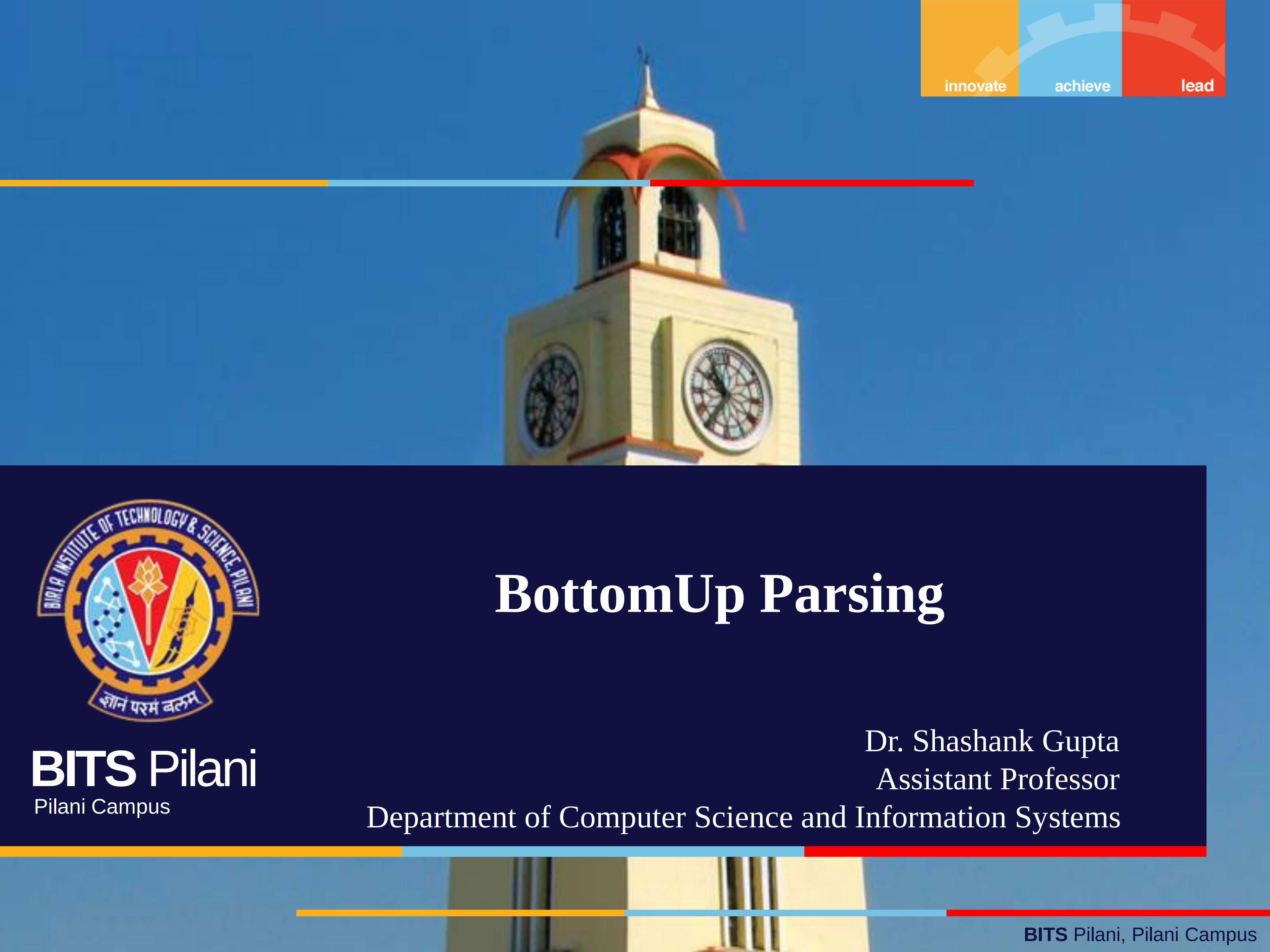

BITS Pilani
Pilani Campus

BottomUp Parsing

Dr. Shashank Gupta
Assistant Professor
Department of Computer Science and Information Systems

Bottom-up Parsing

Design a parse tree for an input string starting from leaf nodes and going towards the root.

- OR

Reduce a string w of input to start symbol of the given grammar.

Example of Bottom-up Parsing

Consider the following grammar and reduce the input string abbcde to the start symbol of Grammar.

$$S \rightarrow a\,A\,B\,e$$

$$A \rightarrow A\,b\,c \mid b$$

$$B \rightarrow d$$

Example of Bottom-up Parsing

$$S \rightarrow a\,A\,B\,e$$

$$A \rightarrow A\,b\,c \mid b$$

$$B \rightarrow d$$

It can be defined as an attempt to reduce the input string w to the start symbol of grammar by tracing out the rightmost derivations of w in reverse.

Reduction

$$a\,\underline{b}\,b\,c\,d\,e$$

$$a\,\underline{A\,b\,c}\,d\,e$$

$$a\,A\,\underline{d}\,e$$

$$\underline{a\,A\,B\,e}$$

$$S$$

Right most derivation

$$S \rightarrow a\,A\,\underline{B}\,e$$

$$\rightarrow a\,\underline{A}\,d\,e$$

$$\rightarrow a\,\underline{A}\,b\,c\,d\,e$$

$$\rightarrow a\,b\,b\,c\,d\,e$$

Bottom-up Parsing

Also known as Shift-Reduce Parsing.

Stack is going to be utilized for performing the reductions.

Shift Reduce Parsing

Split the string being parsed into two parts

- Two parts are separated by a special character “.”
- Left part is a string of terminals and non terminals.
- Right part is a string of terminals.

Initially the input is **.w**

Actions of Shift Reduce Parsing: Shift and Reduce

Shift: It moves terminal symbol from right part of string to the left part of string.

- if string before shift is `α.pqr` then string after shift is `αp.qr`

Reduce: It occurs immediately on the left of "." and identifies a string same as RHS of a production and replaces it by LHS.

- If string before reduce action is `αβ.pqr` and `A -> β` is a production then string after reduction is `αA.pqr`

Shift Reduce Parsing

Symbols on the left of "." are kept on a stack.

- Shift **pushes** a terminal on the stack.

Reduce **pops** symbols (RHS of production) and **pushes** a non terminal (LHS of production) onto the stack.

- **Reduce operation must be executed only if that reduction could lead to a Start symbol of the Grammar.**

Properties of Bottom-up Parsing

It can handle left recursive grammars.

Natural expression of programming language syntax.

Example

Consider the following Grammar and parse the input `id * id + id`

$$E \rightarrow E + E \mid E * E \mid id$$

Example

String to be Parsed id * id + id

$$E \rightarrow E + E \mid E * E \mid id$$

String	Action
.id * id + id	Shift
id. * id + id	Reduce by E->id
E. * id + id	Shift
E*. id + id	Shift
E*id. + id	Reduce by E->id
E*E. + id	Reduce by E->E*E
E. + id	Shift
E+.id	Shift
E+id.	Reduce by E->id
E+E.	Reduce by E->E+E
E.	Accept

String	Action
.id * id + id	Shift
id. * id + id	Reduce by E->id
E. * id + id	Shift
E*. id + id	Shift
E*id. + id	Reduce by E->id
E*E. + id	Shift
E*E+.id	**Shift**
E*E+id.	**Reduce by E->id**
E*E+E.	**Reduce by E->E+E**
E*E.	**Reduce by E->E*E**
E.	**Accept**

Handle

A string that matches right hand side of a production and whose replacement gives a step in the reverse of right most derivation.

Always reduce the handle and not any RHS.

Handle Pruning

- If β is a handle and $A \rightarrow \beta$ is a production then replace β by A
- A right most derivation in reverse can be obtained by handle pruning.

Shift Reduce Parsers

It is the process of detecting the handles and reducing them.

Different bottom-up parsers differ in the way they detect handles.

Conflicts

Shift-Reduce Conflict

- Which action to take if both shift and reduce are valid?

Reduce-Reduce Conflict

- Which rule to use for production if reduction is possible by more than one rule?

Shift-Reduce Conflict

Consider the Grammar and the input `id + id * id`

$$E \rightarrow E + E \mid E * E \mid id$$

Stack	Input	Action
E+E	***id**	**Reduce by E->E+E**
E	*id	Shift
E*	id	Shift
E*id		Reduce by E->id
E*E		Reduce by E->E*E
E		

Stack	Input	Action
E+E	***id**	**Shift**
E+E*	id	Shift
E+E*id		Reduce by E->id
E+E*E		Reduce by E->E*E
E+E		Reduce by E->E+E
E		

Reduce-Reduce Conflict

Consider the Grammar and the input

$$M \rightarrow R+R \mid R+c \mid R$$

$$R \rightarrow c$$

Stack	Input	Action
	c+c	Shift
c	+c	Reduce by R->c
R	+c	Shift
R+	c	Shift
R+c		**Reduce by R->c**
R+R		Reduce by M->R+R
M		

Stack	Input	Action
	c+c	Shift
c	+c	Reduce by R->c
R	+c	Shift
R+	c	Shift
R+c		**Reduce by M->R+c**
M		

Issues in Bottom-up Parser

Whether to shift or reduce?

Which production to use for reduction?