Creating an LL(1) Parser

- 1. Analyze grammar *G* assuring that the **Predict Sets** of production rules grouped by common LHS are **pairwise disjoint**.
- 2. Create an *LLT*, the *parsing table*.
- 3. Using LLTabularParsing generate a parse tree from a token stream providing tuples (TOKENTYPE, srcValue) eg: (TYPE, int), (STRING, "RofL"), (REAL, 3.14159).

The text's algorithms for steps 1 and 2 are straight forward and reasonable.

Predict Sets

Predict Set The **predict set** for a rule $A \to \alpha\beta$ is $First(\alpha\beta) \underbrace{\bigcup Follow(A)}_{\text{if } \alpha\beta \text{ and } \alpha\beta}$

#	$p \in P$	Computed By	Predict Set		
1	$S \rightarrow AM\$$	FirstSet(RHS)	b, m, n, p, s		
2	$A \rightarrow BC$	FirstSet(RHS)	b		
3	$A \rightarrow CM$	FirstSet(RHS)	m, n, p, s		
4	$B \rightarrow b g h$	FirstSet(RHS)	b		
5	$C \rightarrow s t$	FirstSet(RHS)	S		
6	$C o\lambda$	FollowSet(LHS)	m, n, p		
7	$M \rightarrow m$	FirstSet(RHS)	m		
8	$M \rightarrow n$	FirstSet(RHS)	n		
9	$M \rightarrow p$	FirstSet(RHS)	р		

An LL(1) Parsing Table

LL(1) Parsing Table An LL(1) parsing table (LLT) has **rows** for the non-terminals of N, **columns** for the terminals of Σ and Σ .

Each **cell** identifies the production rule $p \in P$ to use in the next step of a derivation to verify the correctness (syntax) of input symbols (tokens).

Look ma! we're gonna be parsing soon!

Predict Sets

#	$p \in P$	Computed By	Predict Set		
1	$S \rightarrow AM$ \$	FirstSet(RHS)	b, m, n, p, s		
2	$A \rightarrow BC$	FirstSet(RHS)	b		
3	$A \rightarrow CM$	FirstSet(RHS)	m, n, p, s		
4	$B \rightarrow b g h$	FirstSet(RHS)	b		
5	$C \rightarrow s t$	FirstSet(RHS)	S		
6	$C o \lambda$	FollowSet(LHS)	m, n, p		
7	$M \rightarrow m$	FirstSet(RHS)	m		
8	$M \rightarrow n$	FirstSet(RHS)	n		
9	$M \rightarrow p$	FirstSet(RHS)	р		

LL(1) Parsing Table

	b	g	h	m	n	р	S	t	\$
S	1			1	1	1	1		
\boldsymbol{A}	2			3	3	3	3		
M				7	8	3 9			
$\begin{array}{c c} B \\ C \end{array}$	4								
C				6	6	6	5		

Operation: begin

Parse Tree

Root

The **QUEUE** is the input sequence of *tokens*. Push the grammar **starting symbol** *S* onto a **STACK**. Create the root node of the raw **PARSE TREE**, make it the *Current* node in the tree.

QUEUE

STACK

(b)

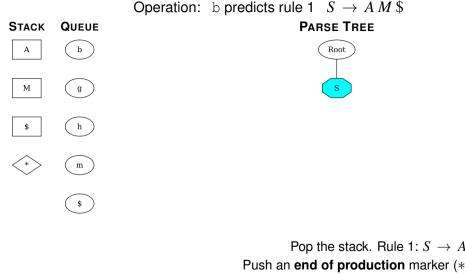
g

h

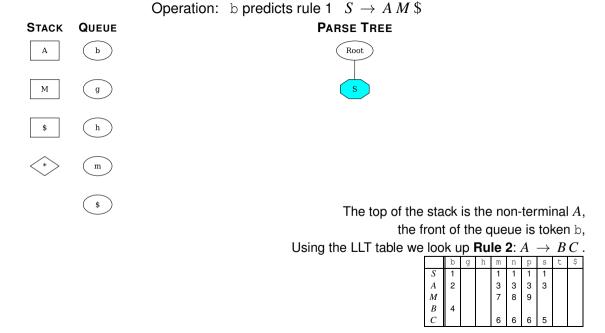
m

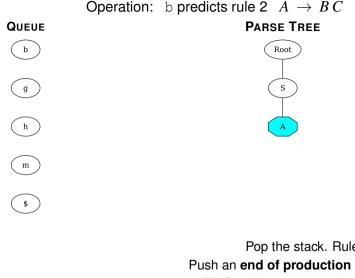
(\$

Operation: begin											
STACK	QUEUE	Parse Tree									
S	b	Root									
	g										
	h	The Queue is the input sequence of <i>tokens</i> .									
	m	Push the grammar starting symbol S onto a STACK. Create the root node of the raw PARSE TREE,									
make it the Current node in th											
	The stack top is the non-terminal S the front of the queue is token be										
	Using the LLT table we look up Rule 1 : $S \rightarrow AM$ \$.										
		$\left egin{array}{c c} M \\ B \end{array} \right \left \begin{array}{c c} 7 \end{array} \right \left \begin{array}{c c} 8 \end{array} \right \left \begin{array}{c c} 9 \end{array} \right \left \begin{array}{c c} \end{array} \right $									



Pop the stack. Rule 1: $S \to AM$ \$ is predicted. Push an **end of production** marker (*) onto the stack, push the RHS of the rule onto the stack **from right to left**, add the LHS as the right-most child of the *Current* tree node, make this new child the *Current* node of the tree.





Pop the stack. Rule 2: $A \rightarrow BC$ is predicted. Push an **end of production** marker (*) onto the stack, push the RHS of the rule onto the stack **from right to left**, add the LHS as the right-most child of the *Current* tree node, make this new child the *Current* node of the tree.

Operation: b predicts rule 2 $A \to BC$ STACK QUEUE

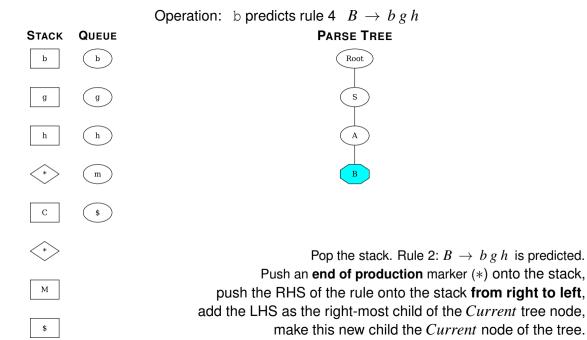
PARSE TREE

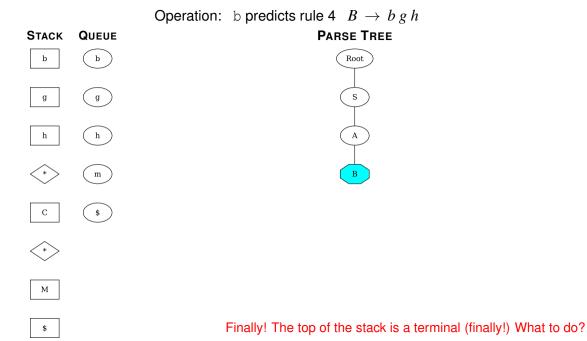
Root

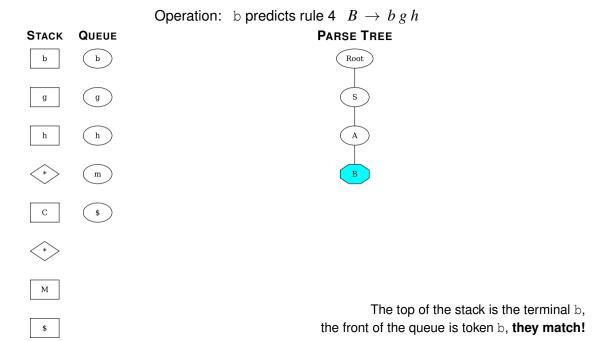
S
S

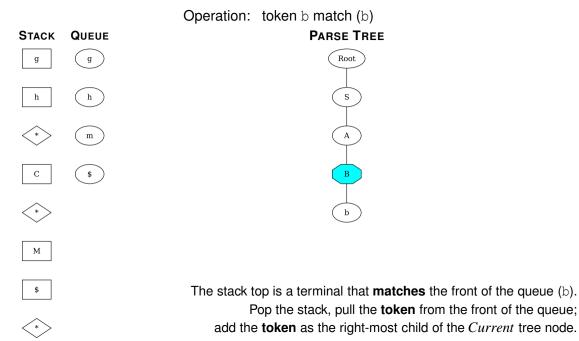


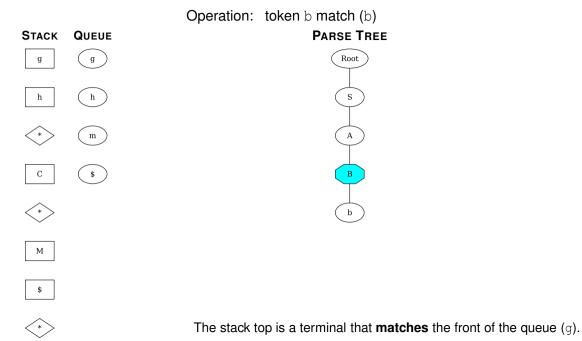


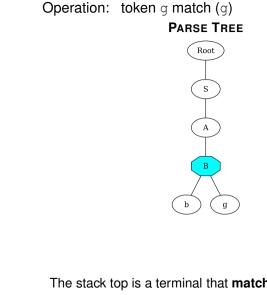








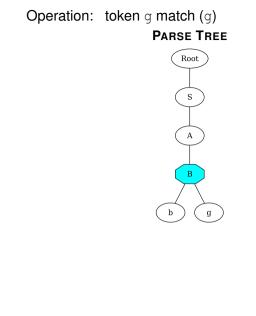




M

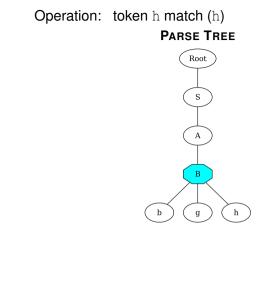
QUEUE

The stack top is a terminal that **matches** the front of the queue (g). Pop the stack, pull the **token** from the front of the queue; add the **token** as the right-most child of the *Current* tree node.



QUEUE

The stack top is a terminal that **matches** the front of the queue (h).



M

QUEUE

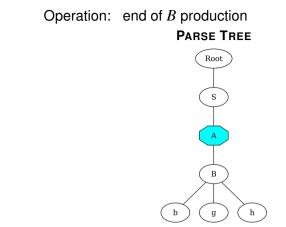
The stack top is a terminal that **matches** the front of the queue (h).

Pop the stack, pull the **token** from the front of the Queue; add the **token** as the right-most child of the *Current* tree node.

Operation: token h match (h) PARSE TREE QUEUE Root

STACK

The top of the stack is the **end of production** marker.

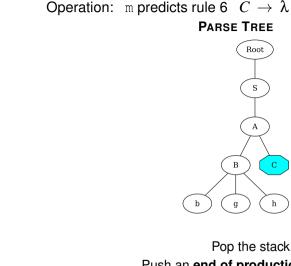


QUEUE

The top of the stack is the **end of production** marker, pop the stack, and set the **parent** of Current as the new current node. $Current \leftarrow Current.parent$

STACK QUEUE PARSE TREE Root The top of the stack is the non-terminal C, the front of the gueue is token m, Using the LLT table we look up Rule 6: $C \, o \, \lambda$.

Operation: end of *B* production



M

QUEUE

Pop the stack. Rule 6: $C \to \lambda$ is predicted. Push an **end of production** marker (*) onto the stack, push the RHS of the rule onto the stack **from right to left**, add the LHS as the right-most child of the *Current* tree node, make this new child the *Current* node of the tree.

Operation: m predicts rule 6 $C \rightarrow \lambda$

STACK QUEUE





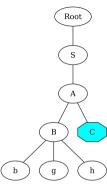








PARSE TREE



Operation: λ consumed from stack





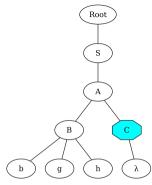


М

\$

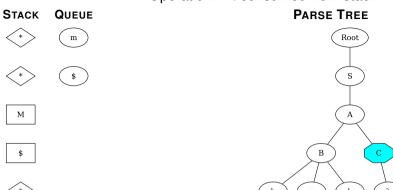


Parse Tree



 $\label{eq:lambda} \text{The top of the stack is } \lambda.$ Pop the stack, place a λ as the right-most child of the Current tree node.

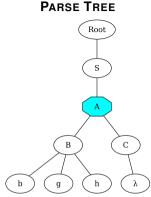
Operation: λ consumed from stack



The top of the stack is the **end of production** marker.

STACK QUEUE

Operation: end of ${\it C}$ production



The top of the stack is the **end of production** marker, pop the stack, and set the **parent** of Current as the new current node. $Current \leftarrow Current.parent$

Operation: end of *C* production

STACK QUEUE

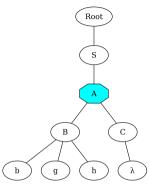




\$



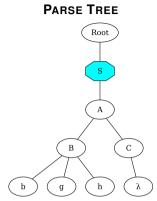
PARSE TREE



The top of the stack is the **end of production** marker.

STACK QUEUE

Operation: end of \boldsymbol{A} production



The top of the stack is the **end of production** marker, pop the stack, and set the **parent** of Current as the new current node. $Current \leftarrow Current.parent$

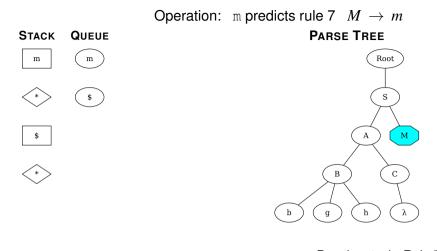
Operation: end of A production PARSE TREE Root

STACK

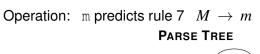
QUEUE

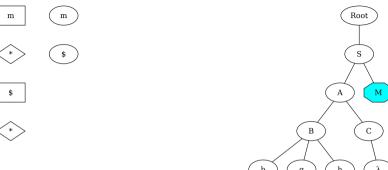
The top of the stack is the non-terminal M, the front of the queue is token m, Using the LLT table we look up **Rule 7**: $M \to m$.

Hook up Ruie 7: $M \rightarrow m$									
b	g	h	m	n	р	S	t	\$	
1			1	1	1	1			
2			3	3	3	3			
			7	8	9				
4									
			6	6	6	5			
	b 1	b g	b g h	b g h m 1 1 3 7 4 - Francisco 1 1	b g h m n 1 1 1 1 2 3 3 7 8	b g h m n p 1 1 1 1 1 2 3 3 3 3 7 8 9	b g h m n p s 1 1 1 1 1 1 1 2 3 3 3 3 3 7 8 9	b g h m n p s t 1 1 1 1 1 1 2 3 3 3 3 3 7 8 9	



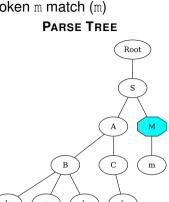
Pop the stack. Rule 7: $M \to m$ is predicted. Push an **end of production** marker (*) onto the stack, push the RHS of the rule onto the stack **from right to left**, add the LHS as the right-most child of the *Current* tree node, make this new child the *Current* node of the tree.





QUEUE

The stack top is a terminal that matches the front of the queue (m).



The stack top is a terminal that **matches** the front of the queue (m). Pop the stack, pull the **token** from the front of the queue:

add the **token** as the right-most child of the *Current* tree node.

Operation: token m match (m)

QUEUE

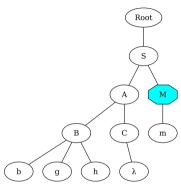






STACK

PARSE TREE

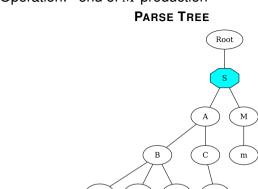


The top of the stack is the **end of production** marker.

Operation: end of M production

STACK

QUEUE



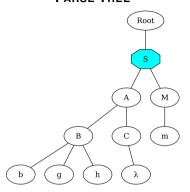
The top of the stack is the **end of production** marker, pop the stack, and set the **parent** of Current as the new current node. $Current \leftarrow Current.parent$

Operation: end of M production

UE PARSE TREE

STACK QUEUE

\$
\$



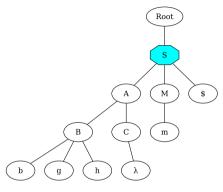
The top of the stack is the **end of input** grammar symbol (\$).

Operation: token \$ match (\$)

QUEUE



PARSE TREE



The stack top \$ matches the front of the queue. Pop the stack, pull the \$ from the front of the queue;

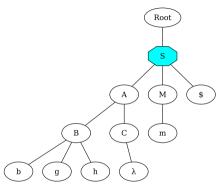
place a \$ as the right-most child of the *Current* tree node.

Operation: token \$ match (\$)

STACK QUEUE



Parse Tree

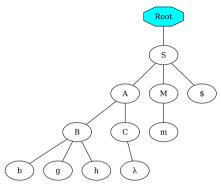


The top of the stack is the **end of production** marker.

STACK QUEUE

Operation: end of *S* production

PARSE TREE



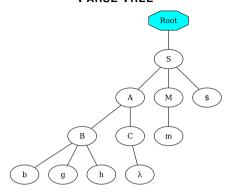
The top of the stack is the **end of production** marker, pop the stack, and set the **parent** of Current as the new current node. $Current \leftarrow Current.parent$

Operation: end of S production

STACK

QUEUE

PARSE TREE



The **stack** is **empty**, *Current* is the root of the parse tree. Below Current is the "raw" parse tree for the input b g h m. b g h m is a valid sentence of the grammar.

LL Parsing Verifies Input with Leftmost Derivations

Watch the same input being parsed, but his time we will keep track of the derivational steps being performed.

Operation: begin

STACK QUEUE

b

PARSE TREE

Root

g

h

 $\binom{m}{}$

(\$

Operation: b predicts rule 1 $S \rightarrow AM$ \$

STACK QUEUE

A b

M g

\$ h

* m

(\$

PARSE TREE



 $S \Rightarrow AM$ \$

Operation: b predicts rule 2 $A \rightarrow BC$

STACK QUEUE

3 b

C g

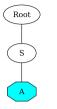
* h

M m

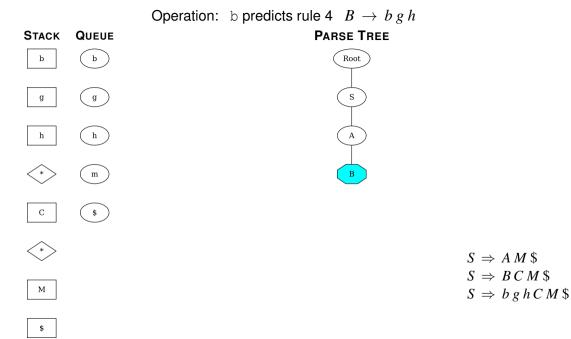
\$ \$

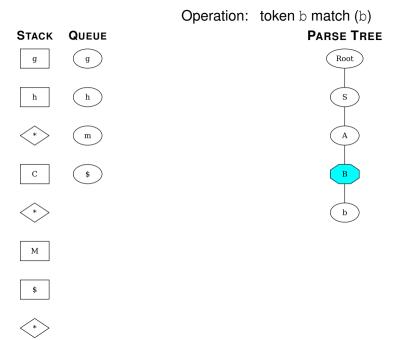


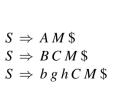
Parse Tree



 $S \Rightarrow AM \$$ $S \Rightarrow BCM \$$







Operation: token g match (g)

QUEUE

h

* m

C \$

*

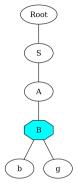
STACK

М

\$



Parse Tree



 $S \Rightarrow AM \$$ $S \Rightarrow BCM \$$ $S \Rightarrow bghCM \$$

Operation: token h match (h)

STACK QUEUE





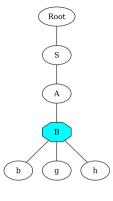


М

\$



PARSE TREE



 $S \Rightarrow AM \$$ $S \Rightarrow BCM \$$ $S \Rightarrow bghCM \$$

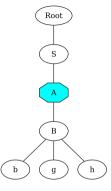
Operation: end of *B* production

QUEUE

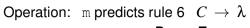
STACK



PARSE TREE



 $S \Rightarrow AM$ \$ $S \Rightarrow BCM$ \$ $S \Rightarrow b g h C M$ \$



QUEUE

λ m

* (\$

*

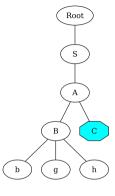
STACK

М

\$







 $S \Rightarrow A M \$$ $S \Rightarrow B C M \$$ $S \Rightarrow b g h C M \$$ $S \Rightarrow b g h \lambda M \$$

Operation: λ consumed from stack







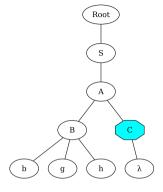
М

STACK

\$



Parse Tree



 $S \Rightarrow AM \$$ $S \Rightarrow BCM \$$ $S \Rightarrow bghCM \$$ $S \Rightarrow bghM \$$

Operation: end of C production

١



QUEUE

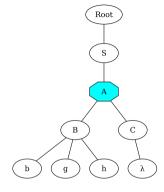




STACK



Parse Tree



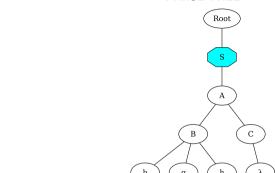
 $S \Rightarrow AM \$$ $S \Rightarrow BCM \$$ $S \Rightarrow bghCM \$$ $S \Rightarrow bghM \$$

Operation: end of A production

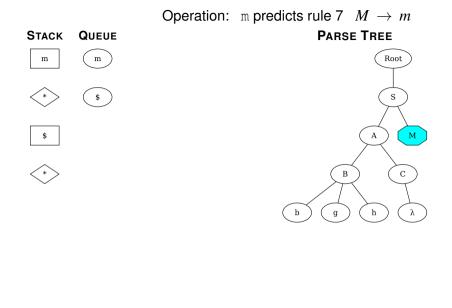
QUEUE

STACK

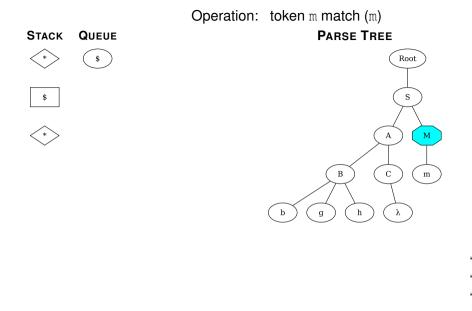




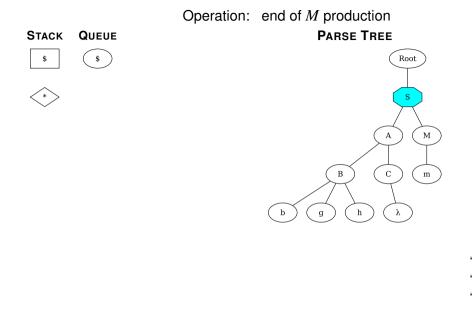
 $S \Rightarrow AM \$$ $S \Rightarrow BCM \$$ $S \Rightarrow bghCM \$$ $S \Rightarrow bghM \$$



 $S \Rightarrow AM \$$ $S \Rightarrow BCM \$$ $S \Rightarrow bghCM \$$ $S \Rightarrow bghM \$$ $S \Rightarrow bghM \$$



 $S \Rightarrow A M \$$ $S \Rightarrow B C M \$$ $S \Rightarrow b g h C M \$$ $S \Rightarrow b g h M \$$ $S \Rightarrow b g h m \$$

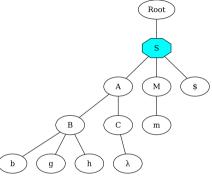


 $S \Rightarrow A M \$$ $S \Rightarrow B C M \$$ $S \Rightarrow b g h C M \$$ $S \Rightarrow b g h M \$$ $S \Rightarrow b g h m \$$

Operation: token \$ match (\$)

QUEUE PARSE TREE

STACK

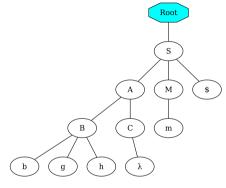


 $S \Rightarrow AM\$$ $S \Rightarrow BCM\$$ $S \Rightarrow bghCM\$$ $S \Rightarrow bghM\$$ $S \Rightarrow bghM\$$ $S \Rightarrow bghM\$$

Operation: end of S production

QUEUE PARSE TREE

STACK



As promised, this is a $S \Rightarrow BCM\$$ **leftmost** parse of the input $S \Rightarrow bghCM\$$

sentence. $S \Rightarrow b g h M \$$ $S \Rightarrow b g h m \$$

 $S \Rightarrow AM$ \$

Pseudo Code for LLT Parsing

lltableparse.pdf

Example Fischer 5-2

Predict Sets

#	$p \in P$	Computed By	Predict Set
1	$S \rightarrow AC$ \$	FirstSet(RHS)	a, b, c, q, \$
2	$C \rightarrow c$	FirstSet(RHS)	С
3	$C o \lambda$	FollowSet(LHS)	d, \$
4	$A \rightarrow aBCd$	FirstSet(RHS)	а
5	$A \rightarrow BQ$	$FirstSet(RHS) \bigcup FollowSet(LHS)$	b, c, q, \$
6	B o b B	FirstSet(RHS)	b
7	$B o\lambda$	FollowSet(LHS)	c, d, q, \$
8	Q o q	FirstSet(RHS)	q
9	$Q o \lambda$	FollowSet(LHS)	с, \$

Example Fischer 5-2

LL(1) Parsing Table

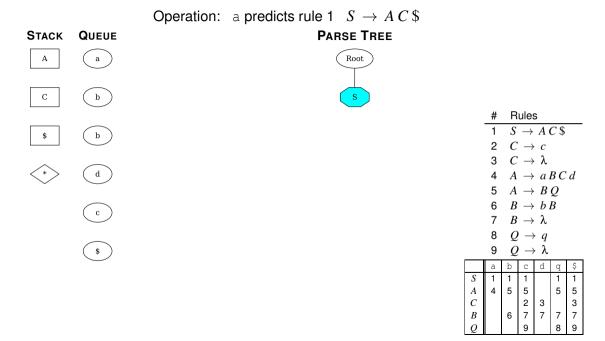
Predict Sets

	а	b	С	d	q	\$
S	1	1	1		1	1
\boldsymbol{A}	4	5	5		5	5
C			2	3		3
\boldsymbol{B}		6	7	7	7	7
Q			9		8	9

_	#	$p \in P$	Computed By	Predict Set
	1	$S \rightarrow AC$ \$	FirstSet(RHS)	a, b, c, q, \$
	2	$C \rightarrow c$	FirstSet(RHS)	С
	3	$C o\lambda$	FollowSet(LHS)	d, \$
	4	$A \rightarrow aBCd$	FirstSet(RHS)	a
	5	$A \rightarrow BQ$	$FirstSet(RHS) \cup FollowSet(LHS)$	b, c, q, \$
	6	B obB	FirstSet(RHS)	b
	7	$B o\lambda$	FollowSet(LHS)	c, d, q, \$
	8	Q o q	FirstSet(RHS)	q
_	9	$Q o\lambda$	FollowSet(LHS)	с,\$

Operation: begin QUEUE PARSE TREE STACK Root Rules 1 $S \rightarrow AC$ \$ 2 $C \rightarrow c$ 3 $C \rightarrow \lambda$ 4 $A \rightarrow aBCd$ 5 $A \rightarrow BQ$ 6 $B \rightarrow b B$

7	B	\rightarrow	λ		
8	Q	\rightarrow	q		
9	Q	\rightarrow	λ		
а	b	С	d	q	\$
1	1	1		1	1
4	5	5 2 7		5	5
		2	3 7		5 3 7
	6	7	7	7	7
		9		8	9

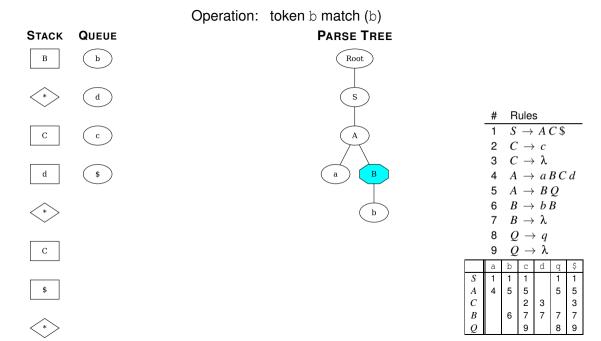


Operation: a predicts rule 4 $A \rightarrow a B C d$ STACK QUEUE PARSE TREE Root Rules $S \rightarrow AC$ \$ $C \rightarrow C$ 3 $C \rightarrow \lambda$ 4 $A \rightarrow aBCd$ 5 $A \rightarrow BQ$ 6 $B \rightarrow b B$ 7 $B \rightarrow \lambda$

Operation: token a match (a) STACK QUEUE PARSE TREE Root Rules $S \rightarrow AC$ \$ $C \rightarrow C$ 3 $C \rightarrow \lambda$ $A \rightarrow aBCd$ 5 $A \rightarrow BQ$ $B \rightarrow b B$ 7 $B \rightarrow \lambda$

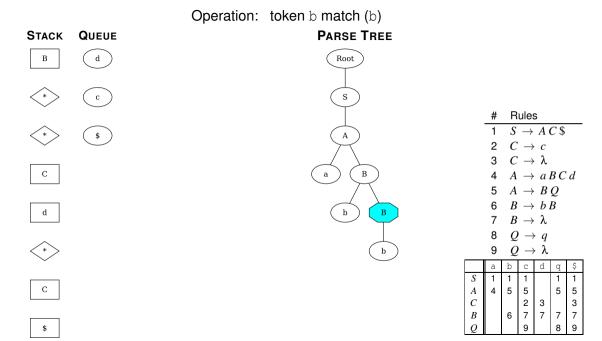
Operation: b predicts rule 6 $B \rightarrow b B$ PARSE TREE STACK QUEUE Root Rules $S \rightarrow AC$ \$ 2 $C \rightarrow c$ 3 $C \rightarrow \lambda$ 4 $A \rightarrow aBCd$ 5 $A \rightarrow BQ$

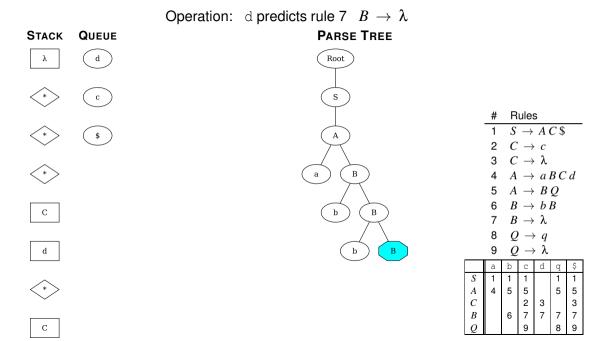
d \$					$\frac{b}{\lambda}$	3	
*			Q Q		q		
v		а	b	С	d	q	-
	S	1	1	1		1	
C	A	4	5	5		5	ţ
	C	1 !		2	3		

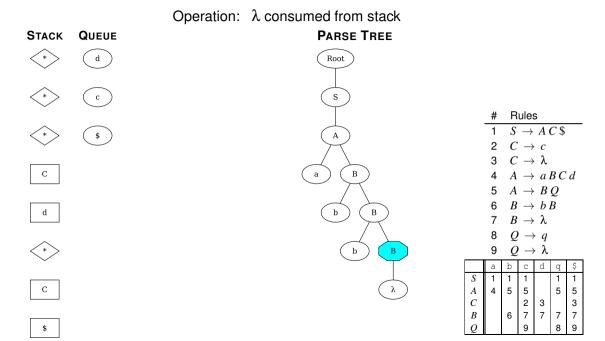


Operation: b predicts rule 6 $B \rightarrow b B$ PARSE TREE STACK QUEUE Root Rules 1 $S \rightarrow AC$ \$ $C \rightarrow C$ 3 $C \rightarrow \lambda$ 4 $A \rightarrow aBCd$ a 5 $A \rightarrow BQ$

С	b		6 7 8		λ		
d	_			\tilde{Q} –			
			a k	С	d	q	\$
^		S	1 1	1		1	1
<*>		A	4 5	5		5	5
~		C		2	3		3
		B	6	7	7	7	7
C		Q		9		8	9

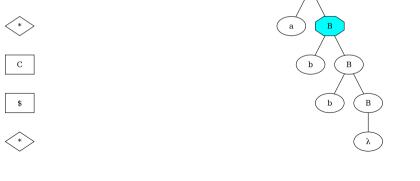






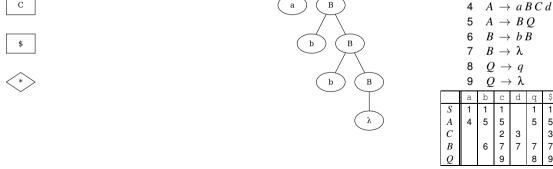
Operation: end of *B* production STACK QUEUE PARSE TREE Root Rules $S \rightarrow AC$ \$ $C \rightarrow C$ 3 $C \rightarrow \lambda$ $A \rightarrow aBCd$ 5 $A \rightarrow BQ$ $B \rightarrow b B$ $B \rightarrow \lambda$

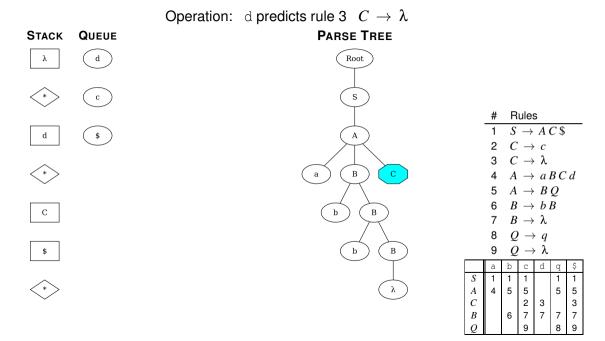
Operation: end of *B* production STACK PARSE TREE QUEUE Root Rules $S \rightarrow AC$ \$ 2 $C \rightarrow c$ 3 $C \rightarrow \lambda$ 4 $A \rightarrow aBCd$



	-	71		u.	$D \subset$	α
	5	\boldsymbol{A}	\rightarrow	B	Q	
	6	B	\rightarrow	b.	B	
	7	B	\rightarrow	λ		
	8	Q	! →	q		
	9	Q	? —	λ		
	а	b	С	d	q	\$
S	1	1	1		1	1
Α	4	5	5		5	5
A C B			5 2 7	3		5 3 7
В		6	7	7	7	7

Operation: end of *B* production PARSE TREE STACK QUEUE Root Rules $S \rightarrow AC$ \$ $C \rightarrow C$ 3 $C \rightarrow \lambda$





Operation: λ consumed from stack PARSE TREE Root Rules $S \rightarrow AC$ \$ $C \rightarrow C$ 3 $C \rightarrow \lambda$ 4 $A \rightarrow aBCd$ 5 $A \rightarrow BQ$ 6 $B \rightarrow b B$ 7 $B \rightarrow \lambda$ 8 $Q \rightarrow a$

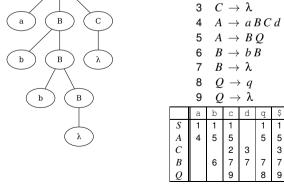
С	a B C
\$	b B λ
*	b
	λ

QUEUE

STACK

	O	\mathcal{L}	/	9		
	9	Q	<i>?</i> →	λ		
	а	b	O	d	q	\$
S	1	1	1		1	1
4	4	5	5		5	5
C			2	3 7		5 3
В		6	7	7	7	7
2			9		8	9

Operation: end of *C* production PARSE TREE STACK QUEUE Root $S \rightarrow AC$ \$ $C \rightarrow C$ В С



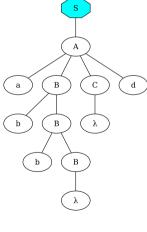
Rules

Operation: token d match (d) STACK QUEUE PARSE TREE Root Rules $S \rightarrow AC$ \$ $C \rightarrow C$ 3 $C \rightarrow \lambda$ С $A \rightarrow aBCd$ a $A \rightarrow BQ$ $B \rightarrow b B$ λ $B \rightarrow \lambda$ 6

Operation: end of A production PARSE TREE Root $C \rightarrow C$ 3 $C \rightarrow \lambda$

STACK

QUEUE



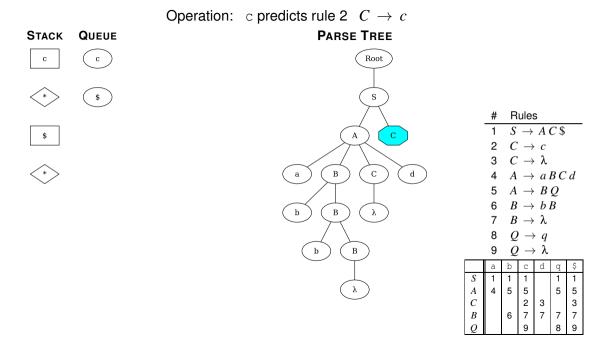
Rules $S \rightarrow AC$ \$

 $A \rightarrow aBCd$

 $A \rightarrow BQ$ $B \rightarrow b B$

 $B \rightarrow \lambda$

6



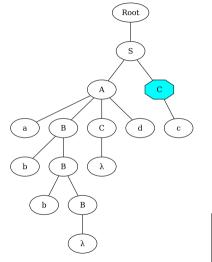
Operation: token c match (c) PARSE TREE











Rules $S \rightarrow AC$ \$ $C \rightarrow C$ 3 $C \rightarrow \lambda$

7 $B \rightarrow \lambda$

6

 $A \rightarrow aBCd$ $A \rightarrow BQ$ $B \rightarrow b B$

Operation: end of *C* production PARSE TREE

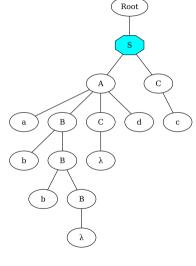
QUEUE





STACK





Rι	ıles	;	
S	\rightarrow	\overline{A}	<i>C</i> \$
C		_	

 $\begin{array}{ccc}
2 & C \to c \\
3 & C \to \lambda
\end{array}$

 $A \rightarrow a B C d$ $A \rightarrow B Q$

 $\begin{array}{ccc} 6 & B \rightarrow b B \end{array}$

 $\begin{array}{ccc} 7 & B \rightarrow \lambda \\ \end{array}$

 $egin{array}{ccc} {\sf 8} & Q
ightarrow q \ {\sf 9} & Q
ightarrow \lambda \end{array}$

	а	b	O	d	q	05
• 1	1	1	1		1	1
١	4	5	5		5	Ę
7			2	3		6)
3		6	7	7	7	7
9			9		8	ç

Operation: token \$ match (\$)

QUEUE

PARSE TREE

Rules $S \rightarrow AC$ \$

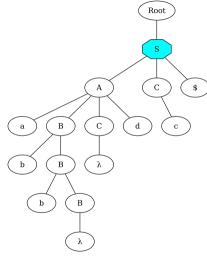
 $C \rightarrow C$ 3 $C \rightarrow \lambda$

 $B \rightarrow \lambda$

6

 $A \rightarrow aBCd$ $A \rightarrow BQ$ $B \rightarrow b B$





Operation: end of S production

QUEUE

STACK

Parse Tree

Rules $S \rightarrow AC$ \$

2 $C \rightarrow c$ 3 $C \rightarrow \lambda$

 $A \rightarrow a B C d$ $A \rightarrow B Q$ $B \rightarrow b B$

 $B \rightarrow \lambda$

6

