Course 5

Context free grammars (cfg)

Context free grammar (cfg)

- Produtions of the form: A $\rightarrow \alpha$, A∈N, α∈(N∪$\boldsymbol{\Sigma}$)*

- More powerful

- Can model programming language:

  G = (N, $\boldsymbol{\Sigma}$,P,S) s.t. L(G) = programming language

Syntax tree

Definition: A syntax tree corresponding to a cfg $G = (N, \Sigma, P, S)$ is a tree obtained in the following way:

1. Root is the starting symbol S
2. Nodes $\in N \cup \Sigma$:
    1. Internal nodes $\in N$
    2. Leaves $\in \Sigma$
3. For a node A the descendants in order from left to right are $X_1, X_2, \ldots, X_n$ only if $A \rightarrow X_1 X_2 \ldots X_n \in P$

Remarks:

a) Parse tree = syntax tree – result of parsing (syntatic analysis)

b) Derivation tree – condition 2.2 not satisfied

c) Abstract syntax tree (AST) ≠ syntax tree (semantic analysis)

Syntax tree (cont)

Property: In a cfg $G = (N, \Sigma, P, S)$, $w \in L(G)$ <u>if and only if</u> there exists a syntax tree with frontier w.

Proof: HW

Definition: A cfg G = (N, $\boldsymbol{\Sigma}$,P,S) is ambigous if for a w ∈ L(G) there exists 2 distinct syntax tree with frontier w.

Example:

Parsing (syntax analysis) modeled with cfg:

cfg $G = (N, \Sigma, P, S)$:

- N – nonterminal: syntactical constructions: declaration, statement, expression, a.s.o.
- Σ – terminals; elements of the language: identifiers, constants, reserved words, operators, separators
- P – syntactical rules – expressed in BNF – simple transformation
- S – syntactical construct corresponding to program

THEN

Program syntactical correct <=> $w \in L(G)$

Equivalent transformation of cfg

Unproductive symbols

Definition

A nonterminal A este ***unproductive*** in a cfg if does not generate any word: $\{w \mid A \Rightarrow^* w, w \in \Sigma^*\} = \varnothing$.

Algorithm 1: Elimination of unproductive symbols

input: $G = (N,\Sigma,P,S)$

output: $G' = (N',\Sigma,P',S)$, $L(G) = L(G')$

// idea: build $N_0,N_1,\ldots$ recursively (until saturation)

step 1: $N_0 = \varnothing$; $i:=1$;

step 2: $N_i = N_{i-1} \cup \{A \mid A\rightarrow\alpha \in P, \alpha \in (N_{i-1} \cup \Sigma)^*\}$

step 3: if $N_i <> N_{i-1}$ then $i:=i+1$; goto step 2

else $N' = N_i$

step 4: if $S \notin N'$ then $L(G) = \varnothing$

else $P' = \{A\rightarrow\alpha \mid A\rightarrow\alpha \in P \text{ and } A \in N'\}$

Example

$G = (\{S,A,B,C,D\}, \{a,b,c\}, P,S)$

P: $S \rightarrow aA \mid aC$

$A \rightarrow AB$

$B \rightarrow b$

$C \rightarrow aC \mid CD$

$D \rightarrow b$

Inaccesible symbols

<u>Definition</u>

A symbol $X \in N \cup \Sigma$ is ***inaccesible*** in a cfg if X does not appear in any sentential form: $\forall\ S \Rightarrow^* \alpha,\ X \notin \alpha$

Algorithm 2: Elimination of inaccessible symbols

input: $G = (N,\Sigma,P,S)$

output: $G' = (N',\Sigma',P',S)$, $L(G) = L(G')$ *and*

$$\forall X \in N \cup \Sigma\ \exists \alpha, \beta \in (N' \cup \Sigma')^* \text{ s.t. } S \Rightarrow^*_{G'} \alpha X \beta.$$

step 1: $V_0 = \{S\}$; $i:=1$;

step 2: $V_i = V_{i-1} \cup \{X \mid \exists\ A \to \alpha X \beta \in P,\ A \in V_{i-1}\}$

step 3: if $V_i <> V_{i-1}$ then $i:=i+1$; goto step 2

else $N' = N \cap V_i$

$\Sigma' = \Sigma \cap V_i$

$P' = \{A \to \alpha \mid A \to \alpha \in P,\ A \in N',\ \alpha \in (N \cup \Sigma)^*\}$

Example

$G = (\{S,A,B,C,D\}, \{a,b,c,d\}, P,S)$

P: $S \rightarrow aA \mid aC$

$A \rightarrow AB$

$B \rightarrow b$

$C \rightarrow aC \mid bCb$

$D \rightarrow bB \mid d$

ε-productions

Definition

A cfg $G=(N,\Sigma,P,S)$ is without ***ε-productions*** if

1. $P \not\ni A \rightarrow \varepsilon$ (ε-productions)

OR

2. $\exists\, S\rightarrow\varepsilon$ si $S \notin rhs(p), \forall p \in P$

Algorithm 3: Elimination of ε-productions

input: cfg $G = (N,\Sigma,P,S)$

output: cfg $G' = (N',\Sigma,P',S')$

step 1: construct $\overline{N} = \{A \mid A \in N, A \Rightarrow^{+} \varepsilon\}$

1.a. $N_0 := \{A \mid A\rightarrow\varepsilon \in P\}$;

$i := 1$;

1.b. $N_i := N_{i-1} \cup \{A \mid A\rightarrow\alpha \in P, \alpha \in N^{*}_{i-1}\}$

1.c. **if** $N_i \neq N_{i-1}$ **then** $i:=i+1$; **goto** step 1.b

else $\overline{N} = N_i$

step 2: Let P' = set of productions built:

2.a. **if** $A\rightarrow\alpha_0B_1\alpha_1B_2\alpha_2 \ldots B_k\alpha_k \in P$, $k>=0$

and for $i := 1,k$ $B_i \in \overline{N}$

and $\alpha_j \notin \overline{N}$, $j:=0,k$

then add to P' all prod of the form

$A\rightarrow\alpha_0X_1\alpha_1X_2\alpha_2 \ldots X_k\alpha_k$

where X_i is B_i or ε (not $A\rightarrow\varepsilon$)

2.b **if** $S \in \overline{N}$' **then** add S' to N' and $S'\rightarrow S \mid \varepsilon$ to P

else $N' := N$; $S' := S$.

Example

$G = (\{S,A,B\}, \{a,b\},P,S)$

P: $S \to aA \mid aAbB$

$A \to aA \mid B$

$B \to bB \mid \boldsymbol{\varepsilon}$

Single productions

Definition

O production of the form $A \rightarrow B$ is called single production or renaming rule.

Algorithm 4 : Elimination of single productions

Input: cfg G, without ε-productions

Output: G' s.t. L(G) = L(G')

For each $A \in N$ build the set $N_A = \{B \mid A \Rightarrow^* B\}$:

1.a. $N_0 := \{A\}$, $i := 1$

1.b. $N_i := N_{i-1} \cup \{C \mid B \rightarrow C \in P$ si $B \in N_{i-1}\}$

1.c. **if** $N_i \neq N_{i-1}$ **then** $i := i+1$ **goto** 1.b.

else $N_A := N_i$

P': **for** all $A \in N$ **do**

for all $B \in N_A$ **do**

if $B \rightarrow \alpha \in P$ **and not** "single" **then** $A \rightarrow \alpha \in P'$

$G' = (N, \Sigma, P', S)$

Example

$G = (\{E,T,F\},\{a,(,),+,*\},P,E)$

P: $E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid a$