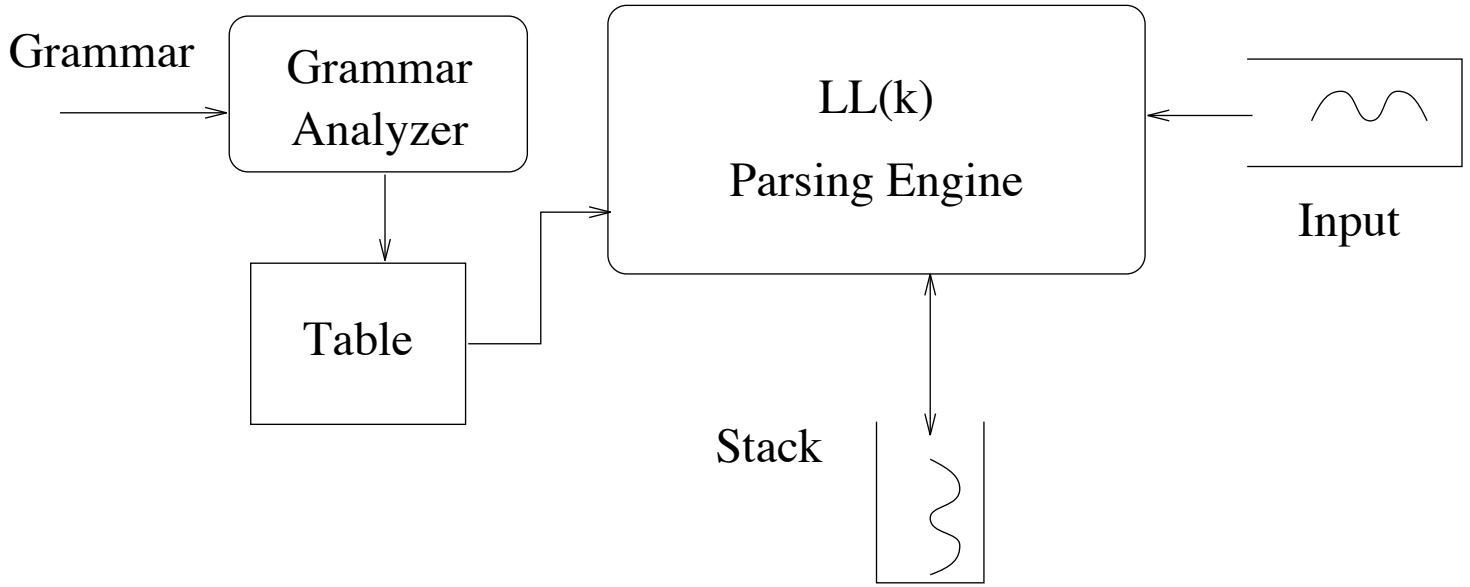
Table-driven $LL(k)$ parsing

Our recursive descent parser contained a procedure for each nonterminal. The generation of these procedures could be automated—through the construction and testing of $First$ and $Follow$ sets—for any grammar free of left recursion.

Another equally automatable approach is to use a simple *parsing engine* that is driven by tables constructed by similar analysis of the grammar.



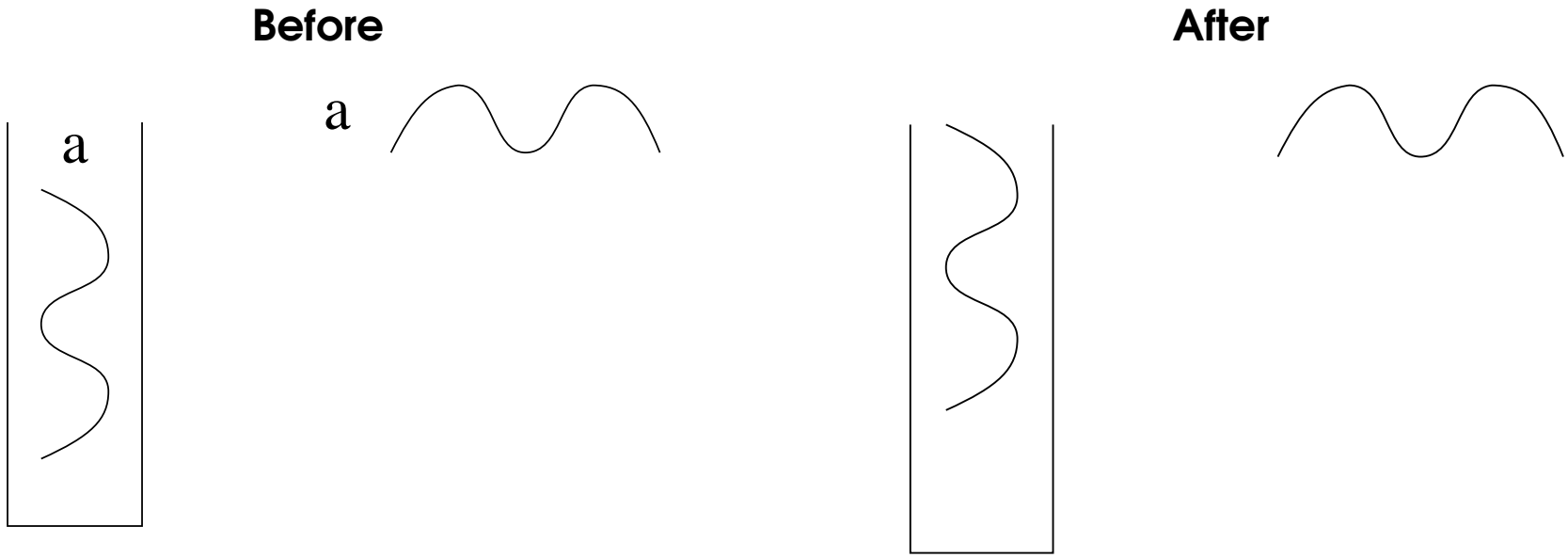
The parsing engine begins by pushing the start symbol S onto the stack. Each subsequent action is one of the following:

Match: pairs an input symbol a an a on top-of-stack.

Apply: replaces the nonterminal N with ω, where $(N \rightarrow \omega) \in P$.

Match

If the top-of-stack contains the terminal symbol "a", then the parsing engine must find an "a" as the next input symbol; the stack is popped, and the input is advanced.

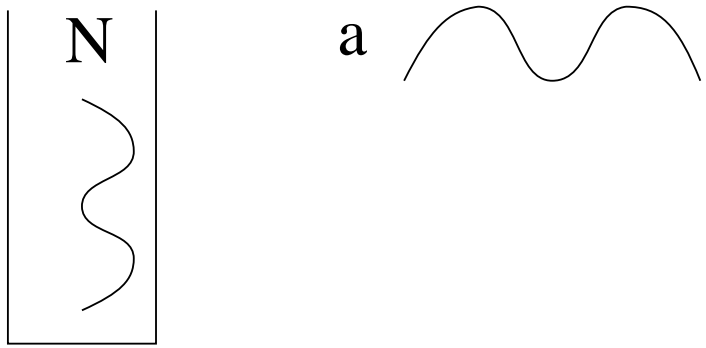


- If a match simultaneously empties the stack and exhausts the input stream, then the string is *accepted* by the parser.
- If a match is attempted, but the symbols disagree, then an error is declared.

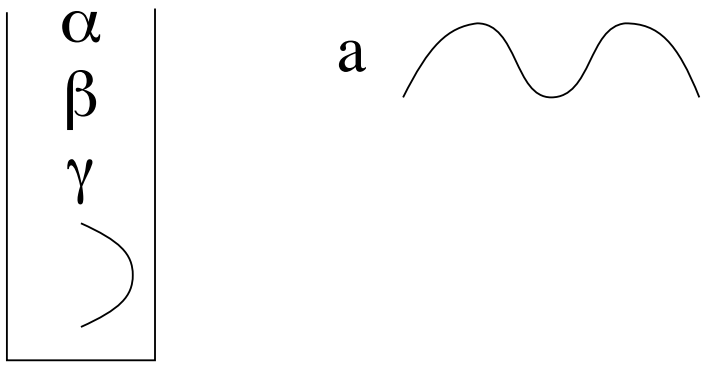
Apply

If the top-of-stack contains a nonterminal N, then the parsing engine must choose the appropriate rule for N, say $N \rightarrow \alpha\beta\gamma$. The stack is popped of symbol N, and the symbols α, β, and γ are pushed onto the stack, such that α is the new top-of-stack.

Before



After



Since a match is always required when a terminal is exposed on top-of-stack, the only information that must be coded in our table is the rule that should be applied when a nonterminal appears on top-of-stack. As with our recursive descent parser, this decision can be based on k symbols of lookahead into the input stream.

Constructing the table

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

	First	Follow
S	$\{ a, b, d, c, \$ \}$	$\{ \}$
A	$\{ a, b, d, \lambda \}$	$\{ c, \$ \}$
B	$\{ b, d \}$	$\{ c, d, q \}$
C	$\{ c, \lambda \}$	$\{ d, \$ \}$
Q	$\{ q \}$	$\{ c, \$ \}$

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

The nonblank entries in the above table indicate the number of the rule that should be applied, given a nonterminal on top-of-stack and an input symbol as lookahead.

Using the table

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

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

Below is shown the stack activity in parsing the input string "abbddc$".

		a	b	b			
		B	B	B	d		
		C	C	C	C		
	A	d	d	d	d	d	
	C	C	C	C	C	C	c
S	$	$	$	$	$	$	$
	1	4	7	7	8	3	2

Input string

a b b d d c $