Компиляция лямбда-выражений в языке программирования PascalABC.NET

Студент 4 курса 1 группы
кафедры алгебры и дискретной математики
Саушкин Р.С.
Научный руководитель: доц. Михалкович С.С.

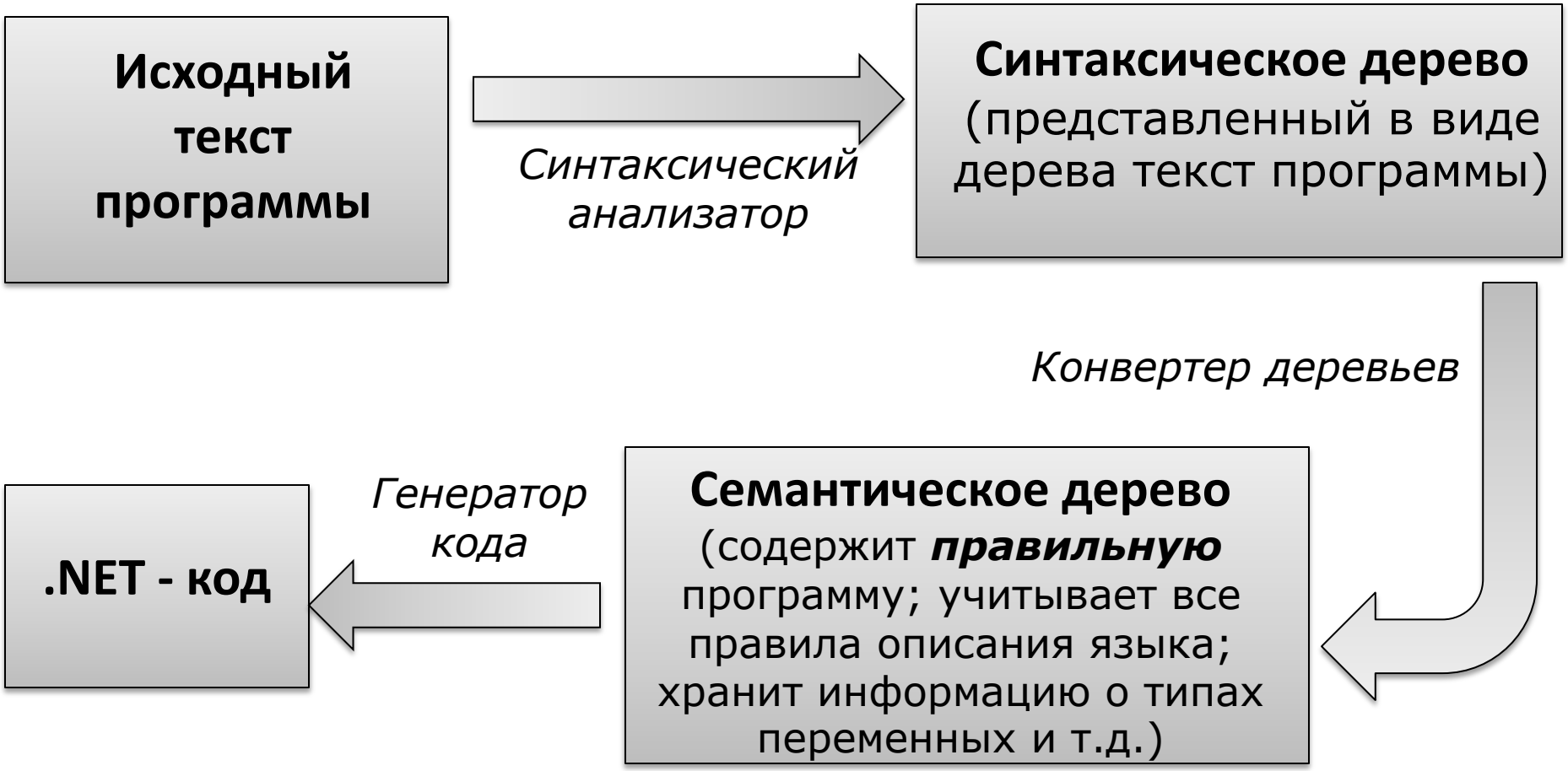
Постановка задачи

Внедрение лямбда-выражений в язык PascalABC.NET

Подзадачи:

- ✓ Введение грамматических конструкций лямбда-выражений, которые бы не конфликтовали с исходной грамматикой PascalABC.NET
- ✓ Реализация автоматического вывода типов формальных параметров и возвращаемого значения лямбда-выражений
- ✓ Реализация захвата переменных из внешнего контекста лямбда-выражений

Стадии компиляции в PascalABC.NET



Как были реализованы λ-выражения раньше

Преобразования, связанные с λ-выражениями, производились ***только*** *на* синтаксическом уровне.

Минусы такого подхода:

- **Невозможно определять тип переменных** → Автоматический вывод типов параметров и возвращаемого значения лямбды ***невозможен***
- **Невозможно определять контекст, в котором описана переменная** → Захват переменных из внешнего контекста ***невозможен***

ЧТО СДЕЛАНО

Что сделано

- ✓ Полностью переписана грамматика, связанная с лямбдами
- ✓ Реализована поддержка λ-выражений
- ✓ Реализован вывод типов формальных параметров и возвращаемого значения λ-выражений
- ✓ Реализован захват переменных из внешнего контекста

Что сделано

- ✓ Полностью переписана грамматика, связанная с лямбдами
- ✓ Реализована поддержка λ-выражений
- ✓ Реализован вывод типов формальных параметров и возвращаемого значения λ-выражений
- ✓ Реализован захват переменных из внешнего контекста

GPPG и LR(1)-грамматики

Генератор синтаксических анализаторов GPPG строит **LR(1) - анализаторы**

L — означает сканирование входного потока слева направо,
R — означает построение правого порождения
1 — количество предпросматриваемых символов входного потока, необходимое для принятия решения, равно 1

Грамматические правила λ-выражений: первоначальные решения, конфликт Reduce/Reduce

func_decl_lambda

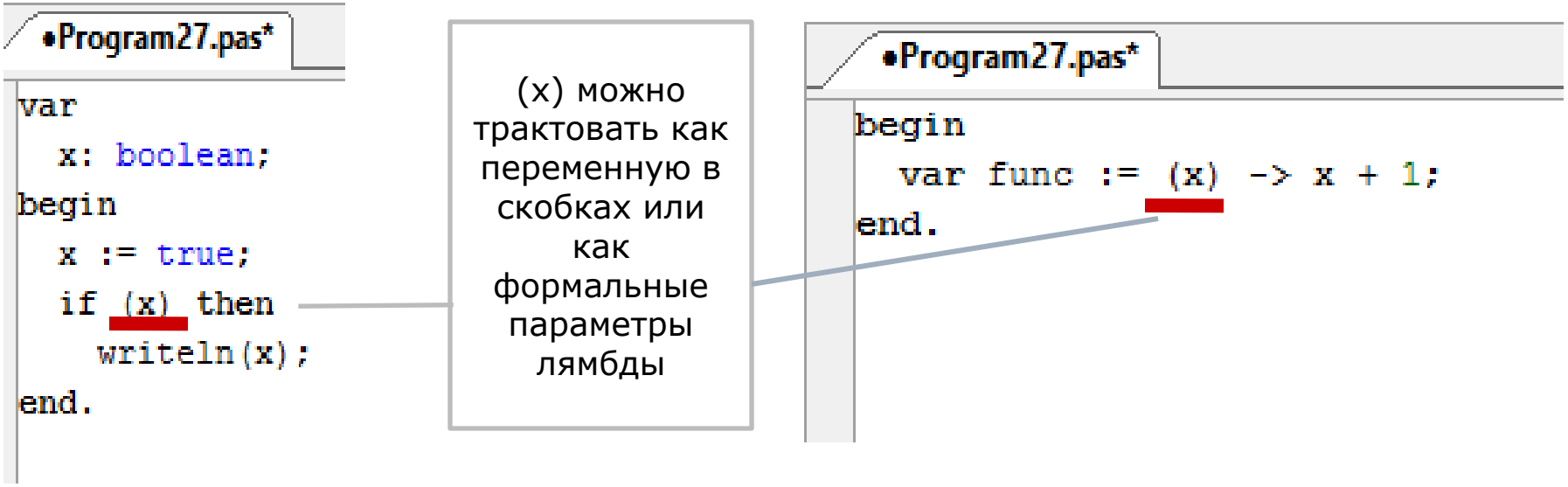
: lambda_formal_parameters_list lambda_return_type tkArrow lambda_body

lambda_formal_parameters_list

: tkRoundOpen ident tkRoundClose

expr

: ident | tkRoundOpen expr tkRoundClose | func_decl_lambda



Грамматические правила λ-выражений: первоначальные решения

typed_const : record_const | func_decl_lambda
record_const : tkRoundOpen const_field tkRoundClose
const_field : ident tkColon typed_const

lambda_formal_parameters_list
: tkRoundOpen ident tkColon fptype tkRoundClose

•Program27.pas*

```
type
  GenderType = (male, female);
  Person = record
    Name: string;
    Age, Weight: integer;
    Gender: GenderType;
  end;

const p: Person = (Name: 'Петрова'; Age: 18; Weight: 55; Gender: female);
```

На каком-то шаге разбора парсер может пойти либо по правилу «инициализатор записи» (тогда в строке Age: 18 будет ошибка), либо по правилу «список параметров лямбды»

Грамматические правила λ-выражений: окончательное решение

Program27.pas

```
type
  ft = function(x:integer): integer;

procedure proc(f: ft; x: integer);
begin
  writeln(f(x));
end;

begin
  var f :ft := function (x: integer): integer -> x + 1;
  proc((x: integer) -> x * x, 5);
end.
```

Что сделано

- ✓ Полностью переписана грамматика, связанная с лямбдами
- ✓ Реализована поддержка λ-выражений
- ✓ Реализован вывод типов формальных параметров и возвращаемого значения λ-выражений
- ✓ Реализован захват переменных из внешнего контекста

Примеры программ с использованием лямбд

```
begin
  var a := function (x: integer) -> x * x;
  var x: integer;
  write('Введите целое число: ');
  readln(x);
  writeln(x, ' ^ 2 = ', a(x));
end.
```

Окно вывода

```
Введите целое число: 45
45 ^ 2 = 2025
```

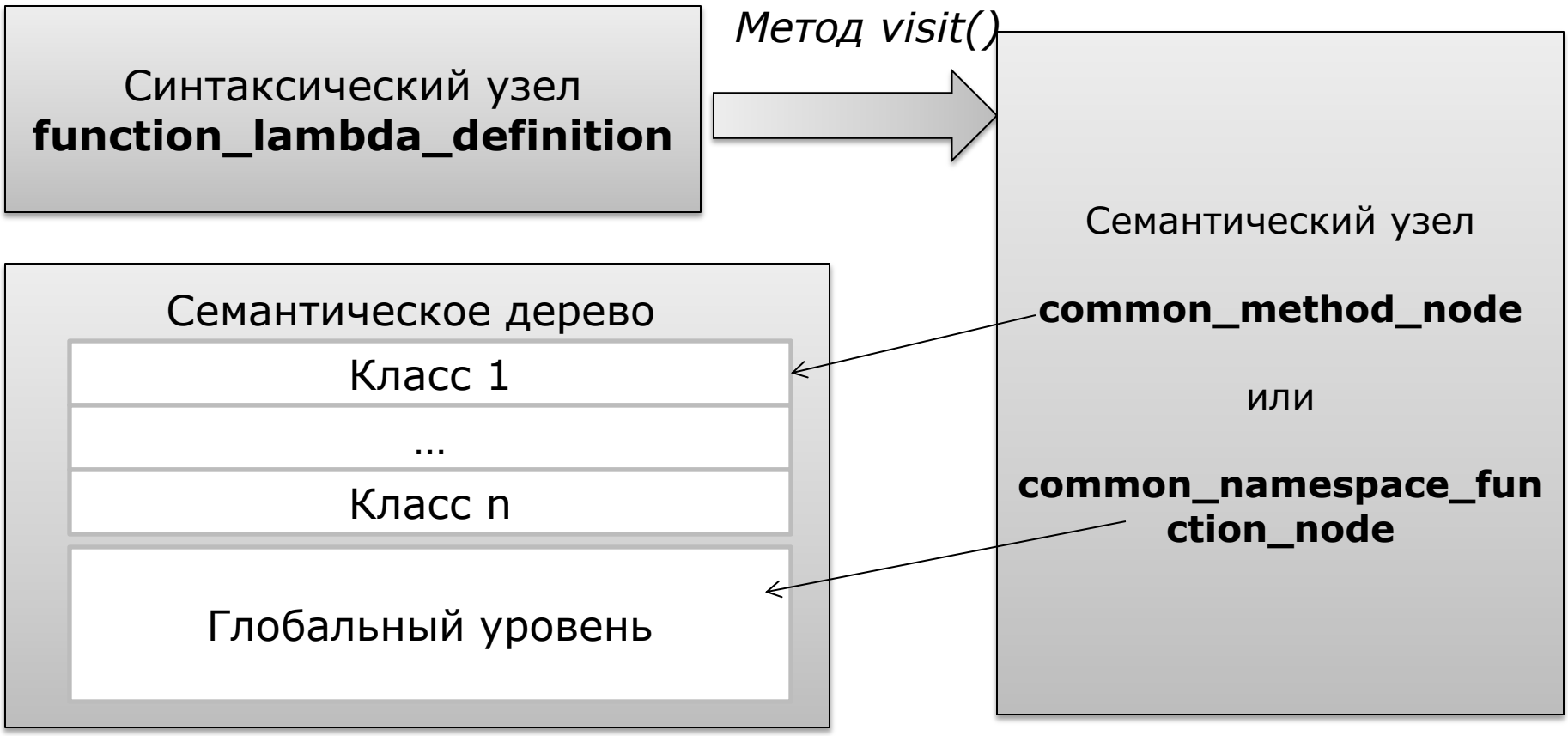
Примеры программ с использованием лямбд

```
const
  n = 50;
begin
  var a := new integer[n];
  for var i := 0 to n - 1 do
    a[i] := random(100);
  var aa := a.Where(x -> x mod 2 = 0).OrderBy(x -> x);
  foreach x: integer in aa do
    write(x, ' ');
end.
```

Окно вывода

```
0 2 4 6 10 16 24 30 32 34 34 38 40 40 54 60 74 82 82 92 94 98
```

Реализация лямбд



Что сделано

- ✓ Полностью переписана грамматика, связанная с лямбдами
- ✓ Реализована поддержка λ-выражений
- ✓ Реализован вывод типов формальных параметров и возвращаемого значения λ-выражений
- ✓ Реализован захват переменных из внешнего контекста

Вывод типов

1

```
type
  ft = function(x:integer): integer;

begin
  var f :ft := function (x) -> x + 1;
end.
```

2

```
const
  n = 50;
begin
  var a := new integer[n];
  for var i := 0 to n - 1 do
    a[i] := random(100);
  var aa := a.Where(x -> x mod 2 = 0).OrderBy(x -> x);
  foreach x: integer in aa do
    write(x, ' ');
end.
```

Алгоритм автовывода типов в случае наличия нескольких версий перегруженных функций

1. Всем параметрам лямбды, тип которых не указан явно, присвоить тип <Any>
2. Провести стандартный поиск перегруженной версии с модификацией: возвращаемые значения лямбд не учитываются, тип <Any> приводим неявно к любому типу.
3. Если в качестве потенциальных осталось несколько функций, вывести ошибку о том, что возникла ситуация неоднозначности выбора функции
4. Иначе произвести ОБРАТНЫЙ вывод типов параметров у лямбды (фактического параметра) по типам формальных параметров вызываемой функции
5. Вывести тип возвращаемого значения лямбды и сопоставить его с типом возвращаемого значения формального параметра вызываемой функции

Теорема. Приведенный алгоритм завершим.

Пример использования алгоритма

- Происходит вызов функции f(x -> x + 1, 5), типы параметров и тип возвращаемого значения <Any>:

  f((x:<Any>):<Any> -> x + 1, 5)

- Имеем перегруженные версии функции со следующими заголовками

  f(g: integer->integer; x: integer)

  f((g, h: integer)->integer; x: integer)

  f(g: integer->integer)

Первый этап: Поиск перегруженной версии функции (возвращаемые значения лямбд не учитываются). Из приведенных версий функции подходит лишь

f(g: integer->integer; x: integer)

Второй этап: Вывод типов параметров и возвращаемого значения лямбды:

f((x:integer):integer -> x + 1, 5)

Возвращаемый тип лямбды согласуется с возвращаемым типом формального параметра вызываемой функции f

Что сделано

- ✓ Полностью переписана грамматика, связанная с лямбдами
- ✓ Реализована поддержка λ-выражений
- ✓ Реализован вывод типов формальных параметров и возвращаемого значения λ-выражений
- ✓ Реализован захват переменных из внешнего контекста

Захват переменных из внешнего контекста

```
var
  t := 5;
type
  ft = function(x: integer): integer;
type cl = class
  private a: integer := 6;
  public function aaa: integer;
  begin
    var lf: ft := function(x) -> 5 + x + a + t;
    writeln(lf(5));
  end;
end;

var
  d := new cl();
begin
  d.aaa;
  var g := function(x: integer):integer -> 5 + x + t;
  writeln(g(55));
end.
```

Итоги:

- ✓ Полностью переписана грамматика, связанная с лямбдами
- ✓ Реализована поддержка λ-выражений
- ✓ Реализован вывод типов формальных параметров и возвращаемого значения λ-выражений
- ✓ Реализован захват переменных из внешнего контекста

Спасибо за внимание