Type declaration in yacc:

```
%union    {
       int  ival;
       double  dval;
       INTERVAL  vval;
       }
%token <ival> DREG VREG /* indices into dreg, vreg arrays */
%token <dval> CONST     /* floating point constant */
%type <dval> dexp       /* expression */
%type <vval> vexp       /* interval expression */

%union {
 float num;
 char *id;
 exp_node *expnode;
}
%type <expnode> Start Stmt Stmts E T
```

Figure 7.17 shows the semantic actions for building the AST. (yacc input)

1. $Start \rightarrow Stmt\,\$$

. $\{$ $$ = $1; $\}$

2. $Stmt \rightarrow id\ assign\ E$

. $\{$ $$ = MakeTree(ASSIGN, MakeNode($1), $3, NULL); $\}$

3. $Stmt \rightarrow if\ lparen\ E\ rparen\ Stmt\ fi$

. $\{$ $$ = MakeTree(IF, $3, $5, NULL); $\}$

4. $Stmt \rightarrow if\ lparen\ E\ rparen\ Stmt\ else\ Stmt\ fi$

. $\{$ $$ = MakeTree(IF, $3, $5, $7); $\}$

5. $Stmt \rightarrow while\ lparen\ E\ rparen\ do\ Stmt\ od$

. $\{$ $$ = MakeTree(WHILE, $3, $6, NULL); $\}$

6. $Stmt \rightarrow begin\ Stmts\ end$

. $\{$ $$ = MakeTree(BLOCK, $2, NULL, NULL); $\}$

7. $Stmts \rightarrow Stmts\ semi\ Stmt$

. $\{$ $$ = Append($1, $3); $\}$

8. $Stmts \rightarrow Stmt$

. $\{$ $$ = MakeList($1); $\}$

9. $E \rightarrow E\ plus\ T$

. $\{$ $$ = MakeTree(PLUS, $1, $3, NULL); $\}$

10. $E \to T$

. $\{$ $$ = $1; $\}$

11. $T \to id$

. $\{$ $$ = MakeNode($1); $\}$

12. $T \to num$

. $\{$ $$ = MakeNode($1); $\}$

Figure 7.17 Semantic actions for building the ast for the grammar in Figure 7.14