Parsing

Aditya Thakur

Top-down parsing

- Constructs parse tree for input string starting from the root
- Finds the leftmost derivation for an input string

Context-free grammar for arithmetic expressions

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \mid \varepsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \varepsilon \\
F &\rightarrow (E) \mid id
\end{aligned}
$$

Can we derive the string $id + id * id$?

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \mid \varepsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow * F\ T' \mid \varepsilon \\
F &\rightarrow (E) \mid id
\end{aligned}
$$

$E \rightarrow T\ E'$

$E'\ \rightarrow\ +T\ E' \mid\ \varepsilon$

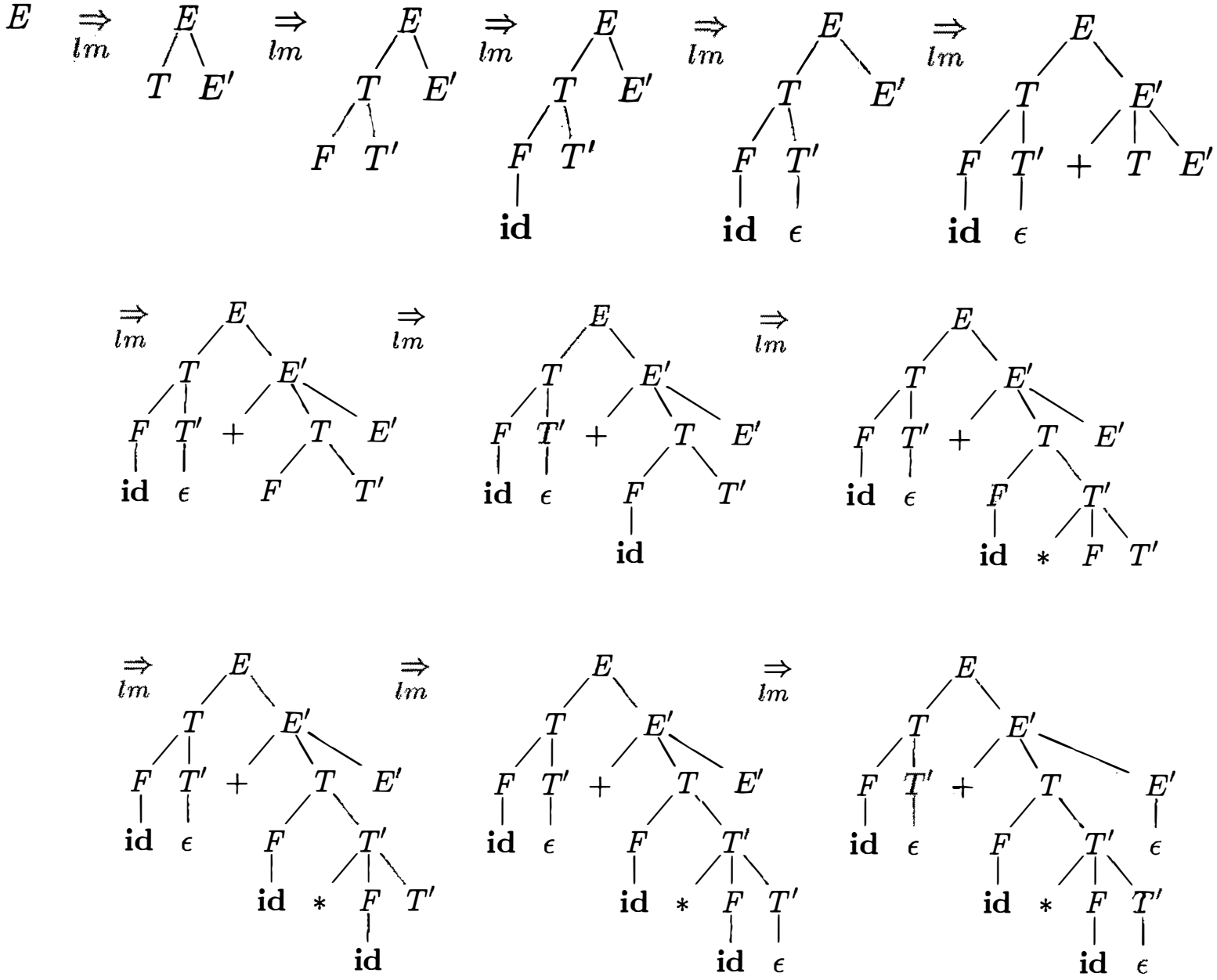
$T\ \rightarrow F\ T'$

$T' \rightarrow *\ F\ T' \mid\ \varepsilon$

$F\ \rightarrow (E) \mid id$

E

$$E \rightarrow T\ E'$$
$$E' \rightarrow +T\ E' \mid \varepsilon$$
$$T \rightarrow F\ T'$$
$$T' \rightarrow *\ F\ T' \mid \varepsilon$$
$$F \rightarrow (E) \mid id$$



Top-down parsing

- Constructs parse tree for input string starting from the root
- Finds the leftmost derivation for an input string
- Two types:
  - Recursive descent parsing
  - Predictive parsing

Recursive-descent parsing

- Consists of a procedure for each nonterminal
- Execution begins with the procedure for the start symbol

```
    void A() {
1)      Choose an A-production, A → X1 X2 ··· Xk;
2)      for ( i = 1 to k ) {
3)          if ( Xi is a nonterminal )
4)              call procedure Xi();
5)          else if ( Xi equals the current input symbol a )
6)              advance the input to the next symbol;
7)          else /* an error has occurred */;
        }
    }
```

Nondeterministic, might require backtracking

Typical procedure for nonterminal A

Predictive parser

- Top-down parser that can correctly choose A-production by looking at at the next k symbols in the input
- No backtracking during parsing

LL(1) parser

- Predictive parser that only looks at the next input symbol
- Derives the leftmost derivation

LL(1) parser

- Constructs parsing table using FIRST and FOLLOW sets of the grammar

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	$F \to \mathbf{id}$			$F \to (E)$		

FIRST sets

- $FIRST(\alpha)$ is the set of terminals that begin strings derived from the string of grammar symbols α

FIRST sets

- $FIRST(\alpha)$ is the set of terminals that begin strings derived from the string of grammar symbols α
- If X is a terminal, $FIRST(X) = \{X\}$
- If X is a nonterminal and $X \rightarrow Y_1 Y_2 \ldots Y_k$ is a production, then everything in $FIRST(Y_1)$ is in $FIRST(X)$.
  If Y_1 derives ε, then add $FIRST(Y_2)$ to $FIRST(X)$, and so on.
  If ε is in all $FIRST(Y_j)$ then add ε to $FIRST(X)$
- If $X \rightarrow \varepsilon$ is a production, add ε to $FIRST(X)$.

FIRST sets

- $FIRST(\alpha)$ is the set of terminals that begin strings derived from the string of grammar symbols α

$$
\begin{aligned}
&E \rightarrow T\ E' \\
&E' \rightarrow\ +T\ E' \mid\ \varepsilon \\
&T\ \rightarrow F\ T' \\
&T' \rightarrow *\ F\ T' \mid\ \varepsilon \\
&F\ \rightarrow (E) \mid id
\end{aligned}
$$

FIRST sets

- $FIRST(\alpha)$ is the set of terminals that begin strings derived from the string of grammar symbols α

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \mid \varepsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \varepsilon \\
F &\rightarrow (E) \mid id
\end{aligned}
$$

	First
E	
E’	
T	
T’	
F	

FIRST sets

- $FIRST(\alpha)$ is the set of terminals that begin strings derived from the string of grammar symbols α

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \mid \varepsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \varepsilon \\
F &\rightarrow (E) \mid id
\end{aligned}
$$

	First
E	
E’	
T	
T’	
F	(id

FIRST sets

- $FIRST(\alpha)$ is the set of terminals that begin strings derived from the string of grammar symbols α

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \mid \varepsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \varepsilon \\
F &\rightarrow (E) \mid id
\end{aligned}
$$

	First
E	
E’	
T	(id
T’	
F	(id

FIRST sets

- $FIRST(\alpha)$ is the set of terminals that begin strings derived from the string of grammar symbols α

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \mid \varepsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \varepsilon \\
F &\rightarrow (E) \mid id
\end{aligned}
$$

	First
E	(id
E’	
T	(id
T’	
F	(id

FIRST sets

- $FIRST(\alpha)$ is the set of terminals that begin strings derived from the string of grammar symbols α

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \mid \varepsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \varepsilon \\
F &\rightarrow (E) \mid id
\end{aligned}
$$

	First
E	(id
E’	+ ε
T	(id
T’	
F	(id

FIRST sets

- $FIRST(\alpha)$ is the set of terminals that begin strings derived from the string of grammar symbols α

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \mid \varepsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \varepsilon \\
F &\rightarrow (E) \mid id
\end{aligned}
$$

	First
E	(id
E’	+ ε
T	(id
T’	* ε
F	(id

FOLLOW

- $FOLLOW(A)$ is the set of terminals that can appear immediately to the right of a nonterminal A in a derivation

FOLLOW

- $FOLLOW(A)$ is the set of terminals that can appear immediately to the right of a nonterminal A in a derivation
- \$ is in $FOLLOW(S)$ where S is the start symbol and \$ is an endmarker
- If there is a production $A \rightarrow \alpha B \beta$, then $FIRST(\beta) \backslash \varepsilon$ is in $FOLLOW(B)$
- If there is a production $A \rightarrow \alpha B$ or $A \rightarrow \alpha B \beta$, where $FIRST(\beta)$ contains ε, then everything in $FOLLOW(A)$ is in $FOLLOW(B)$.

FOLLOW

- $FOLLOW(A)$ is the set of terminals that can appear immediately to the right of a nonterminal A in a derivation

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +T\ E' \mid \varepsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \varepsilon \\
F &\rightarrow (E) \mid id
\end{aligned}
$$

	FOLLOW
E	) $
E’	) $
T	+) $
T’	+) $
F	+ *) $

Parsing table

- Parsing table $M: N \times T \rightarrow R$ where N is the set of nonterminals, T is the set of terminals, R is the set of production rules

For each production $A \rightarrow \alpha$:

- For each terminal a in $FIRST(\alpha)$, $\mathrm{M[A, a]} = A \rightarrow \alpha$
- If $\varepsilon \in FIRST(\alpha)$ and $b \in FOLLOW(A)$, $M[A, b] = A \rightarrow \alpha$

$$E \rightarrow T\ E'$$
$$E' \rightarrow +T\ E' \mid \varepsilon$$
$$T \rightarrow F\ T'$$
$$T' \rightarrow *\ F\ T' \mid \varepsilon$$
$$F \rightarrow (E) \mid id$$

	First
E	(id
E’	+ ε
T	(id
T’	* ε
F	(id

	FOLLOW
E	) $
E’	) $
T	+) $
T’	+) $
F	+ *) $

NON - TERMINAL	INPUT SYMBOL					
	id	+	*	(	)	$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow$ **id**			$F \rightarrow (E)$		

Table-driven predictive parser

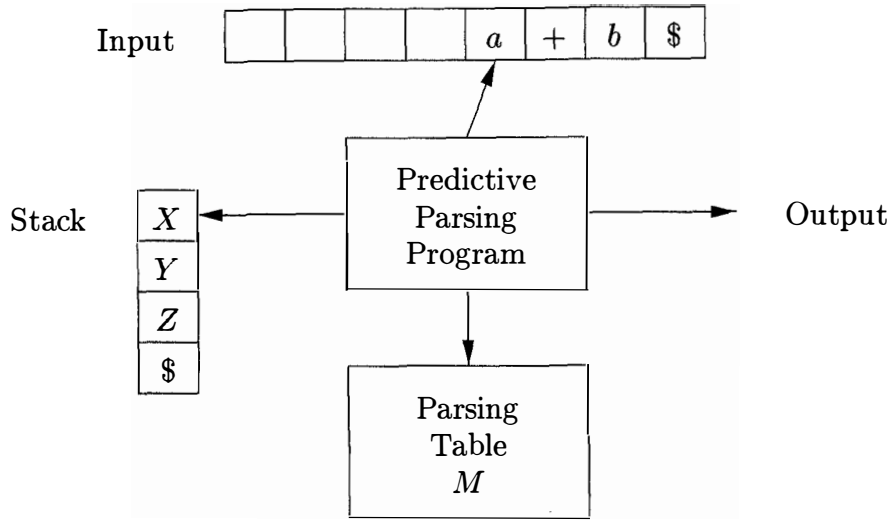
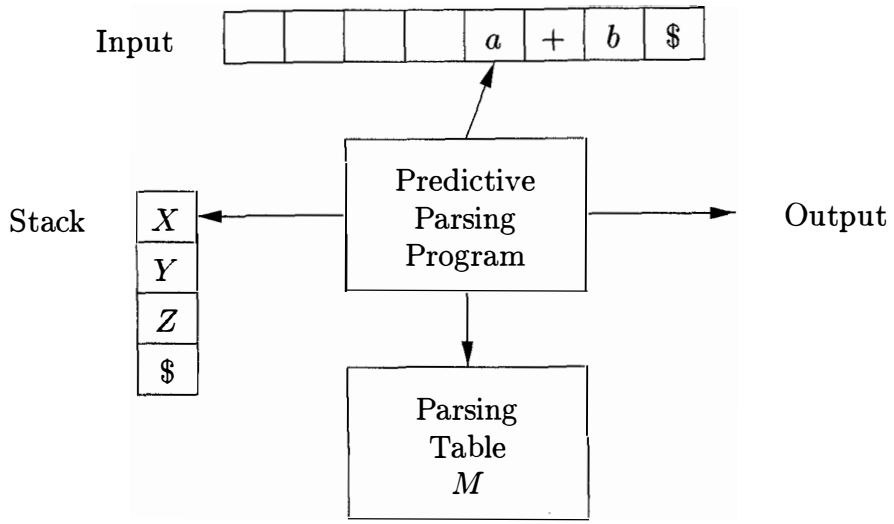


Table-driven predictive parser



INPUT: A string w and a parsing table M for grammar G.

OUTPUT: If w is in $L(G)$, a leftmost derivation of w; otherwise, an error indication.

set ip to point to the first symbol of w;
set X to the top stack symbol;
while ($X \neq \$$) { /* stack is not empty */
    if (X is a) pop the stack and advance ip;
    else if (X is a terminal) *error*();
    else if ($M[X, a]$ is an error entry) *error*();
    else if ($M[X, a] = X \rightarrow Y_1 Y_2 \cdots Y_k$) {
        output the production $X \rightarrow Y_1 Y_2 \cdots Y_k$;
        pop the stack;
        push $Y_k, Y_{k-1}, \ldots, Y_1$ onto the stack, with Y_1 on top;
    }
    set X to the top stack symbol;
}

Deriving the string $id + id * id$

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	$F \to \mathbf{id}$			$F \to (E)$		

Parsing table

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	

Moves made by the parser

Deriving the string $id + id * id$

Parsing table

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	$F \to \mathbf{id}$			$F \to (E)$		

Moves made by the parser

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	output $E \to TE'$

Deriving the string $id + id * id$

Parsing table

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	$F \to \mathbf{id}$			$F \to (E)$		

Moves made by the parser

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 $T \to FT'$

Deriving the string $id + id * id$

Parsing table

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	$F \to \mathbf{id}$			$F \to (E)$		

Moves made by the parser

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 $F \to \mathbf{id}$

Deriving the string $id + id * id$

Parsing table

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	$F \to \mathbf{id}$			$F \to (E)$		

Moves made by the parser

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 $F \to \mathbf{id}$
id	$T'E'\$$	$+\ \mathbf{id} * \mathbf{id}\$$	match **id**

Deriving the string $id + id * id$

Parsing table

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	$F \to \mathbf{id}$			$F \to (E)$		

Moves made by the parser

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 $F \to \mathbf{id}$
id	$T'E'\$$	$+\ \mathbf{id} * \mathbf{id}\$$	match **id**
id	$E'\$$	$+\ \mathbf{id} * \mathbf{id}\$$	output $T' \to \epsilon$

Deriving the string $id + id * id$

Parsing table

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	$F \to \mathbf{id}$			$F \to (E)$		

Moves made by the parser

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 $F \to \mathbf{id}$
id	$T'E'\$$	$+\ \mathbf{id} * \mathbf{id}\$$	match **id**
id	$E'\$$	$+\ \mathbf{id} * \mathbf{id}\$$	output $T' \to \epsilon$
id	$+\ TE'\$$	$+\ \mathbf{id} * \mathbf{id}\$$	output $E' \to +\ TE'$

Deriving the string $id + id * id$

Parsing table

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	$F \to \mathbf{id}$			$F \to (E)$		

Moves made by the parser

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 $F \to \mathbf{id}$
id	$T'E'\$$	$+\ \mathbf{id} * \mathbf{id}\$$	match **id**
id	$E'\$$	$+\ \mathbf{id} * \mathbf{id}\$$	output $T' \to \epsilon$
id	$+\ TE'\$$	$+\ \mathbf{id} * \mathbf{id}\$$	output $E' \to +\ TE'$
id +	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match +

Deriving the string $id + id * id$

Parsing table

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	$F \to \mathbf{id}$			$F \to (E)$		

Moves made by the parser

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

Deriving the string $id + id * id$

Parsing table

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	$F \to \mathbf{id}$			$F \to (E)$		

Moves made by the parser

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

Deriving the string $id + id * id$

Parsing table

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	$F \to \mathbf{id}$			$F \to (E)$		

Moves made by the parser

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

Deriving the string $id + id * id$

Parsing table

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	$F \to \mathbf{id}$			$F \to (E)$		

Moves made by the parser

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

Deriving the string $id + id * id$

Parsing table

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	$F \to \mathbf{id}$			$F \to (E)$		

Moves made by the parser

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

Deriving the string $id + id * id$

Parsing table

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	$F \to$ **id**			$F \to (E)$		

Moves made by the parser

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

Deriving the string $id + id * id$

Parsing table

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	$F \to$ **id**			$F \to (E)$		

Moves made by the parser

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

Deriving the string $id + id * id$

Parsing table

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	$F \to$ **id**			$F \to (E)$		

Moves made by the parser

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

Deriving the string $id + id * id$

Parsing table

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	$F \to \mathbf{id}$			$F \to (E)$		

Moves made by the parser

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 $E' \to \epsilon$

Not all grammars are LL(1)

Grammar

$$
\begin{aligned}
S &\rightarrow iEtSS' \mid a \\
S' &\rightarrow eS \mid \epsilon \\
E &\rightarrow b
\end{aligned}
$$

Parsing table

NON-TERMINAL	INPUT SYMBOL					
	a	b	e	i	t	$\$$
S	$S \rightarrow a$			$S \rightarrow iEtSS'$		
S'			$S' \rightarrow \epsilon$ $S' \rightarrow eS$			$S' \rightarrow \epsilon$
E		$E \rightarrow b$				

Multiple rules possible in the same entry in the parsing table, because of ambiguity.

All LL(1) grammars are unambiguous.

Optional reading

Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. Compilers: principles, Techniques and tools.

Q: What type of parser is used in the Go compiler?