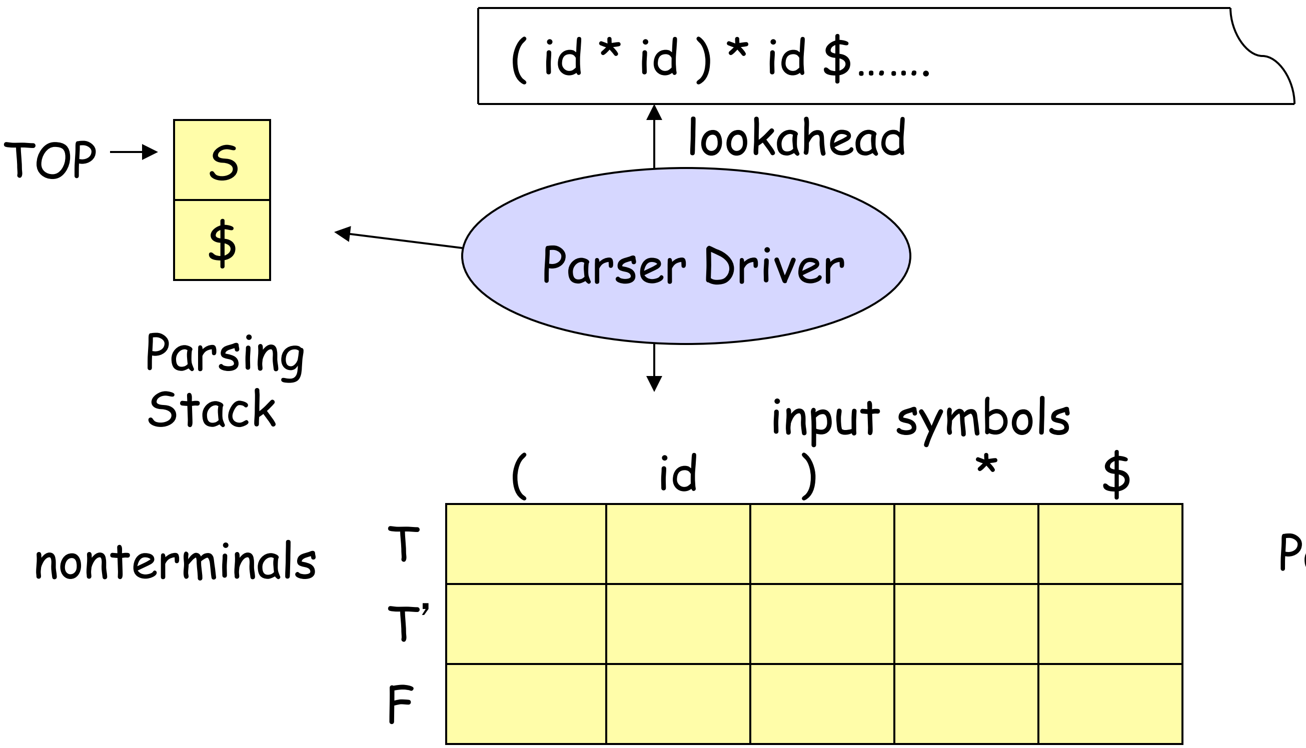
Class 9

Table-driven Top Down Parsing



Token stream

Parsing Table[A,x] =
production to apply
for A on input x

TOP = lookahead = $ -> ACCEPT!
TOP = lookahead <> $ -> POP stack; ADVANCE lookahead;
TOP is terminal<> lookahead -> error
TOP is nonterminal -> Lookup(Table[TOP,lookahead])
For production X -> Y1 Y2 … Yn
POP X; PUSH Yn Yn-1… Y2 Y1

Predictive Parser (Top Down)

Nonterminal Input Token

	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)		

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

↑

Initial Stack:

E
$

Any Questions?

- how the top-down parsing works?

- what you need to do to the grammar to use a top-down parser that is predictive (non-backtracking)

On to how to build that parse table…

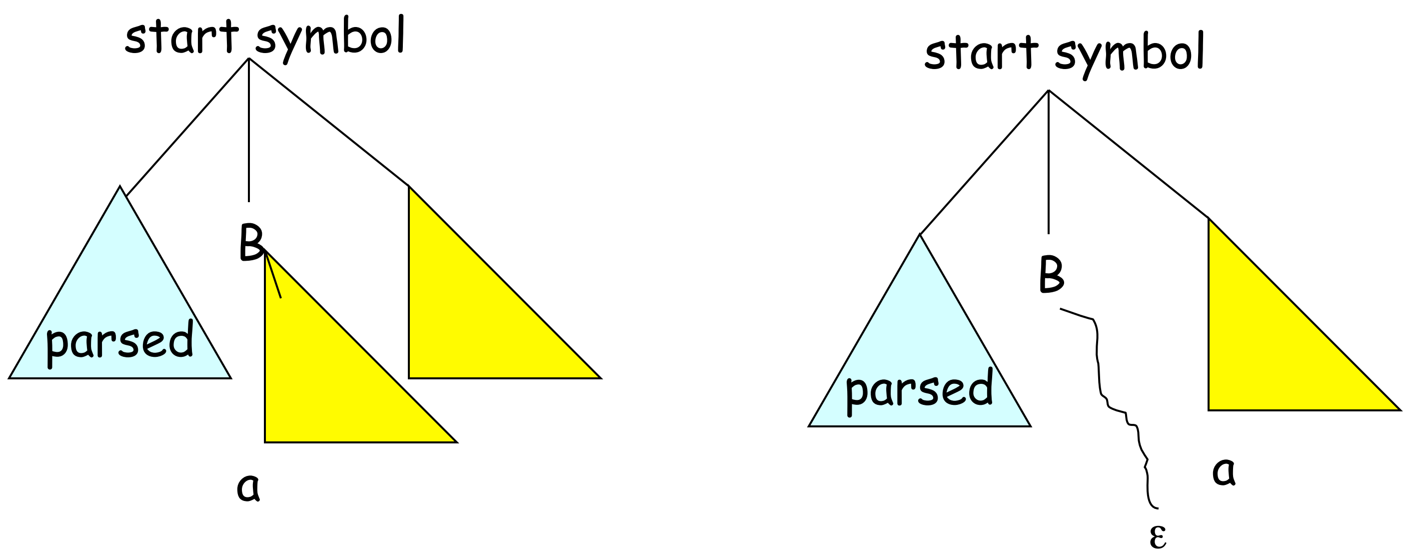
Predictive LL(1) Parse Table Build

Key Insight:

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

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

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 | $\alpha \overset{*}{\Rightarrow} a\beta$ for some β }

E -> T E'
E' -> + T E' | ε
T -> FT'
T' -> *FT' | ε
F -> (E) | id

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

FOLLOW(N):

For A -> α N β, Add FIRST(β), except ε, to FOLLOW(N)

For A -> α N β and FIRST(β) has ε, or A -> αN, Add FOLLOW(A) to FOLLOW(N)

Add $ to FOLLOW(START SYMBOL)

Let's look at some grammars…

Example 1:

S -> ABC
A -> a | Cb | ε
B -> c | dA | ε
C -> e | f

Example 2:

S -> uBDz
B -> Bv | w
D -> E F
E -> y | ε
F -> x | ε