CSC 3201 Compiler Construction

Department of Computer Science

SZABIST (Islamabad Campus)

Week 5 (Lecture 2)

Predictive Parsing

- If a top down parser picks the wrong production, it may need to backtrack.
- Alternative: Look-ahead in input and use context to pick the production to use correctly.
- How much look-ahead is needed? In general, an arbitrarily large amount of look-ahead symbols are required.
- Fortunately, large classes of CFGs can be parsed with limited lookahead.

Predictive Parsing

- A top-down method of syntax analysis in which a set of recursive procedures is used to process the input.
- One procedure is associated with each nonterminal of a grammar.
- The sequence of procedure calls during the analysis of an input string implicitly defines a parse tree for the input, and can be used to build an explicit parse tree, if desired.

Predictive Parsing

```
1.  goal   → expr
2.  expr   → term expr’
3.  expr’  → + term expr’
4.  expr’  → - term expr’
5.         | ε
6.  term   → factor term’
7.  term’  → * factor term’
8.  term’  → / factor term’
9.         | ε
10. Factor → number
11.        | id
12.        | (expr)
```

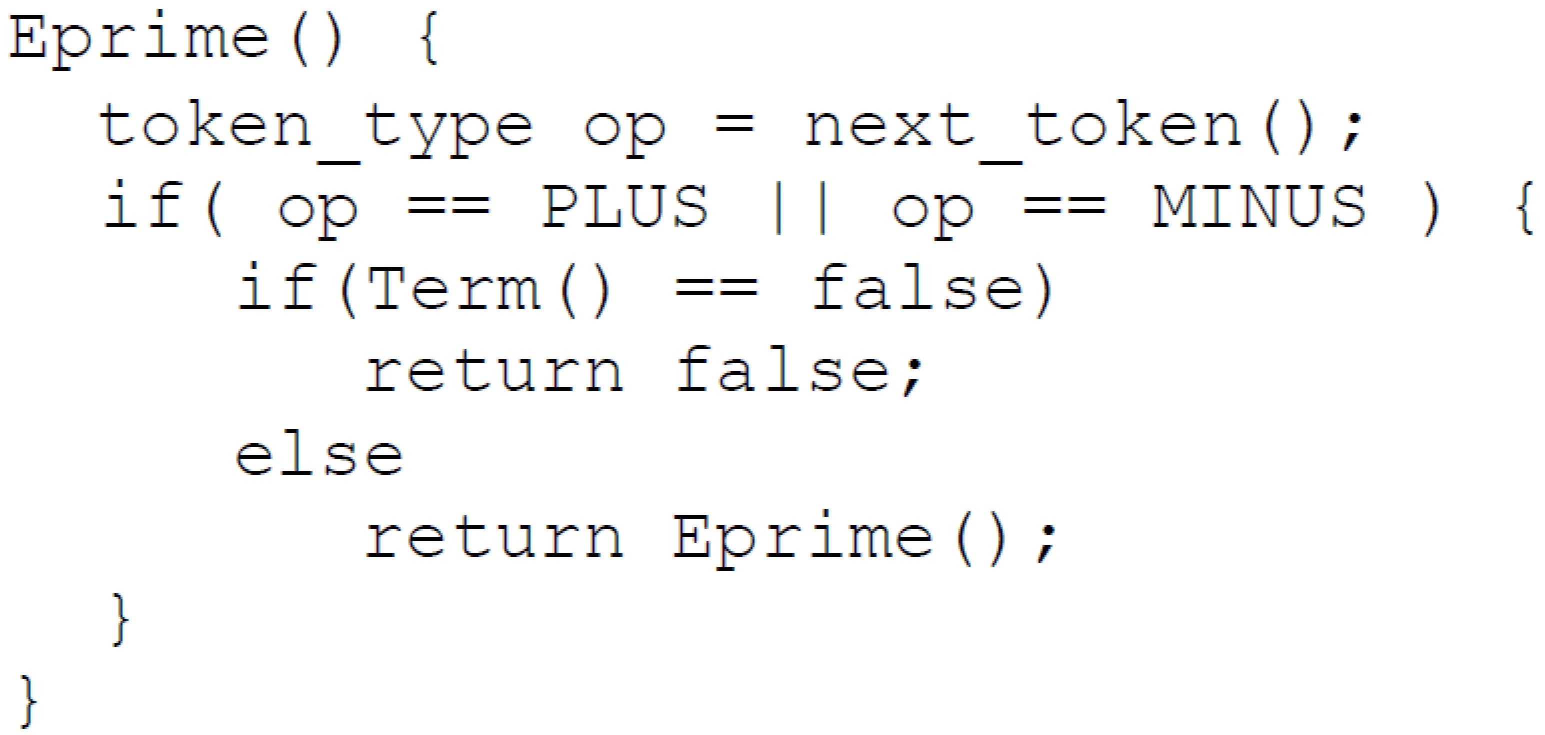
Predictive Parsing

```
Goal() {
  token = next_token();
  if(Expr() == true && token == EOF)
     next compilation step
  else {
    report syntax error;
    return false;
  }
}
```

Predictive Parsing

```
Expr()
{
  if(Term() == false)
     return false;
  else
    return Eprime();
}
```

Predictive Parsing

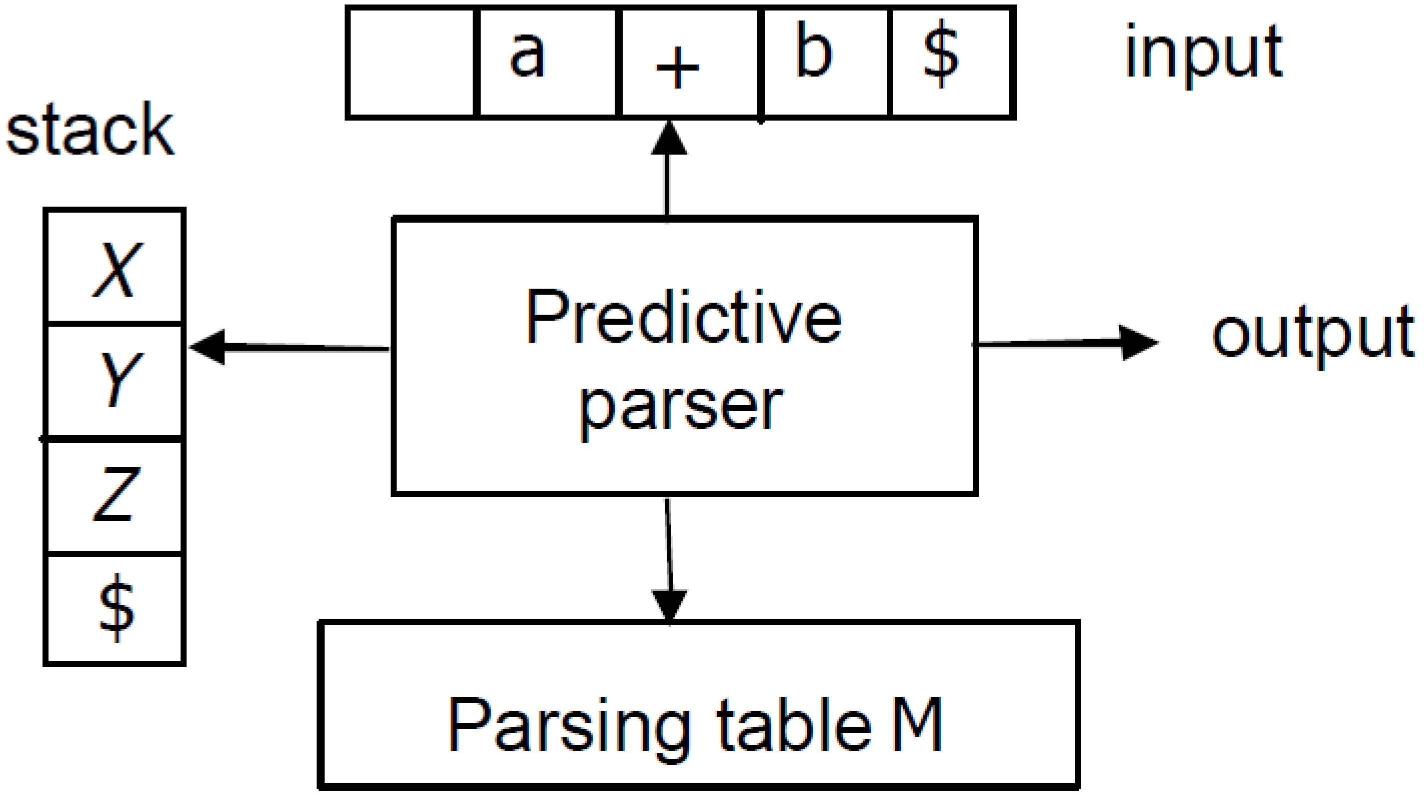


```
Eprime() {
  token_type op = next_token();
  if( op == PLUS || op == MINUS ) {
     if(Term() == false)
        return false;
     else
        return Eprime();
  }
}
```

Predictive Parsing

- Non-recursive predictive parser:
  - Done by maintaining an explicit stack and using a table.
  - Such a parser is called a table-driven parser.
  - The non-recursive LL(1) parser looks up the production to apply by looking up a parsing table.
  - The LL(1) table has one dimension for current non-terminal to expand and another dimension for next token.
  - Each table cell contains one production.

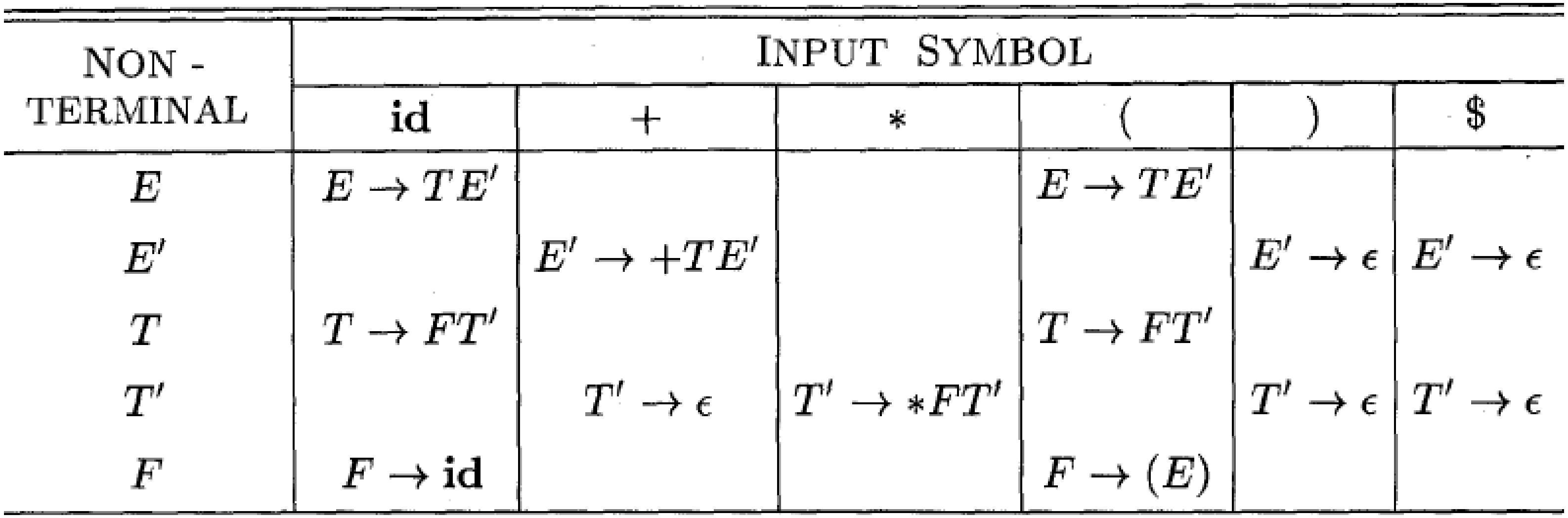
Predictive Parsing



Predictive Parsing

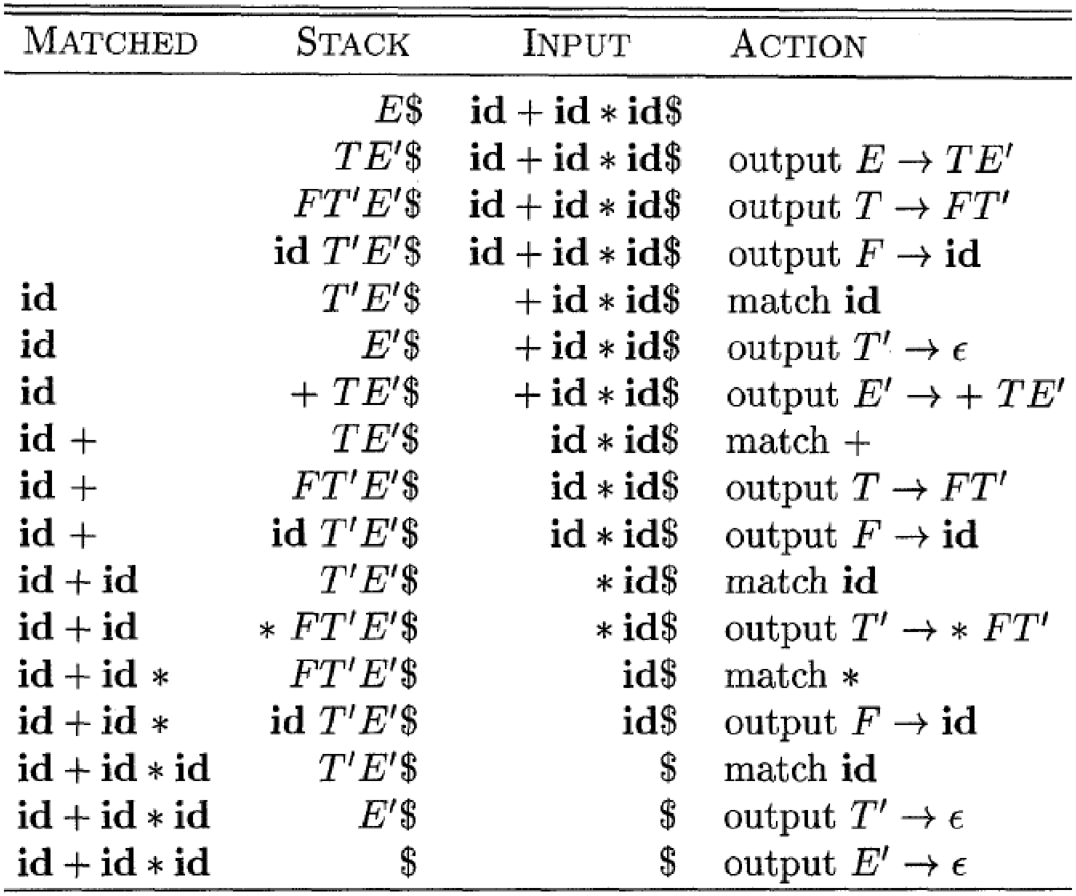
$$
\begin{array}{lcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E'\ \mid\ \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T'\ \mid\ \epsilon \\
F & \rightarrow & (\ E\ )\ \mid\ \mathbf{id}
\end{array}
$$

Predictive Parsing



NON - TERMINAL	INPUT SYMBOL					
	id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \epsilon$	$E' \to \epsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \epsilon$	$T' \to *FT'$		$T' \to \epsilon$	$T' \to \epsilon$
F	$F \to \mathbf{id}$			$F \to (E)$		

Predictive Parsing



MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	output $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	output $T \to FT'$
	$\mathbf{id}\ T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	output $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+ \mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+ \mathbf{id} * \mathbf{id}\$$	output $T' \to \epsilon$
$\mathbf{id}$	$+ TE'\$$	$+ \mathbf{id} * \mathbf{id}\$$	output $E' \to + TE'$
$\mathbf{id} +$	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match $+$
$\mathbf{id} +$	$FT'E'\$$	$\mathbf{id} * \mathbf{id}\$$	output $T \to FT'$
$\mathbf{id} +$	$\mathbf{id}\ T'E'\$$	$\mathbf{id} * \mathbf{id}\$$	output $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$T'E'\$$	$* \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$* FT'E'\$$	$* \mathbf{id}\$$	output $T' \to * FT'$
$\mathbf{id} + \mathbf{id} *$	$FT'E'\$$	$\mathbf{id}\$$	match $*$
$\mathbf{id} + \mathbf{id} *$	$\mathbf{id}\ T'E'\$$	$\mathbf{id}\$$	output $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$T'E'\$$	$\$$	match $\mathbf{id}$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$E'\$$	$\$$	output $T' \to \epsilon$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$\$$	$\$$	output $E' \to \epsilon$