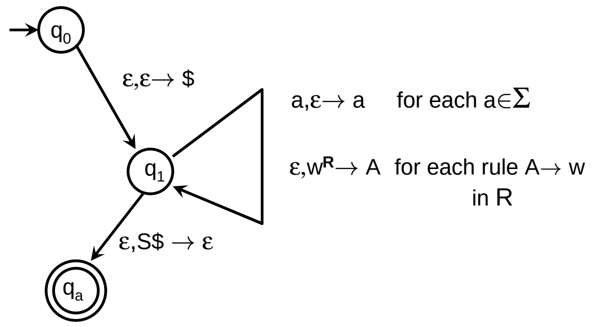
CSE 322
Introduction to Formal Models in Computer Science
Bottom-up (Shift-Reduce) Parsing

The parsing method given in the text to produce a PDA from a CFG is *top-down* in that it begins by putting S (from the top of the parse tree) on the stack and then applies the rules on the stack, matching terminal symbols with the input as they appear on the top of the stack.

Another parsing method that is the basis for most parsers in practice works *bottom-up* in that tries to work from the bottom of the parse tree converting terminal symbols to non-terminals, eventually trying to produce a start symbol S on the stack.

This method has two kinds of rules: rules to *shift* symbols from the input onto the stack, and rules to *reduce* the right-hand sides of rules on the stack to their left-hand sides. Because the shifting rules reverse the input the reduce rules take the reversed form of right-hand sides of rules. Intuitively, on input w a bottom-up parser follows a *post-order* traversal of a parse tree for w, applying shift of the symbol at each leaf node and reduce when it traverses each internal node.

Thus given a grammar $G = (V, \Sigma, R, S)$ we get the following PDA.



Bottom-up parsing for the grammar

$$S \rightarrow aSb \mid \epsilon$$

yields the following sequence of configurations for parsing $aabb$:

state	input remaining	stack
q_0	$aabb$	ε
q_1	$aabb$	$\$$
q_1	abb	$a\$$
q_1	bb	$aa\$$
q_1	bb	$Saa\$$
q_1	b	$bSaa\$$
q_1	b	$Sa\$$
q_1	ε	$bSa\$$
q_1	ε	$S\$$
q_a	ε	ε