THEORY OF COMPUTATION AND COMPILERS

Unit - II

CONTEXT FREE GRAMMARS AND PARSING

- Introduction
- Context-Free Grammars - Derivation, Parse trees, Ambiguity
- Types of Parsers
- **LL(K) grammars and LL(1) parsing** 
- Bottom-up Parsing - handle pruning
- LR Grammar Parsing
- LALR parsing
- Parsing ambiguous grammars
- Error Recovery in Parsing
- YACC programming specification

Dr. R. Madana Mohana
Professor, Artificial Intelligence & Data Science | I/c-Head, Artificial Intelligence & Machine Learning
CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY
Hyderabad - 500 075, Telangana, INDIA
www.cbit.ac.in

Unit-II: Syntax Analysis (or) Parser

Lecture-17. **Top-Down Parsing**: Predictive Parser (Part-1)

Outline:

- Predictive Parser: Introduction & Model
- Working of the Predictive Parser
- Sequence of Moves made by the Predictive Parser
- Example Problem

Predictive Parser

Predictive Parser / Recursive-Descent Parser without Backtracking / Non-Recursive-Descent Parser:

- **Predictive parser** is a **top-down parser**.
- It is an efficient way of implementing a **recursive-descent parser** by maintaining a **stack** explicitly rather than implicitly via **recursive calls**.
- The **predictive parser** can correctly **guess** or **predict** which production to used if two or more alternative productions are there.

Predictive Parser

Model of Predictive Parser:

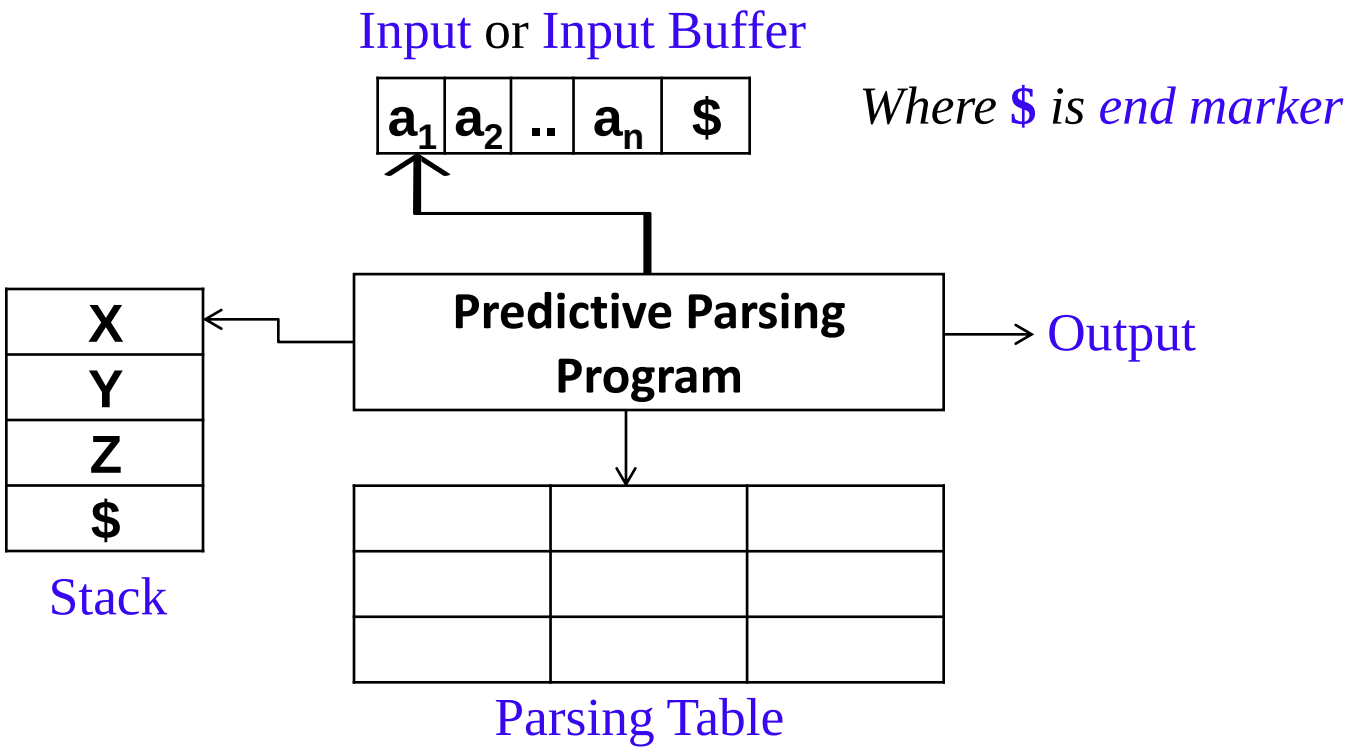

Figure: Model of Predictive Parser

Predictive Parser

Model of Predictive Parser:

Input:

The **input buffer** contains the string to be parsed and the input string ends with **$**. Where **$** indicates the end of the input.

Stack:

It contains sequence of grammar symbols and **$** is placed initially on top of the stack. When **$** is on top of the stack, it indicates that stack is empty.

Predictive Parser

Model of Predictive Parser:

Parsing Table:

It is a two-dimensional array **M[A, a]**. Where **A** is a non-terminal and '**a**' is terminal or **$**. The non-terminal **A** represent the row index and terminal '**a**' represent the column index. The entry in **M[A, a]** contains either a production or blank entry.

Predictive Parser

Model of Predictive Parser:

Parser Program:

It is a program which takes different actions based on **X** which is the symbol on top of the stack and the current input symbol '**a**'.

Output:

As output, the productions that are used are displayed using which the parse tree can be constructed.

Predictive Parser

Working of Predictive Parser:

The various actions performed by the **predictive parser** are shown below:

1. If **X = a = $**, i.e., if the symbol on top of the stack and the current input symbol is equal to **$**, then parsing is successful.
2. If **X = a ≠ $**, i.e., if the symbol on top of the stack is same as the current input symbol '**a**' but not equal to **$**, then pop X from the stack and advance the input pointer to point to next input symbol.

Predictive Parser

Working of Predictive Parser:

3. If **X** is a terminal and **≠ a**, i.e., the symbol on top of the stack is not equal to the current input symbol '**a**', then **error()**.
4. If **X** is a non-terminal and '**a**' is the input symbol, the parser consults the parsing table **M[X, a]** which contains either an **X-Production** or an **error entry**.

If **X → UVW** is the corresponding production, the **parser pops X** from the **stack** and **pushes U**, **V**, **W** in **reverse order** onto the **stack**.

Predictive Parser

Working of Predictive Parser:

The initial configuration of the parser:

Stack	Input
$S	w$

where **S** is the **Starting Variable**.

The final configuration of the parser:

Stack	Input
$	$

Predictive Parser

Example:

Show the sequence of moves made by the predictive parser for the string **id+id*id** during parsing using the given grammar and parsing table.

Given Grammar:

1. E → TE′
2. E′ → +TE′ | ϵ
3. T → FT′
4. T′ → *FT′| ϵ
5. F → (E) | id

Predictive Parser

Example:

Given Predictive Parsing Table:

	id	+	*	(	)	$
E	E → TE′			E → TE′		
E′		E′→ +TE′			E′→ϵ	E′→ϵ
T	T → FT′			T → FT′		
T′		T′→ϵ	T′ → *FT′		T′→ϵ	T′→ϵ
F	F → id			F → (E)		

Predictive Parser

Sequence of moves made by the Predictive Parser:

Stack (X)	Input (a)	Output (M[A, a])	Action
$E	id+id*id$	M[E, id] = E → TE′	Pop E and Push TE′ in reverse.
$E′T	id+id*id$	T → FT′	Pop T and Push FT′ in reverse.
$E′T′F	id+id*id$	F → id	Pop F and Push id.

Predictive Parser

Sequence of moves made by the Predictive Parser:

Stack (X)	Input (a)	Output (M[A, a])	Action
$E′T′id	id+id*id$	Match(id)	Remove id from stack and input; increment input pointer.
$E′T′	+id*id$	T′→ϵ	Pop T′ from stack.
$E′	+id*id$	E′→ +TE′	Pop E′ from stack and push +TE′ in reverse.

Predictive Parser

Sequence of moves made by the Predictive Parser:

Stack (X)	Input (a)	Output (M[A, a])	Action
$E′T+	+id*id$	Match(+)	Remove + from stack and input; increment input pointer.
$E′T	id*id$	T → FT′	Pop T from stack and push FT′ in reverse.
$E′T′F	id*id$	F → id	Pop F from stack and push id.

Predictive Parser

Sequence of moves made by the Predictive Parser:

Stack (X)	Input (a)	Output (M[A, a])	Action
$E′T′id	id*id$	Match(id)	Remove id from stack and input; increment input pointer.
$E′T′	*id$	T′ → *FT′	Pop T′ from stack and push *FT′ in reverse.

Predictive Parser

Sequence of moves made by the Predictive Parser:

Stack (X)	Input (a)	Output (M[A, a])	Action
$E′T′F*	*id$	Match(*)	Remove * from stack and input; increment input pointer.
$E′T′F	id$	F → id	Pop F from stack and push id.

Predictive Parser

Sequence of moves made by the Predictive Parser:

Stack (X)	Input (a)	Output (M[A, a])	Action
$E′T′id	id$	Match(id)	Remove id from stack and input; increment input pointer.
$E′T′	$	T′→ϵ	Pop T′ from stack.
$E′	$	E′→ϵ	Pop E′ from stack.
$	$	ACCEPT	Parsing Successfully done.
Since the stack and input pointer contains $, *the string* id+id*id *is parsed successfully.*			

Summary...

Top-Down Parsing : Predictive Parser (Part-1)

- Predictive Parser: Introduction & Model
- Working of the Predictive Parser
- Sequence of Moves made by the Predictive Parser
- Example Problem

Reading: Aho2, Section 4.4.4

Next Lecture: Predictive parser cont.