Lab 2. Part I. Parser for cool.

Mipt (llab), 7.11.2018

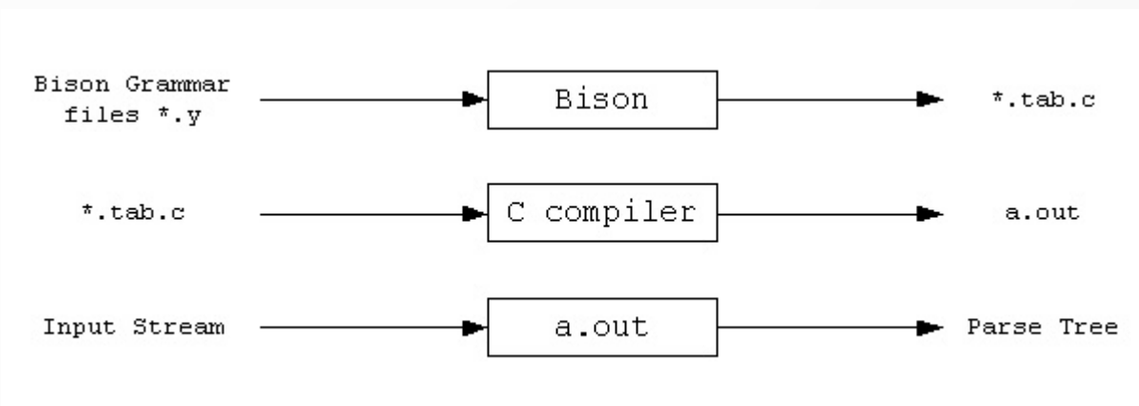
Parser

- Лексер
  - Может поймать опечатки в ключевых словах
  - Может отловить ошибки в структурах, типа строк
- Парсер
  - Ловит грамматические ошибки (незакрытые скобки, неправильное с точки зрения языка использование ключевых слов)
  - Не может отловить ошибки несоответствия типов
- И как бонус AST

Gnu Bison

- Bottom up Parser
- LALR(1) - Look-Ahead LR parser
  - LR - Left-to-right, Rightmost derivation in reverse

- Repository with assignment
- Включает в себя:
  - Задание
  - Cool manual
  - Cool toor



Структура

Bison - генератор синтаксических анализаторов.

%{

Объявления C

%}

Объявления Bison

%%

Правила грамматики

%%

Дополнительный код на C

Example 1. (to console)

```
%{
   #include <stdio.h>
   #include <string.h>
   void yyerror(const char *str){ fprintf(stderr,"ошибка: %s\n",str);}
   int yywrap(){return 1;}
   main(){ yyparse(); }
%}
%token NUMBER TOKHEAT STATE TOKTARGET TOKTEMP
%%

commands: /*empty*/ | commands command ;

command: heat_switch | target_set ;

target_set: TOKTARGET TOKTEMP NUMBER  { printf("\t %d град\n",$3); } ;

heat_switch: TOKHEAT STATE  { if($2) printf("вкл\n"); else
printf("выкл\n"); } ;
```

Объявления Bison

- **%union** - уже используется в шаблонном cool.y - на выходе генерируются уже знакомые **yylval**.*** (**symbol, bool, errmsg** e.t.c)
- **%start** - установление начального правила разбора. Необязателен.
- **%token** - терминальные символы - знакомы по cool-parce.h.
- **%type** - определения нетерминальных символов. Надо будет дополнить.
- **%left , %right, %nonassoc** - приоритеты терминальных символов.

Example 2. (to console)

```
%{
...
%}
%union {
        int ival;
}
%token<ival> T_INT
%token T_PLUS T_MINUS T_MULTIPLY
            T_LEFT T_RIGHT
%token T_NEWLINE T_QUIT
%left T_PLUS T_MINUS
%left T_MULTIPLY
%type<ival> expression
%start calculation
```

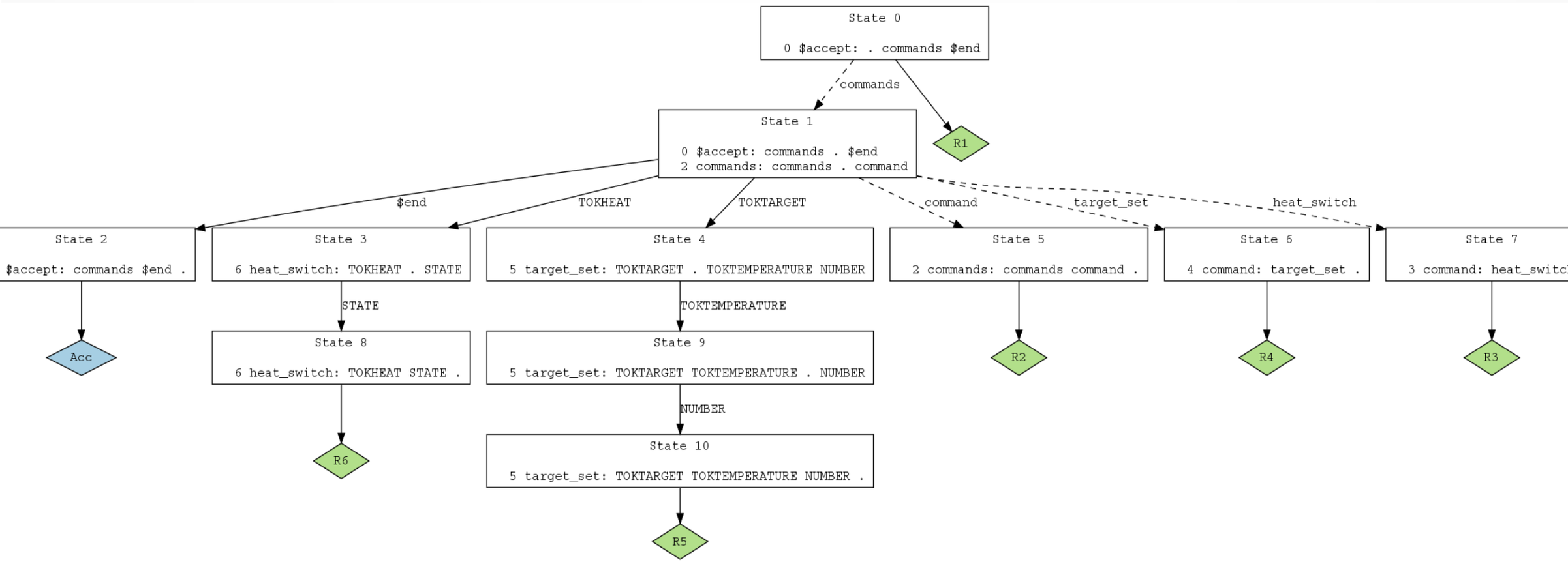
```
%%
calculation:
            | calculation line
;
line: T_NEWLINE
    | expression T_NEWLINE { printf("\tResult: %i\n", $1); }
    | T_QUIT T_NEWLINE { printf("bye!\n"); exit(0); }
;

expression: T_INT                                { $$ = $1; }
          | expression T_PLUS expression    { $$ = $1 + $3; }
          | expression T_MINUS expression { $$ = $1 - $3; }
          | expression T_MULTIPLY expression   { $$ = $1 * $3; }
          | T_LEFT expression T_RIGHT              { $$ = $2; }
;
%%

...
```

github.com/meyerd/flex-bison-example.git

Bison --graph (to console)

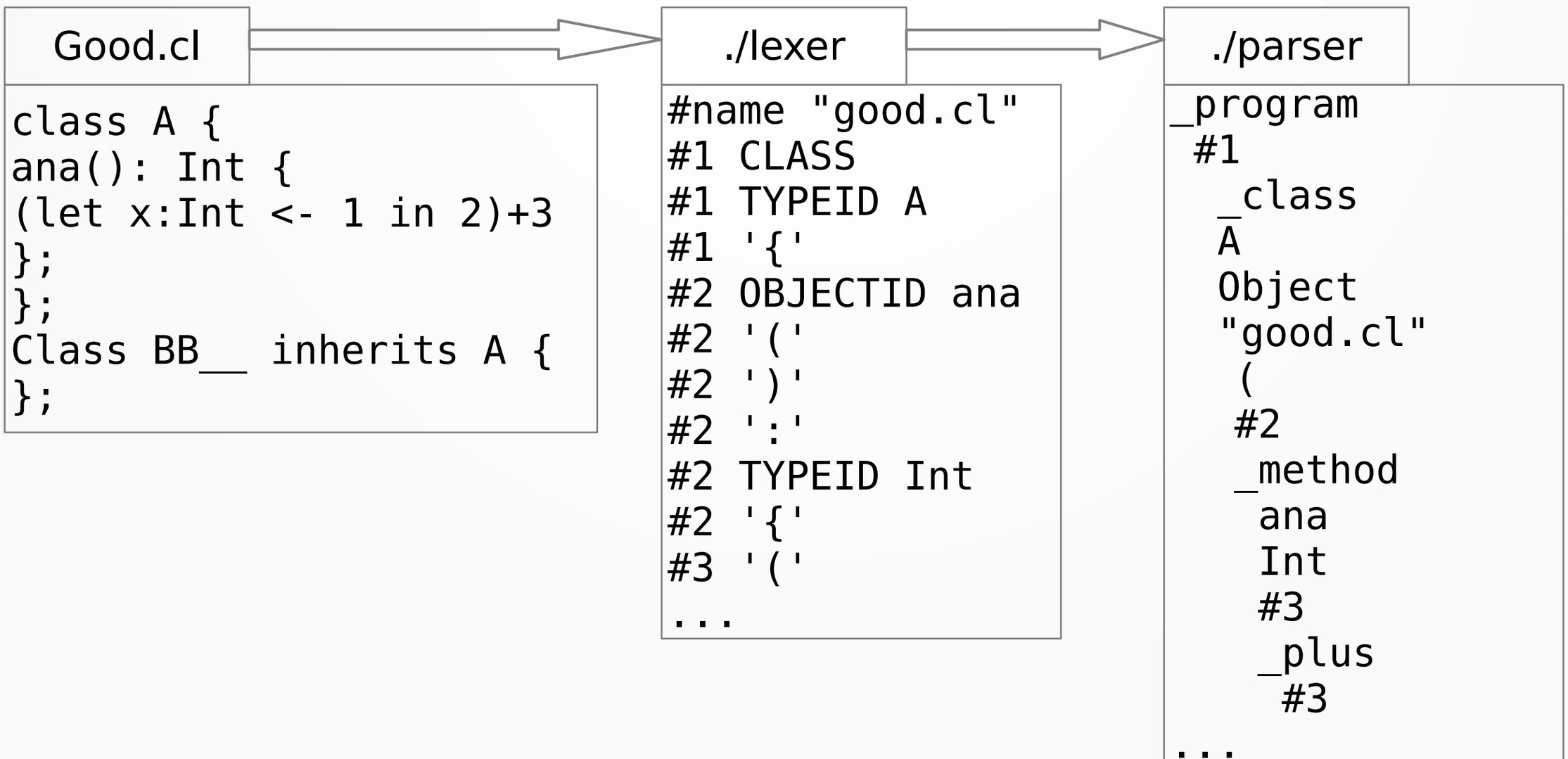


```
program ::= [[class; ]]+
  class ::= class TYPE [inherits TYPE] { [[feature; ]]}
feature ::= ID( [ formal [[, formal]] ] ) : TYPE { expr }
          | ID : TYPE [ <- expr ]
 formal ::= ID : TYPE
   expr ::= ID <- expr
          | expr[@TYPE].ID( [ expr [[, expr]] ] )
          | ID( [ expr [[, expr]] ] )
          | if expr then expr else expr fi
          | while expr loop expr pool
          | { [[expr; ]]+}
          | let ID : TYPE [ <- expr ] [[, ID : TYPE [ <- expr ]]] in expr
          | case expr of [[ID : TYPE => expr; ]]+esac
          | new TYPE
          | isvoid expr
          | expr (+|-|*|/) expr     | ~expr          | expr < expr
          | expr <= expr            | expr = expr    | not expr
          | (expr)                  | ID             | integer
          | string                  | true
          | false
```

Задача 1—скопипастить и заменить все лексемы на токены

(Cool manual Figure 1: Cool syntax. Page 17)

Cool parser (to console)



tree_node.h + doxygen + to console

Дерево разбора (to pdf)

- tree.h + cool-tree.h
  - Класс tree_node

    example: class Expression_class : public tree_node

    => **plus_class** : public Expression_class {

    содержит

    Expression e1;

    Expression e2;
    - **include/cool-tree.h + 818** находим конструктор для плюса

      Expression plus(Expression, Expression);
  - cool-toor.pdf + 6.5 The Constructors - описание конструкторов для создания дерева разбора. (in src cool-tree.cc + cool-tree.h)

AST example %union

Например:

Expression expression;

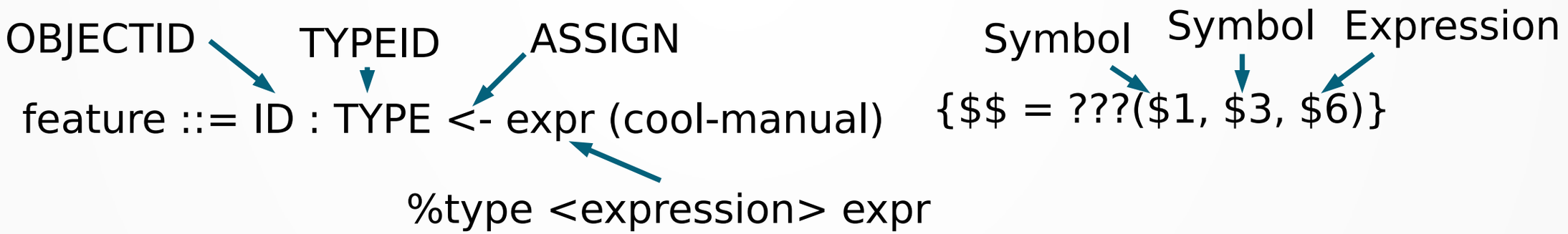
Это определение выражения. На основе union мы создаем нетерминал

%type <expression> expr

Это означает, что если мы хотим реализовать правило для expr – возвращаемым типом будет «Expression» (где возвращаемое значение – это $$)

Аналогично, если в другом правиле используется expr, то нужно его преобразовывать, например:

Что должно стоять на месте ???



Для ответа:

assigngetstype.test :

1. Необходимо понимать, что означает запись:

class Hello { foo:Int <- 89; ...};

2. Открыть cool-tree.h / cool-tree.aps (или cool-tour.pdf 6.5 The Constructors) и найти метод, который реализует эту логику.

При этом возвращает объект типа Feature и принимает два символа и Expression.

Cool parser example

```
addedlet.test.out
#3
_program
  #3
  _class
    Foo             $$ = singleClasses($1)
    Object
    "addedlet.test"
    (
    #2
    _method         $$ = method($1, nil_Formals(), $5, $7);
      bar           $1 = bar
      Int           $5 = Int (TYPEID)
      #2            $7 = let... (EXPRESSION)
      _let
        a
        Int
        #2
        _no_expr
        : _no_type
        #2
        _plus
```

addedlet.test

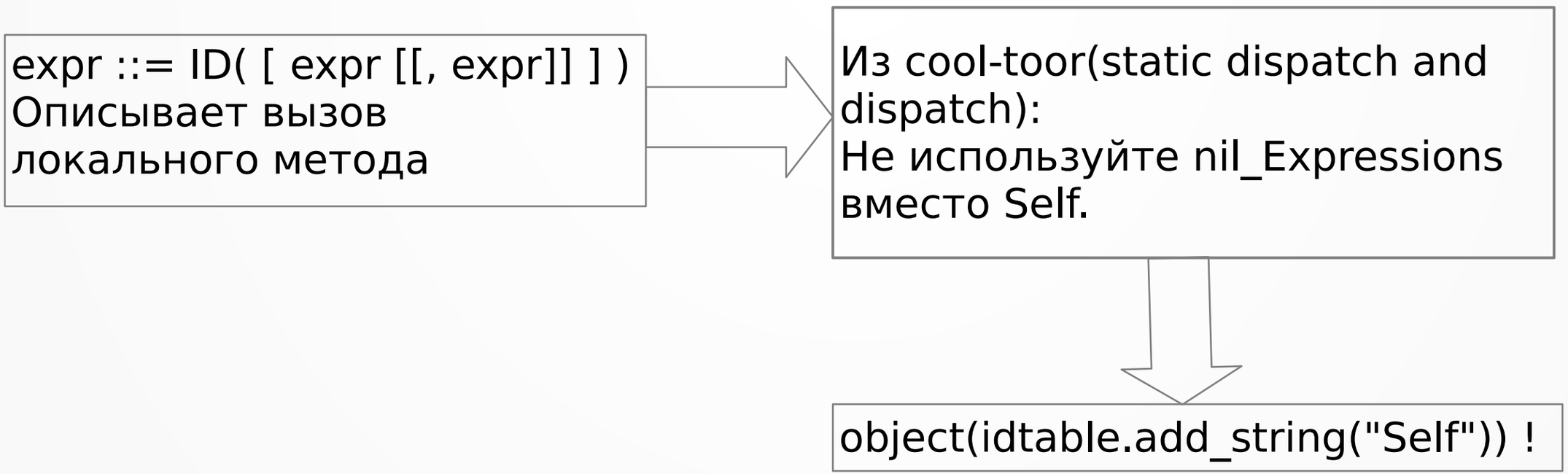
```
class Foo {
    bar():Int{
        let a:Int in a +
        let b:String in b
    };
};
```

Cool.y

- Заполненная секция %union
- Заполненная секция с %token
- *Заготовка секции с %type*
- *Заготовка правил грамматики*

Для решения надо будет сначала реализовать все правила и только потом бодаться с ошибками.

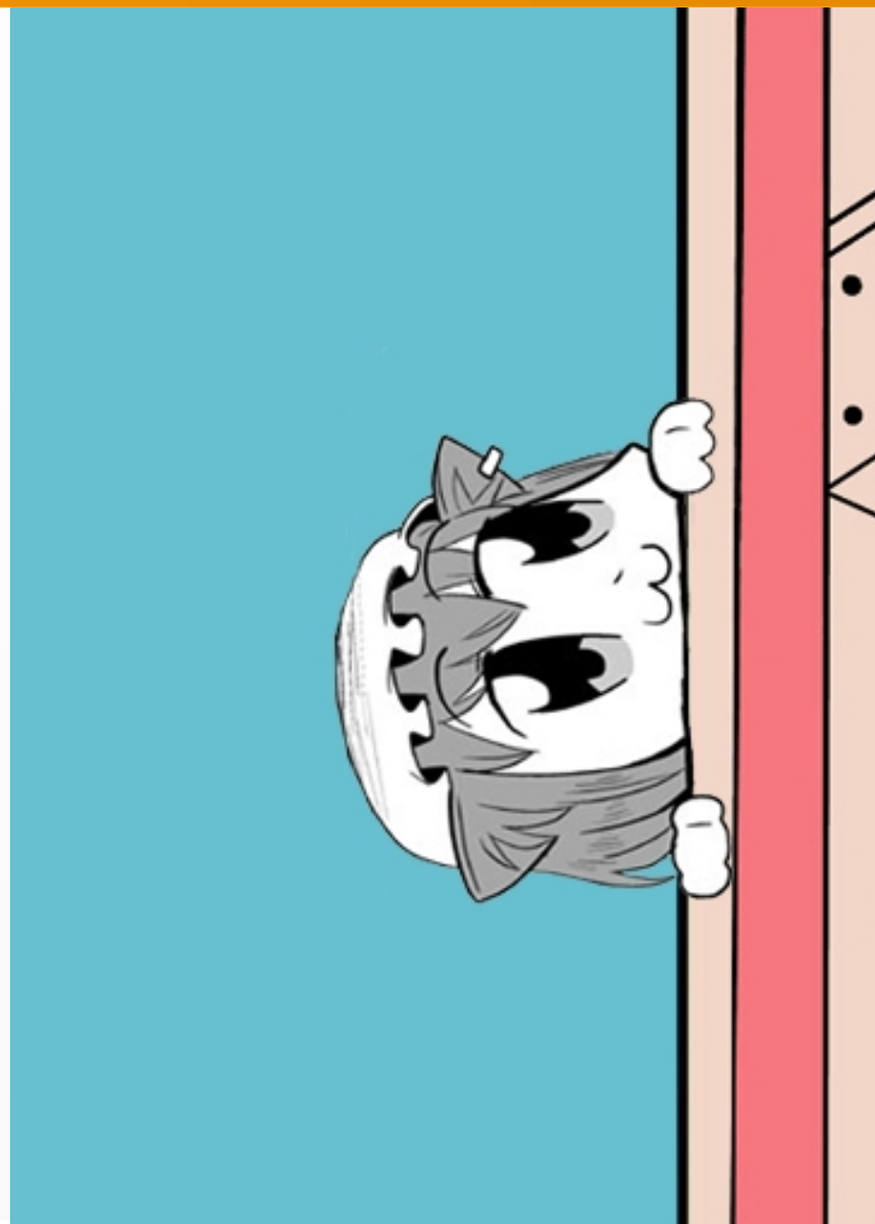
Example



- То есть не забываем, что вызов без указания типа – это вызов локального метода, поэтому в качестве обьекта необходимо передать текущий обьект

Приоритет операций

- cool-manual + 11.1 Precedence
- %left, %right, %noassoc
- порядок в бизоне обратный
- Описание приоритета в док е к бизону



Обработка ошибок

- Добавляйте правило error (bison-default) для выдачи сообщений об ошибках.
- Пример:

```
feature_list : ...
                | feature_list error ‘;' {}
                | error ';'  {}
```

- Если вы видите, что при разборе после ошибки процесс разбора прекращается, значит вы забыли добавить обработку ошибки.

Статьи о bison

- Open net doc
- Habr

  (цыкл статей о компиляторе)