18CSC 301T - Formal Language & Automata
CLA-2 Answer key

prepared by
Dr-R.Rathna/IT

1. S → NP VP
S → Aux NP VP
S → VP
NP → Det NOM
NOM → Noun
NOM → Noun NOM
VP → verb
VP → verb NP
Det → that | this | a | the
Noun → book | flight | meal | man
Verb → book | include | read
Aux → does

CNF (Already in CNF)

S → NP · VP
NP → Det · NOM
NOM → NOUN · NOM
VP → verb · NP
Det → that
Det → this
Det → a
Det → the
Noun → book
Noun → flight
Noun → meal
Noun → man
Verb → book
Verb → include
Verb → read
Aux → does

i) How many productions in the given CFGs are already in CNF? (1)

16

ii) The given production are type 2 grammar (1)

a) 0
b) 1
c) 2 ✓
d) 3

iii) List the terminal and Non-Terminal symbols. (3)

Terminals

that, this, a, the
book, flight, meal,
man, include, read,
does

Non. Terminals

S, NP, NOM, VP,
Det, Noun, Verb, Aux

iv) conversion of ~~PD~~. CFG to PDA requires Grammar simplification first, So let us do vi) simplify the grammar (7)

elimination of unit production

$S \rightarrow VP$
$Nom \rightarrow Noun$ } unit productions
$VP \rightarrow verb$

$S \rightarrow VP$

$S \rightarrow Verb\ NP$

$S \rightarrow book\ NP \mid include\ NP \mid read\ NP$

$Nom \rightarrow Noun$

$Nom \rightarrow book \mid flight \mid meal \mid man$

$VP \rightarrow Verb$

$VP \rightarrow book \mid include \mid read$

There is no other null production, left recursive production and useless production

Dr. R.R

vii) CFG to CNF (4)

productions not in CNF are

$S \rightarrow Aux \cdot NP \cdot VP$

$S \rightarrow Aux \cdot S$

Remaining other productions are already converted to CNF in simplification step.

iv) The CFG should be converted to GNF as the first step for creating PDA.

conversion of CFG to GNF

$S \rightarrow book \cdot NP$
$S \rightarrow include \cdot NP$
$S \rightarrow read \cdot NP$
(in GNF)

$S \rightarrow NP \cdot VP$

$S \rightarrow Det \cdot NOM \cdot VP$

$S \rightarrow that \cdot NOM \cdot VP$
$S \rightarrow this \cdot NOM \cdot VP$
$S \rightarrow a \cdot NOM \cdot VP$
$S \rightarrow the \cdot NOM \cdot VP$
(in now GNF)

$S \rightarrow Aux \cdot S$

$S \rightarrow does \cdot S$ (in now GNF)

$NP \rightarrow Det \cdot NOM$

$NP \rightarrow that \cdot NOM$
$NP \rightarrow this \cdot NOM$
$NP \rightarrow a \cdot NOM$
$NP \rightarrow the \cdot NOM$
(now in GNF)

$Nom \rightarrow book$
$Nom \rightarrow flight$
$Nom \rightarrow meal$
$Nom \rightarrow man$
(already in GNF from (iv))

$Nom \rightarrow Noun \cdot NOM$

$Nom \rightarrow book \cdot NOM$
$Nom \rightarrow flight \cdot NOM$
$Nom \rightarrow meal \cdot NOM$
$Nom \rightarrow man \cdot NOM$
in GNF

Vp → book
Vp → include
Vp → read

(Now in GNF)

Vp → Verb . NP

Vp → book . NP
Vp → include . NP
Vp → read . NP

in GNF

$\delta(q_1, book, Vp) = \delta(q_1, NP)$
$\delta(q_1, include, Vp) = \delta(q_1, NP)$
$\delta(q_1, read, VP) = \delta(q_1, NP)$

Det → that
Det → this
Det → a
Det → the

(in GNF)

Noun → book
Noun → flight
Noun → meal
Noun → man

(in GNF)

Nom → book . NOM
$\delta(q_1, book, Nom) = \delta(q_1, NOM)$
Nom → flight . NOM
$\delta(q_1, flight, Nom) = \delta(q_1, NOM)$
Nom → meal . NOM
$\delta(q_1, meal, Nom) = \delta(q_1, NOM)$
Nom → man . NOM
$\delta(q_1, man, Nom) = \delta(q_1, NOM)$

Verb → book
Verb → include
Verb → read

(in GNF)

Aux → does (in GNF)

production rules to δ(ID) of PDA (5)

S → book . NP
$\delta(q_1, book, S) = \delta(q_1, NP)$
S → include NP
$\delta(q_1, include, S) = \delta(q_1, NP)$
S → read . NP
$\delta(q_1, read, S) = \delta(q_1, NP)$
S → that . NOM . VP
$\delta(q_1, that, S) = \delta(q_1, NOM . VP)$
S → this . NOM . VP
$\delta(q_1, this, S) = \delta(q_1, NOM . VP)$
S → a . NOM . VP
$\delta(q_1, a, S) = \delta(q_1, NOM . VP)$
S → the . NOM . VP
$\delta(q_1, the, S) = \delta(q_1, NOM . VP)$
S → does . S
$\delta(q_1, does, S) = \delta(q_1, S)$

NP → that . NOM
$\delta(q_1, that, NP) = \delta(q_1, NOM)$
NP → this . NOM
$\delta(q_1, this, NP) = \delta(q_1, NOM)$
NP → a . NOM
$\delta(q_1, a, NP) = \delta(q_1, NOM)$
NP → the . NOM
$\delta(q_1, the, NP) = \delta(q_1, NOM)$
Nom → book
$\delta(q_1, book, Nom) = \delta(q_1, \epsilon)$
Nom → flight
$\delta(q_1, flight, Nom) = \delta(q_1, \epsilon)$
Nom → meal
$\delta(q_1, meal, Nom) = \delta(q_1, \epsilon)$
Nom → man

Dr. R.R

VP → book

$\delta(q_1, \text{book}, VP) = \delta(q_1, \varepsilon_f)$

VP → include

$\delta(q_1, \text{include}, VP) = \delta(q_1, \varepsilon_f)$

VP → read

$\delta(q_1, \text{read}, VP) = \delta(q_1, \varepsilon_f)$

Det → that

$\delta(q_1, \text{that}, Det) = \delta(q_1, \varepsilon_f)$

Det → this

$\delta(q_1, \text{this}, Det) = \delta(q_1, \varepsilon_f)$

Det → the

$\delta(q_1, \text{the}\ Det) = \delta(q_1, \varepsilon_f$

Noun → book

$\delta(q_1, \text{book}, Noun) = \delta(q_1, \varepsilon_f)$

Noun → flight

$\delta(q_1, \text{flight}, Noun) = \delta(q_1, \varepsilon_f)$

Noun → meal

$\delta(q_1, \text{meal}, Noun) = \delta(q_1, \varepsilon_f)$

Noun → man

$\delta(q_1, \text{man}, Noun) = \delta(q_1, \varepsilon_f)$

Verb → book

$\delta(q_1, \text{book}, verb) = \delta(q_1, \varepsilon_f)$

verb → include

$\delta(q_1, \text{include}, verb) = \delta(q_1, \varepsilon_f)$

verb → read

$\delta(q_1, \text{read}, verb) = \delta(q_1, \varepsilon_f)$

Aux → does

$\delta(q_1, \text{does}, Aux) = \delta(q_1, \varepsilon_f)$

vii) Check if the above grammar could generate the string "does this flight include a meal"

$\vdash (q_1, \text{does this flight include a meal}, S)$

$= \vdash (q_1, \text{this flight include a meal}, S)$

$= \vdash (q_1, \text{flight include a meal}, NOM.VP)$

$= \vdash (q_1, \text{include a meal}, VP)$

$= \vdash (q_1, \text{a meal}, NP)$

$= \vdash (q_1, \text{meal}, NOM)$

$= \vdash (q_1, \varepsilon_f, Z_0)$

So the string can be accepted

Dr. R.R

2. 'There are 2 color cubes (Red & Yellow) equal in number. Red cubes need to be stacked. Then for each yellow cube, remove one red cube. Make sure stack should be cleared.

i) what is the maximum stack size for a PDA ? (1)

a) n
b) 2^n
c) infinite ✓
d) n * n

ii) Is the language generated for the given scenario is regular ? (1)

a) yes ✓
b) No

iii) Generate the accepting language for above scenario.

$$\{L = r^n y^n \mid n > 1\} \quad (3)$$

iv) Construct CFG

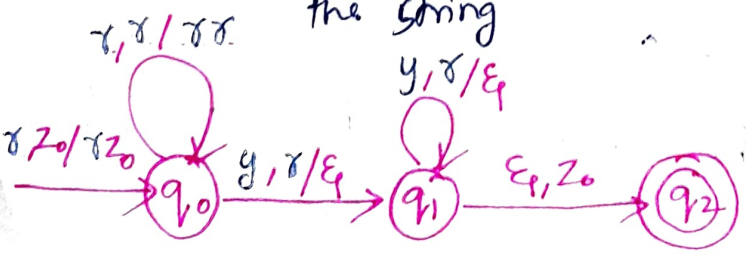
$S \rightarrow rSy \mid rA \mid yA$

$A \rightarrow r \mid y \mid \varepsilon$

v) construct PDA

Logic : (i) push Red color cubes 'R' into the stack
(ii) when a yellow color cube comes, Stop the push, & start 'pop'.
(iii) If both the input & stack becomes empty, accept the string

vi) PDA Diagram (4)



Dr.R.R

DY · R · R

δ(ID)

$\delta(q_0, R, Z_0) = \delta(q_0, RZ_0)$

$\delta(q_0, R, R) = \delta(q_0, RR)$

$\delta(q_0, Y, R) = \delta(q_1, R)$

$\delta(q_1, Y, R) = \delta(q_1, R)$

$\delta(q_1, \varepsilon, Z_0) = \delta(q_2, \varepsilon)$

vi) CFG - (V, T, P, S) Tuple representations (4)

({S, A}, {r, y}, P, S)

P S → rsy | rA | yA

A → r | y | ε

PDA - (Q, Σ, T, q_0, Z_0, δ, F)

({q_0, q_1, q_2}, {r, y}, {Z_0, r}, q_0, Z_0, δ, q_2)

δ - written above

viii) Check whether 3 consecutive yellow followed by 3 consecutive red balls (cubes) can be taken? (3)

yyyrrr

$(q_1, yyyrrr, Z_0) = ?$ no δ(ID) for this y & Z_0

Hence yyyrrr will not be accepted.

3. Consider the CFG

DR·R·R

BLOCK → STMT | { STMTS }

STMTS → ε | STMT STMTS

STMT → EXPR | if (EXPR) BLOCK | while (EXPR) BLOCK |

do BLOCK while (EXPR) | BLOCK

EXPR → a | constant | EXPR + EXPR | EXPR - EXPR | EXPR * EXPR | EXPR / EXPR

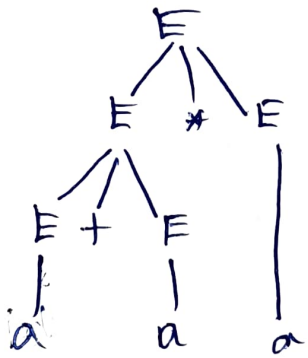
i) Given string is a+a*a

Let us take EXPR as E

E → E*E

→ E+E*E

→ a+a*a

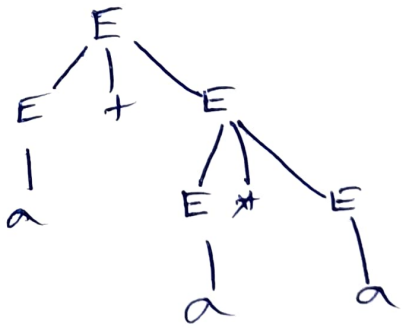


E → E+E

→ E+E*E

→ a+E*E

→ a+a*a



Since we obtain two different parse trees for the string a+a*a

a) it is ambiguous for the string a+a*a (1 Mark)

ii) Which of the following is not true about ambiguous grammar.

c) It is sufficient to derive one leftmost and one rightmost derivation to prove its ambiguity

(1 Mark)

iii) Remove the null productions (3 Marks)

STMTS → ε is the null production

If we remove it then the productions will be

BLOCK → STMT | { STMTS }

STMTS → STMT STMTS

STMT → EXPR | if (EXPR) BLOCK | while (EXPR) BLOCK

| do BLOCK while (EXPR) BLOCK

EXPR → a | constant | EXPR + EXPR | EXPR - EXPR |

EXPR * EXPR | EXPR / EXPR

iv) Remove the unit production (4 marks)

BLOCK → STMT — unit

BLOCK → if (EXPR) BLOCK | while (EXPR) BLOCK

| do BLOCK while (EXPR) BLOCK

STMTS → STMT | STMT STMTS ✓

STMT → EXPR — unit

STMT → a | constant | EXPR + EXPR | EXPR - EXPR |

EXPR also correct EXPR * EXPR | EXPR / EXPR

v) Removal of useless symbols (4 Marks)

We can call a Non Terminal as useless, if it's production is nowhere use. But here there is no such useless symbols.

vi) convert into GNF. (12 Marks) Dr.R.R

BLOCK → has already been in GNF

because of removing unit production

BLOCK → { STMTS } (in GNF { , } are non terminals)

STMTS → STMT . STMTS

STMTS → if (EXPR) BLOCK STMTS (in GNF)

→ while (EPR) BLOCK STMTS (in GNF)

→ do BLOCK while (EXPR) BLOCK STMTS (in GNF)

STMTS → EXPR . STMTS

→ a . STMTS | constant STMTS

| a + a | a - a | a * a | a / a

(all are in GNF)

BLOCK → if (EXPR) BLOCK (in GNF)

while (EXPR) BLOCK (in GNF)

do BLOCK while (EXPR) BLOCK (in GNF)

STMT → a | constant | a + a | a - a | a * a | a / a

(in GNF)

— x —

Key prepared by

nuali

20110122

Dr.R.Rathna/IT