

## Section: LL Parsing

### LL(k) Parser:

- top-down parser - starts with start symbol on stack, and repeatedly replace nonterminals until string is generated.
- predictive parser - predict next rewrite rule
- first L of LL means - read input string left to right
- second L of LL means - produces leftmost derivation
- k - number of lookahead symbols

## LL parsing process:

- convert CFG to PDA (different method than before)
- Use the PDA and lookahead symbols
- Lookahead symbol is next symbol in input string

## Convert CFG to NPDA

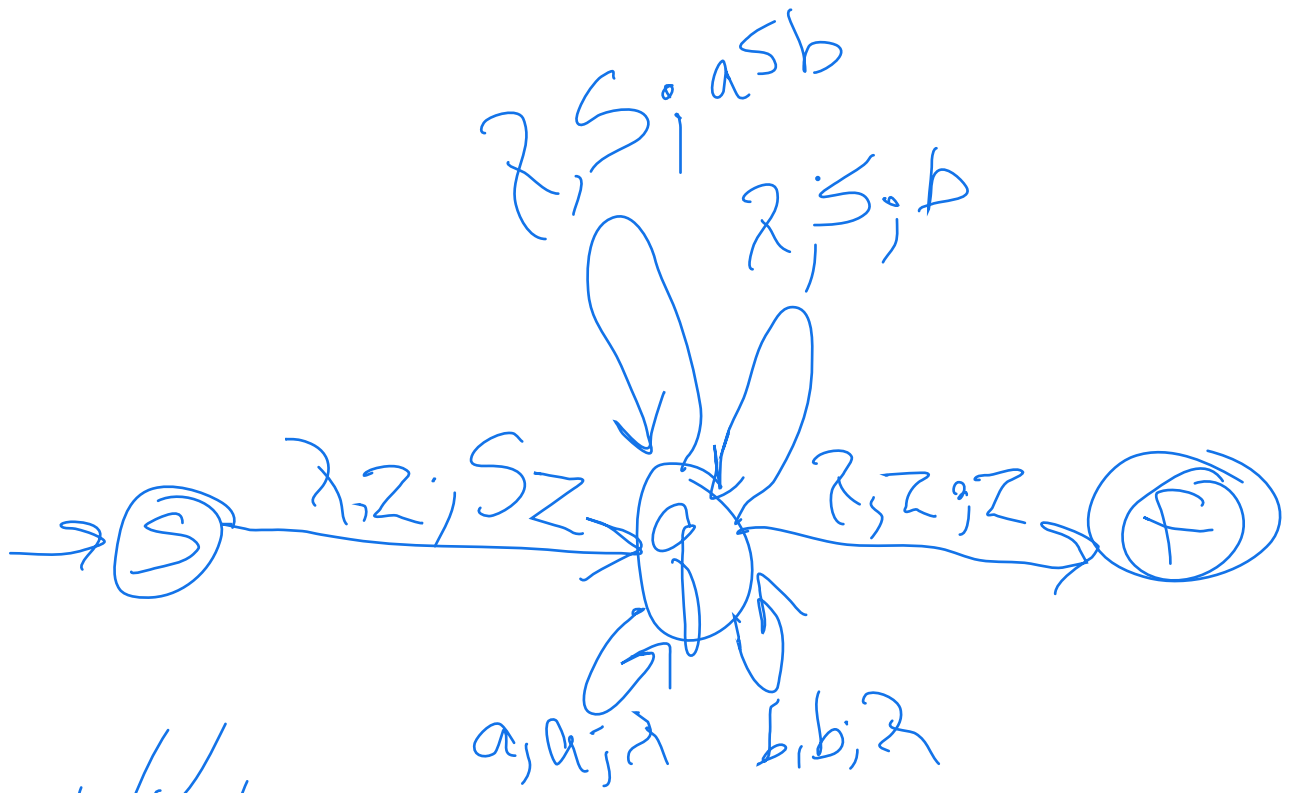
### The constructed NPDA:

- Three states:  $s$ ,  $q$ ,  $f$   
start in state  $s$   
push  $S$  on stack, move into  $q$   
all rewrite rules in state  $q$ : If lhs of rewrite rule on top of stack, replace it with rhs of rewrite rule and stay in state  $q$   
additional rules in  $q$  to recognize terminals: read input symbol, pop input symbol, stay in state  $q$   
pop  $z$  from stack, move into  $f$ , accept

### Example:

$$L = \{a^n b b^n : n \geq 0\}$$

$S \rightarrow aSb \mid b$



tracce

$\checkmark \checkmark \checkmark \checkmark$   
~~a a b b b b~~  
~~↑ ↑ ↑ ↑ ↑ ↑~~  
 a  
 S  
 S b  
 S b b b b b b  
z z z z z z z z z

a  
 S S b  
 S b b b b  
 b b b b b b  
z z z z z z z z z

$aaabbb \in L$

```

state = s
push(S)
state = q
read(symbol)
while top-of-stack  $\neq$  z do
  case top-of-stack of
    S: if symbol == a then
        {pop(); push(aSb)}
      else if symbol == b then
        {pop(); push(b)}
      else error
    a: if symbol  $\neq$  a, then error
        else {pop(); read(symbol)}
    b: if symbol  $\neq$  b, then error
        else {pop(); read(symbol)}
  end case
end while
pop()
if symbol  $\neq$  $ then error
state = f

```

## LL Parse Table - 2-dim array

- rows - variables
- cols - terminals, \$ (end of string marker)
- $LL[i,j]$

contains rhs of a rule

### Example: Parse table for

$$L = \{a^n b b^n : n \geq 0\}$$

$$S \rightarrow aSb \mid b$$

	a	b	\$
S	aSb	b	

blank means error

## A generic parsing routine

```
push(S)
read(symbol)
while stack not empty do
  case top-of-stack of
    terminal:
      if top-of-stack == symbol
        then {pop(); read(symbol)}
      else
        error
    variable:
      if LL[top-of-stack, symbol] ≠ error
        then {pop()
              push(LL[top-of-stack,symbol])}
      else
        error
  end case
end while
if symbol ≠ $, then error
```

**Example:**

$$\begin{aligned} S &\rightarrow aSb \\ S &\rightarrow c \end{aligned}$$

	a	b	c	\$
S	aSb	error	c	error

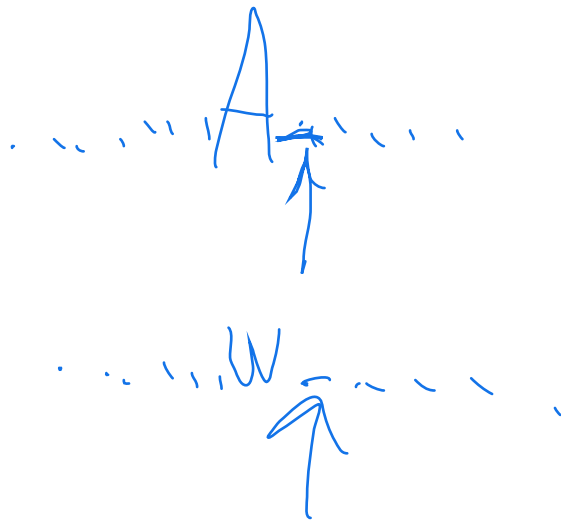
**Example:**

$$\begin{aligned} S &\rightarrow Ac \mid Bc \\ A &\rightarrow aAb \mid \lambda \\ B &\rightarrow b \end{aligned}$$



To construct an LL parse table  
 $LL[\text{rows}, \text{cols}]$ :

1. For each rule  $A \rightarrow w$ 
  - (a) For each  $a$  in  $\text{FIRST}(w)$   
add  $w$  to  $LL[A, a]$
  - (b) If  $\lambda$  is in  $\text{FIRST}(w)$   
add  $w$  to  $LL[A, b]$  for each  $b$  in  
 $\text{FOLLOW}(A)$
2. Each undefined entry is error.



Example:

$$\begin{aligned} S &\rightarrow aSc \mid B \\ B &\rightarrow b \mid \lambda \end{aligned}$$

	FIRST	FOLLOW
S	a,b, $\lambda$	\$,c
B	b, $\lambda$	\$,c

To Compute the LL Parse Table for this example:

- For  $S \rightarrow aSc$ ,  
 $\text{FIRST}(aSc) = \{a\}$
- For  $S \rightarrow B$ ,  
 $\text{FIRST}(B) = \{b, \lambda\}$   
 $\text{FOLLOW}(S) = \{\$, c\}$
- For  $B \rightarrow b$ ,  
 $\text{FIRST}(b) = \{b\}$
- For  $B \rightarrow \lambda$ ,  
 $\text{FIRST}(\lambda) = \{\lambda\}$

## LL(1) Parse Table:

	a	b	c	\$
S	aSc	B	B	B
B		b	$\lambda$	$\lambda$

Parse string: aacc

Trace aabcc



		a						
	a	S	S	B	b			
	S	S	c	c	c	c	c	

Stack: S c c c c c c c c

last char

symbol: a a a' a' b b b c c' \$

second a

aabcc ∈ L

Example: Construct Parse Table for:

$$S \rightarrow AcB$$

$$A \rightarrow aAb$$

$$A \rightarrow \lambda$$

$$B \rightarrow aBb$$

$$B \rightarrow c$$

$$\text{FIRST}(A) = \{a, \lambda\}$$

$$\text{FIRST}(S) = \{a, c\}$$

$$\text{FIRST}(B) = \{a, c\}$$

$$\text{FOLLOW}(A) = \{b, c\}$$

$$\text{FOLLOW}(S) = \{\#\}$$

$$\text{FOLLOW}(B) = \{b, \#\}$$

$$S \# \rightarrow ACB\#$$

To compute the parse table:

- For  $S \rightarrow AcB$ ,  
 $\text{FIRST}(AcB) = \{a, c\}$
- For  $A \rightarrow aAb$ ,  
 $\text{FIRST}(aAb) = \{a\}$
- For  $A \rightarrow \lambda$ ,  
 $\text{FIRST}(\lambda) = \{\epsilon\}$
- For  $B \rightarrow aBb$ ,  
 $\text{FIRST}(aBb) = \{a\}$
- For  $B \rightarrow c$ ,  
 $\text{FIRST}(c) = \{c\}$
- All other entries are errors.

# LL(1) Parse Table:

	a	b	c	\$
S	A <sub>c</sub> B		A <sub>c</sub> B	
A	aAb	$\lambda$	$\lambda$	
B	aBb		c	

In JFLAP, start with grammar, Then "Input", then build LL(1) Parse Table

The screenshot shows the JFLAP software interface with the 'Build LL(1) Parse' window open. The window title is 'JFLAP: <untitled3>'. The menu bar includes 'File', 'Input', 'Test', 'Convert', and 'Help'. The 'Editor' tab is active, and the 'Build LL(1) Parse' sub-tab is selected. Below the menu bar are buttons for 'Do Selected', 'Do Step', 'Do All', 'Next', and 'Parse'. There are three 'Table Text Size' sliders. The main area is divided into two panes. The left pane shows the grammar rules: S → AcB, A → aAb, A → λ, B → aBb, and B → c. The right pane shows the generated LL(1) parse table. It has columns for 'FIRST' and 'FOLLOW' sets, and a grid for the parse table entries. The 'FIRST' and 'FOLLOW' sets are: A: {λ, a}, B: {a, c}, S: {a, c}; FOLLOW: A: {b, c}, B: {b, \$}, S: {\$}. The parse table grid has columns for terminals 'a', 'b', 'c', '\$' and rows for non-terminals 'A', 'B', 'S'. The entries are: A: aAb, λ, λ; B: aBb, c; S: AcB, AcB.

	FIRST	FOLLOW
A	{λ, a}	{b, c}
B	{a, c}	{b, \$}
S	{a, c}	{\$}

	a	b	c	\$
A	aAb	$\lambda$	$\lambda$	
B	aBb		c	
S	AcB		AcB	

# Use table to parse string abcacb

JFLAP : <untitled3>

File Input Test Convert Help

Editor Build LL(1) Parse LL(1) Parsing

Table Text Size

	a	b	c	\$
A	aAb	$\lambda$	$\lambda$	
B	aBb		c	
S	AcB		AcB	

Start Step Derivation Table

Input abcacb

Input Remaining abcacb\$

Stack aAbcB

Input Field Text Size (For optimization, move one o...)

---

Table Text Size

LHS		RHS
S	→	AcB
A	→	aAb
A	→	$\lambda$
B	→	aBb
B	→	c

Table Text Size

	S
S → AcB	AcB
A → aAb	aAbcB

Replacing A with aAb.



# Example:

$$S \rightarrow AcB$$

$$A \rightarrow aAb \mid ab$$

$$B \rightarrow aBb \mid acb$$

	FIRST	FOLLOW
S	a	
A	a	
B	a	

	a	b	c	\$
S	AcB			
A	a <sup>ab</sup> aAb			
B	a <sup>acb</sup> aBb			

not good

one lookahead  
doesn't work

not LL(1)

# LL(2) Parse Table:

use 2 lookahead

	aa	ab	ac	a\$	b	c	\$
S	AcB	AcB	error	error	error	error	error
A	aAb	ab	error	error	error	error	error
B	aBb	error	acb	error	error	error	error

this is LL(2) grammar

parse string: aabbcacb

			a	a							
			A	A	b	b					
		A	b	b	b	b	b		a		
		c	c	c	c	c	c	c	c	c	
Stack:	<u>S</u>	<u>B</u>	<u>B</u>	<u>B</u>	<u>B</u>	<u>B</u>	<u>B</u>	<u>B</u>	<u>B</u>	<u>b</u>	<u>b</u>
symbol:	aa	aa	aa	ab	ab	bb	bc	ca	ac	ac	cb

↑ 2 lookahead

**Example:**

$$L = \{a^n : n \geq 0\} \cup \{a^n b^n : n \geq 0\}$$

$$\mathbf{S} \rightarrow \mathbf{A}$$

$$\mathbf{S} \rightarrow \mathbf{B}$$

$$\mathbf{A} \rightarrow \mathbf{aA}$$

$$\mathbf{A} \rightarrow \lambda$$

$$\mathbf{B} \rightarrow \mathbf{aBb}$$

$$\mathbf{B} \rightarrow \lambda$$

This is  $LL(k)$  what  
is  $k$ ?  
not  $LL(k)$  for any  $k$

**Example:**

$$L = \{a^n : 0 \leq n \leq 10\} \cup \{a^n b^n : 0 \leq n \leq 10\}$$

LL(1)

# Example

$$\begin{aligned} S &\rightarrow bbCd \mid Bcc \rightarrow k=5 \\ B &\rightarrow bB \mid b \rightarrow 2 \\ C &\rightarrow cC \mid c \rightarrow 2 \end{aligned}$$

what is  $k$ ?  $L(k)$

$$k=5$$

bbcccd\$

$S \rightarrow bbcd$

bbcc\$

$S \rightarrow Bcc$