Adrian Copta | FAF-223 | Varian 6

1. **Present the Context Free Grammar in Chomsky Normal Form**

G = (VN,VT, P, S), VN= {S,A,B}, VT = {a,b}
P= {1. S → a B
2. S → b a A
3. A→ a
4. A→ a S
5. A→ b A a B
6. B→ A
7. B→ B S
8. B → ε }.

1) Removing ε productions:

a) Nε = ∅
b) For the production B → ε, Nε = ∅ ∪ {B}, Nε={B}.
Removing this production there are obtained:
P'= { S → a B | b a A | a
A→ a | a S | b A a B | b A a
B→ A | B S }.

2) Removing of the unit productions:

The unit production from P' is B→ A.
After elimination of this production there are obtained:
P''= { S → a B | b a A | a
A→ a | a S | b A a B | b A a
B→ a | a S | b A a B | b A a | B S }.

3) Eliminating nonproductive symbols.

Step I. Pr ={}.
Step II. Pr = Pr {S, A, B}={S, A, B}.
Thus, there are no nonproductive symbols.

4) Eliminating the inaccessible symbols.

Step I. Ac = {S}.
Step II. Ac = Ac ∪ {A, B, a, b}={S, A, B, a, b}.
Step III. Ac = {S, A, B, a, b}.
Step IV. I=(VN ∪ VT)\ Ac=({S, A, B} ∪ {a, b})\ {S, A, B, a, b}= ∅.
Thus, all the symbols are accessible.

5) Converting productions to CNF.

P'''= { S → X_1 B | X_3 A | a
A→ a | X_1 S | X_4 X_5 | X_4 X_1
B→ a | X_1 S | X_4 X_5 | X_4 X_1 | B S
X_1→ a
X_2→ b
X_3→ X_2 X_1
X_4→ X_2 A
X_5→ X_1 B}.

VT = {a, b},
VN = {S, A, B, X_1, X_2, X_3, X_4, X_5}.

2. **Converting the given grammar to the Greibach Normal Form**

G=(VN, VT, P, S,), VN={S, A, B, C}, VT ={a, b},

P={ 1. S → B C

2. C → B B

3. C → a

4. A → b

5. B → C A }.

1) Conversing into the Chomsky Normal Form.

The gramma is already in CNF

2) Renaming all non-terminal symbols in A1, A2,..., An.

A1 = S;

A2 = A;

A3 = B

A4 = C

P'={ A1 → A3 A4

A2 → b

A3 → A4 A2

A4 → A3 A3 | a }.

3) Verifying all indexes.

1 < 3, 3 < 4, but at the production A4 → A3 A3, 4 > 3

After substitution we obtain:

P''={ A1 → A3 A4

A2 → b

A3 → A4 A2

A4 → A4 A2 A3 | a }.

4) Eliminating left recursions.

A4 → A4 A2 A3 is a left recursion

eliminating we get: A4 → a Z; Z → A2 A3 Z | A2 A3

After substitution we obtain:

P''={ A1 → A3 A4

A2 → b

A3 → A4 A2

A4 → a Z

Z → A2 A3 Z | A2 A3}.

5) Converting to GNF.

P''={ A1 → a Z A2 A4

A2 → b

A3 → a Z A2

A4 → a Z

Z → b A3 Z | b A3}.

Substituting we get:

P'''={ S → a Z B C

A → b

B → a Z A

C → a Z

Z → b B Z | b B},

VN={S, A, B, C, Z}, VT ={a, b},

3. For the given grammar G=(VN, VT, P, S,), VN ={S, A, B, D}, VT ={a,b,c,d,f,e}, P={ 1. S → A a b H 2. H → R c 3. R → f 4. R → R d f 5. A → e 6. A → A d e }, build the matrix of simple precedence, analyze the *edeabfdfc* string and construct the derivation tree.

a) Building the sets FIRST and LAST

	FIRST	LAST
S	A, e	H, c
H	R, f	c
R	f, R	f
A	e, A	e

b) Building the matrix of precedence relations

1. x1 = x2
   S → A a b H: A=a, a=b, b=H
   H → R c: R=c
   R → R d f: R=d, d=f
   A → A d e: A=d, d=e
2. x1<x2
   S → A a b H: b < FIRST(H)={R, f}
   $ < FIRST(S) = {A, e}
3. x1>x2
   S → A a b H: LAST(A) > a; {e} > a
   H → R c: LAST(R) > c; {f} > c
   R → R d f: LAST(R) > d; {f} > d
   A → A d e: LAST(A) > d; {e} > d
   LAST(S) > $; {H, c} > $

	S	H	R	A	a	b	c	d	e	f	$
S											
H											>
R							=	=			
A					=			=			
a						=					
b		=	<							<	
c											>
d									=	=	
e					>			>			
f							>	>			
$				<					<		

c) Analysis of string <edeabfdfc>

$<e>d<e>a=b<f>d=f>c>$

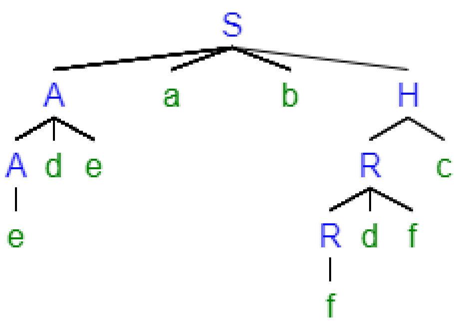
$<A=d<e>a=b<R=d=f>c>$

$<A=a=b>R=c>$

$<A=a=b=H>$

$<S>$

d) Building the derivation tree



4. Construct pushdown automata for the following language:

$L = \{a^n b^n a^k \mid n \in N,\ k = n - 1\}$ **Present the analysis of the word.**

Answer:

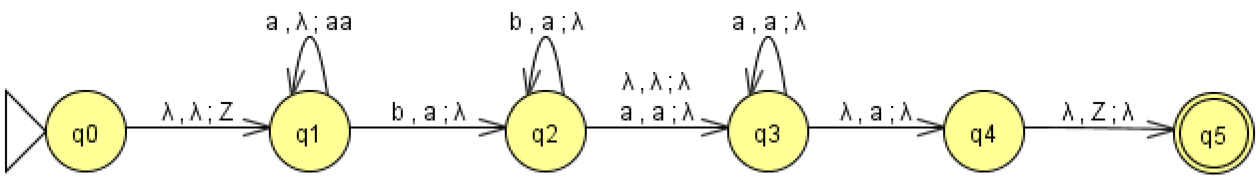


Fig (1). PDA for the given language

We formally express the PDA as a 6-tuple (Q, Σ, Γ, δ, q0, F), where

- Q = {q0, q1, q2, q3, q4, q5}
- Σ = {a, b}
- Γ = {a, Z}
- q0 = {q0}
- Z0= {Z}
- transition function δ : Q × Σλ × Γλ → P(Q × Γλ) is defined by the table:

δ	a	b	λ
→q0	-	-	q1 λ | Z
q1	q1 λ | aa	q2 a | λ	-
q2	q3 a | λ	q2 a | λ	-
q3	q3 a | λ	-	q4 a | λ
q4	-	-	q5 Z | λ
*q5	-	-	-

- F = {q5}

Analysis of the word:

Let's consider the word: aaabbbaa

[q0, aaabbbaa, λ]Ⱶ[q1, aaabbbaa, Z] Ⱶ[q1, aabbbaa, aaZ] Ⱶ[q1,abbbaa, aaaaZ] Ⱶ[q1, bbbaa, aaaaaaZ] Ⱶ[q2, bbaa, aaaaaZ] Ⱶ[q2, baa, aaaaZ] Ⱶ[q2, aa, aaaZ] Ⱶ[q3, a, aaZ] Ⱶ[q3, λ, aZ] Ⱶ[q4, λ, Z] Ⱶ[q5, λ, λ].

Verifying the PDA:

Using JFLAP software we computed our PDA in the program (fig 1) and ran some strings which respect the rules of the language and which do not.

Table Text Size

Input	Result
aaabbbaa	Accept
aabba	Accept
ab	Accept
aaaaaaaabbbbbbbaaaaaaa	Accept
aabbaaa	Reject
aabbaa	Reject
aba	Reject

Fig (2). Testing PDA using JFLAP