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 $A \rightarrow \alpha\beta$ is $First(\alpha\beta) \underbrace{\bigcup Follow(A)}_{\text{if } \alpha\beta \Rightarrow^* \lambda}$

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

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 \$.

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!

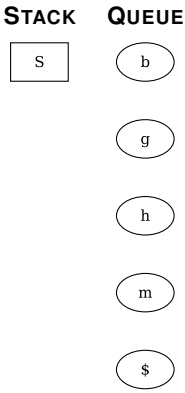
Predict Sets

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

LL(1) Parsing Table

	b	g	h	m	n	p	s	t	\$
S	1			1	1	1	1		
A	2			3	3	3	3		
M				7	8	9			
B	4								
C				6	6	6	5		

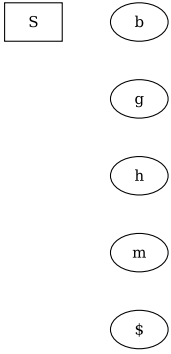
Operation: begin



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.

Operation: begin

STACK **QUEUE**

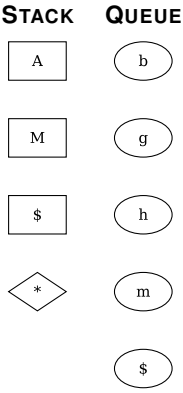


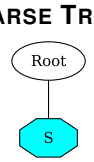
PARSE TREE

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.
The stack top is the non-terminal S,
the front of the queue is token b,
Using the LLT table we look up **Rule 1**: $S \rightarrow A\ M\ \$$.

	b	g	h	m	n	p	s	t	$
S	1			1	1	1	1		
A	2			3	3	3	3		
M				7	8	9			
B	4								
C				6	6	6	5		

Operation: b predicts rule 1 $S \rightarrow A\ M\ \$$

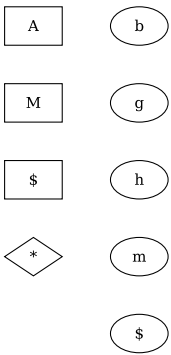




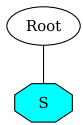
Pop the stack. Rule 1: $S \rightarrow A\ M\ \$$ is predicted.
Push an **end of production** marker ($*$) onto the stack,
push the RHS of the rule onto the stack **from right to left**,
add the LHS as the right-most child of the *Current* tree node,
make this new child the *Current* node of the tree.

Operation: b predicts rule 1 $S \rightarrow A\,M\,\$$

STACK **QUEUE**



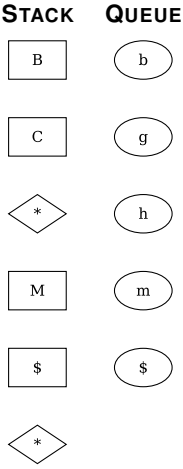
PARSE TREE

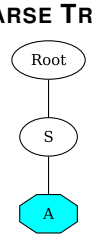


The top of the stack is the non-terminal A,
the front of the queue is token b,
Using the LLT table we look up **Rule 2**: $A \rightarrow B\,C$.

	b	g	h	m	n	p	s	t	$
S	1			1	1	1	1		
A	2			3	3	3	3		
M				7	8	9			
B	4								
C				6	6	6	5		

Operation: b predicts rule 2 $A \rightarrow B\,C$

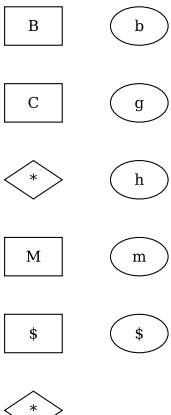




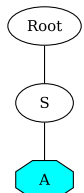
Pop the stack. Rule 2: $A \rightarrow B\,C$ is predicted.
Push an **end of production** marker ($*$) onto the stack,
push the RHS of the rule onto the stack **from right to left**,
add the LHS as the right-most child of the *Current* tree node,
make this new child the *Current* node of the tree.

Operation: b predicts rule 2 $A \rightarrow B\,C$

STACK **QUEUE**



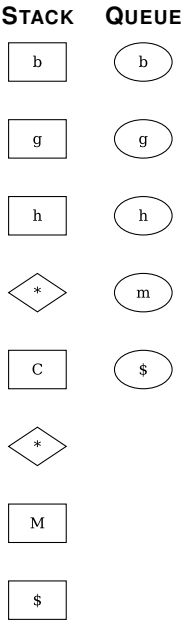
PARSE TREE

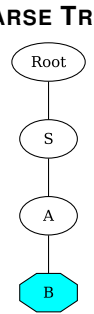


The top of the stack is the non-terminal B,
the front of the queue is token b,
Using the LLT table we look up **Rule 4**: $B \rightarrow b\,g\,h$.

	b	g	h	m	n	p	s	t	$
S	1			1	1	1	1		
A	2			3	3	3	3		
M				7	8	9			
B	4								
C				6	6	6	5		

Operation: b predicts rule 4 $B \rightarrow b\ g\ h$

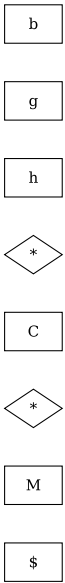




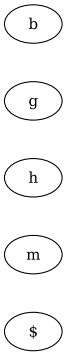
Pop the stack. Rule 2: $B \rightarrow b\ g\ h$ is predicted.
Push an **end of production** marker ($*$) onto the stack,
push the RHS of the rule onto the stack **from right to left**,
add the LHS as the right-most child of the *Current* tree node,
make this new child the *Current* node of the tree.

Operation: b predicts rule 4 $B \rightarrow b\ g\ h$

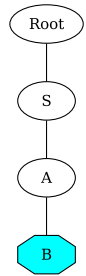
STACK



QUEUE

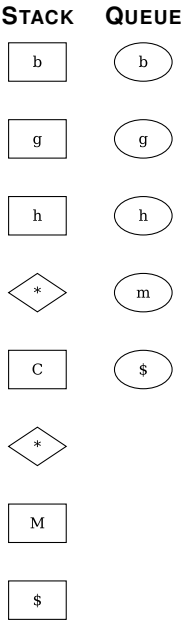


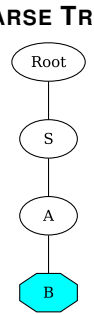
PARSE TREE



Uh oh — the top of the stack is a terminal! What to do?

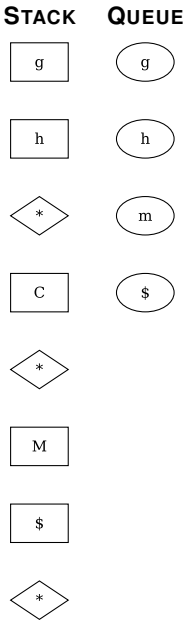
Operation: b predicts rule 4 $B \rightarrow b\ g\ h$

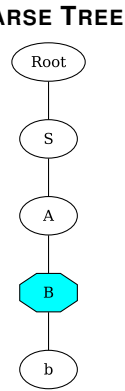




The top of the stack is the terminal b,
the front of the queue is token b, **they match!**

Operation: token b match (b)



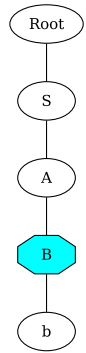


The stack top is a terminal that **matches** the front of the queue (b).
Pop the stack, pull the **token** from the front of the queue;
add the **token** as the right-most child of the *Current* tree node.

Operation: token b match (b)

STACK **QUEUE**

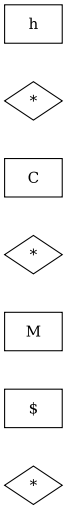
PARSE TREE



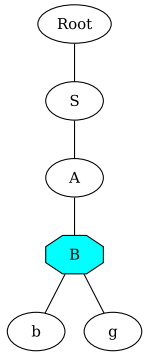
The stack top is a terminal that **matches** the front of the queue (g).

Operation: token g match (g)

STACK **QUEUE**



PARSE TREE



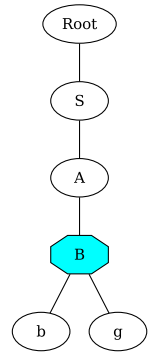
The stack top is a terminal that **matches** the front of the queue (g).
Pop the stack, pull the **token** from the front of the queue;
add the **token** as the right-most child of the *Current* tree node.

Operation: token g match (g)

STACK

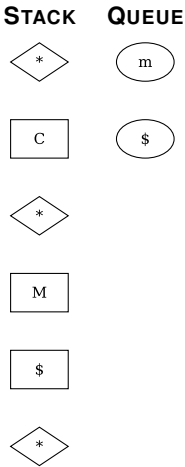
QUEUE

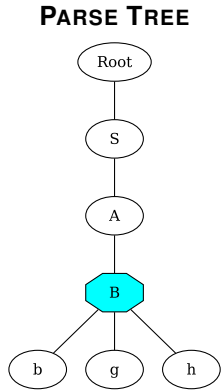
PARSE TREE



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

Operation: token h match (h)

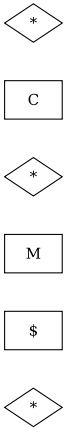




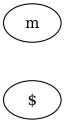
The stack top is a terminal that **matches** the front of the queue (h).
Pop the stack, pull the **token** from the front of the Queue;
add the **token** as the right-most child of the *Current* tree node.

Operation: token `h` match (`h`)

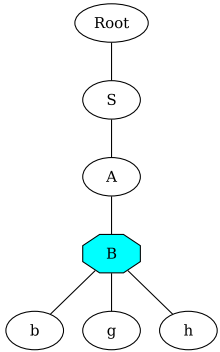
STACK



QUEUE



PARSE TREE

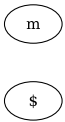


The top of the stack is the **end of production** marker.

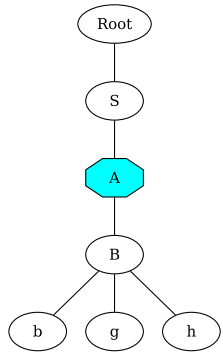
Operation: end of B production

STACK

QUEUE



PARSE TREE

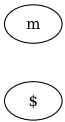


The top of the stack is the **end of production** marker,
pop the stack, and set the **parent** of $Current$ as the new current node.

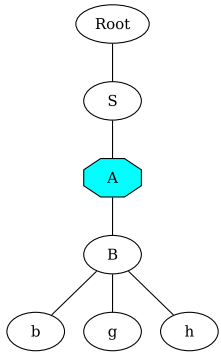
$$Current \leftarrow Current.parent$$

Operation: end of B production

STACK **QUEUE**



PARSE TREE

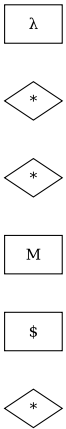


The top of the stack is the non-terminal C, the front of the queue is token m, Using the LLT table we look up **Rule 6**: $C \rightarrow \lambda$.

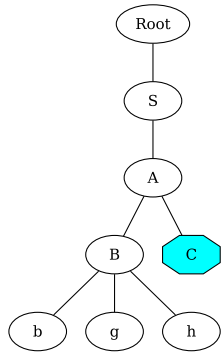
	b	g	h	m	n	p	s	t	$
S	1			1	1	1	1		
A	2			3	3	3	3		
M				7	8	9			
B	4								
C				6	6	6	5		

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

STACK **QUEUE**



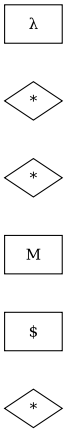
PARSE TREE



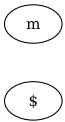
Pop the stack. Rule 6: $C \rightarrow \lambda$ is predicted.
Push an **end of production** marker ($*$) onto the stack,
push the RHS of the rule onto the stack **from right to left**,
add the LHS as the right-most child of the *Current* tree node,
make this new child the *Current* node of the tree.

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

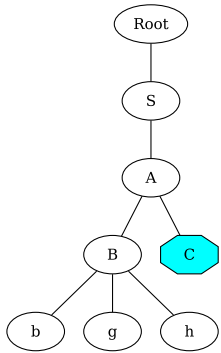
STACK



QUEUE



PARSE TREE

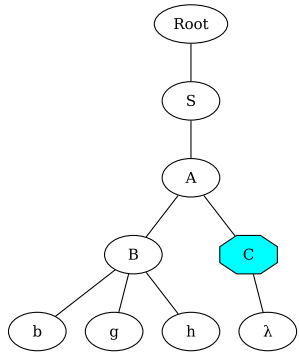


The top of the stack is λ! What to do?

Operation: λ consumed from stack

STACK **QUEUE**

PARSE TREE

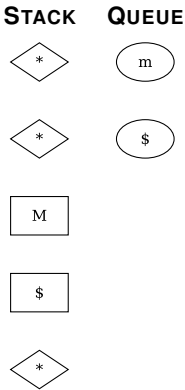


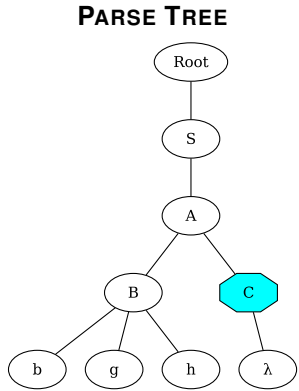
The top of the stack is λ.

Pop the stack,

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

Operation: λ consumed from stack



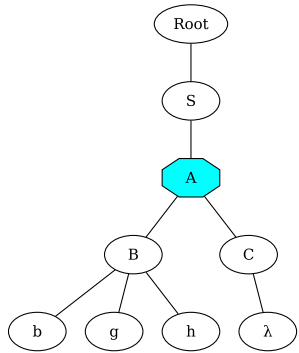


The top of the stack is the **end of production** marker.

Operation: end of C production

STACK **QUEUE**

PARSE TREE

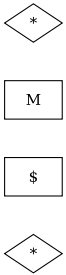


The top of the stack is the **end of production** marker,
pop the stack, and set the **parent** of $Current$ as the new current node.

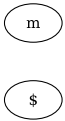
$$Current \leftarrow Current.parent$$

Operation: end of C production

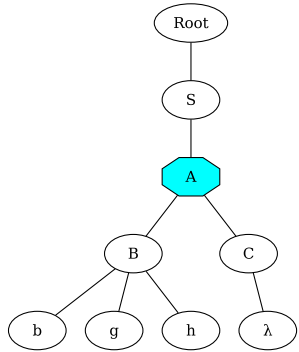
STACK



QUEUE



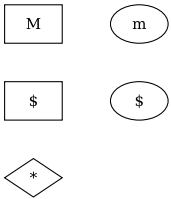
PARSE TREE



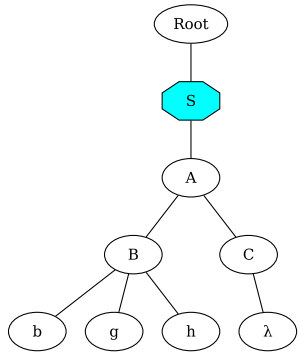
The top of the stack is the **end of production** marker.

Operation: end of A production

STACK **QUEUE**



PARSE TREE

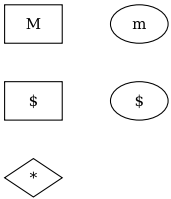


The top of the stack is the **end of production** marker,
pop the stack, and set the **parent** of $Current$ as the new current node.

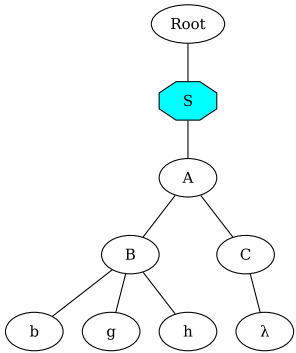
$$Current \leftarrow Current.parent$$

Operation: end of A production

STACK **QUEUE**



PARSE TREE

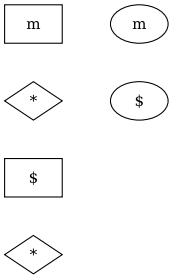


The top of the stack is the non-terminal M, the front of the queue is token m, Using the LLT table we look up **Rule 7**: $M \rightarrow m$.

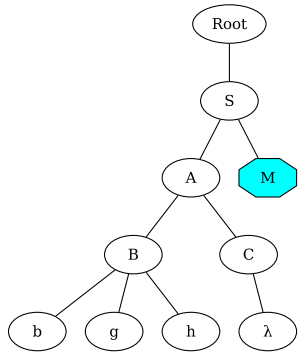
	b	g	h	m	n	p	s	t	$
S	1			1	1	1	1		
A	2			3	3	3	3		
M				7	8	9			
B	4								
C				6	6	6	5		

Operation: $\mathtt{m}$ predicts rule 7 $M \rightarrow m$

STACK **QUEUE**



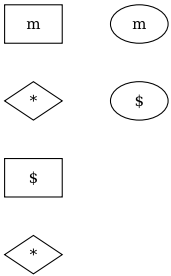
PARSE TREE



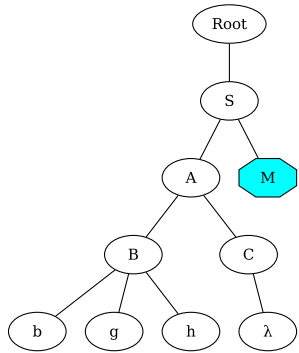
Pop the stack. Rule 7: $M \rightarrow m$ is predicted.
Push an **end of production** marker ($*$) onto the stack,
push the RHS of the rule onto the stack **from right to left**,
add the LHS as the right-most child of the $Current$ tree node,
make this new child the $Current$ node of the tree.

Operation: m predicts rule 7 $M \rightarrow m$

STACK **QUEUE**



PARSE TREE

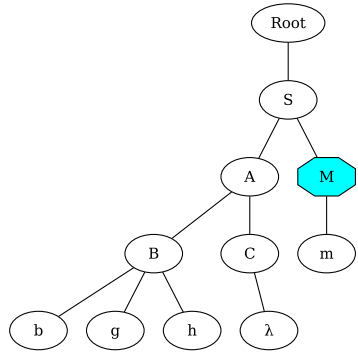


The stack top is a terminal that **matches** the front of the queue (m).

Operation: token `m` match (`m`)

STACK **QUEUE**

PARSE TREE

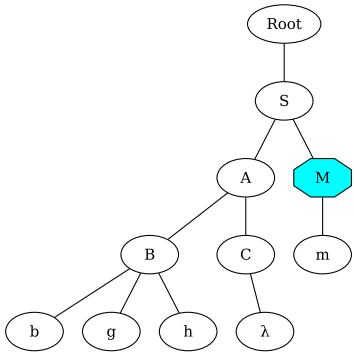


The stack top is a terminal that **matches** the front of the queue (`m`).
Pop the stack, pull the **token** from the front of the queue;
add the **token** as the right-most child of the *Current* tree node.

Operation: token `m` match (`m`)

STACK **QUEUE**

PARSE TREE

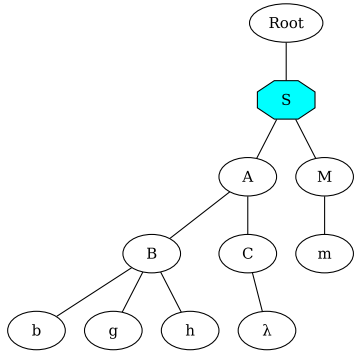


The top of the stack is the **end of production** marker.

Operation: end of M production

STACK **QUEUE**

PARSE TREE



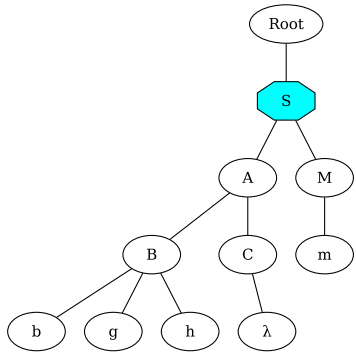
The top of the stack is the **end of production** marker, pop the stack, and set the **parent** of $Current$ as the new current node.

$$Current \leftarrow Current.parent$$

Operation: end of M production

STACK **QUEUE**

PARSE TREE

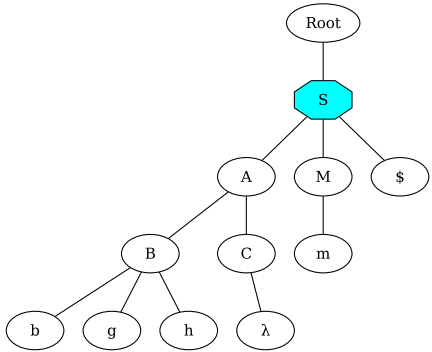


The top of the stack is the **end of input** grammar symbol ($).

Operation: token $ match ($)

STACK **QUEUE**

PARSE TREE

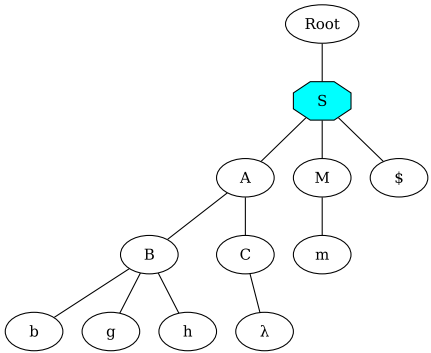


The stack top $ matches the front of the queue.
Pop the stack, pull the $ from the front of the queue;
place a $ as the right-most child of the *Current* tree node.

Operation: token $ match ($)

STACK **QUEUE**

PARSE TREE

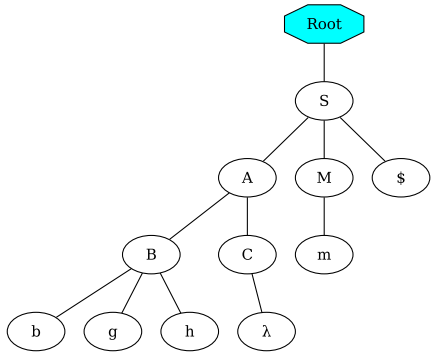


The top of the stack is the **end of production** marker.

Operation: end of S production

STACK **QUEUE**

PARSE TREE



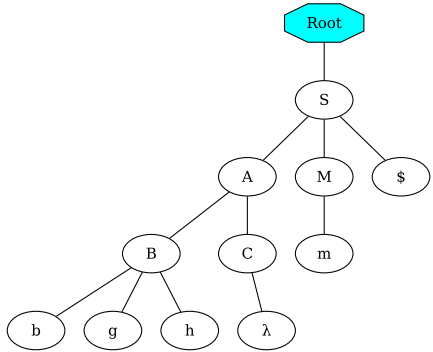
The top of the stack is the **end of production** marker,
pop the stack, and set the **parent** of $Current$ as the new current node.

$$Current \leftarrow Current.parent$$

Operation: end of S production

STACK **QUEUE**

PARSE TREE

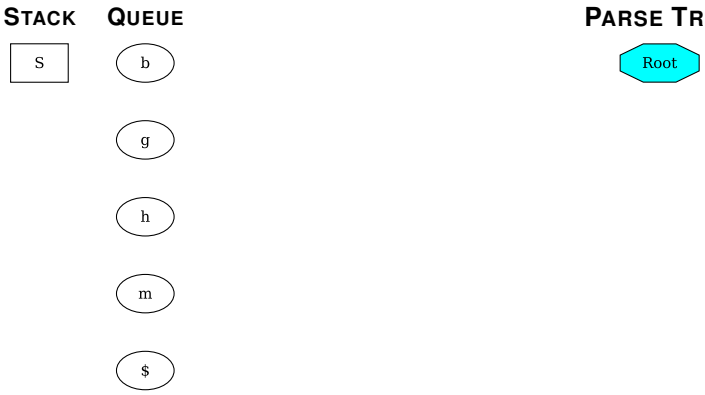


The **stack is empty**, *Current* is the root of the parse tree.
Below *Current* is the **"raw" parse tree** for the input `b g h m`.
`b g h m` is a valid sentence of the grammar.

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

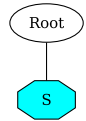


Operation: b predicts rule 1 $S \rightarrow A\ M\ \$$

STACK

QUEUE

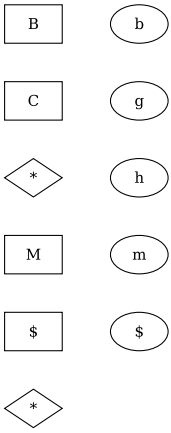
PARSE TREE



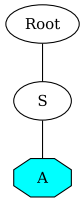
$S \Rightarrow A\ M\ \$$

Operation: b predicts rule 2 $A \rightarrow B\,C$

STACK **QUEUE**



PARSE TREE

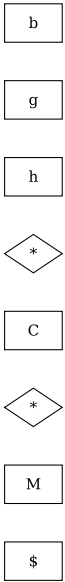


$S \Rightarrow A\,M\,\$$

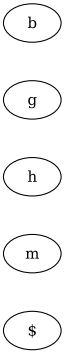
$S \Rightarrow B\,C\,M\,\$$

Operation: b predicts rule 4 $B \rightarrow b\ g\ h$

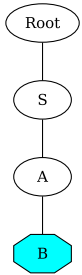
STACK



QUEUE



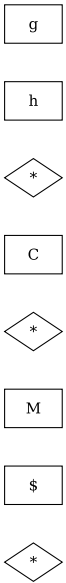
PARSE TREE



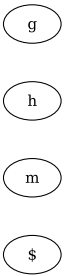
$S \Rightarrow A\ M\ \$$
$S \Rightarrow B\ C\ M\ \$$
$S \Rightarrow b\ g\ h\ C\ M\ \$$

Operation: token b match (b)

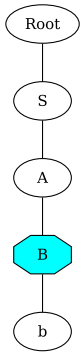
STACK



QUEUE



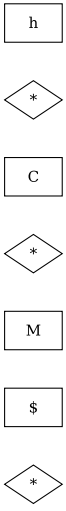
PARSE TREE



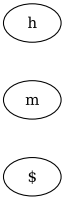
$$S \Rightarrow A\,M\,\$$$
$$S \Rightarrow B\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,C\,M\,\$$$

Operation: token g match (g)

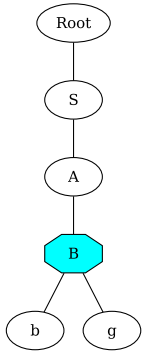
STACK



QUEUE



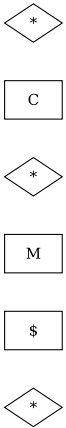
PARSE TREE



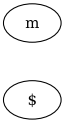
$$S \Rightarrow A\,M\,\$$$
$$S \Rightarrow B\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,C\,M\,\$$$

Operation: token `h` match (`h`)

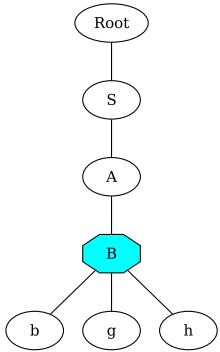
STACK



QUEUE



PARSE TREE

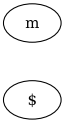


$$S \Rightarrow A\,M\,\$$$
$$S \Rightarrow B\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,C\,M\,\$$$

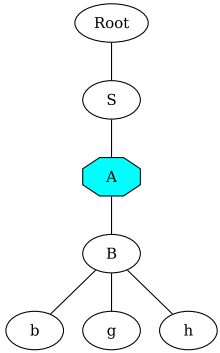
Operation: end of B production

STACK

QUEUE



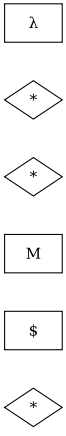
PARSE TREE



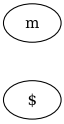
$$S \Rightarrow A\,M\,\$$$
$$S \Rightarrow B\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,C\,M\,\$$$

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

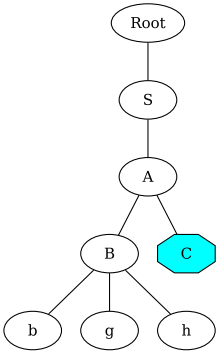
STACK



QUEUE



PARSE TREE

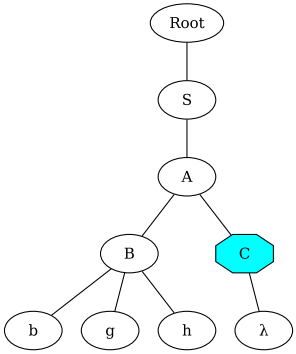


$S \Rightarrow A\,M\,\$$
$S \Rightarrow B\,C\,M\,\$$
$S \Rightarrow b\,g\,h\,C\,M\,\$$
$S \Rightarrow b\,g\,h\,\lambda\,M\,\$$

Operation: λ consumed from stack

STACK **QUEUE**

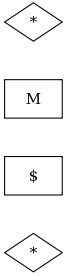
PARSE TREE



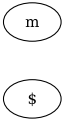
$$S \Rightarrow A\,M\,\$$$
$$S \Rightarrow B\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,M\,\$$$

Operation: end of C production

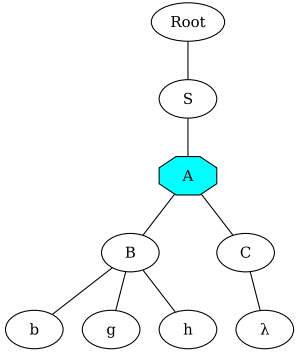
STACK



QUEUE



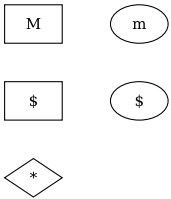
PARSE TREE



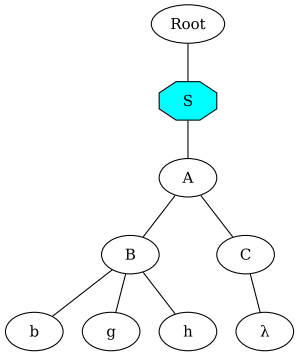
$$S \Rightarrow A\,M\,\$$$
$$S \Rightarrow B\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,M\,\$$$

Operation: end of A production

STACK **QUEUE**



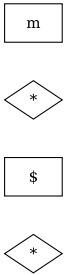
PARSE TREE



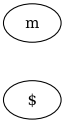
$$S \Rightarrow A\,M\,\$$$
$$S \Rightarrow B\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,M\,\$$$

Operation: m predicts rule 7 $M \rightarrow m$

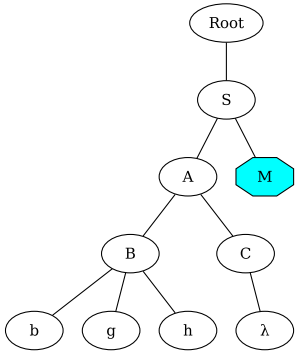
STACK



QUEUE



PARSE TREE



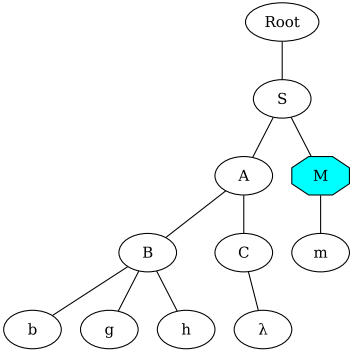
$S \Rightarrow A\,M\,\$$
$S \Rightarrow B\,C\,M\,\$$
$S \Rightarrow b\,g\,h\,C\,M\,\$$
$S \Rightarrow b\,g\,h\,M\,\$$
$S \Rightarrow b\,g\,h\,m\,\$$

Operation: token `m` match (`m`)

STACK

QUEUE

PARSE TREE

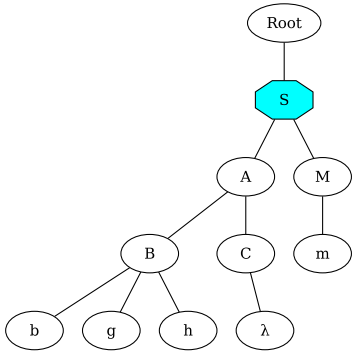


$$S \Rightarrow A\,M\,\$$$
$$S \Rightarrow B\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,M\,\$$$
$$S \Rightarrow b\,g\,h\,m\,\$$$

Operation: end of M production

STACK **QUEUE**

PARSE TREE

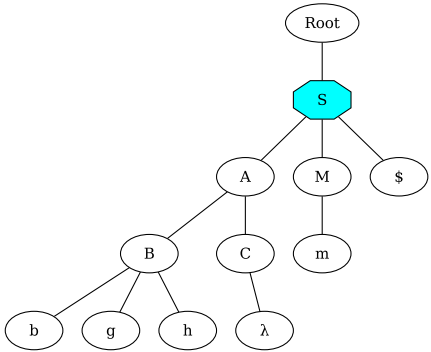


$S \Rightarrow A\,M\,\$$
$S \Rightarrow B\,C\,M\,\$$
$S \Rightarrow b\,g\,h\,C\,M\,\$$
$S \Rightarrow b\,g\,h\,M\,\$$
$S \Rightarrow b\,g\,h\,m\,\$$

Operation: token $ match ($)

STACK **QUEUE**

PARSE TREE

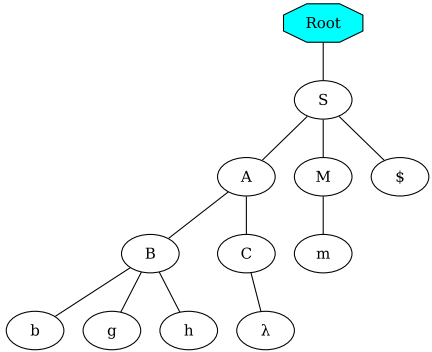


$$S \Rightarrow A\,M\,\$$$
$$S \Rightarrow B\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,C\,M\,\$$$
$$S \Rightarrow b\,g\,h\,M\,\$$$
$$S \Rightarrow b\,g\,h\,m\,\$$$

Operation: end of S production

STACK **QUEUE**

PARSE TREE



As promised, this is a **leftmost** parse of the input sentence.

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

Pseudo Code for LLT Parsing

lltableparse.pdf

Example Fischer 5-2

Predict Sets

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

Example Fischer 5-2

LL(1) Parsing Table

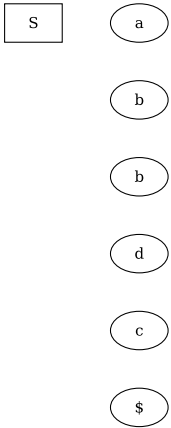
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Predict Sets

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

Operation: begin

STACK **QUEUE**



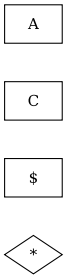
PARSE TREE

#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

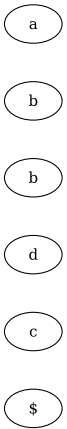
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: a predicts rule 1 $S \rightarrow A\,C\,\$$

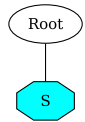
STACK



QUEUE



PARSE TREE

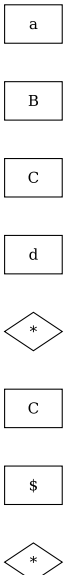


#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

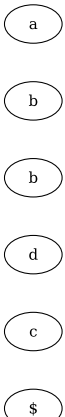
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: a predicts rule 4 $A \rightarrow a\,B\,C\,d$

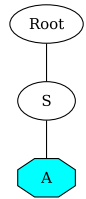
STACK



QUEUE



PARSE TREE

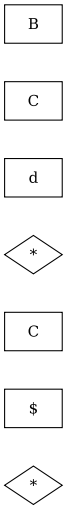


#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

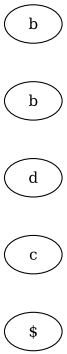
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: token a match (a)

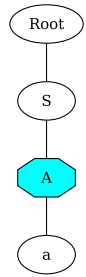
STACK



QUEUE



PARSE TREE

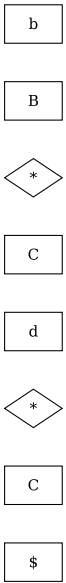


#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

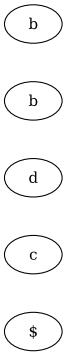
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: b predicts rule 6 $B \rightarrow b\,B$

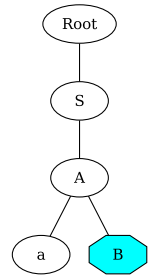
STACK



QUEUE



PARSE TREE

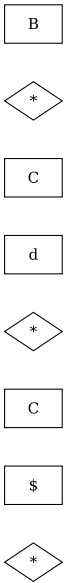


#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

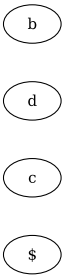
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: token b match (b)

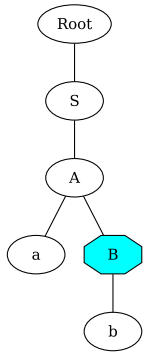
STACK



QUEUE



PARSE TREE

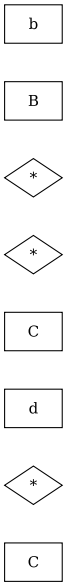


#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

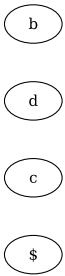
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: b predicts rule 6 $B \rightarrow b\,B$

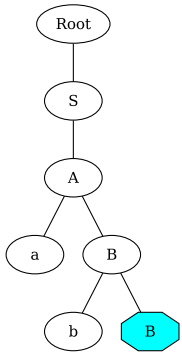
STACK



QUEUE



PARSE TREE

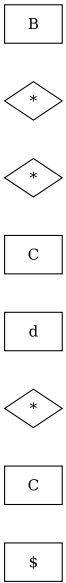


#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

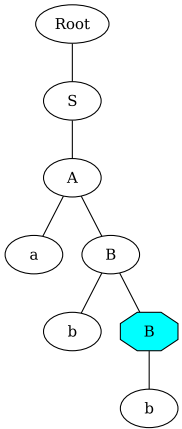
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: token b match (b)

STACK **QUEUE**



PARSE TREE

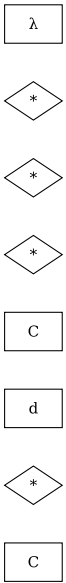


#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

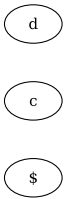
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: d predicts rule 7 $B \rightarrow \lambda$

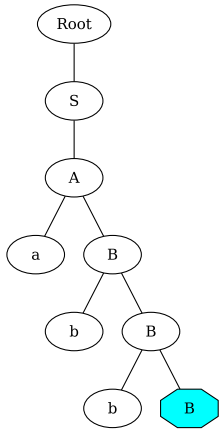
STACK



QUEUE



PARSE TREE



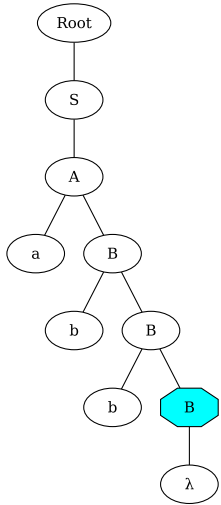
#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: λ consumed from stack

STACK **QUEUE**

PARSE TREE

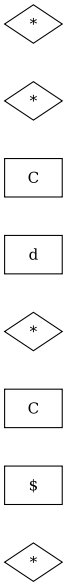


#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

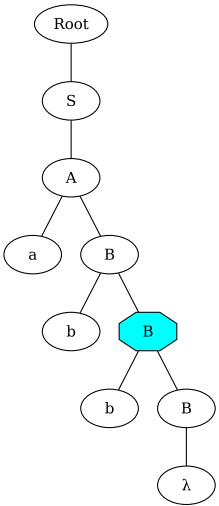
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: end of B production

STACK **QUEUE**



PARSE TREE



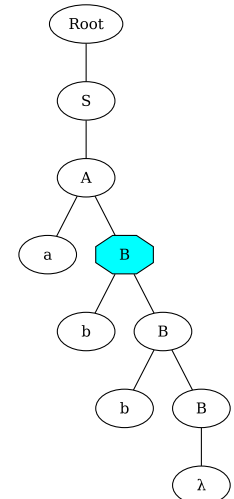
#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: end of B production

STACK **QUEUE**

PARSE TREE

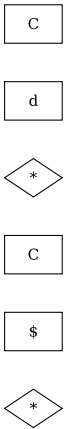


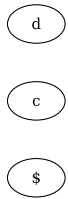
#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

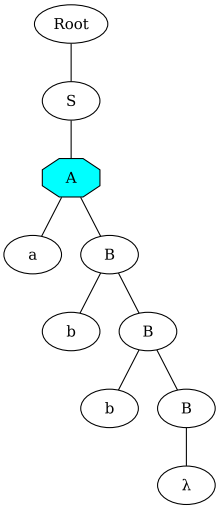
Operation: end of B production

STACK **QUEUE**





PARSE TREE

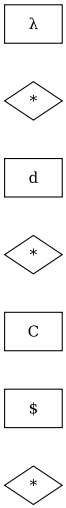


#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

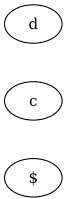
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: d predicts rule 3 $C \rightarrow \lambda$

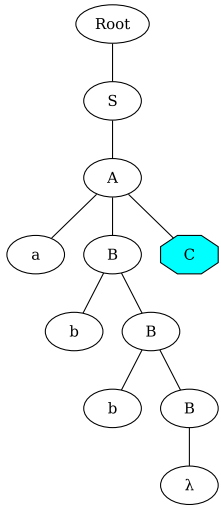
STACK



QUEUE



PARSE TREE

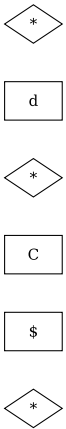


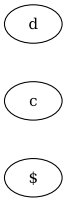
#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

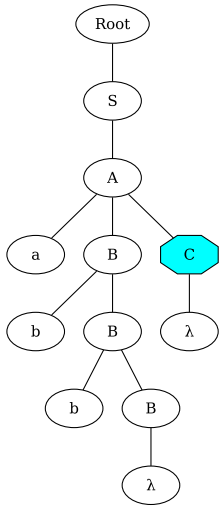
Operation: λ consumed from stack

STACK **QUEUE**





PARSE TREE

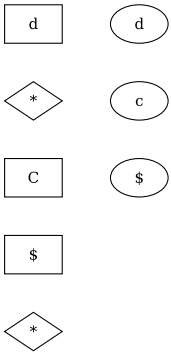


#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

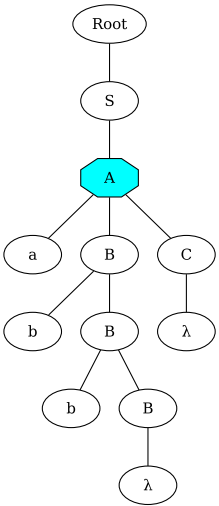
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: end of C production

STACK **QUEUE**



PARSE TREE

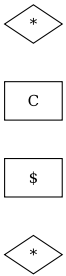


#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

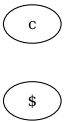
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: token d match (d)

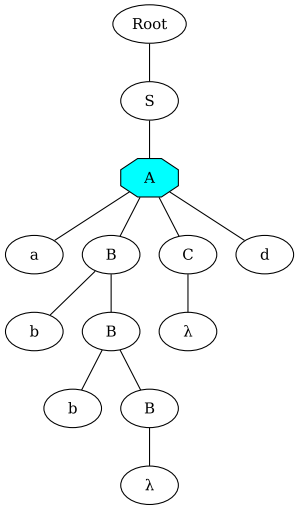
STACK



QUEUE



PARSE TREE

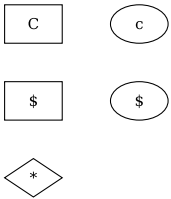


#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

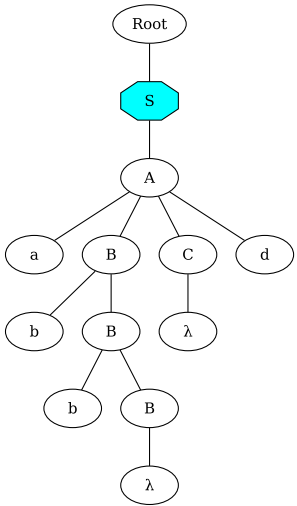
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: end of A production

STACK **QUEUE**



PARSE TREE

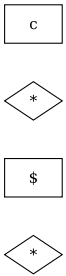


#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

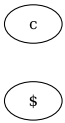
	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: c predicts rule 2 $C \rightarrow c$

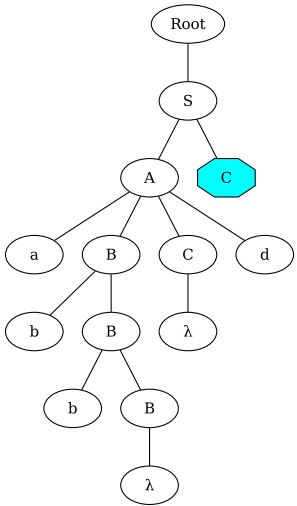
STACK



QUEUE



PARSE TREE



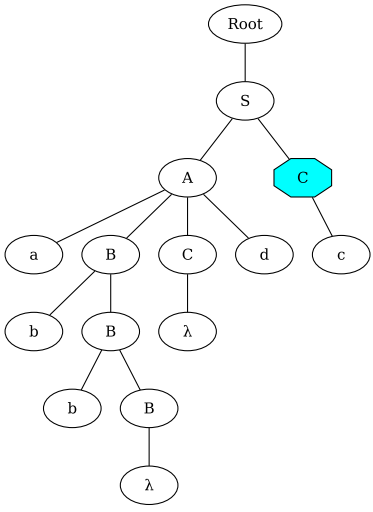
#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: token c match (c)

STACK **QUEUE**

PARSE TREE



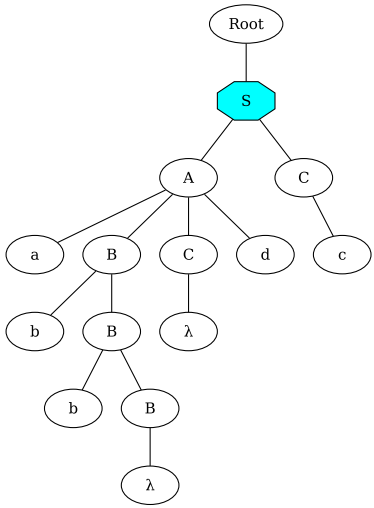
#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: end of C production

STACK **QUEUE**

PARSE TREE



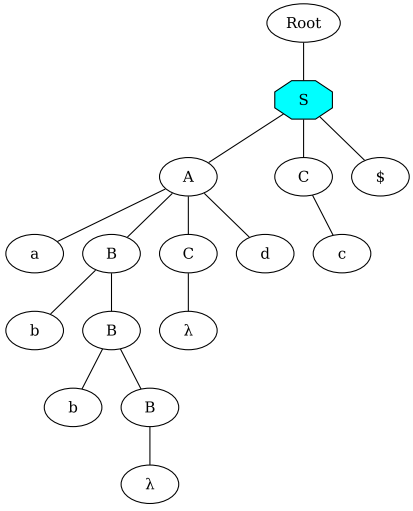
#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: token $ match ($)

STACK **QUEUE**

PARSE TREE



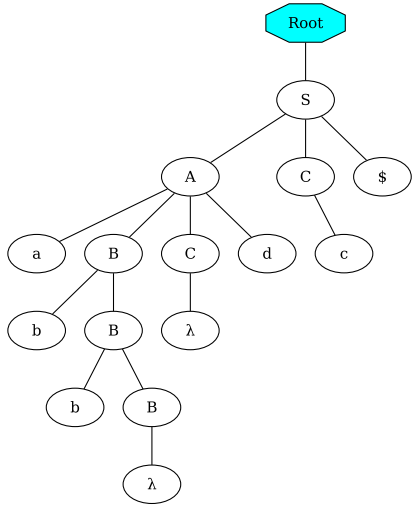
#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9

Operation: end of S production

STACK **QUEUE**

PARSE TREE



#	Rules
1	$S \rightarrow A\,C\,\$$
2	$C \rightarrow c$
3	$C \rightarrow \lambda$
4	$A \rightarrow a\,B\,C\,d$
5	$A \rightarrow B\,Q$
6	$B \rightarrow b\,B$
7	$B \rightarrow \lambda$
8	$Q \rightarrow q$
9	$Q \rightarrow \lambda$

	a	b	c	d	q	$
S	1	1	1		1	1
A	4	5	5		5	5
C			2	3		3
B		6	7	7	7	7
Q			9		8	9