Page No.
Date : / /

Properties of LL(K) & LR(K)

Ex - $S \to aA|aB$, $A \to a$, $B \to a$

1) LL(1)
2) LL(2)
3) LR(3)
4) ✓ None of these (as the grammar ambiguous)

Ex - $S \to a|b$ → LL(1)

A Grammar G1 is LL(K) if and only if "K" symbol (including the current symbol) are presented to the compiler, the production that must be used in every step of derivation is 'UNIQUELY' determined & the derivation is used ~~to RMT~~ is LMD.

LL(2) → 2 symbols are presented at a time to compiler.

[K-min-1 as current symbol has to be shown.]

LL(1) - $S \to a|b$

	a	b	ε
S	S→a	S→b	

LL(1) predic parsing table

A Grammar is LL(K) if the table contain uniquely entries only.

Ex - $S \to aA|aB$, $A \to a$, $B \to b$

	a	b	ε
S	S→aA S→aB	-	-
A	A→a		
B		B→b	

DCFL but not LL(1)

but in LL(2)

-	aa	bb	ab	ba	ε
S	S→aA	-	S→aB	-	
A					
B					

→ No confusion so it is LL(2)

EXCELLENT

checking ambiguity of CFG is undecidable

• Every UA is represented by LL(k) or LR(k) – False

w → . . . .

LR(k) → .....

Page No.

Date : / /

Ex- S → abA | abB

A → a

B → b

→ LL(3) as two symbol are not sufficient to remove confusion.

To ck

problem to check whether a grammar is LL(k) is decidable.

LR(K) Grammar- • k does not count current symbol. it does count of look ahead symbol only.

• LR(0) exist.

• the dire derivation used is RMD.

• in LL(k), derivation is done from root to vert but LR(k) is done from Bottom to top. ie Bottom-up parser.

• LR(k) uses deduction tries to find out what can happen but LR(0) LL(k) does predictive work.

Properties of LL(k) or LR(k) -

1. LL(k) & LR(k) both are unambiguous
2. ——— both are O(n)
3. ∀ DCFL ⟶ ∃ LR(k)

∀ DCFL → LL(k) may or may not exist.

4. for each LL(k), LR(k) grammar there exist DCFL.
5. If a grammar is LL(k) then it is also LL(k') where k' ≥ k.

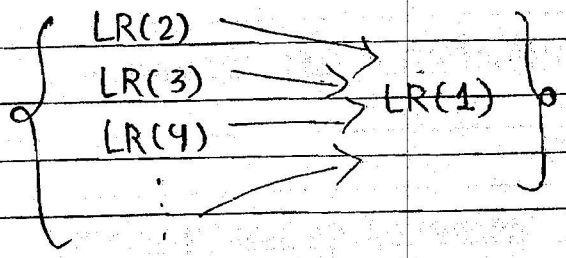
Ex- each LL(2) is also a LL(3) grammar.

6. LR(K) ⇒ LR(K') for K' ≥ K.
7. LL(K) ⇏ LL(K') if K' < k
8. Given a Grammar, to check whether it is LL(k) or LR(k) is decidable

EXCELLENT

Page No.
Date : / /

9. if a grammar is LR(k), for $k \geq 2$, there exist an equivalent grammar LR(1)



Conversion of Grammar from LR(2) to LR(1) is possible.

[For each DCFL, there exist atleast LR(0) or LR(1)
∴ each DCFL will surely have LR(0) or LR(1) grammar.

Algorithm in CFG.

- Given CFG ⟹ ≡ CFG without Null production is possible false
- ——— w/o unit ——— True
- ——— w/o useless ——— True
- ——— w/o left Recursion ——— True
- ——— w/o left factoring ——— True
- ≡ CNF False
- ≡ GNF False

- Given CFG(ε-free) → ≡ CFG w/o Null production is possible — True
- ——— CNF → true
- ——— GNF true.

[If a Grammar has no ε production, then ε does not belong language ∝ corresponds to the grammar]

Grammar
If a language do contain ε, language may or may not have null.

Page No.
Date

- if ϵ is present in Language, grammar must have null production.

- for all CFG G, G' with null production exist having
$$L(G) = L(G') - \{\epsilon\}$$

An Grammar ~~with~~ when applied removal of useless production O/P will be removal of null, unit, useless production

Cover Conversion of CFG to CNF -

1. Removal of null & unit production.
2. Conversion

[CFG without any null production will be converted to equivalent CNF]

Left factoring & Left recursion are removed to minimize the chance of ambiguity

- How to know that ambiguity is removed?
  check the obtained grammar for LL(K) & LR(K)
  Problem for checking for removal of ambiguity is undecidable because it is not possible to write LL(k) or LR(k) for each unambiguous grammar.

ϵ- Unit production are removed to convert a grammar to CNF or GNF
- or to decide the stopping cond^n.

- Useless production are written so that compiler can work faster.

- CFG to CNF - for application of CYK
- CFG to GNF - for conversion to PDA

Page No.
Date

Removal of Null Productions -

1. Identify all the nullable variable. $A \overset{*}{\Rightarrow} \epsilon$

Ques - $S \to ABa|aAb|aC$
$A \to aB|\epsilon$
$B \to bA|\epsilon$
$C \to a|b$

No of Nullable variable = 2 (A, B, S)

2) it create a production set $\hat{P} = P - \{\epsilon \text{ production}\}$

$S \to ABa|aAb|aC$
$A \to aB$
$B \to bA$
$C \to a|b$

3, Add all the production to $\hat{P}$ obtain by putting Nullable variable to Null.

$S \to Ba|Aa|a|ab$
$A \to a$
$B \to b$

∴ New production set is

$S \to ABa|aAb|ac|Ba|Aa|a|ab$
$A \to aB|a$
$B \to bA|b$
$C \to a|b$

to check whether a grammar is Null free if S goes to null it is not null free.

Ex - $S \to AB \mid aAb \mid ac$
$A \to aB \mid \epsilon$
$B \to bA \mid \epsilon$
$C \to a \mid b$

$\Rightarrow$
$S \to A \mid B \mid ab \mid ac \mid \epsilon$
$A \to aB \mid a$
$B \to bA \mid b$
$C \to a \mid b$

$\Rightarrow$ So do not ~~[illegible]~~ & Grammar G' will be

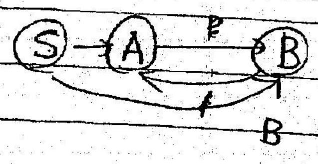
$L(G) = L(G') - \{\epsilon\}$

~~If A → a~~

Removal of Unit Productions -

Ex - $S \to A \mid B$
$A \to aa \mid bb \mid B$ $B \to A \mid ab \mid ba$

1) Identify all the Unit production by using unit production dependency Graph



$S \overset{*}{\Rightarrow} A$
$S \overset{*}{\Rightarrow} B$
$A \overset{*}{\Rightarrow} B$
$B \overset{*}{\Rightarrow} A$

if G is $S \to A$
$A \to aa \mid bb \mid B$
$B \to ab \mid ba \mid A$

$\Rightarrow$ $S \overset{*}{\Rightarrow} A$
$S \overset{*}{\Rightarrow} B$ → indirect paths are also covered.
$A \overset{*}{\Rightarrow} B$
$B \overset{*}{\Rightarrow} A$

Ex - $S \to A \mid aAb \mid ac$
$A \to aa \mid bb \mid B$
$B \to ab \mid ba \mid A$
$C \to c \mid cC$

2. $\hat{P} = P - \{$unit production$)$

$S \to aAb \mid ac$
$A \to aa \mid bb$
$B \to ab \mid ba$
$C \to cC \mid c$

3. Add all the production obtained by mixing unit production with $\hat{P}$

$S \overset{*}{\Rightarrow} A$ $S \overset{*}{\Rightarrow} B$
$S \to aAb \mid ac \mid aa \mid bb \mid ab \mid ba$
$A \to aa \mid bb \mid ab \mid ba$ — $A \overset{*}{\Rightarrow} B$
$B \to ab \mid ba \mid aa \mid bb$ — $B \overset{*}{\Rightarrow} A$
$C \to c \mid cC$

Page No.
Date : / /

Removal of Useless Production:-

Removal of
- the production which have atleast one useless variable either on right or left side.
- A variable is ~~useful~~ useless if it does never occur in any sentential form.
- If it does not generate a sentence i.e terminal string

Ex - $C \to cC|dc$

or if generate a sentence which is no use of language

$$\left\{\begin{array}{l} S \to abB|C \\ C \to cc|dc \end{array}\right\}$$

Ex - $S \to aAb|bA$
$B \to aB|c|aC$
$A \to aA|b|B$
$C \to cC|dC$

Step1 - Useful which can directly go to terminal

$U = \{B, A\}$

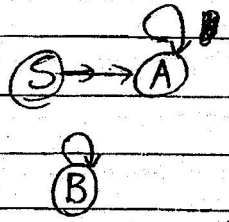
Step2 - Try to include another variable by checking where they are one which are useful.

$U = \{A, B, S\}$

C is not added then C is the useless symbol

$$\therefore \left\{\begin{array}{l} S \to aAb|bA \\ B \to aB|c \\ A \to aA|b \end{array}\right\}$$

Step3 - Draw a variable dependency graph



the thing which are not reachable from S remove them so B is removed.

$$\left\{\begin{array}{l} S \to aAb|bA \\ A \to aA|b \end{array}\right\}$$

Removal of Left Recursion-

If any production is of the form

$A \to A\alpha_1 | A\alpha_2 | A\alpha_3 | \dots | A\alpha_m | \beta_1 | \beta_2 | \beta_3 | \dots \beta_n$

its equivalent Grammar is

$A \to \beta_1 A' | \beta_2 A' | \beta_3 A' | \dots | \beta_n A'$

$A' \to \alpha_1 A' | \alpha_2 A' | \alpha_3 A' | \dots | \alpha_m A' | \epsilon$

We put β in starting as β are the stopping condn & then α can go on.

$A \quad A_1\alpha_2 \quad A_1\alpha_3 \quad \beta_1 \quad \beta_2 \quad \beta_3$

Ex $S \to S*S | S+S | a | b | c$

$S \to *$

$$\begin{cases} S \to aS' | bS' | cS' \\ S' \to *SS' | +SS' | \epsilon \end{cases}$$

If $S \to S*S | S+S | a | b | \epsilon$

$$\begin{cases} S \to *aS' | bS' | S' \\ S' \to *SS' | +SS' | \epsilon \end{cases}$$

Removal of Left factoring-

If the production is of the form

$S \to \alpha\beta | \alpha\gamma$

Its equivalent grammar is

$$\begin{cases} S \to \alpha X \\ X \to \beta | \gamma \end{cases}$$

Ex- $S \to a0 | a1$

equivalent- $S \to aS'$

$S \to 0 | 1$

Page No.
Date

Ex: $S \to ABa \mid ABCd \mid ADE$

∴ equivalent -

$S \to AX$
$X \to BY \mid DE$
$Y \to a \mid cd$

Conversion of CFG to CNF

Step1 - if CFG is ε-free, remove ε & unit production.

Step2 - Conversion to CNF

Ex - $S \to aABb \mid aB \mid BaC$
$A \to ab \mid bA \mid aAB$
$B \to a \mid BA \mid b$

Its equivalent CNF is -

Step1 - identify all thing which are already CNF

$A \to AB$
$B \to a \mid BA \mid b$

Step2 - convert them to CNF by fully subscripted variable.
i.e conversion of terminal.

a) $S \to D_aABD_b \mid D_aB \mid BD_aC$
$A \to D_aD_b \mid D_bA$
$D_a \to a$
$D_b \to b$

∴ New Grammar is

$S \to D_aABD_b \mid D_aB \mid BD_aC$
$A \to D_aD_b \mid D_bA$
$D_a \to a$
$D_b \to b$
$A \to AB$
$B \to a \mid BA \mid b$

~~$S \to X_1 X_2$~~

~~$S \to D_a BD_2$~~

$S \to D_a D_3 \mid D_a B \mid BD_1$

$A \to D_a D_b \mid D_b A \mid AB$

$B \to BA \mid a \mid b$

$D_a \to a$

$D_b \to b$

$D_1 \to D_a C$

$D_2 \to BD_b$

$D_3 \to AD_3$

Conversion of ~~a~~ CFG to GNF- ~~No~~

Properties of CNF & GNF-

1. CNF for CYK & GNF or PDA.
2. G_1 is ~~a~~ CNF then every derivation of string length n with have exactly $(2n-1)$ steps

n → steps to replaced from terminals ABCDE

$n-1$ steps → to obtain final state like $n-1$

3. G_1 is CNF then min. height of derivation tree for w of length n is $\lceil \log_2 n \rceil + 1$ ↓ max. height can be n

↓ bcoz each step will covers the 2 variable. ie. 2 length string height /2 in each step

3. In CNF, ~~ear~~ every derivation tree is a binary tree.
4. If G_1 is in GNF, every derivation of string of length n will have exactly n steps as each time we will be having one terminal.

EXCELLENT

CFG, every derivation is binary tree - false

Page No.
Date :

Ex - "abcde" each time you will get exactly one variable
∴ total no of steps = n

for same string, GNF always have less no of steps.
GNF is not always a binary tree.

1-May-2018 ~~PDAL programming~~

Theorem - When conversion of CFG to CNF is done with production P in CFG & terminal T_0 & the size of ~~long~~ longest string in RHS of CFG is k then Max no of production in $\hat{P}$ (for CNF) is $(k-1)|P| + |T|$

∴ Max. ~~prod~~ No of production $\leq (k-1)|P| + |T|$

(k-1) is taken as

Ex - $S \to ABCDE$

$S \to D_3E$
$D_3 \to D_2D$
$D_2 \to D_1C$
$D_1 \to AB$

} 4 production

|P| it may be possible that all production are of maximum length.

|T| - as it is possible that each variable has to be replaced

$D_a \to a$
$D_b \to b$
$D_c \to c$

EXCELLENT