Parsing and Parsing Table

Course Code: CSC3220 Course Title: Compiler Design

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:		**Week No:**		**Semester:**	
Lecturer:					

Lecture Outline

1. Quiz
2. Parsing
3. Parsing Technique(LL1 Grammar)
4. Parsing Table Construction Technique
5. Examples
6. Exercises

Objective and Outcome

Objective:

- To provide an overview of parsing and parsing types.
- To give an overview of predictive parser
- To demonstrate the predictive parsing table construction for predictive / LL(1) parser from a given CFG

Outcome:

- After this lecture the students will be able to understand basics of predictive and LL (1) parser.
- The students will be capable of constructing a predictive parsing table from given CFG

Quiz

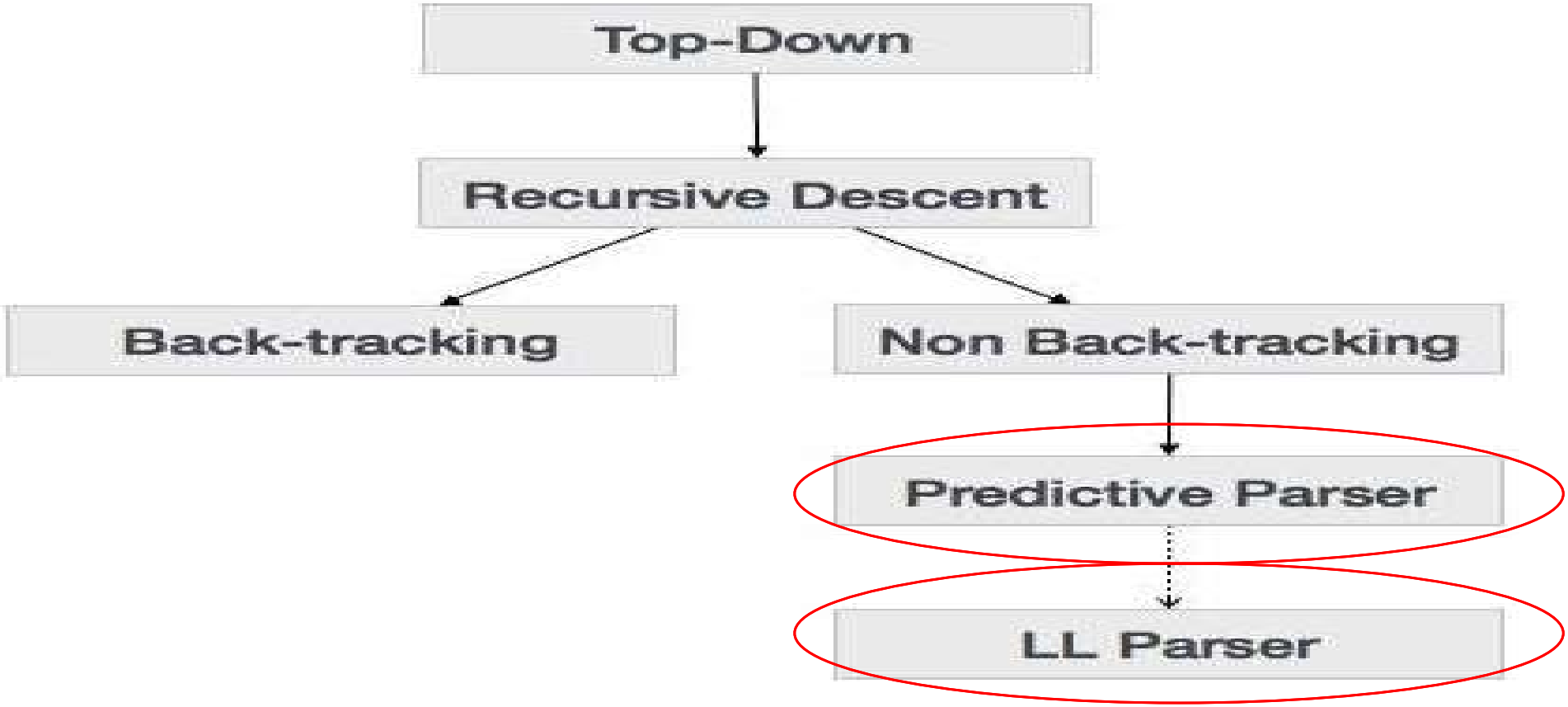

AMERICAN INTERNATIONAL UNIVERSITY
PRAESIDIUM
DISCIPLINA
CIVITATIS
1994
BANGLADESH

- Quiz

Parsing

- The process of determining if a string of terminals (tokens) can be generated by a grammar.
- Time complexity:
  - For any CFG there is a parser that takes at most $O(n^3)$ time to parse a string of n terminals.
  - Linear algorithms suffice to parse essentially all languages that arise in practice.
- Two kinds of methods:
  - Top-down: constructs a parse tree from root to leaves
  - Bottom-up: constructs a parse tree from leaves to root

Types of Parsing



Parsing Table Overview

- A Parsing table collects information from FIRST and FOLLOW set.
- A Parsing table provides a direction/predictive guideline for generating a parse tree from a grammar.
- A Parsing table provide information to create moves made by a predictive parser on a specific input.

LL(k) LL(1) Parser Design Prerequisite

- Make the grammar suitable for top-down parser. By performing the elimination of left recursion. And by performing left factoring.
- Find the FIRST and FOLLOW of the variables.
- Create Parsing table based on the information from FIRST and FOLLOW sets.

Predictive (LL1) Parsing Table Construction Rule

- Collect information from FIRST and FOLLOW sets into a predictive parsing Table M[A,a]
- M[A,a] is a 2D array where
  - A nonterminal
  - A is a terminal or the symbol $, the input endmarker
- The Production A -> a is chosen if the next input symbol a is in First (a).
- If a = ε, we should again choose A-> a, if the current input symbol is in FOLLOW (A) or if the $ on the input has been reached and $ is in the FOLLOW(A)

Predictive (LL1) Parsing Table Construction Rule

- From a Grammar Find out First and Follow
- Take a production; Row should be left hand side and column should be first of right and side
- If we see epsilon in first of right hand side, place the production in follow also
- If first of right hand side terminal, directly place in table
- If the first of right hand side is epsilon, directly place in follow of left hand side

Parsing Table Construction (Example)

LL(1) grammar ('' is ε):

```
E -> T E'
E' -> + T E'
E' -> ''
T -> F T'
T' -> * F T'
T' -> ''
F -> ( E )
F -> id
```

FIRST	FOLLOW	Nonterminal	+	*	(	)	id	$
{(,id}	{$,)}	E			E -> T E'		E -> T E'	
{+,''}	{$,)}	E'	E' -> + T E'			E' -> ''		E' -> ''
{(,id}	{+,$,)}	T			T -> F T'		T -> F T'	
{*,''}	{+,$,)}	T'	T' -> ''	T' -> * F T'		T' -> ''		T' -> ''
{(,id}	{*,+,$,)}	F			F -> (E)		F -> id	

Predictive parsing table for the grammar (Example 1)

S → +SS | *SS | a;

FIRST(s) = {+, *, a}
FOLLOW(s) = {+, *, a, $}

Nonterminal	Input Symbol a	+	*	$
S	S → a	S → +SS	S → *SS	error

Predictive parsing table for the grammar (Example 2)

$$S \rightarrow (S) S \mid \varepsilon$$

$$FIRST(S) = \{(, \varepsilon\}$$
$$FOLLOW(S) = \{), \$\}$$

Nonterminal	Input Symbol (	)	$
S	$S \rightarrow (S)S$	$S \rightarrow \varepsilon$	$S \rightarrow \varepsilon$

Predictive parsing table for the grammar (Example 3)

S → S (S) | ε

FIRST(S) = {(, ε}
FOLLOW(S) = {(,), $}

Nonterminal	Input Symbol (	)	$
S	S → S(S) S → ε	S → ε	S → ε

Parsing Table Construction (Problem)

Consider the following LL(1) grammar, which has the set of terminals *T* = *f***a***;* **b***;* **ep***;* **+***; *; (;)g*. This grammar generates regular expressions over *f*a, b*g*, with **+** meaning the RegExp OR operator, and **ep** meaning the ε symbol. (Yes, this is a context free grammar for generating regular expressions!)

$$
\begin{aligned}
E &\rightarrow TE' \\
E' &\rightarrow +E \mid \epsilon \\
T &\rightarrow FT' \\
T' &\rightarrow T \mid \epsilon \\
F &\rightarrow PF' \\
F' &\rightarrow *F' \mid \epsilon \\
P &\rightarrow (E) \mid \mathbf{a} \mid \mathbf{b} \mid \mathbf{ep}
\end{aligned}
$$

Parsing Table Construction (Solution)

FIRST and FOLLOW sets

$$
\begin{array}{ll}
\text{First}(E) = \{(, a, b, ep\} & \text{Follow}(E) = \{), \$\} \\
\text{First}(E') = \{+, \epsilon\} & \text{Follow}(E') = \{), \$\} \\
\text{First}(T) = \{(, a, b, ep\} & \text{Follow}(T) = \{+,), \$\} \\
\text{First}(T') = \{(, a, b, ep, \epsilon\} & \text{Follow}(T') = \{+,), \$\} \\
\text{First}(F) = \{(, a, b, ep\} & \text{Follow}(F) = \{(, a, b, ep, +,), \$\} \\
\text{First}(F') = \{*, \epsilon\} & \text{Follow}(F') = \{(, a, b, ep, +,), \$\} \\
\text{First}(P) = \{(, a, b, ep\} & \text{Follow}(P) = \{(, a, b, ep, +,), *, \$\}
\end{array}
$$

Parsing Table Construction (Solution)

LL (1) Parsing Table

	$($	$)$	a	b	ep	$+$	$*$	$\$$
E	TE'		TE'	TE'	TE'			
E'		ϵ				$+E$		ϵ
T	FT'		FT'	FT'	FT'			
T'	T	ϵ	T	T	T	ϵ		ϵ
F	PF'		PF'	PF'	PF'			
F'	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	$*F'$	ϵ
P	(E)		a	b	ep			

Lecture References

- Carnegie Mellon University Material

https://www.cs.cmu.edu/~fp/courses/15411-f09/lectures/08-predictive.pdf

- Columbia University Material

http://www1.cs.columbia.edu/~aho/cs4115/lectures/13-02-20.htm

- Online Material

https://www.ques10.com/p/8960/construct-predictive-passing-table-for-following-2/

- Online Tutorial

https://www.tutorialspoint.com/compiler_design/compiler_design_top_down_parser.htm

References/ Books

- 1. Compilers-Principles, techniques and tools (2nd Edition) V. Aho, Sethi and D. Ullman
- 2. Principles of Compiler Design (2nd Revised Edition 2009) A. A. Puntambekar
- 3. Basics of Compiler Design Torben Mogensen