Course 5

Context free grammars (cfg)

Context free grammar (cfg)

- Procdutions of the form: A $\rightarrow \alpha$, A$\in$N, $\alpha \in (N \cup \Sigma)^*$
- More powerful
- Can model programming language:

  G = (N, Σ,P,S) s.t. L(G) = programming language

Equivalent transformation of cfg

- Unproductive symbols
- Inaccesible symbols

- ε - productions
- Single productions

1. Determine elements (symbols/ productions): Greedy alg

2. eliminate them: construct equivalent grammar

Unproductive symbols

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

Algorithm 1: Elimination of unproductive symbols

input: G = (N,Σ,P,S)

output: G' = (N',Σ,P',S), L(G) = L(G')

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

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

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

step 3: if $N_i \neq N_{i-1}$ then i:=i+1; goto step 2
else $N' = N_i$

step 4: if $S \notin N'$ then $L(G) = \emptyset$
else $P' = \{A \to \alpha \mid A \to \alpha \in P \text{ and } A \in N'\}$

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

Definition

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,Σ,P,S)

output: G' = (N',Σ',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: $V_i = V_{i-1} \cup \{X \mid \exists A \rightarrow \alpha X \beta \in P, A \in V_{i-1}\}$

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

else $N' = N \cap V_i$

$\Sigma' = \Sigma \cap V_i$

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

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,Σ,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,Σ,P,S)

output: cfg G' = (N',Σ,P',S')

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

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** $N_i <> N_{i-1}$ **then** i:=i+1; **goto** step 1.b
else $\overline{N} = N_i$

A->BC
B->ε
C->ε

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

2.a. **if** $A \rightarrow \alpha_0 B_1 \alpha_1 B_2 \alpha_2 \ldots B_k \alpha_k \in P$, 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
$A \rightarrow \alpha_0 X_1 \alpha_1 X_2 \alpha_2 \ldots X_k \alpha_k$
where X_i is B_i or ε (not $A \rightarrow \varepsilon$)

2.b **if** $S \in N'$ **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: $S \rightarrow aA \mid aAbB$

$A \rightarrow aA \mid B$

$B \rightarrow bB \mid \varepsilon$

Single productions

Definition

O production of the form $A \to B$ 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. $N_i := N_{i-1} \cup \{C \mid B \to C \in P \text{ si } B \in N_{i-1}\}$

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

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

Example

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

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

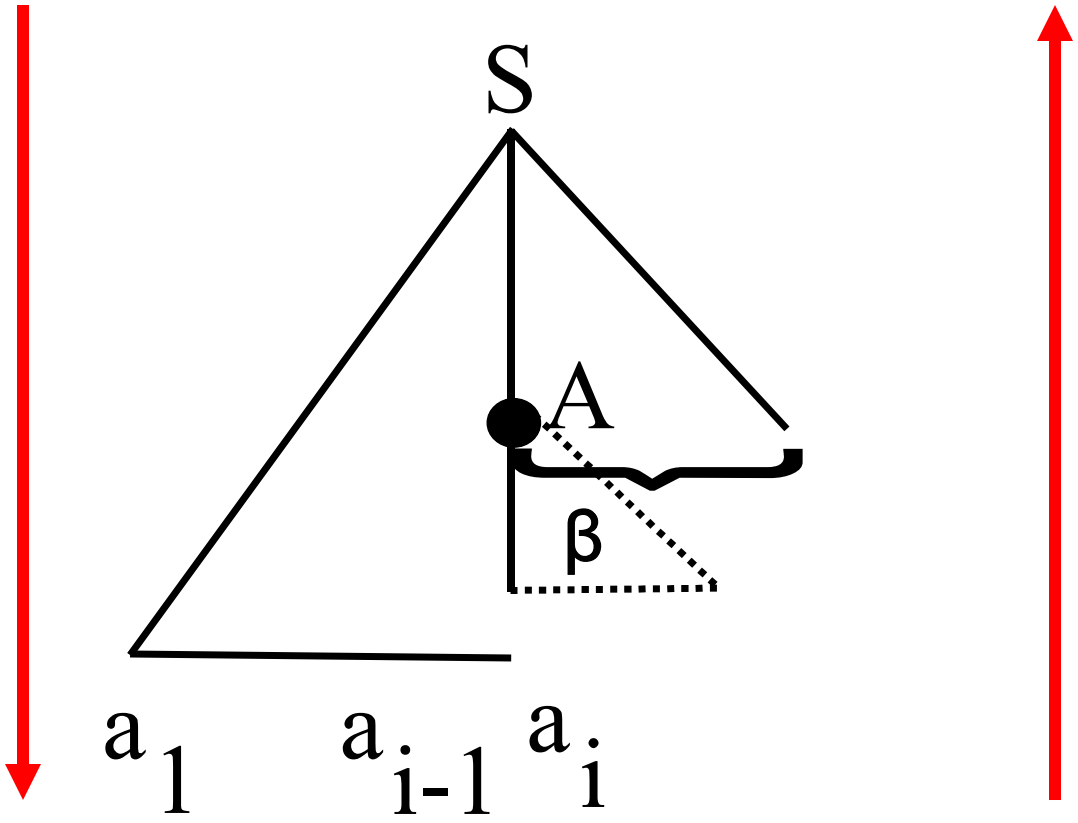
$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid a$

Parsing

- Cfg G = (N, **Σ**, P,S) check if w ∈ L(G)
- Construct parse tree

- How:
  1. Top-down vs. Bottom-up
  2. Recursive vs. linear

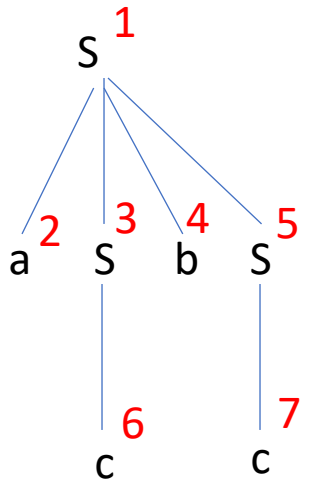


	Descendent	Ascendent
Recursive	Descendent recursive parser	Ascendent recursive parser
Linear	LL(k): LL(1)	LR(k): LR(0), SLR, LR(1), LALR

Result – parse tree -representation

- Arbitrary tree – child sybling representation
- Sequence of derivations S => α_1 => α_2 =>… => α_n = w
- String of production – index associated to prod – which prod is used at each derivation step: 1,4,3,…

index	Info	Parent	Right sibling
1	S	0	0
2	a	1	0
3	S	1	2
4	b	1	3
5	S	1	4
6	c	3	0
7	c	5	0



Example – equivalance of the representation

S -> aSbS | c

$S \Rightarrow^{*} acbacbc$

Sequence of derivation / string of productions / syntax tree