Tutorial 03: CS31003: Compilers:

[M-04] Bison

Indranil Sengupta
Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

isg@iitkgp.ac.in
ppd@cse.iitkgp.ac.in

October 03, 2020

Doubts from the Week

Problem: Simple Calculator

Given the following grammar

$$
\begin{array}{rlcl}
1: & S & \rightarrow & E \\
2: & E & \rightarrow & E + T \\
3: & E & \rightarrow & E - T \\
4: & E & \rightarrow & T \\
5: & T & \rightarrow & T * F \\
6: & T & \rightarrow & T \; / \; F \\
7: & T & \rightarrow & F \\
8: & F & \rightarrow & (E) \\
9: & F & \rightarrow & - F \\
10: & F & \rightarrow & \textbf{num}
\end{array}
$$

and the Bison specs for translation to the Simple Calculator codes (as discussed in the Module 4 Lecture), show the evaluation of the following inputs with sequence of reductions and the stack traces:

1. $7 + 16 * 5$ \$
2. $(9 + 5) * -6$ \$
3. $7 - -3$ \$

Solution: $7 + 16 * 5$

Input:

7 + 16 * 5 $

Reductions

$$
\begin{array}{ll}
 & \underline{\mathbf{num}_7} + \mathbf{num}_{16} * \mathbf{num}_5 \ \$ \\
\Rightarrow & \underline{F} + \mathbf{num}_{16} * \mathbf{num}_5 \ \$ \\
\Rightarrow & \underline{T} + \mathbf{num}_{16} * \mathbf{num}_5 \ \$ \\
\Rightarrow & E + \underline{\mathbf{num}_{16}} * \mathbf{num}_5 \ \$ \\
\Rightarrow & E + \underline{F} * \mathbf{num}_5 \ \$ \\
\Rightarrow & E + T * \underline{\mathbf{num}_5} \ \$ \\
\Rightarrow & E + \underline{T * F} \ \$ \\
\Rightarrow & \underline{E + T} \ \$ \\
\Rightarrow & \underline{E} \ \$ \\
\Rightarrow & S \ \$
\end{array}
$$

Symbols	Values	Input
		7 + 16 * 5 $
$\mathbf{num}_7$	7	+ 16 * 5 $
F	7	+ 16 * 5 $
T	7	+ 16 * 5 $
E	7	+ 16 * 5 $
E +	7 +	16 * 5 $
E + $\mathbf{num}_{16}$	7 + 16	* 5 $
E + F	7 + 16	* 5 $
E + T	7 + 16	* 5 $
E + T *	7 + 16 *	5 $
E + T * $\mathbf{num}_5$	7 + 16 * 5	$
E + T * F	7 + 16 * 5	$
E + T	7 + 80	$
E	87	$
S	87	$

Solution: $(9 + 5) * -6$

Input:

(9 + 5) * -6 $

Reductions

$(\underline{\text{num}_9} + \text{num}_5) * - \text{num}_6 \ \$$
$\Rightarrow (\underline{F} + \text{num}_5) * - \text{num}_6 \ \$$
$\Rightarrow (\underline{T} + \text{num}_5) * - \text{num}_6 \ \$$
$\Rightarrow (E + \underline{\text{num}_5}) * - \text{num}_6 \ \$$
$\Rightarrow (E + \underline{F}) * - \text{num}_6 \ \$$
$\Rightarrow (\underline{E + T}) * - \text{num}_6 \ \$$
$\Rightarrow \underline{(E)} * - \text{num}_6 \ \$$
$\Rightarrow \underline{F} * - \text{num}_6 \ \$$
$\Rightarrow T * - \underline{\text{num}_6} \ \$$
$\Rightarrow T * \underline{- F} \ \$$
$\Rightarrow \underline{T * F} \ \$$
$\Rightarrow \underline{T} \ \$$
$\Rightarrow \underline{E} \ \$$
$\Rightarrow \underline{S} \ \$$

Symbols	Values	Input
		(9 + 5) * - 6 $
(		9 + 5) * - 6 $
(num_9	9	+ 5) * - 6 $
(F	9	+ 5) * - 6 $
(T	9	+ 5) * - 6 $
(E	9	+ 5) * - 6 $
(E +	9 +	5) * - 6 $
(E + num_5	9 + 5	) * - 6 $
(E + F	9 + 5	) * - 6 $
(E + T	9 + 5	) * - 6 $
(E	14	) * - 6 $
(E)	14	* - 6 $
F	14	* - 6 $
T	14	* - 6 $
T *	14 *	- 6 $
T * -	14 * -	6 $
T * - num_6	14 * - 6	$
T * - F	14 * - 6	$
T * F	14 * -6	$
T	-84	$
E	-84	$
S	-84	$

Solution: 7 − −3

Input:

7 - -3 $

Reductions

```
    num7 - - num3 $
=>  F - - num3 $
=>  T - - num3 $
=>  E - - num3 $
=>  E - - F $
=>  E - F $
=>  E - T $
=>  E $
=>  S $
```

Symbols	Values	Input
		7 - - 3 $
$\mathbf{num}_7$	7	- - 3 $
F	7	- - 3 $
T	7	- - 3 $
E	7	- - 3 $
E -	7 -	- 3 $
E - -	7 - -	3 $
E - - $\mathbf{num}_3$	7 - - 3	$
E - - F	7 - - 3	$
E - F	7 - -3	$
E - T	7 - -3	$
E	10	$
S	10	$

Problem: Programmable Calculator

Given the following grammar:

$$
\begin{array}{rlll}
\text{1-2:} & L & \rightarrow & L\ S\ \backslash\text{n} \mid S\ \backslash\text{n} \\
\text{3-4:} & S & \rightarrow & \textbf{id} = E \mid E \\
\text{5-7:} & E & \rightarrow & E + T \mid E - T \mid T \\
\text{8-10:} & T & \rightarrow & T * F \mid T \ / \ F \mid F \\
\text{11-14:} & F & \rightarrow & (E) \mid - F \mid \textbf{num} \mid \textbf{id}
\end{array}
$$

and the Bison specs for translation to the Programmable Calculator codes (as discussed in the Module 4 Lecture), show the evaluation of the following inputs with sequence of reductions and the stack traces:

1.
```
a = 5 - 3
b = 12 * 2
a + b
```

2.
```
a = 4
b = 6
a * b + 2
```

3.
```
a = 3
a - b
```

Solution: Programmable Calculator (1)

Reductions

```
    id_a = num_5 - num_3 \n id_b = num_12 * num_2 \n id_a + id_b \n $
=>  id_a = num_5 - num_3 \n id_b = num_12 * num_2 \n id_a + id_b \n $
=>  id_a = F - num_3 \n id_b = num_12 * num_2 \n id_a + id_b \n $
=>  id_a = T - num_3 \n id_b = num_12 * num_2 \n id_a + id_b \n $
=>  id_a = E - num_3 \n id_b = num_12 * num_2 \n id_a + id_b \n $
=>  id_a = E - F \n id_b = num_12 * num_2 \n id_a + id_b \n $
=>  id_a = E - T \n id_b = num_12 * num_2 \n id_a + id_b \n $
=>  id_a = E \n id_b = num_12 * num_2 \n id_a + id_b \n $
=>  S \n id_b = num_12 * num_2 \n id_a + id_b \n $
=>  L id_b = num_12 * num_2 \n id_a + id_b \n $
=>  L id_b = num_12 * num_2 \n id_a + id_b \n $
=>  L id_b = F * num_2 \n id_a + id_b \n $
=>  L id_b = T * num_2 \n id_a + id_b \n $
=>  L id_b = T * F \n id_a + id_b \n $
=>  L id_b = T \n id_a + id_b \n $
=>  L id_b = E \n id_a + id_b \n $
=>  L S \n id_a + id_b \n $
=>  L id_a + id_b \n $
=>  L F + id_b \n $
=>  L T + id_b \n $
=>  L E + id_b \n $
=>  L E + F \n $
=>  L E + T \n $
=>  L E \n $
=>  L S \n $
=>  L $
```

Solution: Programmable Calculator (1)

Evaluation

Symbols	Values	SymTab
id$_a$	a	[a = ?]
id$_a$ = **num**$_5$	a = 5	[a = ?]
id$_a$ = E	a = 5	[a = ?]
id$_a$ = E - **num**$_3$	a = 5 - 3	[a = ?]
id$_a$ = E - T	a = 5 - 3	[a = ?]
id$_a$ = E	a = 2	[a = ?]
S		[a = 2]
S \n		[a = 2]
L		[a = 2]
L **id**$_b$	b	[a = 2][b = ?]
L **id**$_b$ = **num**$_{12}$	b = 12	[a = 2][b = ?]
L **id**$_b$ = T * **num**$_2$	b = 12 * 2	[a = 2][b = ?]
L **id**$_b$ = T * F	b = 12 * 2	[a = 2][b = ?]

Symbols	Values	SymTab
L **id**$_b$ = T	b = 24	[a = 2][b = ?]
L **id**$_b$ = E	b = 24	[a = 2][b = ?]
L S		[a = 2][b = 24]
L S \n		[a = 2][b = 24]
L		[a = 2][b = 24]
L **id**$_a$	a	[a = 2][b = 24]
L E	2	[a = 2][b = 24]
L E + **id**$_b$	2 + b	[a = 2][b = 24]
L E + T	2 + 24	[a = 2][b = 24]
L E	26	[a = 2][b = 24]
L S		[a = 2][b = 24]
L S \n		[a = 2][b = 24]
L		[a = 2][b = 24]

Solution: Programmable Calculator (2)

Reductions

$$
\begin{array}{ll}
 & \underline{\mathbf{id}_a} = \mathbf{num}_4 \ \backslash n \ \mathbf{id}_b = \mathbf{num}_6 \ \backslash n \ \mathbf{id}_a * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & \mathbf{id}_a = \underline{\mathbf{num}_4} \ \backslash n \ \mathbf{id}_b = \mathbf{num}_6 \ \backslash n \ \mathbf{id}_a * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & \mathbf{id}_a = \underline{F} \ \backslash n \ \mathbf{id}_b = \mathbf{num}_6 \ \backslash n \ \mathbf{id}_a * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & \mathbf{id}_a = \underline{T} \ \backslash n \ \mathbf{id}_b = \mathbf{num}_6 \ \backslash n \ \mathbf{id}_a * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & \underline{\mathbf{id}_a = E} \ \backslash n \ \mathbf{id}_b = \mathbf{num}_6 \ \backslash n \ \mathbf{id}_a * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & \underline{S \ \backslash n} \ \mathbf{id}_b = \mathbf{num}_6 \ \backslash n \ \mathbf{id}_a * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & L \ \underline{\mathbf{id}_b} = \mathbf{num}_6 \ \backslash n \ \mathbf{id}_a * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & L \ \mathbf{id}_b = \underline{\mathbf{num}_6} \ \backslash n \ \mathbf{id}_a * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & L \ \mathbf{id}_b = \underline{F} \ \backslash n \ \mathbf{id}_a * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & L \ \mathbf{id}_b = \underline{T} \ \backslash n \ \mathbf{id}_a * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & L \ \underline{\mathbf{id}_b = E} \ \backslash n \ \mathbf{id}_a * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & \underline{L \ S \ \backslash n} \ \mathbf{id}_a * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & L \ \underline{\mathbf{id}_a} * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & L \ \underline{F} * \mathbf{id}_b + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & L \ T * \underline{\mathbf{id}_b} + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & L \ \underline{T * F} + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & L \ \underline{T} + \mathbf{num}_2 \ \backslash n \ \$ \\
\Rightarrow & L \ E + \underline{\mathbf{num}_2} \ \backslash n \ \$ \\
\Rightarrow & L \ E + \underline{F} \ \backslash n \ \$ \\
\Rightarrow & L \ \underline{E + T} \ \backslash n \ \$ \\
\Rightarrow & L \ \underline{E} \ \backslash n \ \$ \\
\Rightarrow & \underline{L \ S \ \backslash n} \ \$ \\
\Rightarrow & L \ \$
\end{array}
$$

Solution: Programmable Calculator (2)

Evaluation

Symbols	Values	SymTab
id$_a$	a	[a = ?]
id$_a$ = **num**$_4$	a = 4	[a = ?]
id$_a$ = E	a = 4	[a = ?]
S		[a = 4]
S \n		[a = 4]
L		[a = 4]
L **id**$_b$	b	[a = 4][b = ?]
L **id**$_b$ = **num**$_6$	b = 6	[a = 4][b = ?]
L **id**$_b$ = E	b = 6	[a = 4][b = ?]
L S		[a = 4][b = 6]
L S \n		[a = 4][b = 6]
L		[a = 4][b = 6]

Symbols	Values	SymTab
L **id**$_a$	a	[a = 4][b = 6]
L T	4	[a = 4][b = 6]
L T * **id**$_b$	4 * b	[a = 4][b = 6]
L T * F	4 * 6	[a = 4][b = 6]
L T	24	[a = 4][b = 6]
L E	24	[a = 4][b = 6]
L E + **num**$_2$	24 + 2	[a = 4][b = 6]
L E + T	24 + 2	[a = 4][b = 6]
L E	26	[a = 4][b = 6]
L S		[a = 4][b = 6]
L S \n		[a = 4][b = 6]
L		[a = 4][b = 6]

Solution: Programmable Calculator (3)

Reductions

$\underline{\mathbf{id}_a}\ =\ \mathbf{num}_3\ \backslash n\ \mathbf{id}_a - \mathbf{id}_b\ \backslash n\ \$$

$\Rightarrow\ \mathbf{id}_a\ =\ \underline{\mathbf{num}_3}\ \backslash n\ \mathbf{id}_a - \mathbf{id}_b\ \backslash n\ \$$

$\Rightarrow\ \mathbf{id}_a\ =\ \underline{F}\ \backslash n\ \mathbf{id}_a - \mathbf{id}_b\ \backslash n\ \$$

$\Rightarrow\ \mathbf{id}_a\ =\ \underline{T}\ \backslash n\ \mathbf{id}_a - \mathbf{id}_b\ \backslash n\ \$$

$\Rightarrow\ \underline{\mathbf{id}_a\ =\ E}\ \backslash n\ \mathbf{id}_a - \mathbf{id}_b\ \backslash n\ \$$

$\Rightarrow\ \underline{S\ \backslash n}\ \mathbf{id}_a - \mathbf{id}_b\ \backslash n\ \$$

$\Rightarrow\ L\ \underline{\mathbf{id}_a}\ -\ \mathbf{id}_b\ \backslash n\ \$$

$\Rightarrow\ L\ \underline{F}\ -\ \mathbf{id}_b\ \backslash n\ \$$

$\Rightarrow\ L\ \underline{T}\ -\ \mathbf{id}_b\ \backslash n\ \$$

$\Rightarrow\ L\ E\ -\ \underline{\mathbf{id}_b}\ \backslash n\ \$$

$\Rightarrow\ L\ E\ -\ \underline{F}\ \backslash n\ \$$

$\Rightarrow\ L\ \underline{E\ -\ T}\ \backslash n\ \$$

$\Rightarrow\ L\ \underline{E}\ \backslash n\ \$$

$\Rightarrow\ \underline{L\ S\ \backslash n}\ \$$

$\Rightarrow\ L\ \$$

Evaluation

Symbols	Values	SymTab
$\mathbf{id}_a$	a	[a = ?]
$\mathbf{id}_a$ = $\mathbf{num}_3$	a = 3	[a = ?]
$\mathbf{id}_a$ = E	a = 3	[a = ?]
S		[a = 3]
S \n		[a = 3]
L		[a = 3]
L $\mathbf{id}_a$	a	[a = 3]
L E	3	[a = 3]
L E - $\mathbf{id}_b$	3 - b	[a = 3][b = ?]
L E - T	3 - ?	[a = 3][b = ?]
L E	err	[a = 3][b = ?]

Tutorial 03

I Sengupta & P P Das

Weekly Feedback

Simple Calculator

Programmable Calculator

Ambiguous Grammar

Expression

Dangling Else

Practice Problems

Problem: Ambiguous Grammar: Expression

Consider the following grammar:

$$
\begin{array}{llll}
1: & E & \rightarrow & E = E \\
2: & E & \rightarrow & E + E \\
3: & E & \rightarrow & E * E \\
4: & E & \rightarrow & *E \\
5: & E & \rightarrow & +E \\
6: & E & \rightarrow & id
\end{array}
$$

where the precedence is given as:

$$*(unary), +(unary) \gtrdot *(binary) \gtrdot +(binary) \gtrdot =$$

and $*(binary)$ & $+(binary)$ are left-associative and rest are right-associative.

1. Try to construct an LR(0) parser for the above grammar and show the states with shift-reduce conflicts
2. Justify why no LR parser would be possible for this grammar
3. Using the precedence and associativity resolve the conflicts to get an LR parser
4. Write the Bison specs (no action for rules is needed) for the above grammar (without shift-reduce conflict)

Solution: Ambiguous Grammar: Expression


```
I0:
   E' -> . E
   E  -> . E = E
   E  -> . E + E
   E  -> . E * E
   E  -> . * E
   E  -> . + E
   E  -> . id
```

```
I1:
   E' -> E .
   E  -> E . = E
   E  -> E . + E
   E  -> E . * E
```

```
I2:
   E -> + . E
   E -> . E = E
   E -> . E + E
   E -> . E * E
   E -> . * E
   E -> . + E
   E -> . id
```

```
I3:
   E -> * . E
   E -> . E = E
   E -> . E + E
   E -> . E * E
   E -> . * E
   E -> . + E
   E -> . id
```

```
I4:
   E -> id .
```

```
I5:
   E -> E = . E
   E -> . E = E
   E -> . E + E
   E -> . E * E
   E -> . * E
   E -> . + E
   E -> . id
```

```
I6:
   E -> E + . E
   E -> . E = E
   E -> . E + E
   E -> . E * E
   E -> . * E
   E -> . + E
   E -> . id
```

```
I7:
   E -> E * . E
   E -> . E = E
   E -> . E + E
   E -> . E * E
   E -> . * E
   E -> . + E
   E -> . id
```

```
I8:
   E -> * E .
   E -> E . = E
   E -> E . + E
   E -> E . * E
```

```
I9:
   E -> + E .
   E -> E . = E
   E -> E . + E
   E -> E . * E
```

```
I10:
   E -> E = E .
   E -> E . = E
   E -> E . + E
   E -> E . * E
```

```
I11:
   E -> E + E .
   E -> E . = E
   E -> E . + E
   E -> E . * E
```

```
I12:
   E -> E * E .
   E -> E . = E
   E -> E . + E
   E -> E . * E
```

Ambiguous Grammar

```
0: E' → E
1: E → E = E
2: E → E + E
3: E → E * E
4: E → *E
5: E → +E
6: E → id
```

	=	+	*
8	s5/r4	s6/r4	s7/r4
9	s5/r5	s6/r5	s7/r5
10	s5/r1	s6/r1	s7/r1
11	s5/r2	s6/r2	s7/r2
12	s5/r3	s6/r3	s7/r3

- As { +,=,* } ⊂ FOLLOW (E), so no LR parser would be possible for this grammar.

Solution: Ambiguous Grammar: Expression

Ambiguous Grammar

0: E' → E
1: E → E = E
2: E → E + E
3: E → E * E
4: E → *E
5: E → +E
6: E → id

STATE	ACTION					GOTO
	id	=	+	*	$	E
0	s4		s2	s3		1
1		s5	s6	s7	acc	
2	s4		s2	s3		9
3	s4		s2	s3		8
4		r6	r6	r6	r6	
5	s4		s2	s3		10
6	s4		s2	s3		11
7	s4		s2	s3		12
8		r4	r4	r4	r4	
9		r5	r5	r5	r5	
10		s5	s6	s7	r1	
11		r2	r2	s7	r2	
12		r3	r3	r3	r3	

Solution: Ambiguous Grammar: Expression

```
%{
#include <string.h>
#include <iostream>
#include "parser.h"
extern int yylex();
void yyerror(char *s);
#define NSYMS 20 /* max # of symbols */
symboltable symtab[NSYMS];
%}

%union {
    struct symtab *symp;
}

%token <symp> NAME

%left '+'
%left '*'
%right UPLUS, UMULT

%%
expression: expression '=' expression
          | expression '+' expression
          | expression '*' expression
          | '+' expression %prec UPLUS
          | '*' expression %prec UMULT
          | NAME
          ;

%%
```

```
struct symtab *symlook(char *s) {
    char *p;
    struct symtab *sp;
    for(sp = symtab;
        sp < &symtab[NSYMS]; sp++) {
        /* is it already here? */
        if (sp->name &&
            !strcmp(sp->name, s))
            return sp;
        if (!sp->name) {
        /* is it free */
            sp->name = strdup(s);
            return sp;
        }
        /* otherwise continue to next*/
    }
    yyerror("Too many symbols");
    exit(1); /* cannot continue */
} /* symlook */


void yyerror(char *s) {
    std::cout << s << std::endl;
}


int main() {
    yyparse();
}
```

Problem: Dangling Else Ambiguity

In the C language grammar, we have:

```
statement = ...
    | selection-statement

selection-statement = ...
    | IF ( expression ) statement
    | IF ( expression ) statement ELSE statement
```

1. Using the above, show how

   `if (a) if (b) s; else s2;`

   could be ambiguously parsed
2. Illustrate the manifestation of this ambiguity in the LR(1) construction of a parser by Bison
3. In practice in C, the first tree is chosen by associating the `else` with the nearest `if`. How would you get this in an LR parser (in Bison) to resolve the ambiguity?

Solution: Dangling Else Ambiguity

```
Left-most Derivation of =>    if (a) if (b) s; else s2;

=> statement
=> selection-statement
=> IF ( expression ) statement
=> IF ( a ) statement
=> IF ( a ) selection-statement
=> IF ( a ) IF ( expression ) statement ELSE statement
=> IF ( a ) IF ( b ) statement ELSE statement
=> IF ( a ) IF ( b ) s; ELSE statement
=> IF ( a ) IF ( b ) s; ELSE s2;
```

```
Right-most Derivation of =>    if (a) if (b) s; else s2;

=> statement
=> selection-statement
=> IF ( expression ) statement ELSE statement
=> IF ( expression ) statement ELSE s2;
=> IF ( expression ) selection-statement ELSE s2;
=> IF ( expression ) IF ( expression ) statement ELSE s2;
=> IF ( expression ) IF ( expression ) s; ELSE s2;
=> IF ( expression ) IF ( b ) s; ELSE s2;
=> IF ( a ) IF ( b ) s; ELSE s2;
```

Solution: Dangling Else Ambiguity


```
Let:    S = statement    S' = selection-statement     E = expression

statement = ... | selection-statement

selection-statement = ... | IF ( expression ) statement
| IF ( expression ) statement ELSE statement

S  = ... | S'
S' = ... | IF ( E ) S | IF ( E ) S ELSE S
```

```
I1:
S' -> IF ( E ) S . ELSE S
```

```
I3:
S' -> IF ( E ) S .
```

```
I2:
S' -> IF ( E ) S ELSE . S
S  -> ...
...
```

The ambiguity produces the shift/reduce conflict for I1.
{ S' -> IF (E) S . ELSE S } expects shift of ELSE but as FOLLOW(S) contains ELSE, { S' -> IF (E) S . } expects the reduction by { S' -> IF (E) S } on input ELSE.

Resolving ambiguity by associating ELSE with nearest IF
The ambiguity can be solved by breaking the shift/reduce conflict by shifting on ELSE. So that, ELSE is shifted onto the stack and is later reduced as { S' -> IF (E) S ELSE S } ensuring association of ELSE with nearest IF.

Practice Problems

Tutorial 03

I Sengupta & P P Das

1. For Simple Calculator
   1. $8 - 5 * -6$ \$
   2. $(6 + 2)/(1 + 1)$ \$
2. For Programmable Calculator
   1.
   ```
   a = 2 * 4
   a = a + 3 * a
   a + 3
   ```
   2.
   ```
   a = 2 + 3 + 4
   b = 4 - a
   b * 2
   ```
3. For the grammar

$$E \to E\&E \mid E + E \mid E * E \mid \; * E \mid \; + E \mid \&E \mid id$$

where the precedence is given as:

$$*(unary), +(unary), \&(unary) \gtrdot *(binary) \gtrdot +(binary) \gtrdot \&(binary)$$

and binary (unary) operators are left (right)-associative

   1. Show the states with shift-reduce conflicts in LR construction
   2. Justify why no LR parser would be possible for this grammar
   3. Using the precedence and associativity to build an LR parser
   4. Write the Bison specs for the above grammar