

Top Down Parsing Issues

Consider: `procedure id (param list) ;` param list is optional

where `param list => param : type; param : type;...param:type`
`param => var id, id, ..., id`
var is optional

Context-free Grammar:

```
S -> procedure id P ; | ε
P -> ( L ) | ε
L -> R : T | R : T ; L
R -> V D
V -> var | ε
D -> D , id | id
T -> int | real
```

String: `procedure print (var x,y,z: int; a,b: real);`

Recursive-descent Parsing

CFG: $T \rightarrow T * F \mid F$
 $F \rightarrow (E) \mid id$

$\Rightarrow T \rightarrow F T'$
 $T' \rightarrow * F T' \mid \epsilon$
 $F \rightarrow (E) \mid id$

T()
{

F()
{

}

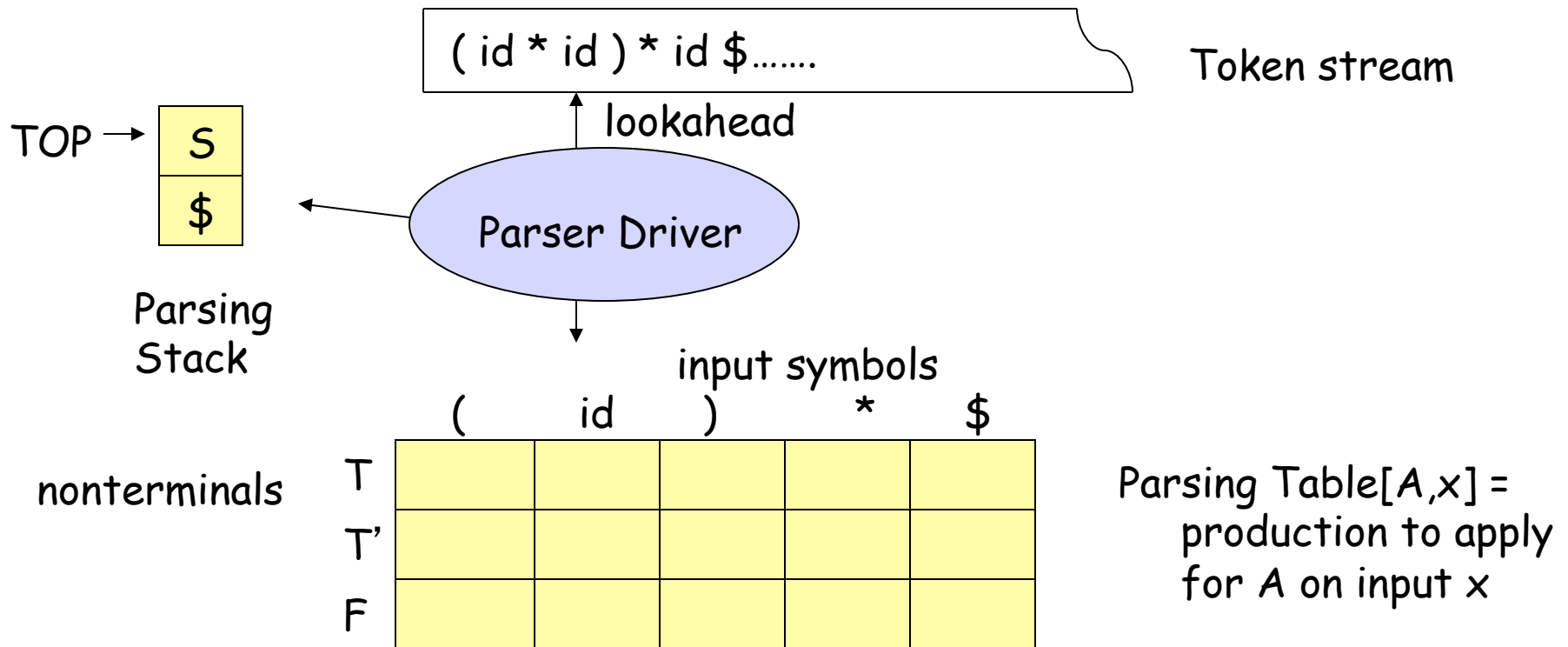
T'()
{

}

}

Consider input string: $a * b * c$

Table-driven Top Down Parsing



- TOP = lookahead = \$ -> ACCEPT!
- TOP = lookahead <> \$ -> POP stack; ADVANCE lookahead;
- TOP is terminal <> lookahead -> error
- TOP is nonterminal -> Lookup(Table[TOP,lookahead])
 For production $X \rightarrow Y_1 Y_2 \dots Y_n$
 POP X; PUSH $Y_n Y_{n-1} \dots Y_2 Y_1$

Predictive Parser (Top Down)

Nonterminal	Input Token					
	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	$F \rightarrow id$			$F \rightarrow (E)$		

Let input token stream be: (id + id) * id \$

Initial Stack:



E
\$

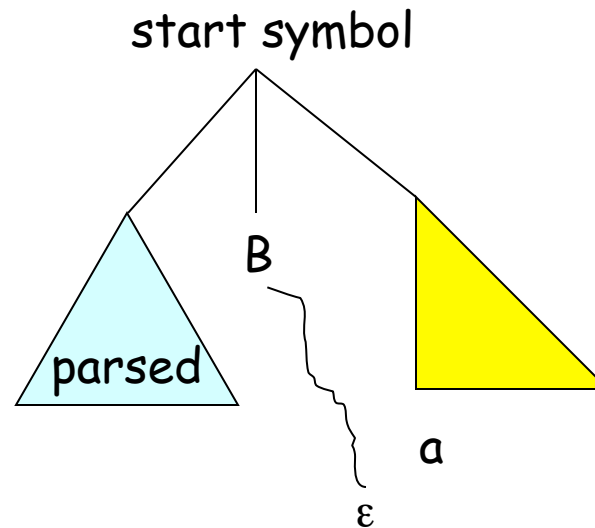
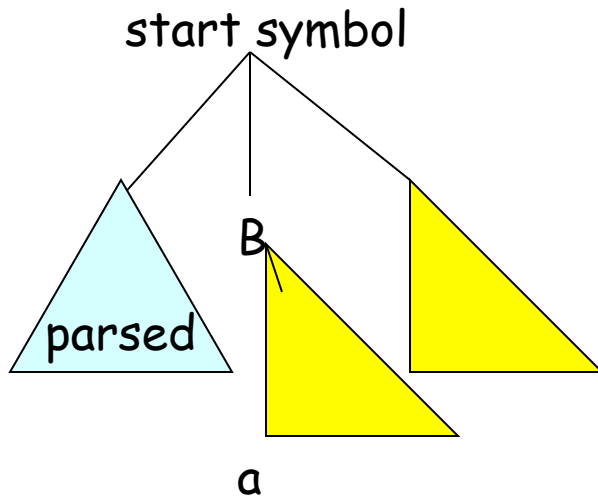
Predictive LL(1) Parse Table Build

Key Insight:

Given input “a” and nonterminal B to be expanded, which one of the alternatives

$$B \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$$

is the unique choice to derive a string starting with “a”?



Computing FIRST and FOLLOW

FIRST(α) = set of terminals that can begin strings derived by α

*

FIRST(α) = { a | $\alpha \Rightarrow a\beta$ for some β }

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \varepsilon$

$T \rightarrow FT'$

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

$F \rightarrow (E) \mid id$

FOLLOW(N) = set of terminals that can immediately follow N in right sentential form

FOLLOW(N):

For $A \rightarrow \alpha N \beta$, Add FIRST(β), except ε , to FOLLOW(N)

For $A \rightarrow \alpha N \beta$ and FIRST(β) has ε , or $A \rightarrow \alpha N$,

Add FOLLOW(A) to FOLLOW(N)

Add \$ to FOLLOW(START SYMBOL)

Let's look at some grammars...

Example 1:

$$\begin{aligned} S &\rightarrow ABC \\ A &\rightarrow a \mid Cb \mid \varepsilon \\ B &\rightarrow c \mid dA \mid \varepsilon \\ C &\rightarrow e \mid f \end{aligned}$$

Example 2:

$$\begin{aligned} S &\rightarrow uBDz \\ B &\rightarrow Bv \mid w \\ D &\rightarrow EF \\ E &\rightarrow y \mid \varepsilon \\ F &\rightarrow x \mid \varepsilon \end{aligned}$$

LL(1) Parse Table Construction

Look at each production, $A \rightarrow \alpha$, and $\text{FIRST}(\alpha)$:

- If terminal a in $\text{FIRST}(\alpha) \implies$

$$\begin{array}{c} a \\ \hline A \mid A \rightarrow \alpha \end{array}$$

- If ϵ in $\text{FIRST}(\alpha)$, then

 If terminal b in $\text{FOLLOW}(A) \implies$

$$\begin{array}{c} b \\ \hline A \mid A \rightarrow \alpha \end{array}$$

 If $\$$ in $\text{FOLLOW}(A) \implies$

$\$$

$$\hline A \mid A \rightarrow \alpha$$

Nonterminal	id	+	*	()	\$
-------------	----	---	---	---	---	----

E
E'
T
T'
F

Is a Grammar LL(1)?

Method:

- * Construct table and look for multidefined entries.
If no multidefined entries, then LL(1) grammar
- * Look at FIRST and FOLLOW sets as follows:

A grammar G is LL(1) iff

whenever there exists $A \rightarrow \alpha \mid \beta$ in G ,
all of the following conditions hold true:

- * $\text{FIRST}(\alpha) \cap \text{FIRST}(\beta) = \emptyset$
- * At most 1 of α and β derive the empty string
- * If β derives the empty string, then
 $\text{FIRST}(\alpha) \cap \text{FOLLOW}(A) = \emptyset$

Summary: Top-down Parsing

- * To avoid backtracking:
 - no left recursion, no common prefixes, no ambiguity -> rewrite
- * Easy to write parser:
 - but sometimes difficult to structure grammar to be LL(1)
- * Error detection:
 - terminal on TOP not equal terminal on input
 - no table entry for [TOP, input]
- * Error Recovery:
 - panic mode:
 - skip input until input in FOLLOW(TOP) or FIRST(TOP)
 - pop terminal and pretend match
 - phrase level
 - error calls in table