Yacc Ambiguities and Conflicts

Lex & Yacc

Jan Pačes Petr Dvořák

Faculty of Information Technology
LTA 2012

December 13, 2012

Yacc pointer model

Single pointer example

```
start: A B C;
```

Yacc pointer model

Single pointer example

```
start: ↑A B C;
```

Yacc pointer model

Single pointer example

```
start: A↑B C;
```

Yacc pointer model

Single pointer example

```
start: A B↑C;
```

Yacc pointer model

Single pointer example

```
start: A B C↑;
```

Yacc pointer model

Single pointer example

```
start: A B C↑;
```

One pointer ⇒ No problem!

Yacc pointer model

Single pointer example

```
start: A B C↑;
```

One pointer ⇒ No problem!

Multiple pointers example

```
start   :   x
        |   y;

x:  ↑A B z R;
x:  ↑A B z S;
z:  C D;
```

Yacc pointer model

Single pointer example

```
start: A B C↑;
```

One pointer ⇒ No problem!

Multiple pointers example

```
start   :   x
        |   y;

x:  A↑B z R;
x:  A↑B z S;
z:  C D;
```

Yacc pointer model

Single pointer example

```
start: A B C↑;
```

One pointer ⇒ No problem!

Multiple pointers example

```
start   :   x
        |   y;

x:  A B↑z R;
x:  A B↑z S;
z:  C D;
```

Yacc pointer model

Single pointer example

```
start: A B C↑;
```

One pointer ⇒ No problem!

Multiple pointers example

```
start   :   x
        |   y;

x:  A B z R;
x:  A B z S;
z:  ↑C D;
```

Yacc pointer model

Single pointer example

```
start: A B C↑;
```

One pointer ⇒ No problem!

Multiple pointers example

```
start   :   x
        |   y;

x:  A B z R;
x:  A B z S;
z:  C↑D;
```

Yacc pointer model

Single pointer example

```
start: A B C↑;
```

One pointer ⇒ No problem!

Multiple pointers example

```
start   :   x
        |   y;

x:  A B z R;
x:  A B z S;
z:  C D↑;
```

Yacc pointer model

Single pointer example

```
start: A B C↑;
```

One pointer ⇒ No problem!

Multiple pointers example

```
start   :   x
        |   y;

x:  A B z R;
x:  A B z S;
z:  C D↑;
```

Multiple pointers + ambigious grammars = conflicts!

Types of conflicts

2 types of conflict

Types of conflicts

2 types of conflict

Reduce/Reduce conflicts

```
start   :   x
        |   y;
x:  A↑;
y:  A↑;
```

Types of conflicts

2 types of conflict

Reduce/Reduce conflicts

```
start   :   x
        |   y;
x:  A↑;
y:  A↑;
```

Shift/Reduce conflicts

```
start   :   x
        |   y R;
x:  A↑R;
y:  A↑;
```

Yacc conflict reporting - Reduce/reduce errors

Errors reported in separete file *output.y* describing parser states.

Grammar with Reduce/Reduce conflict

```
start   :   A B x Z
        |   y Z;
x:  C;
y:  A B C;
```

output.y contents

```
Rules useless in parser due to conflicts
    4 y: A B C
State 7 conflicts: 1 reduce/reduce
...
state 7
    3 x: C .
    4 y: A B C .
    Z         reduce using rule 3 (x)
    Z         [reduce using rule 4 (y)]
    $default  reduce using rule 3 (x)
```

Yacc conflict reporting - Shift/reduce errors

Grammar with Shift/Reduce conflict

```
start   :   x
        |   y R;
x:  A R;
y:  A;
```

output.y contents

```
Rules useless in parser due to conflicts
    4 y: A
State 1 conflicts: 1 shift/reduce
...
state 1
    3 x: A . R
    4 y: A .
    R  shift, and go to state 5
    R  [reduce using rule 4 (y)]
```

Common type of conflicts

- Expression Grammars

Common type of conflicts

- Expression Grammars
- IF-THEN-ELSE

Common type of conflicts

- Expression Grammars
- IF-THEN-ELSE
- Nested list grammar

How to fix these conflicts?

IF-THEN-ELSE (Shift/Reduce)

Problem

Shift/reduce conflict in if-then-else conditions.

```
stmt: IF '(' cond ')' stmt
    | IF '(' cond ')' stmt ELSE stmt
```

```
IF (cond)
  IF (cond) nop(); <-     stmt: IF (cond) stmt <return here>
  ELSE nop();             stmt: IF (cond) stmt <or here?> ELSE stmt
```

IF-THEN-ELSE (Shift/Reduce)

Problem

Shift/reduce conflict in if-then-else conditions.

```
stmt: IF '(' cond ')' stmt
    | IF '(' cond ')' stmt ELSE stmt
```

```
IF (cond)
  IF (cond) nop(); <-     stmt: IF (cond) stmt <return here>
  ELSE nop();             stmt: IF (cond) stmt <or here?> ELSE stmt
```

Solution

Set up precedence of ELSE to choose priorities:

```
IF (cond) { IF (cond) stmt } ELSE stmt
IF (cond) { IF (cond) stmt ELSE stmt }
```

Loop within a loop (Shift/Reduce)

Problem

Do we prefere one outer loop and many inner loops or many outer loops and one inner loop?

```
start: outer Z;
outer: /* empty */
    | outer outerItem ;
outerItem: inner ;
inner: /* empty */
    | inner innerItems ;
innerItem: I
```

Loop within a loop (Shift/Reduce)

Problem

Do we prefere one outer loop and many inner loops or many outer loops and one inner loop?

```
start: outer Z;
outer: /* empty */
     | outer outerItem ;
outerItem: inner ;
inner: /* empty */
     | inner innerItems ;
innerItem: I
```

Solution

Choose one approach only:

```
start: outer Z
outer: /* empty */
     | outer innerItem
innerItem: I;
```

```
start: inner Z
inner: /* empty */
     | inner innerItem ;
innerItem: I;
```

Expression precedence (Shift/Reduce)

Problem

Even one rule can cause conflicts

```
expr - expr - expr
expr: TERMINAL
    | expr '-' expr

Conflict (shift/reduce):
    expr: expr↑- expr
    expr: expr -↑expr
```

Expression precedence (Shift/Reduce)

Problem

Even one rule can cause conflicts

```
expr - expr - expr
expr: TERMINAL
    | expr '-' expr

Conflict (shift/reduce):
    expr: expr↑- expr
    expr: expr -↑expr
```

Solution

Specify whether use left or right associativity using `%left` or `%right` keywords.

Limited lookahead (Shift/Reduce or Reduce/Reduce)

Problem

yacc checks only one following token

```
rule: command optional_keyword '(' identifiers ')';
optional_keyword: /* empty */
  | '(' keyword ')';
```

Limited lookahead (Shift/Reduce or Reduce/Reduce)

Problem

yacc checks only one following token

```
rule: command optional_keyword '(' identifiers ')';
optional_keyword: /* empty */
  | '(' keyword ')';
```

Solution

Precedence does not help. The rule must be flattened:

```
rule: command '(' keyword ')' '(' identifier ')'
    | command '(' identifier ')';
```

Bibliography


D. Brown, J. Levine, and T. Mason.
lex & yacc.
O'Reilly & Associates, Inc., second edition, 1992.