The Scanner and The Parser

Akim Demaille Étienne Renault Roland Levillain
first.*last*@lrde.epita.fr

EPITA — École Pour l'Informatique et les Techniques Avancées

January 25, 2018

The Scanner and The Parser

Symbols

cstats

cstats: Counting Symbols

```
g++ -E -P "$@" \
  | tr -cs '[:alnum:]_' '[\n*]' \
  | grep '^[[:alpha:]]' \
  | grep -v -E -w "$cxx_keywords" > $tmp.1
total=$(wc -lc < $tmp.1 \
        | awk '{print $1 " (" $2 " chars)"}')
sort $tmp.1 \
  | uniq -c \
  | sed 's/^   //;s/\t/ /' \
  | sort -rn >$tmp.2
unique=$(sed -s 's/.* //' $tmp.2 | wc -lc \
         | awk '{print $1 " (" $2 " chars)"}')
echo $total occurrences of $unique symbols.
sed 42q $tmp.2 \
  | pr --page-width=60 --column=3 --omit-header
rm -f $tmp.*
```

Lemon

```
7992 (44454 chars) occurrences of 842 (6950 chars) symbols.
 454 i                  94 next                 59 rule
 262 lemp               93 name                 59 lem
 230 psp                91 h                    59 filename
 193 rp                 91 FILE                 59 config
 142 cfp                82 np                   57 symbol
 137 n                  77 c                    54 type
 118 x                  73 state                50 data
 114 fprintf            73 j                    48 tbl
 113 s                  72 size                 48 errorcnt
 113 out                70 stp                  44 d
 112 cp                 70 array                43 x2a
 112 ap                 69 ht                   42 x4a
 111 sp                 64 size_t               41 lemon
 104 lineno             61 a                    41 argv
```

GCC's C Parser

```
18958 (198353 chars) occurrences of 5835 (89396 chars) symbols.
2676 tree                89 new_type_flag    38 build_nt
1579 ttype               70 cpp_reader       36 itype
1123 yyvsp               69 build_tree_lis   36 build_x_binary
 909 yyval               67 parse            35 yychar
 358 ftype               65 y                35 frob_opname
 247 t                   61 obstack          35 d
 206 gt_pointer_ope      58 GTY              34 e
 200 common              46 identifier       33 tree_code_type
 192 size_t              43 error            33 operator_name_
 175 code                40 cp_global_tree   33 C
 171 tree_code           39 yyn              32 got_scope
 123 FILE                39 s                31 IDENTIFIER_NOD
  97 rtx                 39 lookups          30 tree_class_che
  95 type                38 TREE_LIST        30 global_trees
```

Tiger Compiler's Driver

```
43224 (376096 chars) occurrences of 3191 (35027 chars) symbols.
1520 _CharT           374 __err             243 __y
1063 __first          360 _M_impl           239 __result
 846 _Tp              328 _ForwardIterat    237 _RandomAccessI
 721 ios_base         322 size_t            226 __first1
 706 _Traits          305 __len             212 __a
 698 __x              301 basic_string      211 __io
 675 __last           284 __beg             208 __first2
 657 std              282 iterator          206 _M_node
 636 _Alloc           275 __pos             202 __p
 584 __n              271 __i               194 __end
 455 char_type        258 _InputIterator    184 iostate
 438 size_type        254 _Compare          182 traits_type
 414 __s              248 locale            182 __comp
 404 __c              244 iter_type         176 __middle
```

Symbols

Save Time and Space

One unique occurrence for each identifier:

In C a simple `const char*`

In C++ an iterator in a `std::set`

"Set has the important property that inserting a new element into a set does not invalidate iterators that point to existing elements."

Save space fewer allocations

Save time fewer allocations, easier comparisons

Save nerves easier memory management

Save Time and Space

One unique occurrence for each identifier:

In C a simple `const char*`

In C++ an iterator in a `std::set`

"Set has the important property that inserting a new element into a set does not invalidate iterators that point to existing elements."

Save space fewer allocations

Save time fewer allocations, easier comparisons

Save nerves easier memory management

Save Time and Space

One unique occurrence for each identifier:

In C a simple `const char*`

In C++ an iterator in a `std::set`

"Set has the important property that inserting a new element into a set does not invalidate iterators that point to existing elements."

Save space fewer allocations

Save time fewer allocations, easier comparisons

Save nerves easier memory management

Save Time and Space

One unique occurrence for each identifier:

In C a simple `const char*`

In C++ an iterator in a `std::set`

"Set has the important property that inserting a new element into a set does not invalidate iterators that point to existing elements."

Save space fewer allocations

Save time fewer allocations, easier comparisons

Save nerves easier memory management

Save Time and Space

One unique occurrence for each identifier:

In C a simple `const char*`

In C++ an iterator in a `std::set`

"Set has the important property that inserting a new element into a set does not invalidate iterators that point to existing elements."

Save space fewer allocations

Save time fewer allocations, easier comparisons

Save nerves easier memory management

Save Time and Space

One unique occurrence for each identifier:

In C a simple `const char*`

In C++ an iterator in a `std::set`

"Set has the important property that inserting a new element into a set does not invalidate iterators that point to existing elements."

Save space	fewer allocations
Save time	fewer allocations, easier comparisons
Save nerves	easier memory management

Semantic Values

Parser

Reading tokens in the parser

```
// Allow storing object values.
%define api.value.type variant
// Generate functions to build tokens.
%define api.token.constructor
// Prefix all the tokens with TOK_ to avoid colisions.
%define api.token.prefix {TOK_}

%token <misc::symbol>   ID     "identifier"
%token <int>            INT    "integer"
%token <std::string>    STRING "string"

%printer { yyo << $$; } "identifier" "integer" "string"
%%
// ...
exp:
  INT     { $$ = new IntExp($1); }
| STRING  { $$ = new StringExp($1); }
//...
```

Scanner

Generating tokens from the scanner

```
id          [a-zA-Z][a-zA-Z_0-9]*
int         [0-9]+
string      "\""([^\\]|\\.)*"\""

%%

{id}        return parser::make_ID(yytext);
{int}       return parser::make_INT(atoi(yytext));
{string}    return parser::make_STRING(std::string(yytext + 1,
                                                   yyleng - 2));
```

or even (C++ 11)

```
{string}    return parser::make_STRING({yytext+1, yyleng-2});
```

Locations

Location tracking in the Scanner

Location tracking in Flex

What

loc the current location

How

%initial-action

run at the beginning of yyparse

YY_USER_ACTION

once per scanner match

%{ ... %}

(after the first %%) pasted into yylex.

When at its top when first in the rule section:

- local variables
- code run once per yylex invocation

Location tracking in Flex

What

loc the current location

How

`%initial-action`

run at the beginning of yyparse.

`YY_USER_ACTION`

once per scanner match

`%{ ... %}`

(after the first %%) pasted into `yylex`.

When at its top when first in the rule section:

- local variables
- code run once per `yylex` invocation

Location tracking in Flex

What

loc the current location

How

`%initial-action`

run at the beginning of `yyparse`.

`YY_USER_ACTION`

once per scanner match

`%{ ... %}`

(after the first %%) pasted into `yylex`.

When at its top when first in the rule section:

- local variables
- code run once per `yylex` invocation

Location tracking in Flex

What

loc the current location

How

%initial-action

run at the beginning of `yyparse`.

YY_USER_ACTION

once per scanner match

`%{ ... %}`

(after the first %%) pasted into `yylex`.

When at its top when first in the rule section:

- local variables
- code run once per `yylex` invocation

Location tracking in Flex

What

loc the current location

How

%initial-action

run at the beginning of `yyparse`.

YY_USER_ACTION

once per scanner match

%{ ... %}

(after the first %%) pasted into `yylex`.

When at its top when first in the rule section:

- local variables
- code run once per `yylex` invocation

Location tracking in Flex

What

loc the current location

How

%initial-action

run at the beginning of `yyparse`.

YY_USER_ACTION

once per scanner match

%{ ... %}

(after the first %%) pasted into `yylex`.

When at its top when first in the rule section:

- local variables
- code run once per `yylex` invocation

Location tracking in Flex

What

loc the current location

How

%initial-action

run at the beginning of `yyparse`.

YY_USER_ACTION

once per scanner match

%{ ... %}

(after the first %%) pasted into `yylex`.

When at its top when first in the rule section:

- local variables
- code run once per `yylex` invocation

Location tracking in Flex

What

loc the current location

How

%initial-action

run at the beginning of `yyparse`.

YY_USER_ACTION

once per scanner match

%{ ... %}

(after the first %%) pasted into `yylex`.

When at its top when first in the rule section:

- local variables
- code run once per `yylex` invocation

Location tracking in Flex

```
%{
  /* At each match, adjust the last column. */
# define YY_USER_ACTION  loc.columns(yyleng);
%}
/* ... */
%%
%{
  /* At each call, bring the tail to the head.  */
  loc.step();
%}
        /* Locations of blanks are ignored. */
[ \t]+  loc.step();

        /* Newlines change the current line number,
           but are ignored too. */
\n+     loc.line(yyleng); loc.step();
```

Location tracking in Flex

```
{id}        return parser::make_ID(yytext, loc);
{int}       return parser::make_INT(atoi(yytext), loc);
{string}    return parser::make_STRING({yytext+1, yyleng-2}, loc);
```

Location tracking in the Parser

Using the Location in the Parser

```
%define filename_type {const std::string}
%locations

%%

lvalue.big:
  ID "[" exp "]"
   { $$ = new SubscriptVar
              (@$, new SimpleVar(@1, $1), $3); }
| lvalue.big "[" exp "]"
   { $$ = new SubscriptVar(@$, $1, $3); }
;
```

Error Messages

```
%error-verbose
%%
// ...
%%
void
yy::parser::error(const location_type& l, const std::string& m)
{
  tp.error_ << misc::Error::parse
            << l << ": " << m << std::endl;
}
```

Improving the Scanner/Parser

Error Recovery

Error Recovery

- The `error` token in Yacc/Bison:
  1. dig in the stack to find a nice place
  2. throw away unpleasant lookaheads
  3. reduce as usual
- "Guard" it, put bounds around
- May introduce new conflicts.
- Do as if there were no error: generate dummy values
- Maybe introduce an `Error` class to prevent cascades of errors.

Error Recovery

- The `error` token in Yacc/Bison:
  1. dig in the stack to find a nice place
  2. throw away unpleasant lookaheads
  3. reduce as usual
- "Guard" it, put bounds around
- May introduce new conflicts.
- Do as if there were no error: generate dummy values
- Maybe introduce an `Error` class to prevent cascades of errors.

Error Recovery

- The `error` token in Yacc/Bison:
  1. dig in the stack to find a nice place
  2. throw away unpleasant lookaheads
  3. reduce as usual
- "Guard" it, put bounds around
- May introduce new conflicts.
- Do as if there were no error: generate dummy values
- Maybe introduce an `Error` class to prevent cascades of errors.

Error Recovery

- The `error` token in Yacc/Bison:
  1. dig in the stack to find a nice place
  2. throw away unpleasant lookaheads
  3. reduce as usual
- "Guard" it, put bounds around
- May introduce new conflicts.
- Do as if there were no error: generate dummy values
- Maybe introduce an `Error` class to prevent cascades of errors.

Error Recovery

- The `error` token in Yacc/Bison:
  1. dig in the stack to find a nice place
  2. throw away unpleasant lookaheads
  3. reduce as usual
- "Guard" it, put bounds around
- May introduce new conflicts.
- Do as if there were no error: generate dummy values
- Maybe introduce an `Error` class to prevent cascades of errors.

Error Recovery

- The `error` token in Yacc/Bison:
  1. dig in the stack to find a nice place
  2. throw away unpleasant lookaheads
  3. reduce as usual
- "Guard" it, put bounds around
- May introduce new conflicts.
- Do as if there were no error: generate dummy values
- Maybe introduce an `Error` class to prevent cascades of errors.

Error Recovery

- The `error` token in Yacc/Bison:
  1. dig in the stack to find a nice place
  2. throw away unpleasant lookaheads
  3. reduce as usual
- "Guard" it, put bounds around
- May introduce new conflicts.
- Do as if there were no error: generate dummy values
- Maybe introduce an `Error` class to prevent cascades of errors.

Error Recovery

- The `error` token in Yacc/Bison:
  1. dig in the stack to find a nice place
  2. throw away unpleasant lookaheads
  3. reduce as usual
- "Guard" it, put bounds around
- May introduce new conflicts.
- Do as if there were no error: generate dummy values
- Maybe introduce an `Error` class to prevent cascades of errors.

Error Recovery

parse/parsetiger.yy

```
// Reclaim the memory.
%destructor { delete $$; } exp
%%
exp:
  "nil"           { $$ = new NilExp(@$); }
| "(" exps ")"    { $$ = new SeqExp(@$, $2); }
| "(" error ")"   { $$ = new SeqExp(@$, new exps_t); }
// ...
```

Pure Parser

The Parsing Driver

- Information exchanged with the parser/scanner
  - Input data
    library path, debugging flags, etc.
  - Output data
    The `ast`, the error messages/status
  - Data maintained during the parsing
    Open files
- Coordination
  - Initialize/open the scanner
  - Parse
  - Close the scanner
- Introduce a parsing driver.

The Parsing Driver

- Information exchanged with the parser/scanner
  - Input data
    library path, debugging flags, etc.
  - Output data
    The ast, the error messages/status
  - Data maintained during the parsing
    Open files
- Coordination
  - Initialize/open the scanner
  - Parse
  - Close the scanner
- Introduce a parsing driver.

The Parsing Driver

- Information exchanged with the parser/scanner
  - Input data
    library path, debugging flags, etc.
  - Output data
    The `ast`, the error messages/status
  - Data maintained during the parsing
    Open files
- Coordination
  - Initialize/open the scanner
  - Parse
  - Close the scanner
- Introduce a parsing driver.

The Parsing Driver

- Information exchanged with the parser/scanner
  - Input data
    library path, debugging flags, etc.
  - Output data
    The `ast`, the error messages/status
  - Data maintained during the parsing
    Open files
- Coordination
  - Initialize/open the scanner
  - Parse
  - Close the scanner
- Introduce a parsing driver.

The Parsing Driver

- Information exchanged with the parser/scanner
  - Input data
    library path, debugging flags, etc.
  - Output data
    The `ast`, the error messages/status
  - Data maintained during the parsing
    Open files
- Coordination
  - Initialize/open the scanner
  - Parse
  - Close the scanner
- Introduce a parsing driver.

The Parsing Driver

- Information exchanged with the parser/scanner
  - Input data
    library path, debugging flags, etc.
  - Output data
    The `ast`, the error messages/status
  - Data maintained during the parsing
    Open files
- Coordination
  - Initialize/open the scanner
  - Parse
  - Close the scanner
- Introduce a parsing driver.

The Parsing Driver

- Information exchanged with the parser/scanner
  - Input data
    library path, debugging flags, etc.
  - Output data
    The `ast`, the error messages/status
  - Data maintained during the parsing
    Open files
- Coordination
  - Initialize/open the scanner
  - Parse
  - Close the scanner
- Introduce a parsing driver.

The Parsing Driver

- Information exchanged with the parser/scanner
  - Input data
    library path, debugging flags, etc.
  - Output data
    The `ast`, the error messages/status
  - Data maintained during the parsing
    Open files
- Coordination
  - Initialize/open the scanner
  - Parse
  - Close the scanner
- Introduce a parsing driver.

The Parsing Driver

- Information exchanged with the parser/scanner
  - Input data
    library path, debugging flags, etc.
  - Output data
    The `ast`, the error messages/status
  - Data maintained during the parsing
    Open files
- Coordination
  - Initialize/open the scanner
  - Parse
  - Close the scanner
- Introduce a parsing driver.

The Parsing Driver (parse/tiger-parser.hh)

```
class TigerParser
{
public:
  /// Parse a Tiger program, return its AST.
  ast::Exp* parse_program(...);
  /// Parse a Tiger prelude, return the list of decs.
  ast::decs_list_type* parse_import(...);

private:
  /// The result of the parse.
  ast_type ast_;
  /// Parsing errors handler.
  misc::error error_;
  /// The source to parse.
  input_type input_;
  /// The file library for imports.
  misc::file_library library_;
};
```

The Parsing Driver (parse/tiger-parser.cc)

```
void TigerParser::parse_() {
  std::string* fn = boost::get<std::string>(&input_);
  misc::symbol filename(fn == nullptr ? ""
                        : *fn == "-" ? "standard input" : *fn);
  location_.initialize(&filename.name_get());
  std::shared_ptr<std::istream> in;
  if (fn_ == "-")
    in.reset(&std::cin, [](...){});
  else {
      in = std::make_shared<std::ifstream>(filename);
      // Check for errors...
    }
  scanner_->scan_open(*in);
  parser parser(*this);
  parser.set_debug_level(parse_trace_p_);
  decs_ = nullptr; exp_ = nullptr;
  parser.parse();
  scanner_->scan_close();
}
```

The Parser (parse/parsetiger.yy)

```
%define filename_type {const std::string}
%locations

// The parsing context.
%param { parse::TigerParser& tp }
```

Two Grammars in One

The Parser

parse/parsetiger.yy

```
%token SEED_IMPORT  "seed-import"
%token SEED_SOURCE  "seed-source"
%%
program:
  /* Parsing a source program. */
  "seed-source" exp                  { tp.exp_ = $2; }
| /* Parsing an imported file.  */
  "seed-import" "let" decs "end"     { tp.decs_ = $3; }
;
```

The Scanner: Wrapping yyflex

parse/scantiger.ll

```
int
yylex (yystype *yylval, yy::location *yyloc,
       parse::TigerParser& tp)
{
  if (tp.seed_)
    {
      int res = 0;
      std::swap(res, tp.seed_);
      return res;
    }
  else
    return flex_yylex(yylval, yyloc, tp);
}
```

The Scanner: Using the top of yyflex

parse/scantiger.ll

```
%%
%{
  if (tp.seed_)
    {
      int res = 0;
      std::swap(res, tp.seed_);
      return res;
    }
%}
```

Without Seeds

parse/parsetiger.yy

```
%%
program:
  /* Parsing a source program. */
  exp   { tp.exp_ = $1; }
| /* Parsing an imported file.  */
  decs  { tp.decs_ = $1; }
;
```

Reentrancy

1. Symbols
2. Semantic Values
3. Locations
4. Improving the Scanner/Parser
   - Error Recovery
   - Pure Parser
   - Two Grammars in One
   - Reentrancy

Reentrant Flex Scanner

parse/scantiger.ll

```
void yyFlexLexer::scan_open_(std::istream& f)
{
  yypush_buffer_state(YY_CURRENT_BUFFER);
  yy_switch_to_buffer(yy_create_buffer(&f, YY_BUF_SIZE));
}

void yyFlexLexer::scan_close_()
{
  yypop_buffer_state();
}
```

Recursive Invocation of the Parser

parse/parsetiger.yy

```
importdec: "import" STRING
  {
    $$ = tp.parse_import(take($2), @$);
    // Parsing may have failed.
    if (!$$)
      $$ = new ast::decs_list_type;
  }
;
```