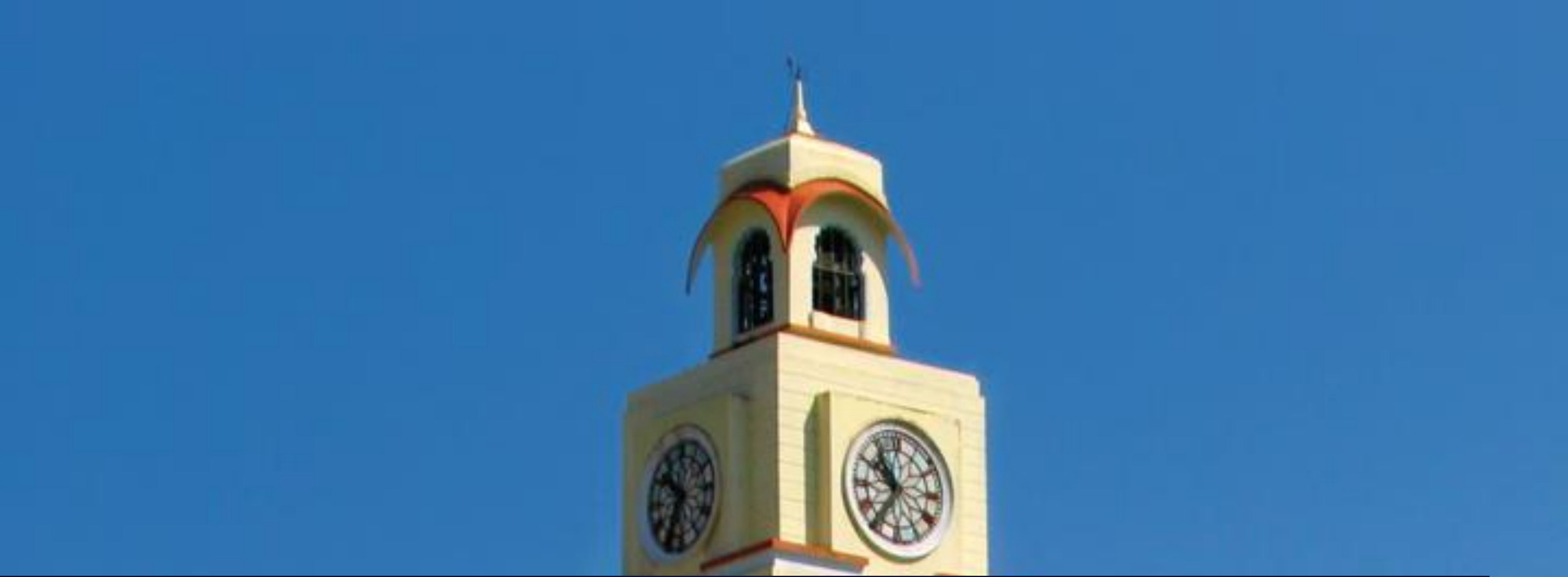


BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Theory of Computation
CS F351

Vishal Gupta

Department of Computer Science and Information Systems

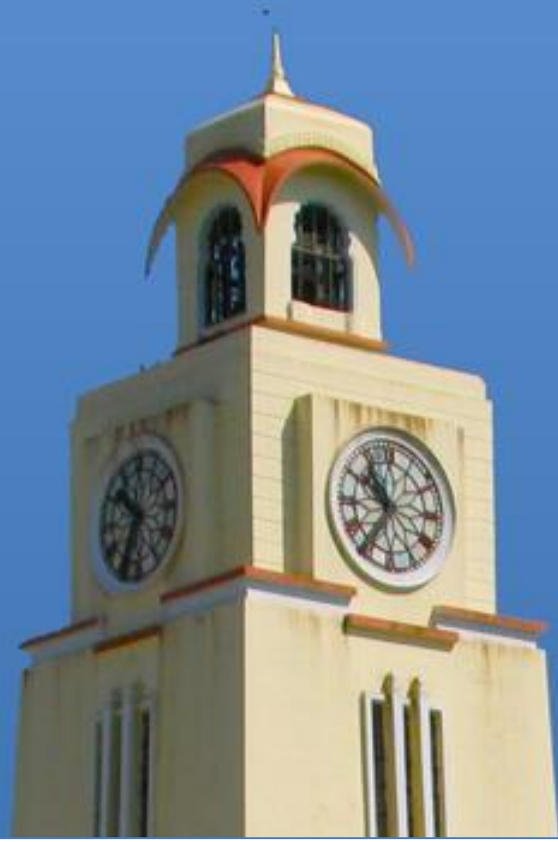
Birla Institute of Technology and Science

Pilani Campus, Pilani

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad



Agenda: 1) Determinism and Parsing 2) Top Down Parsing

Determinism and Parsing

Parser is an Algorithm to determine whether a given string is in the language generated by a given CFG, and, if so, to construct the parse tree of the string.

A PDA is not of IMMEDIATE practical use in parsing because of its non-deterministic nature.

Q. Can we make every PDA deterministic ???

To facilitate making most PDA's deterministic, let us modify the acceptance convention.

" A language is said to be deterministic context free if it is recognized by a deterministic PDA that also has a extra capability of sensing the end of input string".

Let the last symbol by \$ which is not in $\sum$.

Q. Let $L = a^* \cup \{a^n b^n : n \geq 1\}$. Case 1: Try making a deterministic PDA for L.

Case 2: Try making a deterministic PDA for L\$.

"Heuristic" to prove that some CFL's are not deterministic

"The class of deterministic CFL's is closed under complement"

Q. Let L = {$a^n b^m c^p$: m, n, p ≥ 0 and m ≠ n or m ≠ p.
Prove that this language is not deterministic CFL.

Types of PDA's

- Given a deterministic CFG G, we can make following types of PDA's for it:
  - Non-deterministic PDA.
  - Deterministic PDA which does not act as a parser.
  - Deterministic PDA which acts as a top down parser.
  - Deterministic PDA which acts as a bottom up parser.

Top Down Parsing: "Look-ahead" WHY and HOW

Let L = $\{a^n b^n : n \geq 0\}$. Let the corresponding Grammar be:

G = ({a, b, S} , {a, b} , R , S)

Where R contains: S $\rightarrow$ aSb | e [*Note: e is used for empty string*]

The corresponding PDA will be:

M = ({p, q} , {a, b} , {a, b, S} , Δ , p , {q})

Where Δ contains:

((p, e, e) , (q, S))

((q, e, S) , (q, aSb))

((q, e, S) , (q, e))

((q, a, a) , (q, e))

((q, b, b) , (q, e))

The transformed PDA will be:

M_1=({p, q, q_a, q_b, $q_\$$} , {a, b} , {a, b, S} , Δ_1 , p ,{$q_\$$})

Where Δ_1 contains:

((p, e, e) , (q, S))

((q, a, e) , (q_a, e))

((q_a, e, a), (q, e))

((q_a, e, S), (q_a, aSb))

((q, b, e), (q_b, e))

((q_b, e, b), (q, e))

((q_b, e, S), (q_b, e))

((q, \$, e), ($q_\$$, e))

The transformed PDA will be: M_1=({p, q, q_a, q_b, $q_\$$} , {a, b} , {a, b, S} , Δ_1 , p ,{$q_\$$}), Where Δ_1 contains:

((p, e, e) , (q, S)) ((q, a, e) , (q_a, e)) ((q_a, e, a), (q, e)) ((q_a, e, S), (q_a, aSb))

((q, b, e), (q_b, e)) ((q_b, e, b), (q, e)) ((q_b, e, S), (q_b, e))

((q, \$, e), ($q_\$$, e))

State	Unread Input	Stack	Transition used	Rule of G

- Not all CFL's have deterministic PDA's that can be derived from the "*standard non-deterministic*" one via the look ahead idea.
  - For example: L = {w w^R | w ϵ (a U b)*}
- Even for certain deterministic CFL's look ahead of just one symbol may not be sufficient.
  - For example: S -> a S A | e
    A -> a b S | c
- But some languages have certain "always observable" anomalies because of which they cannot become parsers using the idea of look ahead. Let's see these ……

Ques. Can Look-ahead always work??

Consider the following grammar and corres. PDA:

Grammar: G = (V, ∑, R, E)

Where R is

E → E + T

E → T

T → T * F

T → F

F → (E)

F → id

F → id (E)

PDA M = (Q, ∑, Γ, Δ, s, F)

Where Δ is

((p, e, e), (q, E))

((q, e, E), (q, E + T))

((q, e, E), (q, T))

((q, e, T), (q, T*F))

((q, e, T), (q, F))

((q, e, F), (q, (E)))

((q, e, F), (q, id))

((q, e, F), (q, id(E)))

((q, a, a), (q, e)) for all a є ∑

Left Factoring

Left Factoring: Whenever rules of the form

$A \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \ldots\ldots\ldots\ldots\ldots \mid \alpha\beta_n$ where $\alpha \neq e$ and $n \geq 2$

are present, then replace them by the rules:

$A \rightarrow \alpha A'$

$A' \rightarrow \beta_1 \mid \beta_2 \mid \ldots\ldots\ldots\ldots\ldots \mid \beta_n$

Ques. Can Look-ahead always work??

Consider the following grammar and corres. PDA:

Grammar: $G = (V, \Sigma, R, E)$

Where R is

$E \rightarrow E + T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow id$

$F \rightarrow id\ (E)$

PDA $M = (Q, \Sigma, \Gamma, \Delta, s, F)$

Where Δ is

$((p, e, e), (q, E))$

$((q, e, E), (q, E + T))$

$((q, e, E), (q, T))$

$((q, e, T), (q, T*F))$

$((q, e, T), (q, F))$

$((q, e, F), (q, (E)))$

$((q, e, F), (q, id))$

$((q, e, F), (q, id(E)))$

$((q, a, a), (q, e))$ for all $a \in \Sigma$

Ques. Can Look-ahead always work??

Consider the following grammar and corres. PDA:

Grammar: G = (V, ∑, R, E)

Where R is

E → E + T

E → T

T → T * F

T → F

F → (E)

~~F → id~~ F -> id A

~~F → id (E)~~ A -> e | (E)

PDA M = (Q, ∑, Γ, Δ, s, F)

Where Δ is

((p, e, e), (q, E))

((q, e, E), (q, E + T))

((q, e, E), (q, T))

((q, e, T), (q, T*F))

((q, e, T), (q, F))

((q, e, F), (q, (E)))

~~((q, e, F), (q, id))~~ ((q, e, F),(q, idA))

~~((q, e, F), (q, id(E)))~~ ((q, e, A),(q, e))

((q, e, A)(q,(E)))

((q, a, a), (q, e)) for all a ϵ ∑

Ques. Can Look-ahead always work??

innovate achieve lead

Consider the following grammar and corres. PDA:

Grammar: G = (V, ∑, R, E)

Where R is

E → E + T

E → T

T → T * F

T → F

F → (E)

~~F → id~~ F -> id A

~~F → id (E)~~ A -> e | (E)

PDA M = (Q, ∑, Γ, Δ, s, F)

Where Δ is

((p, e, e), (q, E))

((q, e, E), (q, E + T))

((q, e, E), (q, T))

((q, e, T), (q, T*F))

((q, e, T), (q, F))

((q, e, F), (q, (E)))

~~((q, e, F), (q, id))~~ ((q, e, F),(q, idA))

~~((q, e, F), (q, id(E)))~~ ((q, e, A),(q, e))

((q, e, A)(q,(E)))

((q, a, a), (q, e)) for all a ϵ ∑

Left Factoring and Left Recursion

Left Recursion: Whenever rules of the form

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots\ldots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \ldots\ldots\ldots\ldots\ldots \mid \beta_m$$

where n > 0

are present, then replace them by the rules:

$$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \ldots\ldots\ldots \mid \beta_m A'$$

$$A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \alpha_n A' \mid e$$

Ques. Can Look-ahead always work??

Consider the following grammar and corres. PDA:

Grammar: G = (V, ∑, R, E)

Where R is

E → E + T

E → T

T → T * F

T → F

F → (E)

~~F → id~~ F -> id A

~~F → id (E)~~ A -> e | (E)

PDA M = (Q, ∑, Γ, Δ, s, F)

Where Δ is

((p, e, e), (q, E))

((q, e, E), (q, E + T))

((q, e, E), (q, T))

((q, e, T), (q, T*F))

((q, e, T), (q, F))

((q, e, F), (q, (E)))

~~((q, e, F), (q, id))~~ ((q, e, F),(q, idA))

~~((q, e, F), (q, id(E)))~~ ((q, e, A),(q, e))

((q, e, A)(q,(E)))

((q, a, a), (q, e)) for all a ϵ ∑

Ques. Can Look-ahead always work??

Consider the following grammar and corres. PDA:

Grammar: G = (V, ∑, R, E)

Where R is

~~E → E + T~~ E -> TE'

E' -> +TE' | e

~~E → T~~

~~T → T * F~~ T -> FT'

T' -> *FT' | e

~~T → F~~

F → (E)

~~F → id~~ F -> id A

~~F → id (E)~~ A -> e | (E)

PDA M = (Q, ∑, Γ, Δ, s, F)

Where Δ is

~~((p, e, e), (q, E))~~ ???

~~((q, e, E), (q, E + T))~~ ???

~~((q, e, E), (q, T))~~ ???

~~((q, e, T), (q, T*F))~~ ???

~~((q, e, T), (q, F))~~ ???

((q, e, F), (q, (E)))

~~((q, e, F), (q, id))~~ ((q, e, F),(q, idA))

~~((q, e, F), (q, id(E)))~~ ((q, e, A),(q, e))

((q, e, A)(q,(E)))

((q, a, a), (q, e)) for all a ϵ ∑

Deterministic PDA

Thus we can construct the following deterministic PDA M4 that accepts the language L(G)$:

$$M_4 = (K, \Sigma \cup \{\$\}, V', \Delta, p, \{q_\$\}),$$

$$K = \{p, q, q_{\text{id}}, q_+, q_*, q_), q_(, q_\$\},$$

Where delta is given as →→

E -> TE'
E' -> +TE' | e
T -> FT'
T' -> *FT' | e
F → (E)
F -> id A
A -> e | (E)

$((p, e, e), (q, E))$

$((q, a, e), (q_a, e))$ for each $a \in \Sigma \cup \{\$\}$

$((q_a, e, a), (q, e))$ for each $a \in \Sigma$

$((q_a, e, E), (q_a, TE'))$ for each $a \in \Sigma \cup \{\$\}$

$((q_+, e, E'), (q_+, +TE'))$

$((q_a, e, E'), (q_a, e))$ for each $a \in \{), \$\}$

$((q_a, e, T), (q_a, FT'))$ for each $a \in \Sigma \cup \{\$\}$

$((q_*, e, T'), (q_*, *FT'))$

$((q_a, e, T'), (q_a, e))$ for each $a \in \{+,), \$\}$

$((q_(, e, F), (q_(, (E)))$

$((q_{\text{id}}, e, F), (q_{\text{id}}, \text{id}A))$

$((q_(, e, A), (q_(, (E)))$

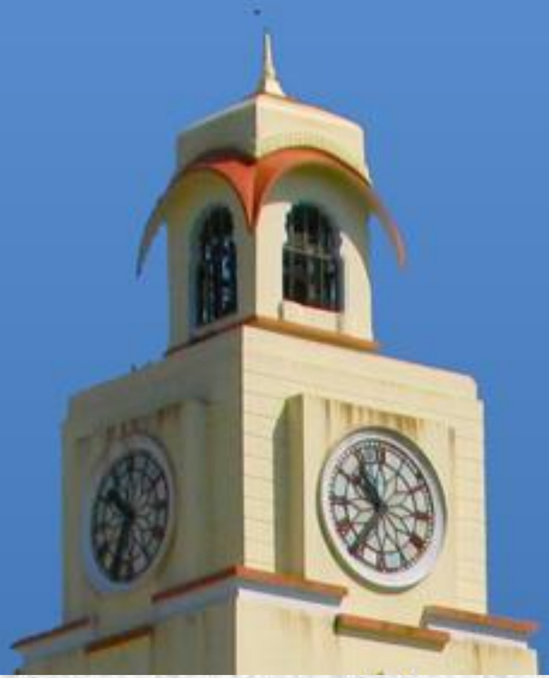
$((q_a, e, A), (q_a, e))$ for each $a \in \{+, *,), \$\}$

Step	State	Unread Input	Stack	Rule of G'
0	p	$\mathsf{id} * (\mathsf{id})\$$	e	
1	q	$\mathsf{id} * (\mathsf{id})\$$	E	
2	q_{id}	$*(\mathsf{id})\$$	E	
3	q_{id}	$*(\mathsf{id})\$$	TE'	1
4	q_{id}	$*(\mathsf{id})\$$	$FT'E'$	4
5	q_{id}	$*(\mathsf{id})\$$	$\mathsf{id}AT'E'$	8
6	q	$*(\mathsf{id})\$$	$AT'E'$	
7	q_*	$(\mathsf{id})\$$	$AT'E'$	
8	q_*	$(\mathsf{id})\$$	$T'E'$	9
9	q_*	$(\mathsf{id})\$$	$*FT'E'$	5
10	q	$(\mathsf{id})\$$	$FT'E'$	
11	$q_($	$\mathsf{id})\$$	$FT'E'$	
12	$q_($	$\mathsf{id})\$$	$(E)T'E'$	7
13	q	$\mathsf{id})\$$	$E)T'E'$	
14	q_{id}	$)\$$	$E)T'E'$	
15	q_{id}	$)\$$	$TE')T'E'$	1
16	q_{id}	$)\$$	$FT'E')T'E'$	4
17	q_{id}	$)\$$	$\mathsf{id}AT'E')T'E'$	8
18	q	$)\$$	$AT'E')T'E'$	
19	$q_)$	$\$$	$AT'E')T'E'$	
20	$q_)$	$\$$	$T'E')T'E'$	10
21	$q_)$	$\$$	$E')T'E'$	6
22	$q_)$	$\$$	$)T'E'$	3
23	q	$\$$	$T'E'$	6
24	$q_\$$	e	$T'E'$	
25	$q_\$$	e	E'	6
26	$q_\$$	e	e	3

(1) $E \to TE'$
(2) $E' \to +TE'$
(3) $E' \to e$
(4) $T \to FT'$
(5) $T' \to *FT'$
(6) $T' \to e$
(7) $F \to (E)$
(8) $F \to \mathsf{id}A$
(9) $A \to e$
(10) $A \to (E)$

Next : ……..

Bottom Up Parsing



innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Thank You