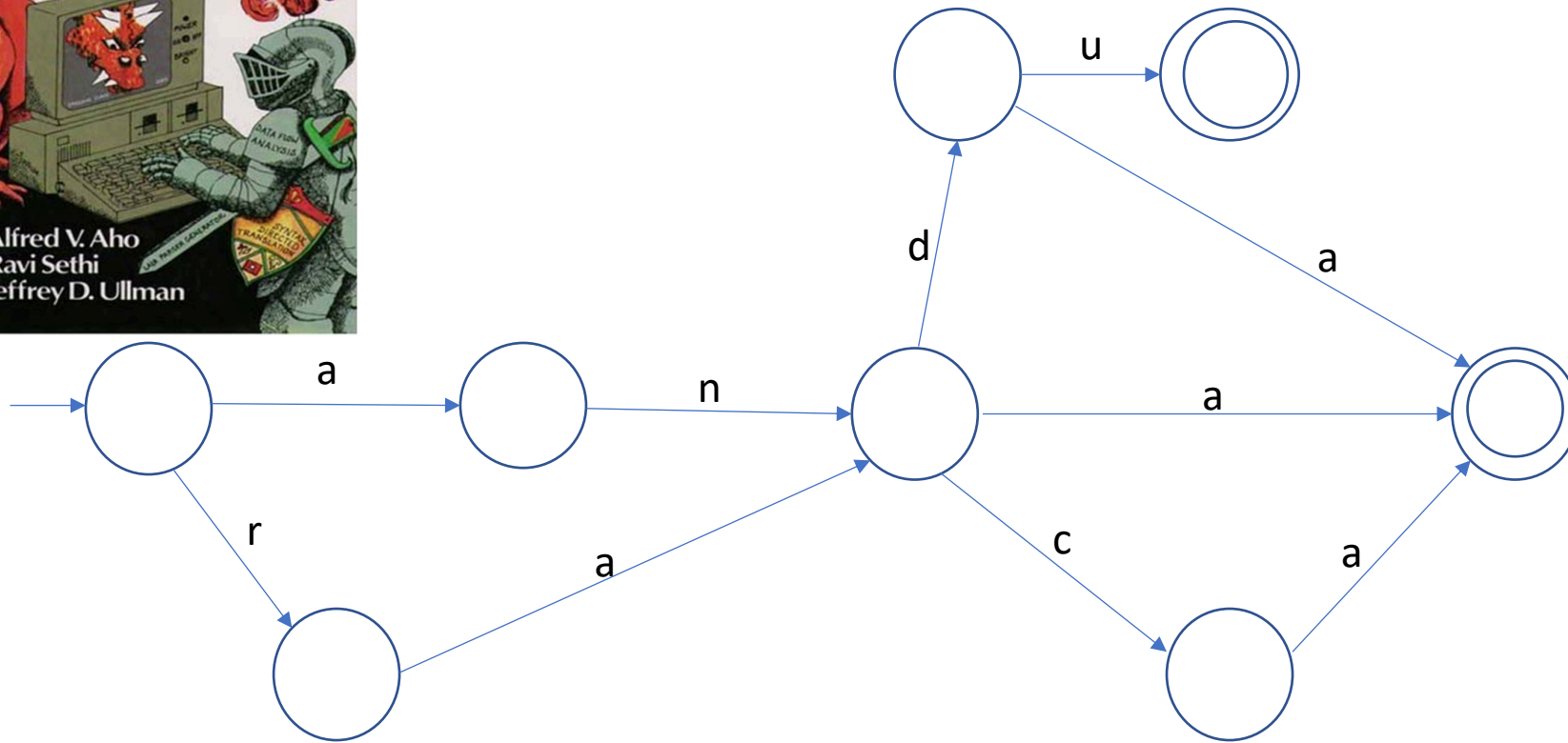
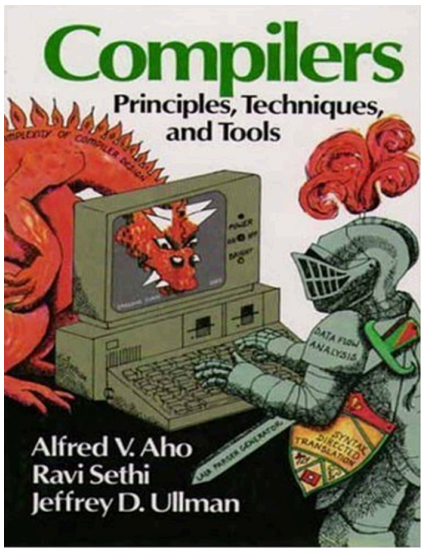
- type 0 : no restriction
- type 1 : context dependent grammar ($x_1Ay_1 \rightarrow x_1\gamma y_1$)
- type 2 : context free grammar ($A \rightarrow \alpha \in P$, where $A \in N$ and $\alpha \in (N \cup \Sigma)^*$)
- type 3 : regular grammar ($A \rightarrow aB \mid a \in P$)

Remark :

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

Notations

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



Problem: The door to the tower is closed by the **Red Dragon**, using a complicated machinery. Prince Charming has managed to steal the plans and is asking for your help. Can you help him determining all the person names that can unlock the door