# Распознавание экранируемых предложений в программе на Рефале-5λ

**Александр Барлука**
МГТУ имени Н. Э. Баумана

Третье совместное рабочее совещание
ИПС имени А.К. Айламазяна РАН и МГТУ имени Н.Э.Баумана
по функциональному языку программирования Рефал

**12 июня 2020 года**

# Постановка задачи

```
Test {
    s._ e._ = True;
    ‘A’  e._ = False
}
```

Экранирующие предложения такого вида могут быть написаны только по ошибке, и эта ошибка может быть достаточно неочевидной.

Но при этом её вполне можно диагностировать автоматически и выдавать соответствующее предупреждение.

# Образцы с условиями

```
Test {
    s._ , <Pred1 …>= True;
    ‘A’  , <Pred2 …> = False
}
```

Случай, когда не выполняется правая часть предложения с условием, сложен для автоматического анализа. Более того, такие предложения могут быть написаны намеренно ради побочного эффекта в условии и отката на последующие предложения.

# Расширенные L-выражения

- Вхождение **t-** и **e-**переменных в выражение допускается не более одного раза;
- В каждом подвыражении данного выражения содержится не более одной не обрамленной структурными скобками **e-**переменной.

# Обобщенное сопоставление

***<GenericMatch (e.Pattern) (e.LPattern)>***

***== Clear (e.Val ':' t.Var)****

***== Contracted ((t.Var ':' e.Val)* (e.Val ':' t.Var))****

***== Failure***

***== Undefined***

Функция ***GenericMatch*** принимает пару образцов и определяет, является ли первый подмножеством второго.

Можно использовать функцию обобщённого сопоставления — если сопоставление чистое, то образец точно вкладывается.

# Особенности реализации

- Новые переменные (“e.X^”)
- Связанные и повторные переменные
- Блоки, замыкания, присваивания и условия

# Построение общего формата

Для двух образцов строить их формат, выбирать из форматов подвыражения-аргументы и далее рассматривать предложенную теорию не на парах образцов,
а на парах кортежей образцов.

# Не L-выражения?

*Якорной* называется t-переменная *t.x* в образце *P*, если:

- либо t.x имеет кратность $\geq 2$;
- либо в P существует подслово A, не содержащее e-переменных, такое, что A = B t.x C, причем и B, и C содержат хотя бы один символ или t-переменную, имеющую кратность $\geq 2$.

В противном случае назовем t.x *плавающей*.

# Нормальная форма образца

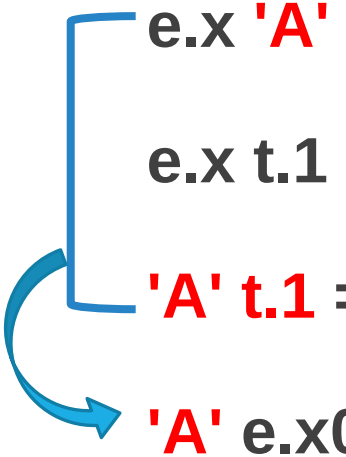
Считаем, что линейный образец P *в нормальной форме*, если

- P не содержит двух и более идущих подряд e-переменных;
- каждая плавающая t-переменная в P предшествует e-переменной и следует за e-переменной.

Нормальная форма образца t.y1 t.y2 e.x1 t.y3 t.y4 t.y2 e.x2 t.y5 есть t.y1 t.y2 e.x1 t.y3 e.x3 t.y4 e.x4 t.y2 e.x2 t.y5 e.x5

# Групповое экранирование

```
Test {
    e.x 'A' e.y s.w s.w e.z = 1;
    e.x 'A' = 2;
    e.x t.1 t.2 t.3 e.y = 3;
    'A' t.1 = 4;
    'A' e.x0 = 5;
}
```



Преобразования:

e.X → e._ s._ e._
e.X → e._ (e._) e._
e.X → /* empty */
e.X → e._ t._ e._

# Групповое экранирование

Образец P может экранироваться объединением образцов P1 и P2, но ни одним из них, если в P1 или в P2 есть плавающие переменные.

Можно заметить, что последовательности якорей образцов P2-P4 похожи на последовательность якорей образца P в смысле преобразований, а у P1 есть «лишний» фрагмент s.w s.w, который такими преобразованиями получиться не может.

# Заключение

- Реализован механизм выдачи предупреждений в компиляторе
- Реализован алгоритм для случаев L-выражений
- Рассмотрена теория языков образцов и приведены соображения для написания алгоритма, покрывающего иные случаи экранирования