

Actions of a YACC-generated Parser

<i>name</i>	shift	<i>n</i>
-------------	--------------	----------

If the lookahead token is *name* then

1. push state *n* on the stack (and also push the value *yylval* associated with *name* on the value stack).
2. clear the lookahead token.

<i>name</i>	reduce	<i>m</i>
-------------	---------------	----------

If the lookahead token is *name*, then refer to the grammar rule *m* :

1. execute the action code associated with the rule if supplied. (The return value of the action code becomes the current value of *yyval*)
2. if there are *k* symbols on the RHS of the rule, pop off *k* states from the stack (and also pop off *k* values from the value stack).
3. the symbol on the LHS of the grammar rule becomes the current nonterminal for **goto** operation that follows.

<i>name</i>	goto	<i>n</i>
-------------	-------------	----------

If the current nonterminal is *name* then

push state *n* on the stack (and also push the value *yyval* on the value stack).

accept

The lookahead token is the *endmarker* (ie. `yylex()` returned 0).

This means that the parser accepts the entire input and so `yparse()` returns 0 to its caller.

error

The parser cannot continue parsing according to the grammar rules given, so it will do the following actions:

1. call `yyerror()` to report an error message
2. attempt to do the error recovery. If successful, the parsing resumes; otherwise the parsing halts and the parser `yparse()` return 1 to its caller.

Note

- Initially, the stack contains only state 0. In other words, the initial state of the parser is 0.
- The top of stack is the current state of the parser. The current state indicates what has been found so far in the input.
- Since *shift* and *goto* push a new state n on the stack, this has the effect of changing the current state to n .
- *Shift* has the effect of "shifting" the lookahead token, while *goto* has the effect of "shifting" the current nonterminal. As a consequence, the lookahead token is cleared by *shift* but unaffected by *goto*.

Example of Parsing Actions

action	Input status	stack
	<code>_DING DONG DELL \$end</code>	<code>0</code>
<code>shift 3</code>	<code>DING_DONG DELL \$end</code>	<code>0 3</code>
<code>shift 6</code>	<code>DING DONG_DELL \$end</code>	<code>0 3 6</code>
<code>reduce 2</code>	<code>_sound DELL \$end</code>	<code>0</code>
<code>goto 2</code>	<code>sound_DELL \$end</code>	<code>0 2</code>
<code>shift 5</code>	<code>sound DELL_\$end</code>	<code>0 2 5</code>
<code>reduce 3</code>	<code>sound_place \$end</code>	<code>0 2</code>
<code>goto 4</code>	<code>sound place_\$end</code>	<code>0 2 4</code>
<code>reduce 1</code>	<code>_rhyme \$end</code>	<code>0</code>
<code>goto 1</code>	<code>rhyme_\$end</code>	<code>0 1</code>
<code>accept</code>	<code>rhyme_\$end</code>	<code>0 1</code>