```
// $Id: ambiguous-else.y,v 1.1 2011-10-28 18:07:07-07 - - $

// Example of solving the problem of the dangling else with an
// ambiguous grammar and precedence declarations.

%verbose

%token IF WHILE
%right ELSE
%start program

%%

program    : program statement
           |
           ;

statement  : ifhead statement ELSE statement
           | ifhead statement %prec ELSE
           | whilehead statement
           | otherstmt
           ;

ifhead     : IF ’(’ expr ’)’
           ;

whilehead  : WHILE ’(’ expr ’)’
           ;

otherstmt  : expr ’;’
           ;

expr       : ’x’
           ;

%%
```

```
bison -v ambiguous-else.y
::::::::::::::
ambiguous-else.stdout
::::::::::::::
::::::::::::::
ambiguous-else.stderr
::::::::::::::
::::::::::::::
ambiguous-else.output
::::::::::::::
Grammar

    0 $accept: program $end

    1 program: program statement
    2        | /* empty */

    3 statement: ifhead statement ELSE statement
    4          | ifhead statement
    5          | whilehead statement
    6          | otherstmt

    7 ifhead: IF ’(’ expr ’)’

    8 whilehead: WHILE ’(’ expr ’)’

    9 otherstmt: expr ’;’

   10 expr: ’x’


Terminals, with rules where they appear

$end (0) 0
’(’ (40) 7 8
’)’ (41) 7 8
’;’ (59) 9
’x’ (120) 10
error (256)
IF (258) 7
WHILE (259) 8
ELSE (260) 3


Nonterminals, with rules where they appear

$accept (10)
    on left: 0
program (11)
    on left: 1 2, on right: 0 1
statement (12)
    on left: 3 4 5 6, on right: 1 3 4 5
ifhead (13)
    on left: 7, on right: 3 4
whilehead (14)
    on left: 8, on right: 5
otherstmt (15)
    on left: 9, on right: 6
expr (16)
    on left: 10, on right: 7 8 9


state 0

```

```
    0 $accept: . program $end

    $default  reduce using rule 2 (program)

    program  go to state 1


state 1

    0 $accept: program . $end
    1 program: program . statement

    $end   shift, and go to state 2
    IF     shift, and go to state 3
    WHILE  shift, and go to state 4
    ’x’    shift, and go to state 5

    statement  go to state 6
    ifhead     go to state 7
    whilehead  go to state 8
    otherstmt  go to state 9
    expr       go to state 10


state 2

    0 $accept: program $end .

    $default  accept


state 3

    7 ifhead: IF . ’(’ expr ’)’

    ’(’  shift, and go to state 11


state 4

    8 whilehead: WHILE . ’(’ expr ’)’

    ’(’  shift, and go to state 12


state 5

   10 expr: ’x’ .

    $default  reduce using rule 10 (expr)


state 6

    1 program: program statement .

    $default  reduce using rule 1 (program)


state 7

    3 statement: ifhead . statement ELSE statement
    4          | ifhead . statement

    IF     shift, and go to state 3
    WHILE  shift, and go to state 4
    ’x’    shift, and go to state 5

    statement  go to state 13
    ifhead     go to state 7
    whilehead  go to state 8
    otherstmt  go to state 9
    expr       go to state 10


state 8

    5 statement: whilehead . statement

    IF     shift, and go to state 3
    WHILE  shift, and go to state 4
    ’x’    shift, and go to state 5

    statement  go to state 14
    ifhead     go to state 7
    whilehead  go to state 8
    otherstmt  go to state 9
    expr       go to state 10


state 9

    6 statement: otherstmt .

    $default  reduce using rule 6 (statement)


state 10

    9 otherstmt: expr . ’;’

    ’;’  shift, and go to state 15


state 11

    7 ifhead: IF ’(’ . expr ’)’

    ’x’  shift, and go to state 5

    expr  go to state 16


state 12

    8 whilehead: WHILE ’(’ . expr ’)’

    ’x’  shift, and go to state 5

    expr  go to state 17


state 13

    3 statement: ifhead statement . ELSE statement
    4          | ifhead statement .

    ELSE  shift, and go to state 18

    $default  reduce using rule 4 (statement)


state 14

    5 statement: whilehead statement .

    $default  reduce using rule 5 (statement)


state 15

    9 otherstmt: expr ’;’ .

    $default  reduce using rule 9 (otherstmt)


state 16

    7 ifhead: IF ’(’ expr . ’)’

    ’)’  shift, and go to state 19


state 17

    8 whilehead: WHILE ’(’ expr . ’)’

    ’)’  shift, and go to state 20


state 18

    3 statement: ifhead statement ELSE . statement

    IF     shift, and go to state 3
    WHILE  shift, and go to state 4
    ’x’    shift, and go to state 5

    statement  go to state 21
    ifhead     go to state 7
    whilehead  go to state 8
    otherstmt  go to state 9
    expr       go to state 10


state 19

    7 ifhead: IF ’(’ expr ’)’ .

    $default  reduce using rule 7 (ifhead)


state 20

    8 whilehead: WHILE ’(’ expr ’)’ .

    $default  reduce using rule 8 (whilehead)


state 21

    3 statement: ifhead statement ELSE statement .

    $default  reduce using rule 3 (statement)
```

```
// $Id: unambiguous-else.y,v 1.1 2011-10-28 18:07:07-07 - - $

// Example of solving the problem of the dangling else with an
// ambiguous grammar and precedence declarations.

%verbose

%token IF WHILE
%right ELSE
%start program

%%

program    : program statement
           |
           ;

statement  : closedstmt
           | openstmt
           ;

closedstmt : ifhead closedstmt ELSE closedstmt
           | whilehead closedstmt
           | otherstmt
           ;

openstmt   : ifhead closedstmt ELSE openstmt
           | ifhead statement
           | whilehead openstmt

ifhead     : IF ’(’ expr ’)’
           ;

whilehead  : WHILE ’(’ expr ’)’
           ;

otherstmt  : expr ’;’
           ;

expr       : ’x’
           ;

%%

```

```
bison -v unambiguous-else.y
::::::::::::::
unambiguous-else.stdout
::::::::::::::
::::::::::::::
unambiguous-else.stderr
::::::::::::::
::::::::::::::
unambiguous-else.output
::::::::::::::
Grammar

    0 $accept: program $end

    1 program: program statement
    2        | /* empty */

    3 statement: closedstmt
    4          | openstmt

    5 closedstmt: ifhead closedstmt ELSE closedstmt
    6           | whilehead closedstmt
    7           | otherstmt

    8 openstmt: ifhead closedstmt ELSE openstmt
    9         | ifhead statement
   10         | whilehead openstmt

   11 ifhead: IF ’(’ expr ’)’

   12 whilehead: WHILE ’(’ expr ’)’

   13 otherstmt: expr ’;’

   14 expr: ’x’


Terminals, with rules where they appear

$end (0) 0
’(’ (40) 11 12
’)’ (41) 11 12
’;’ (59) 13
’x’ (120) 14
error (256)
IF (258) 11
WHILE (259) 12
ELSE (260) 5 8


Nonterminals, with rules where they appear

$accept (10)
    on left: 0
program (11)
    on left: 1 2, on right: 0 1
statement (12)
    on left: 3 4, on right: 1 9
closedstmt (13)
    on left: 5 6 7, on right: 3 5 6 8
openstmt (14)
    on left: 8 9 10, on right: 4 8 10
ifhead (15)
    on left: 11, on right: 5 8 9
```

```
whilehead (16)
    on left: 12, on right: 6 10
otherstmt (17)
    on left: 13, on right: 7
expr (18)
    on left: 14, on right: 11 12 13


state 0

    0 $accept: . program $end

    $default  reduce using rule 2 (program)

    program  go to state 1


state 1

    0 $accept: program . $end
    1 program: program . statement

    $end   shift, and go to state 2
    IF     shift, and go to state 3
    WHILE  shift, and go to state 4
    ’x’    shift, and go to state 5

    statement   go to state 6
    closedstmt  go to state 7
    openstmt    go to state 8
    ifhead      go to state 9
    whilehead   go to state 10
    otherstmt   go to state 11
    expr        go to state 12


state 2

    0 $accept: program $end .

    $default  accept


state 3

   11 ifhead: IF . ’(’ expr ’)’

    ’(’  shift, and go to state 13


state 4

   12 whilehead: WHILE . ’(’ expr ’)’

    ’(’  shift, and go to state 14


state 5

   14 expr: ’x’ .

    $default  reduce using rule 14 (expr)


```

```
state 6

    1 program: program statement .

    $default  reduce using rule 1 (program)


state 7

    3 statement: closedstmt .

    $default  reduce using rule 3 (statement)


state 8

    4 statement: openstmt .

    $default  reduce using rule 4 (statement)


state 9

    5 closedstmt: ifhead . closedstmt ELSE closedstmt
    8 openstmt: ifhead . closedstmt ELSE openstmt
    9         | ifhead . statement

    IF     shift, and go to state 3
    WHILE  shift, and go to state 4
    ’x’    shift, and go to state 5

    statement   go to state 15
    closedstmt  go to state 16
    openstmt    go to state 8
    ifhead      go to state 9
    whilehead   go to state 10
    otherstmt   go to state 11
    expr        go to state 12


state 10

    6 closedstmt: whilehead . closedstmt
   10 openstmt: whilehead . openstmt

    IF     shift, and go to state 3
    WHILE  shift, and go to state 4
    ’x’    shift, and go to state 5

    closedstmt  go to state 17
    openstmt    go to state 18
    ifhead      go to state 9
    whilehead   go to state 10
    otherstmt   go to state 11
    expr        go to state 12


state 11

    7 closedstmt: otherstmt .

    $default  reduce using rule 7 (closedstmt)


```

```
state 12

   13 otherstmt: expr . ’;’

    ’;’  shift, and go to state 19


state 13

   11 ifhead: IF ’(’ . expr ’)’

    ’x’  shift, and go to state 5

    expr  go to state 20


state 14

   12 whilehead: WHILE ’(’ . expr ’)’

    ’x’  shift, and go to state 5

    expr  go to state 21


state 15

    9 openstmt: ifhead statement .

    $default  reduce using rule 9 (openstmt)


state 16

    3 statement: closedstmt .
    5 closedstmt: ifhead closedstmt . ELSE closedstmt
    8 openstmt: ifhead closedstmt . ELSE openstmt

    ELSE  shift, and go to state 22

    $default  reduce using rule 3 (statement)


state 17

    6 closedstmt: whilehead closedstmt .

    $default  reduce using rule 6 (closedstmt)


state 18

   10 openstmt: whilehead openstmt .

    $default  reduce using rule 10 (openstmt)


state 19

   13 otherstmt: expr ’;’ .

    $default  reduce using rule 13 (otherstmt)


```

```
state 20

   11 ifhead: IF '(' expr . ')'

    ')'  shift, and go to state 23


state 21

   12 whilehead: WHILE '(' expr . ')'

    ')'  shift, and go to state 24


state 22

    5 closedstmt: ifhead closedstmt ELSE . closedstmt
    8 openstmt: ifhead closedstmt ELSE . openstmt

    IF     shift, and go to state 3
    WHILE  shift, and go to state 4
    'x'    shift, and go to state 5

    closedstmt  go to state 25
    openstmt    go to state 26
    ifhead      go to state 9
    whilehead   go to state 10
    otherstmt   go to state 11
    expr        go to state 12


state 23

   11 ifhead: IF '(' expr ')' .

    $default  reduce using rule 11 (ifhead)


state 24

   12 whilehead: WHILE '(' expr ')' .

    $default  reduce using rule 12 (whilehead)


state 25

    5 closedstmt: ifhead closedstmt ELSE closedstmt .

    $default  reduce using rule 5 (closedstmt)


state 26

    8 openstmt: ifhead closedstmt ELSE openstmt .

    $default  reduce using rule 8 (openstmt)
```

```
// $Id: reduce-reduce.y,v 1.3 2011-10-28 18:41:59-07 - - $

// Example of a reduce/reduce conflict.

%verbose

%token CHAR INT IDENT
%start program

%%

program     : program statement
            |
            ;

statement   : declaration ’;’
            | expression ’;’
            ;

declaration : basetype ’[’ ’]’ IDENT
            | basetype IDENT
            ;

basetype    : CHAR
            | INT
            | IDENT
            ;

expression  : expression ’[’ expression ’]’
            | IDENT
            ;

%%

```

```
bison -v reduce-reduce.y
::::::::::::::
reduce-reduce.stdout
::::::::::::::
::::::::::::::
reduce-reduce.stderr
::::::::::::::
reduce-reduce.y: conflicts: 1 reduce/reduce
::::::::::::::
reduce-reduce.output
::::::::::::::
State 5 conflicts: 1 reduce/reduce


Grammar

    0 $accept: program $end

    1 program: program statement
    2        | /* empty */

    3 statement: declaration ’;’
    4          | expression ’;’

    5 declaration: basetype ’[’ ’]’ IDENT
    6            | basetype IDENT

    7 basetype: CHAR
    8         | INT
    9         | IDENT

   10 expression: expression ’[’ expression ’]’
   11           | IDENT


Terminals, with rules where they appear

$end (0) 0
’;’ (59) 3 4
’[’ (91) 5 10
’]’ (93) 5 10
error (256)
CHAR (258) 7
INT (259) 8
IDENT (260) 5 6 9 11


Nonterminals, with rules where they appear

$accept (9)
    on left: 0
program (10)
    on left: 1 2, on right: 0 1
statement (11)
    on left: 3 4, on right: 1
declaration (12)
    on left: 5 6, on right: 3
basetype (13)
    on left: 7 8 9, on right: 5 6
expression (14)
    on left: 10 11, on right: 4 10


state 0
```

```

    0 $accept: . program $end

    $default  reduce using rule 2 (program)

    program  go to state 1


state 1

    0 $accept: program . $end
    1 program: program . statement

    $end   shift, and go to state 2
    CHAR   shift, and go to state 3
    INT    shift, and go to state 4
    IDENT  shift, and go to state 5

    statement    go to state 6
    declaration  go to state 7
    basetype     go to state 8
    expression   go to state 9


state 2

    0 $accept: program $end .

    $default  accept


state 3

    7 basetype: CHAR .

    $default  reduce using rule 7 (basetype)


state 4

    8 basetype: INT .

    $default  reduce using rule 8 (basetype)


state 5

    9 basetype: IDENT .
   11 expression: IDENT .

    ’;’       reduce using rule 11 (expression)
    ’[’       reduce using rule 9 (basetype)
    ’[’       [reduce using rule 11 (expression)]
    $default  reduce using rule 9 (basetype)


state 6

    1 program: program statement .

    $default  reduce using rule 1 (program)


state 7
```

```

    3 statement: declaration . ’;’

    ’;’  shift, and go to state 10


state 8

    5 declaration: basetype . ’[’ ’]’ IDENT
    6            | basetype . IDENT

    IDENT  shift, and go to state 11
    ’[’    shift, and go to state 12


state 9

    4 statement: expression . ’;’
   10 expression: expression . ’[’ expression ’]’

    ’;’  shift, and go to state 13
    ’[’  shift, and go to state 14


state 10

    3 statement: declaration ’;’ .

    $default  reduce using rule 3 (statement)


state 11

    6 declaration: basetype IDENT .

    $default  reduce using rule 6 (declaration)


state 12

    5 declaration: basetype ’[’ . ’]’ IDENT

    ’]’  shift, and go to state 15


state 13

    4 statement: expression ’;’ .

    $default  reduce using rule 4 (statement)


state 14

   10 expression: expression ’[’ . expression ’]’

    IDENT  shift, and go to state 16

    expression  go to state 17


state 15

    5 declaration: basetype ’[’ ’]’ . IDENT

    IDENT  shift, and go to state 18


state 16

   11 expression: IDENT .

    $default  reduce using rule 11 (expression)


state 17

   10 expression: expression . ’[’ expression ’]’
   10           | expression ’[’ expression . ’]’

    ’[’  shift, and go to state 14
    ’]’  shift, and go to state 19


state 18

    5 declaration: basetype ’[’ ’]’ IDENT .

    $default  reduce using rule 5 (declaration)


state 19

   10 expression: expression ’[’ expression ’]’ .

    $default  reduce using rule 10 (expression)
```

```

%start S
%token a b c d

%%

S : A a ;
S : b A c ;
S : B c ;
S : b B a ;
A : d ;
B : d ;

%%

```

```
bison -v notlalr1.y
::::::::::::::
notlalr1.stdout
::::::::::::::
::::::::::::::
notlalr1.stderr
::::::::::::::
notlalr1.y: conflicts: 2 reduce/reduce
notlalr1.y:12.5: warning: rule never reduced because of conflicts: B: d
::::::::::::::
notlalr1.output
::::::::::::::
Rules never reduced

    6 B: d


State 2 conflicts: 2 reduce/reduce


Grammar

    0 $accept: S $end

    1 S: A a
    2  | b A c
    3  | B c
    4  | b B a

    5 A: d

    6 B: d


Terminals, with rules where they appear

$end (0) 0
error (256)
a (258) 1 4
b (259) 2 4
c (260) 2 3
d (261) 5 6


Nonterminals, with rules where they appear

$accept (7)
    on left: 0
S (8)
    on left: 1 2 3 4, on right: 0
A (9)
    on left: 5, on right: 1 2
B (10)
    on left: 6, on right: 3 4


state 0

    0 $accept: . S $end

    b  shift, and go to state 1
    d  shift, and go to state 2

    S  go to state 3
    A  go to state 4
    B  go to state 5


state 1

    2 S: b . A c
    4  | b . B a

    d  shift, and go to state 2

    A  go to state 6
    B  go to state 7


state 2

    5 A: d .
    6 B: d .

    a         reduce using rule 5 (A)
    a         [reduce using rule 6 (B)]
    c         reduce using rule 5 (A)
    c         [reduce using rule 6 (B)]
    $default  reduce using rule 5 (A)


state 3

    0 $accept: S . $end

    $end  shift, and go to state 8


state 4

    1 S: A . a

    a  shift, and go to state 9


state 5

    3 S: B . c

    c  shift, and go to state 10


state 6

    2 S: b A . c

    c  shift, and go to state 11


state 7

    4 S: b B . a

    a  shift, and go to state 12


state 8

```

```
    0 $accept: S $end .

    $default  accept


state 9

    1 S: A a .

    $default  reduce using rule 1 (S)


state 10

    3 S: B c .

    $default  reduce using rule 3 (S)


state 11

    2 S: b A c .

    $default  reduce using rule 2 (S)


state 12

    4 S: b B a .

    $default  reduce using rule 4 (S)
```