Table Driven Parsing algorithm

Lecture 9

Predictive Parsing

Assume that the grammar is LL(1)

Backtracking will never be needed

Always know which right hand side to choose (with one look-ahead)

- No Left Recursion
- Grammar is Left-Factored.

Step 1: From grammar, construct table.
Step 2: Use table to parse strings.

Predictive Parsing

Assume that the grammar is LL(1)

Backtracking will never be needed

Always know which right hand side to choose (with one look-ahead)

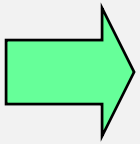
- No Left Recursion
- Grammar is Left-Factored.

$$S \rightarrow aT$$
$$T \rightarrow b \mid Sb$$

Example of FIRST

$S \rightarrow aT$

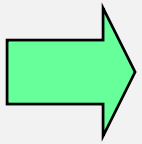
$T \rightarrow b \mid Sb$



$A \rightarrow \alpha$	FIRST(α)
$S \rightarrow aT$	a
$T \rightarrow b \mid Sb$	b, a

Example of FOLLOW

S → aT
T → b | Sb



A → α	FIRST(α)	Follow(α)
S → aT	a	b
T → b|Sb	b, a	b

Pre-computed Parsing Table

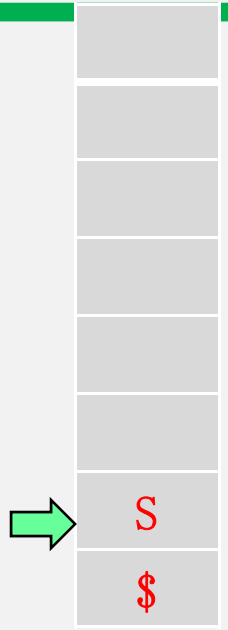
For each production X → α
– for each terminal t in First(α): put α in Table[X,t]
– if ε is in First(α) then:
for each terminal t in Follow(X): put α in Table[X,t]

Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program

bcba

Output



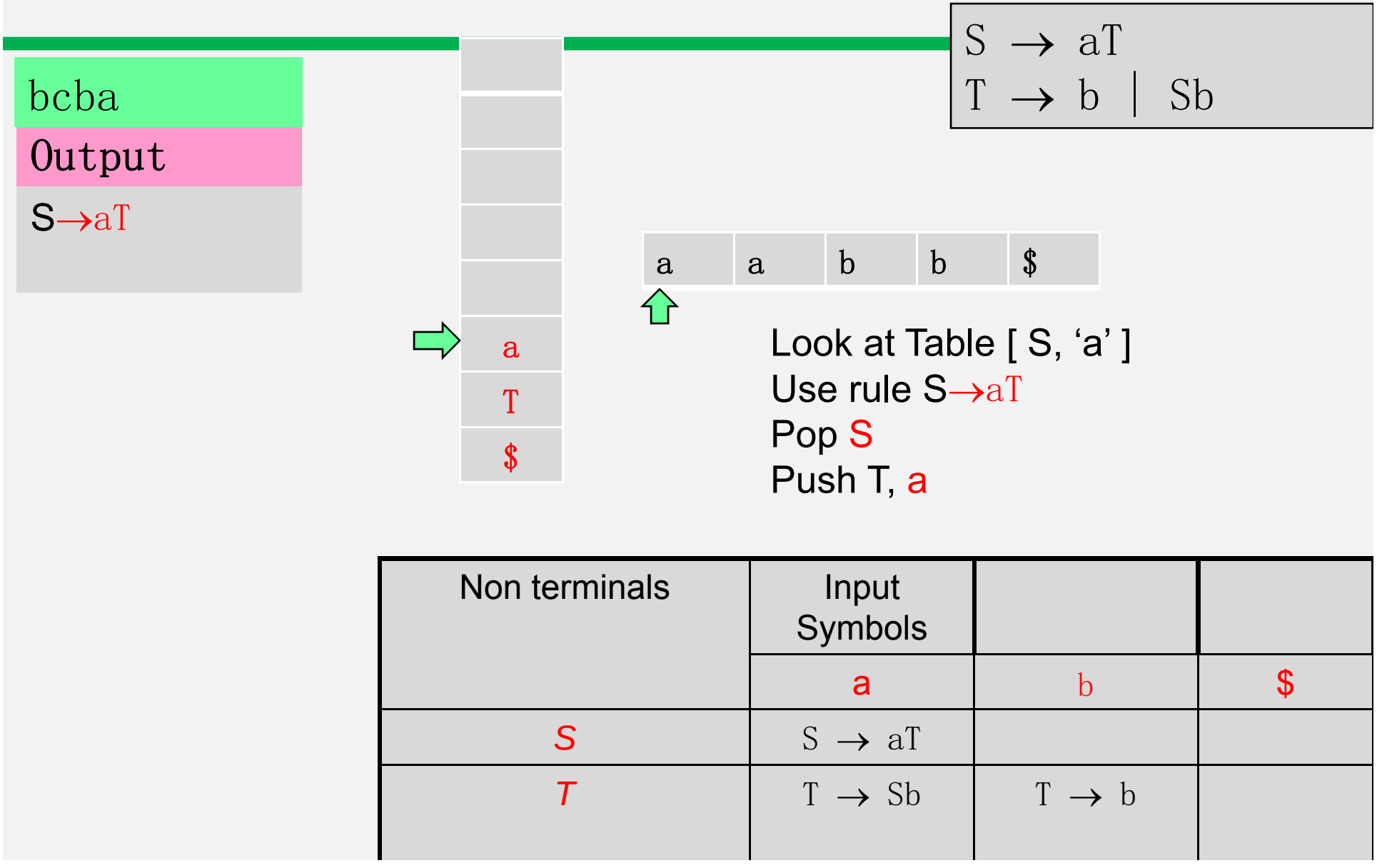
S → aT
T → b | Sb

a	a	b	b	$

Add $ to end of input
Push $
Push S

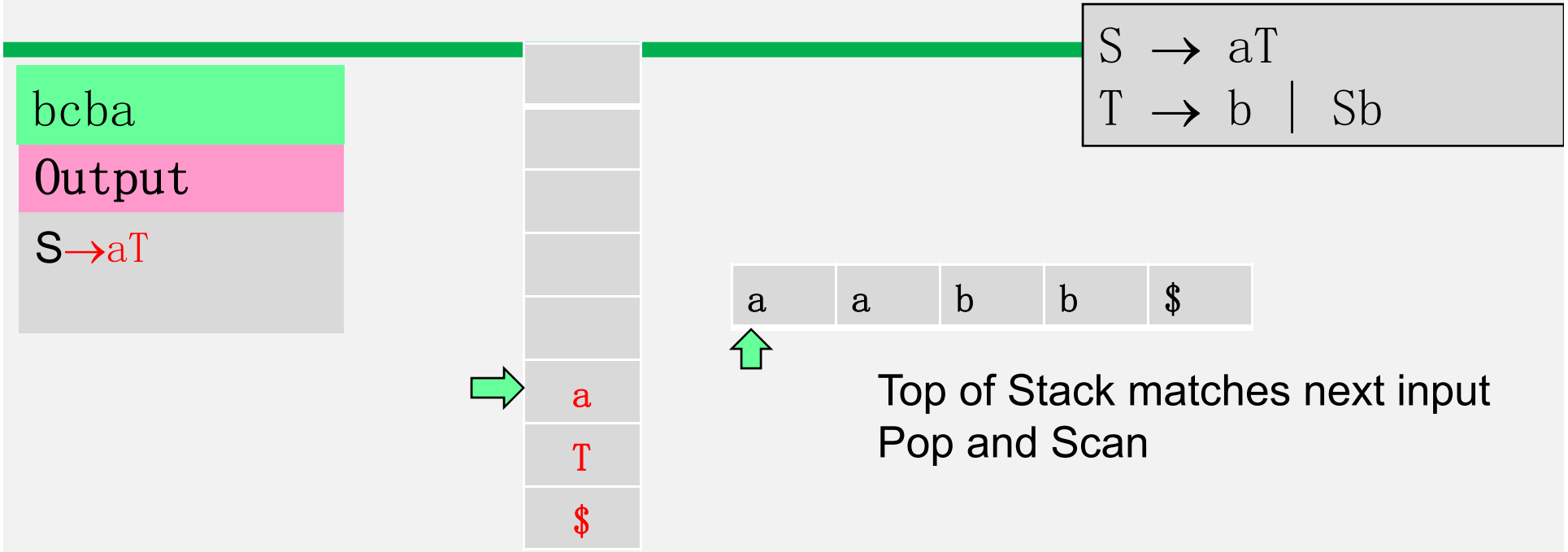
Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program



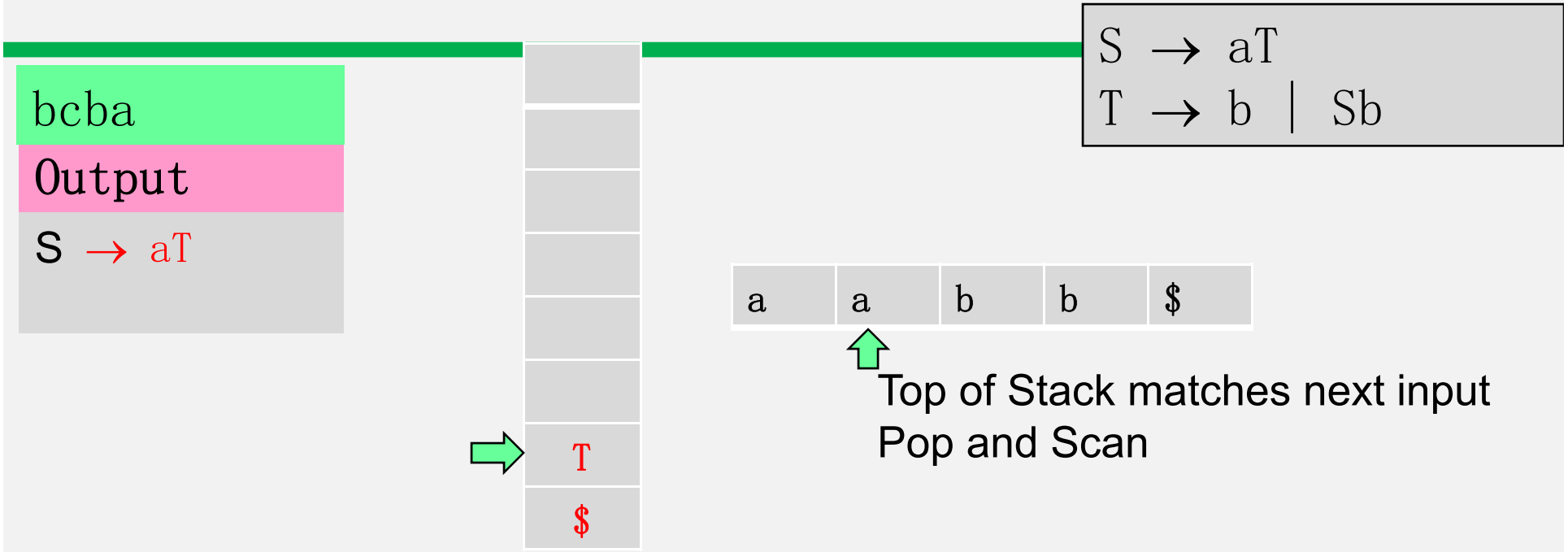
Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program



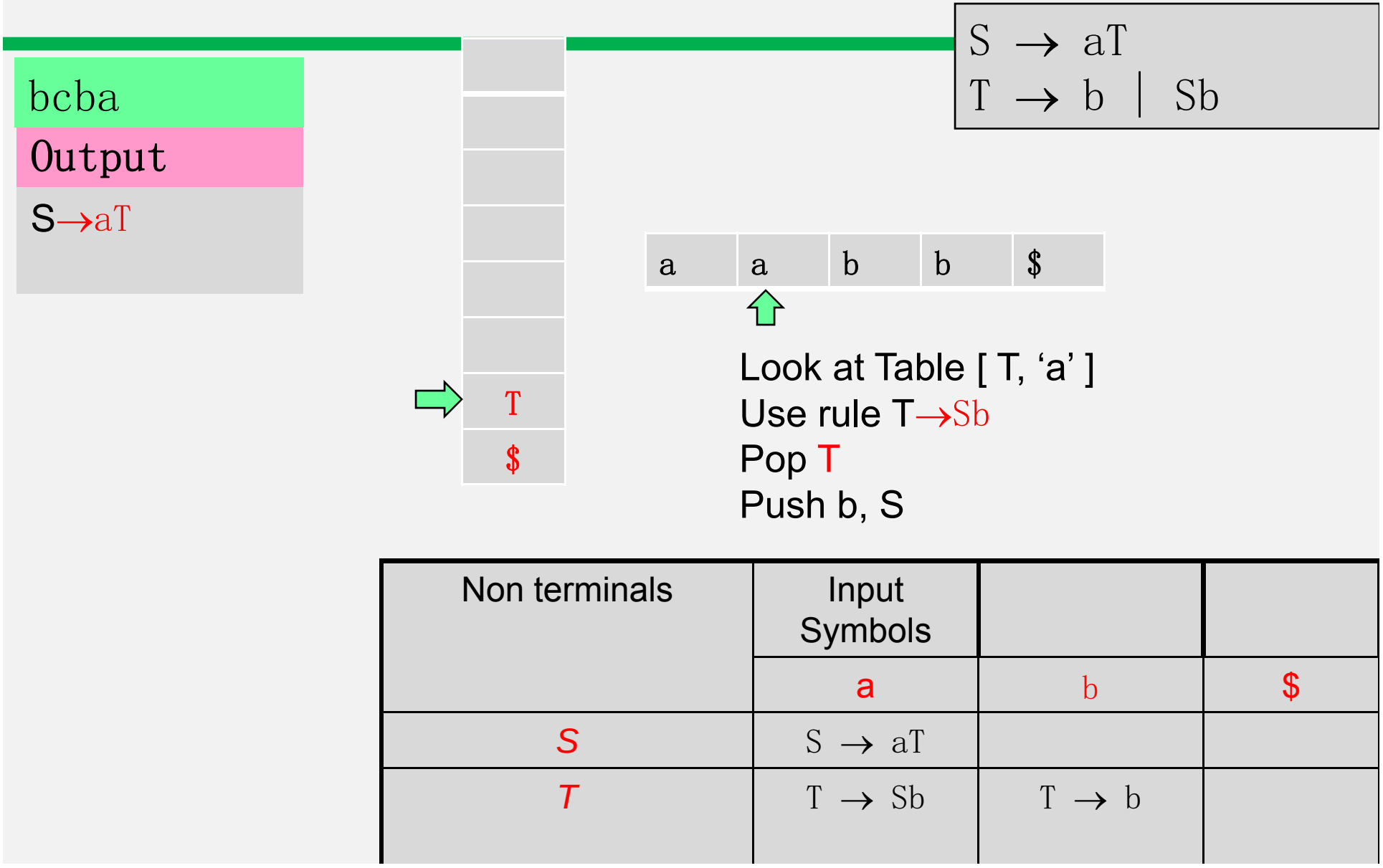
Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program



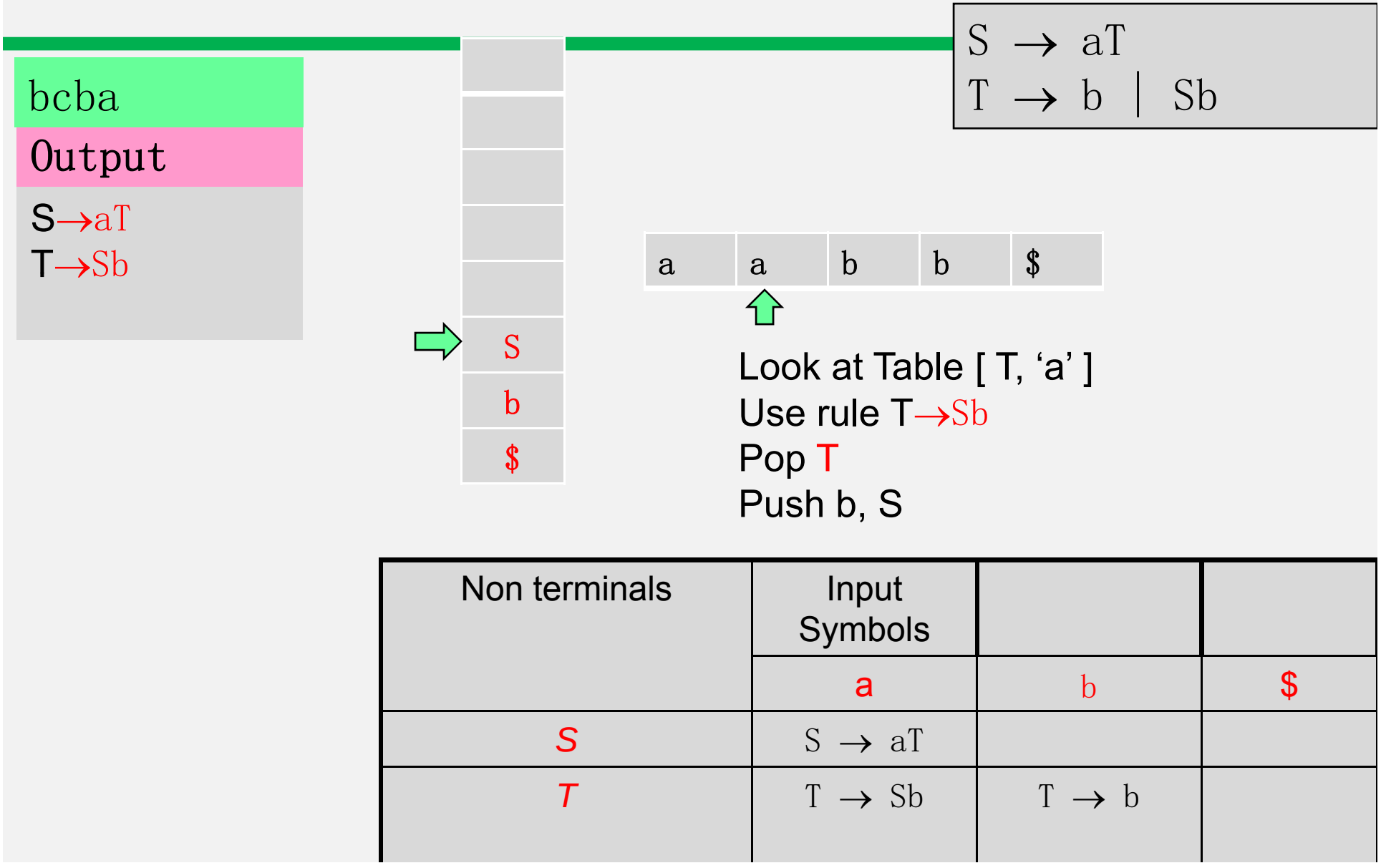
Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program



Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program



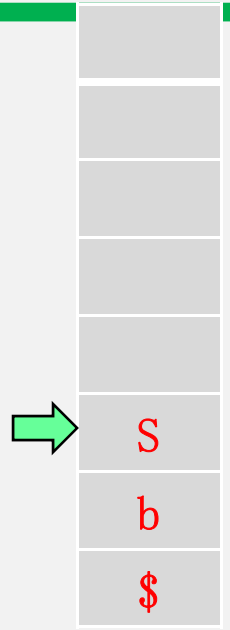
Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program

bcba

Output

S→aT
T→Sb



S → aT
T → b | Sb

a	a	b	b	$

Look at Table [S, 'a']
Use rule S→aT
Pop S
Push T, a

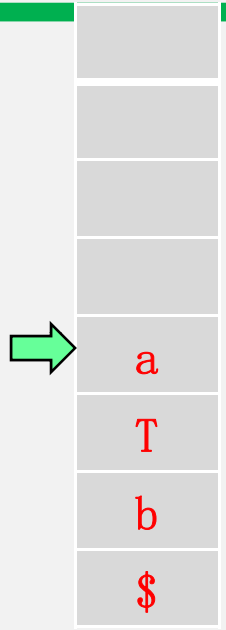
Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program

bcba

Output

S→aT
T→Sb
S→aT



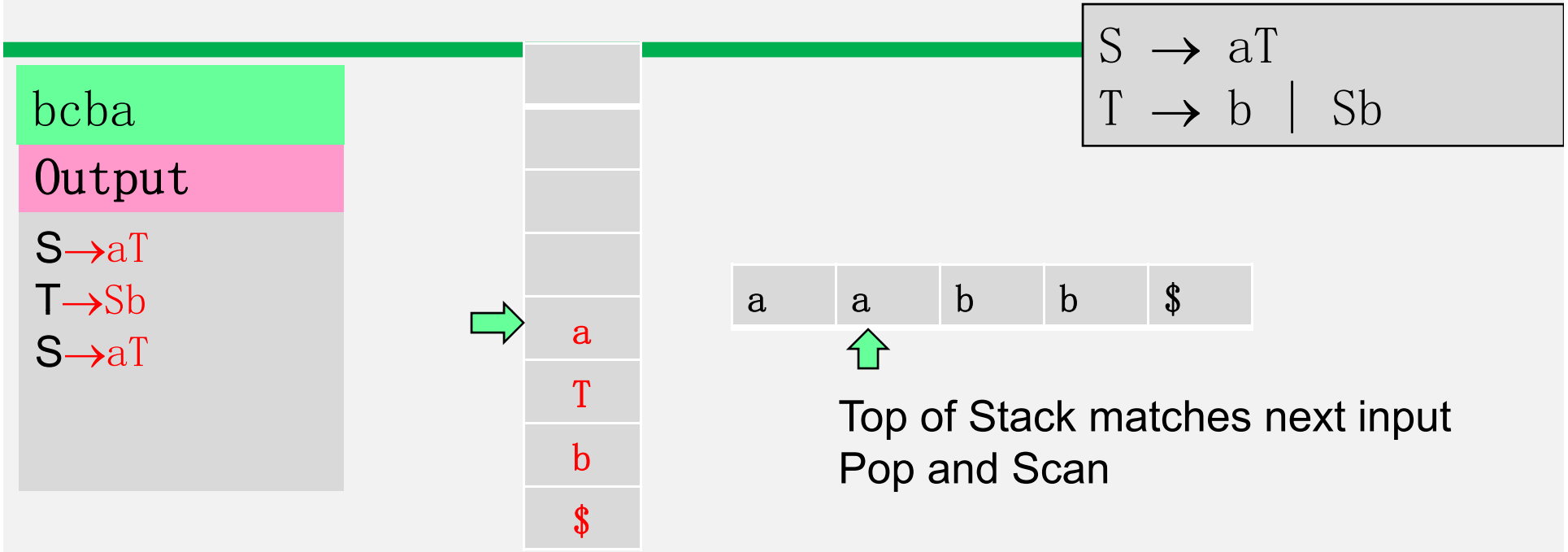
S → aT
T → b | Sb

a	a	b	b	$

Look at Table [S, 'a']
Use rule S→aT
Pop S
Push T, a

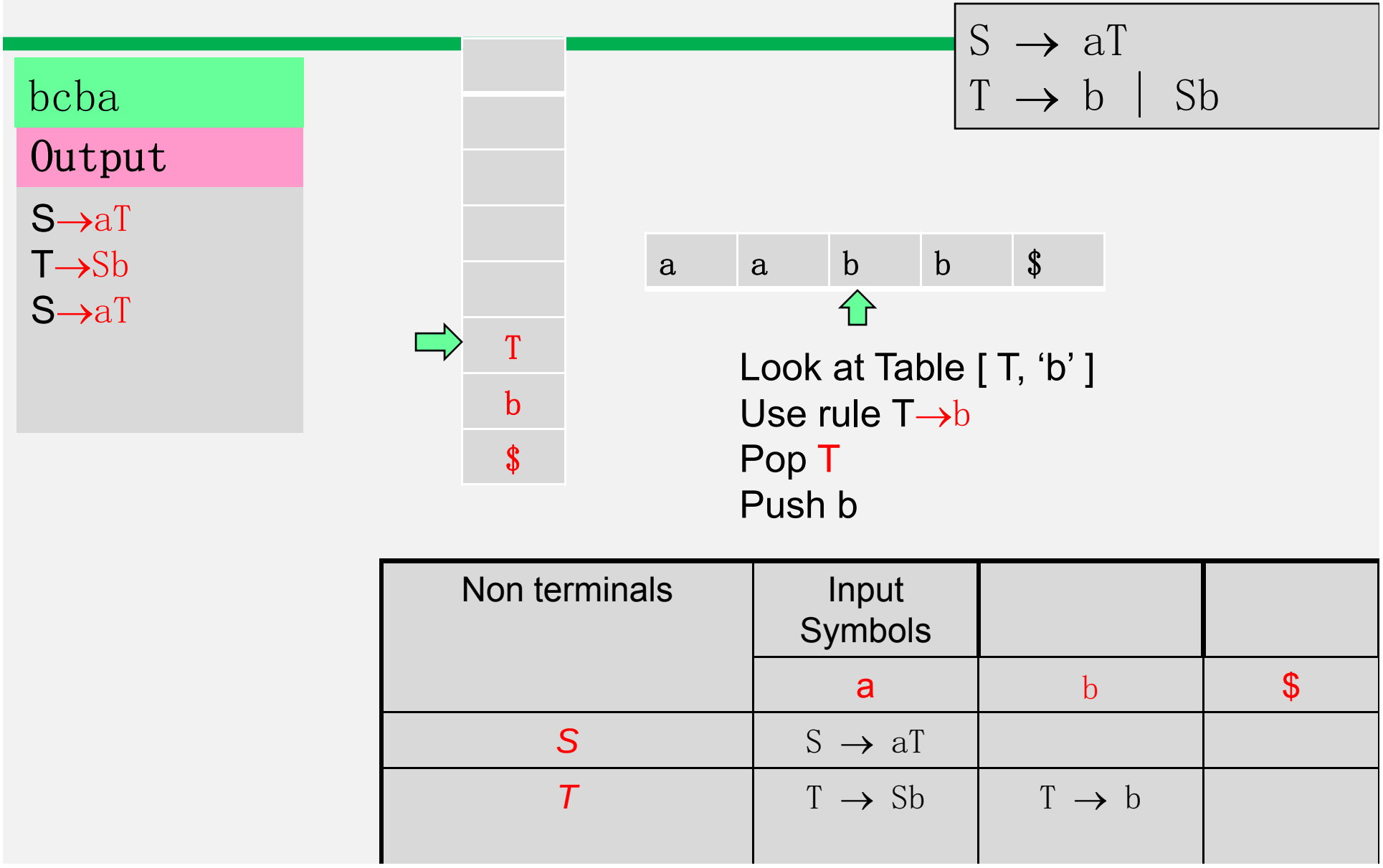
Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program



Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program



Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

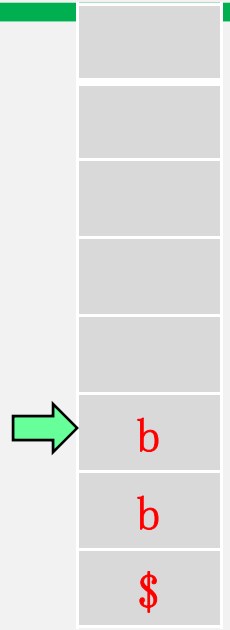
Predictive Parsing Program

S → aT T → b | Sb

bcba

Output

S→aT
T→Sb
S→aT
T→b

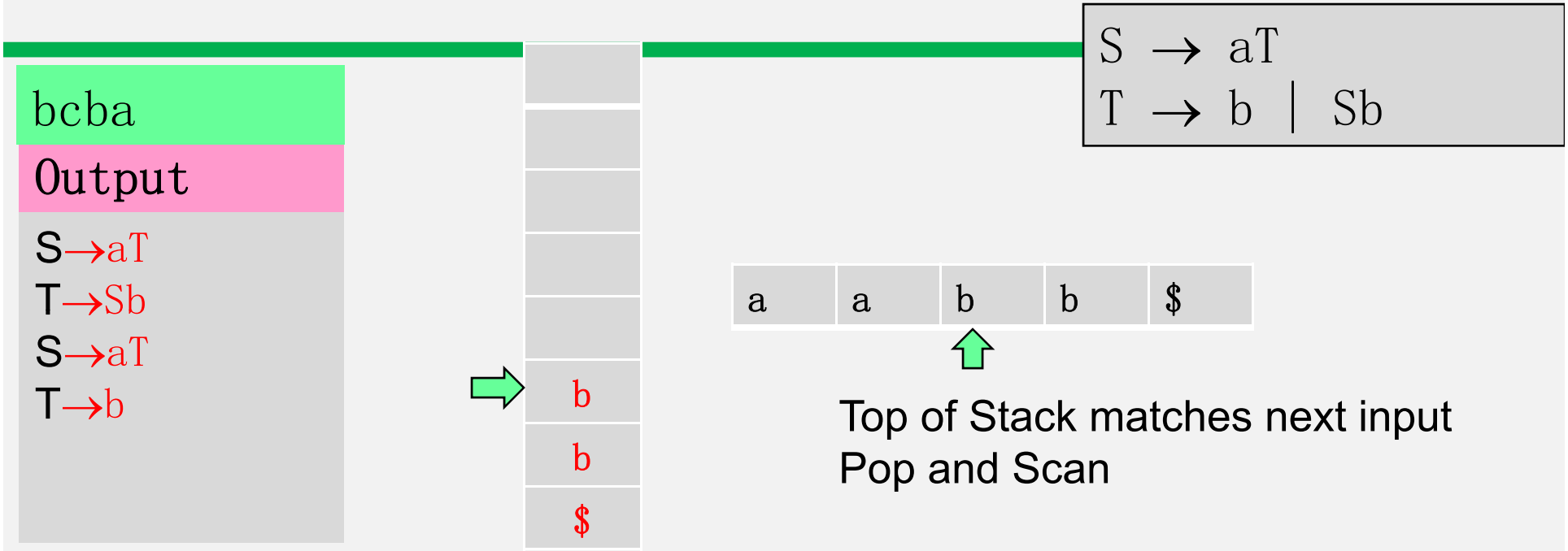


a	a	b	b	$

Look at Table [T, 'b']
Use rule T→b
Pop T
Push b

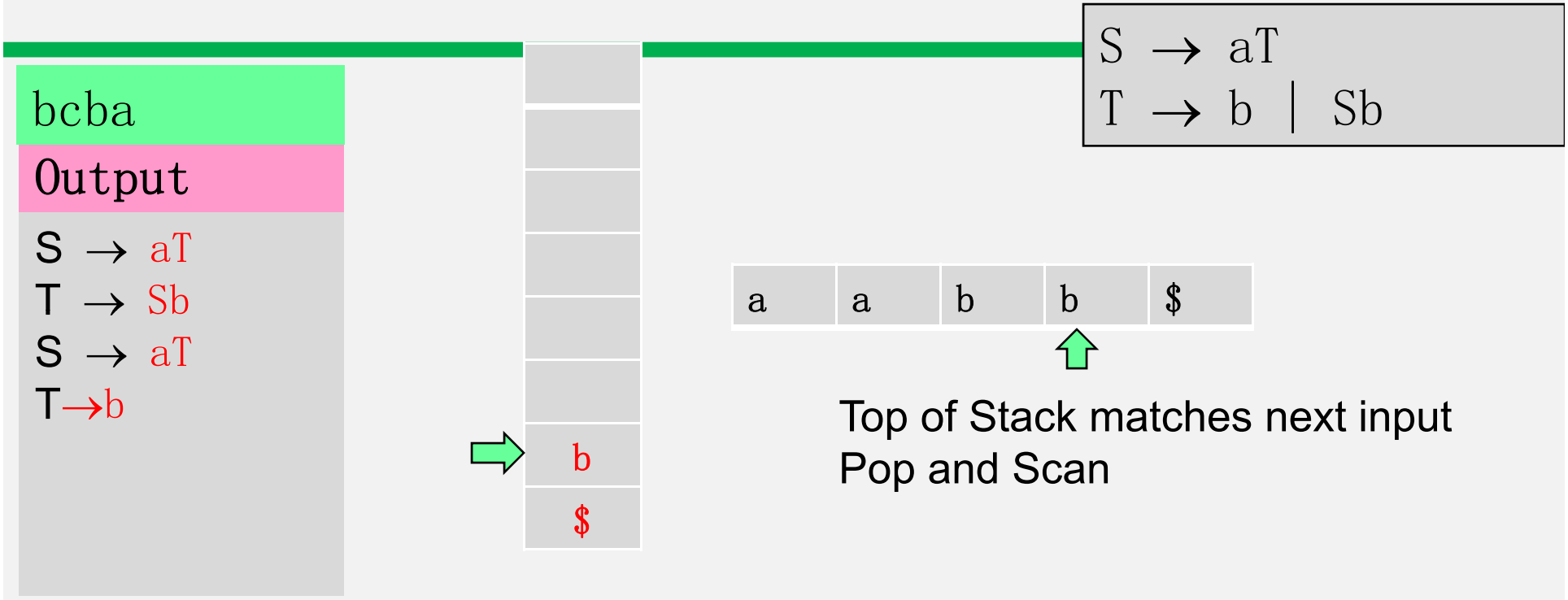
Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program



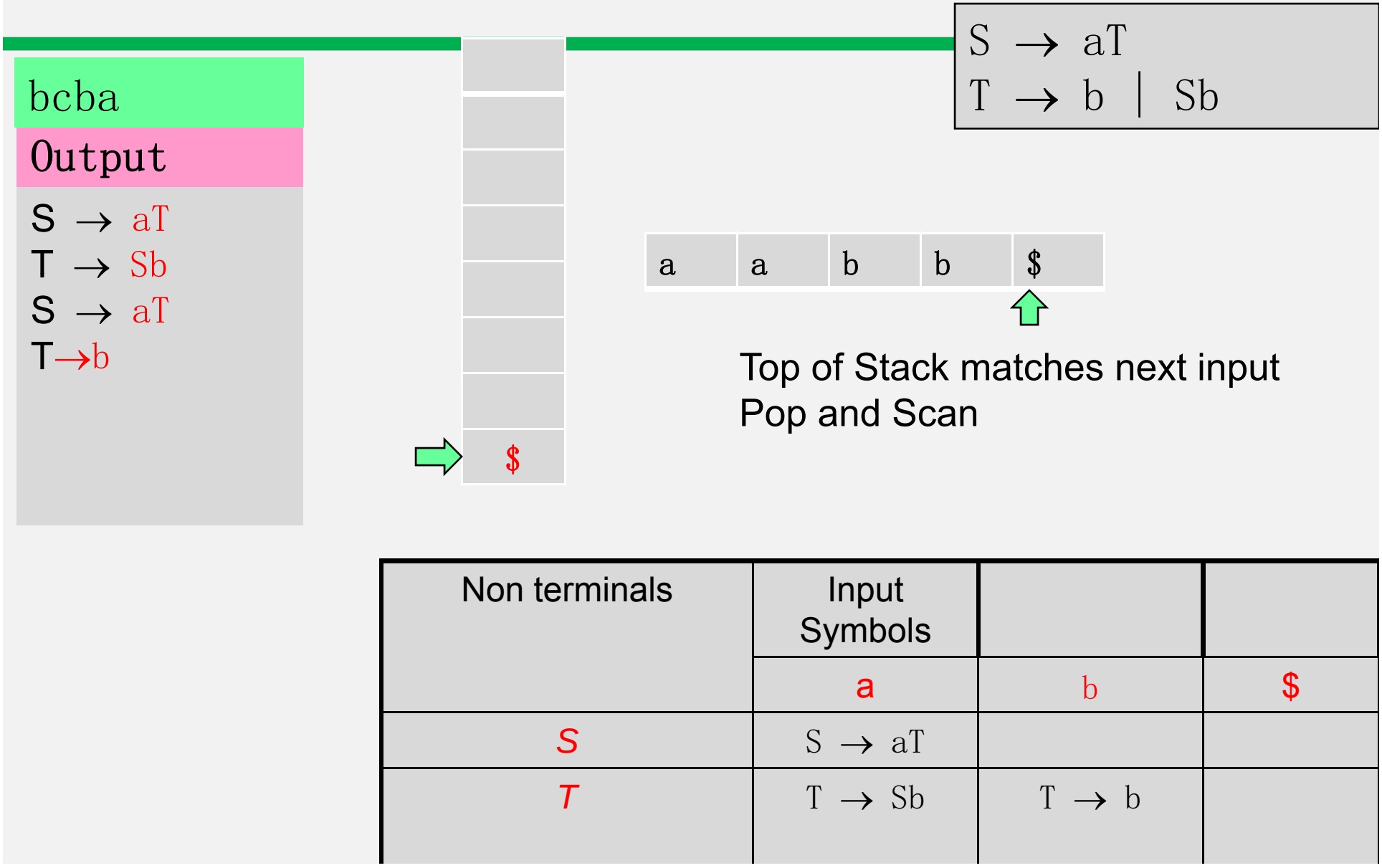
Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program



Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program



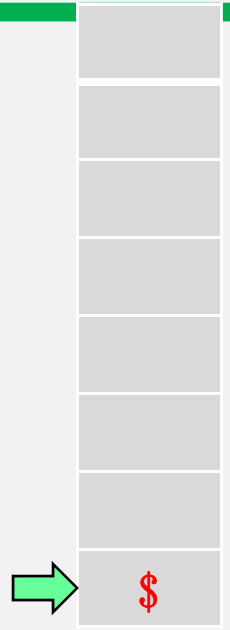
Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing Program

bcba

Output

S → aT
T → Sb
S → aT
T→b



S → aT
T → b | Sb

a	a	b	b	$

Input symbol == $
Top of stack == $
Loop terminates with success

Non terminals	Input Symbols		
	a	b	$
S	S → aT		
T	T → Sb	T → b	

Predictive Parsing

Assume that the grammar is LL(1)

Backtracking will never be needed

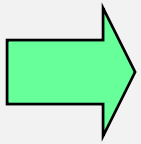
Always know which right hand side to choose (with one look-ahead)

- No Left Recursion
- Grammar is Left-Factored.

$$
\begin{aligned}
S &\rightarrow aAa \mid BAa \mid \varepsilon \\
A &\rightarrow cA \mid bA \mid \varepsilon \\
B &\rightarrow b
\end{aligned}
$$

Example of FIRST

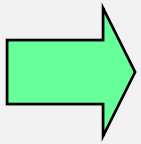
$S \rightarrow aAa|BAa|\varepsilon$
$A \rightarrow cA|bA|\varepsilon$
$B \rightarrow b$



$A \rightarrow \alpha$	FIRST(α)
$S \rightarrow aAa\|BAa\|\varepsilon$	a, b, ε
$A \rightarrow cA\|bA\|\varepsilon$	c, b, ε
$B \rightarrow b$	b

Example of FOLLOW

S → aAa|BAa|ε
A → cA|bA|ε
B → b



A → α	FIRST(α)	Follow(α)
S → aAa|BAa|ε	a, b, ε	
A → cA|bA|ε	c, b, ε	a
B → b	b	

Pre-computed Parsing Table

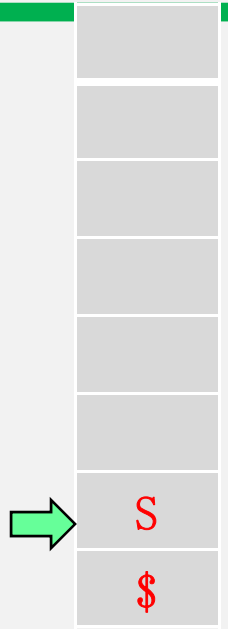
For each production X → α
– for each terminal t in First(α): put α in Table[X,t]
– if ε is in First(α) then:
for each terminal t in Follow(X): put α in Table[X,t]

Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output



$$S \rightarrow aAa \mid BAa \mid \varepsilon$$
$$A \rightarrow cA \mid bA \mid \varepsilon$$
$$B \rightarrow b$$

b	c	b	a	$

Add $ to end of input
Push $
Push S

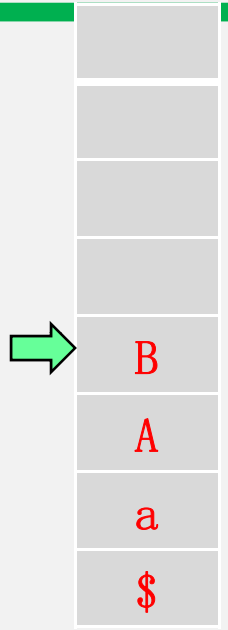
Non terminals	Input Symbols			
	a	b	c	$
S	$S \rightarrow aAa$	$S \rightarrow BAa$		$S \rightarrow \varepsilon$
A	$A \rightarrow \varepsilon$	$A \rightarrow bA$	$A \rightarrow cA$	$A \rightarrow \varepsilon$
B		$B \rightarrow b$		

Predictive Parsing Program

bcba

Output

S → BAa



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Look at Table [S, 'b']
Use rule S→BAa
Pop S
Push a, A, B

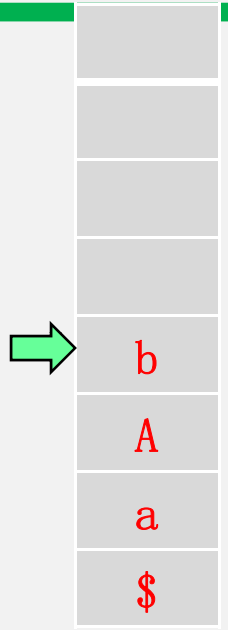
Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output

S → BAa
B → b



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Look at Table [B, 'b']
Use rule B→b
Pop B
Push b

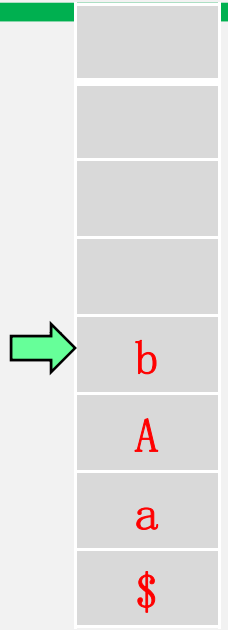
Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output

S → BAa
B → b



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Top of Stack matches next input
Pop and Scan

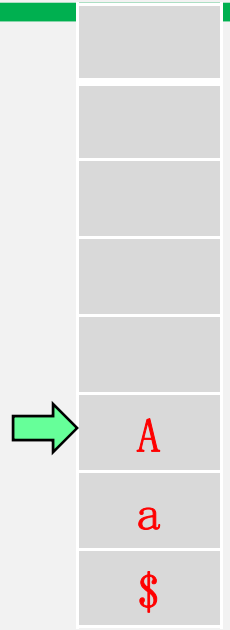
Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output

S → BAa
B → b



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Top of Stack matches next input
Pop and Scan

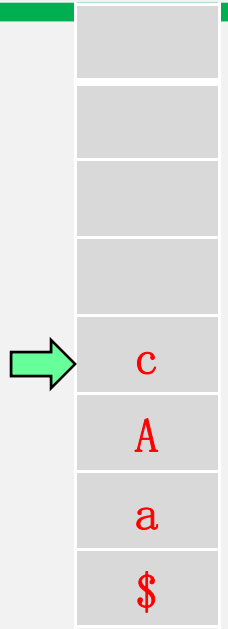
Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output

S → BAa
B → b
A→cA



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Look at Table [A, 'c']
Use rule A→cA
Pop A
Push A, c

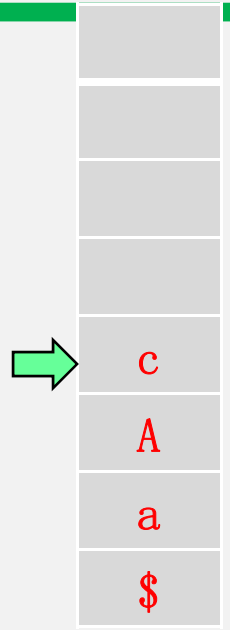
Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output

S → BAa
B → b
A→cA



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Top of Stack matches next input
Pop and Scan

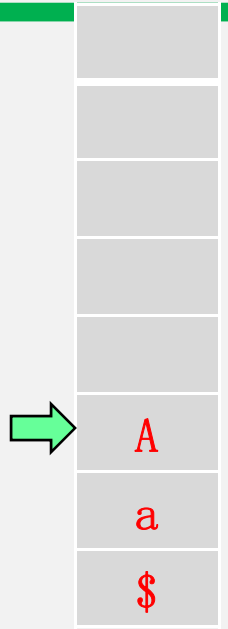
Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output

S → BAa
B → b
A→cA



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Top of Stack matches next input
Pop and Scan

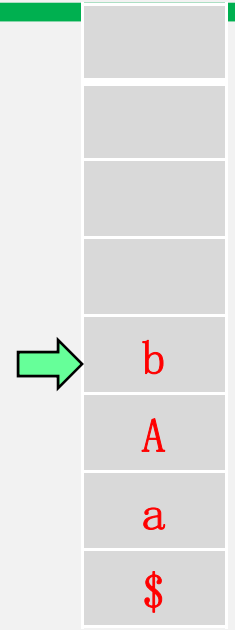
Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output

S → BAa
B → b
A→cA
A→bA



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Look at Table [A, 'b']
Use rule A→bA
Pop A
Push A, b

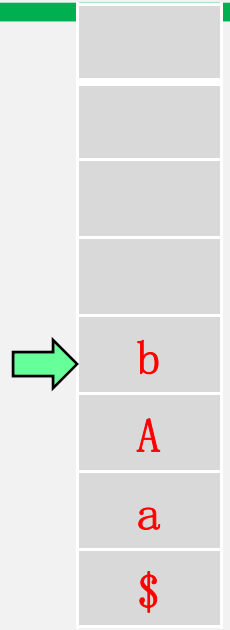
Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output

S → BAa
B → b
A→cA
A→bA



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Top of Stack matches next input
Pop and Scan

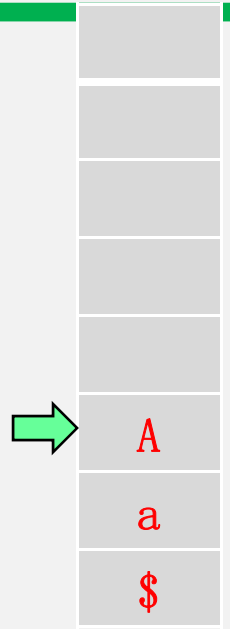
Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output

S → BAa
B → b
A→cA
A→bA



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Top of Stack matches next input
Pop and Scan

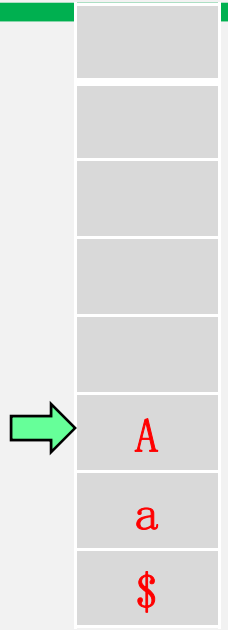
Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output

S → BAa
B → b
A→cA
A→bA
A→ε



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Look at Table [A, 'a']
Use rule A→ε
Pop A

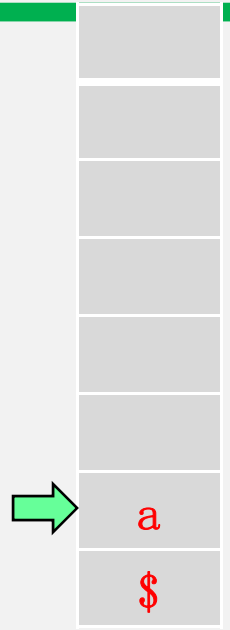
Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output

S → BAa
B → b
A→cA
A→bA
A→ε



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Look at Table [A, 'a']
Use rule A→ε
Pop A

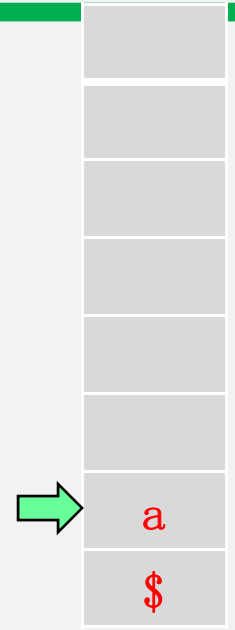
Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output

S → BAa
B → b
A→cA
A→bA
A→ε



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Top of Stack matches next input
Pop and Scan

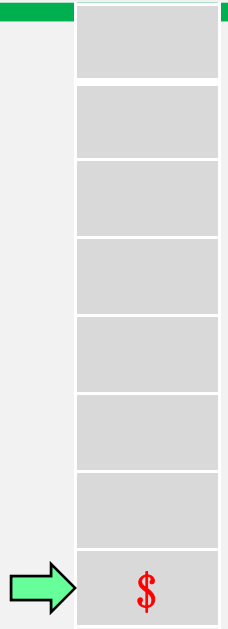
Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Predictive Parsing Program

bcba

Output

S → BAa
B → b
A→cA
A→bA
A→ε



S → aAa|BAa|ε
A → cA|bA|ε
B → b

b	c	b	a	$

Input symbol == $
Top of stack == $
Loop terminates with success

Non terminals	Input Symbols			
	a	b	c	$
S	S → aAa	S → BAa		S → ε
A	A → ε	A → bA	A → cA	A → ε
B		B → b		

Another Example

Construct Parse Tree for the following

$$
\begin{aligned}
S &\rightarrow ABC \\
A &\rightarrow aA \mid \varepsilon \\
B &\rightarrow cB \mid bB \mid \varepsilon \\
C &\rightarrow d
\end{aligned}
$$

Another Example

$S \rightarrow ABC$
$A \rightarrow aA \mid \varepsilon$
$B \rightarrow cB \mid bB \mid \varepsilon$
$C \rightarrow d$

$A \rightarrow \alpha$	FIRST(α)
$S \rightarrow ABC$	a, b, c, d, ε
$A \rightarrow aA \mid \varepsilon$	a, ε
$B \rightarrow cB \mid bB \mid \varepsilon$	c, b, ε
$C \rightarrow d$	d

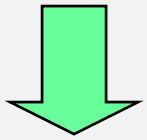
Another Example

$S \rightarrow ABC$
$A \rightarrow aA \mid \varepsilon$
$B \rightarrow cB \mid bB \mid \varepsilon$
$C \rightarrow d$

$A \rightarrow \alpha$	FIRST(α)	Follow(α)
$S \rightarrow ABC$	a, b, c, d	–
$A \rightarrow aA \mid \varepsilon$	a, ε	c, b, d
$B \rightarrow cB \mid bB \mid \varepsilon$	c, b, ε	d
$C \rightarrow d$	d	–

Another Example

$S \rightarrow ABC$
$A \rightarrow aA \mid \varepsilon$
$B \rightarrow cB \mid bB \mid \varepsilon$
$C \rightarrow d$



Non terminals	Input Symbols				
	a	b	c	d	$
S	$S \rightarrow ABC$	$S \rightarrow ABC$	$S \rightarrow ABC$	$S \rightarrow ABC$	
A	$A \rightarrow aA$	$A \rightarrow \varepsilon$	$A \rightarrow \varepsilon$	$A \rightarrow \varepsilon$	$A \rightarrow \varepsilon$
B		$B \rightarrow bB$	$B \rightarrow cB$	$B \rightarrow \varepsilon$	$B \rightarrow \varepsilon$
C				$C \rightarrow d$	

Thank You