МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Н. Э. БАУМАНА

ВЫЧИСЛИТЕЛЬ И ВЕРИФИКАТОР ТИПОВ ФУНКЦИЙ ДЛЯ ЯЗЫКА РЕФАЛ-5

Выполнил:
Иванов Георгий, ИУ9-72

Научный руководитель:
Коновалов Александр Владимирович

МОСКВА, 2018

Цель и задачи

- Целью данной курсовой работы является разработка верификатора типов в РЕФАЛ, который будет проверять корректность вызовов функций в программе, т.е. аргумент в любом вызове функции должен быть совместим с форматом аргумента.
- Задачи – обзор предметной области, разработка алгоритма вывода типов функций

Рефал-5

РЕФАЛ (РЕкурсивных Функций АЛгоритмический язык) - это один из старейших функциональных языков программирования, который ориентирован на символьные преобразования: обработку символьных строк (например, алгебраические выкладки), перевод с одного языка (искусственного или естественного) на другой. РЕФАЛ был впервые реализован в 1968 году в России Валентином Турчиным, где широко используется и поныне, который соединяет в себе математическую простоту с практической ориентацией на написание больших и сложных программ.

Разработка языка типов

Базисом для собственного языка типов в динамическом языке РЕФАЛ-5 будем иметь ввиду статическую типизацию в РЕФАЛ+, а именно для определения типа функции для верификатора спецификация функции будет иметь вид:

$$func\ format_{in} = format_{out}$$

Лексические домены РЕФАЛ-5

EXTERN_KEYWORD=\$EXTERN|\$EXTRN|\$EXTERNAL|\$ENTRY

IDENT=[A-Za-z][-A-Za-z_0-9]*

MACROGIDIT=[0-9]*

VARIABLE=[set]\.[-A-Za-z_0-9]+

COMPOSITE_SYMBOLS=IDENT|([^\\"]|\\([\\nrt"'()<>]|x[0-9a-fA-F]{2}))*

CHARS=([^\\']|\\([\\nrt"'()<>]|x[0-9a-fA-F]{2}))*

MARK_SIGN=[()<>=;:,{}]

LEFT_CALL_BRACKET=<[+-*/\%?]

Грамматика РЕФАЛ-5

Program ::= Global*

Global ::= Externs | Function | ';'

Externs ::= ExternKeyword 'Name' (',' 'Name')* ';'

ExternKeyword ::= '$EXTERN' | '$EXTRN' | '$EXTERNAL'

Function ::= ('$ENTRY')? 'Name' Body

Body ::= '{' Sentences '}'

Sentences ::= Sentence (';' Sentences?)?

Sentence ::= Pattern (('=' Result) | (',' Result ':' (Sentence | Body)))

Pattern ::= PatternTerm*

PatternTerm ::= Common | '(' Pattern ')'

Common ::= 'Name' | "chars" | '123' | 's.Var' | 't.Var' | 'e.Var'

Result ::= ResultTerm*

ResultTerm ::= Common | '(' Result ')' | '<Name' Result '>';

Грамматика языка типов РЕФАЛ-5

File ::= Function*

Function ::= 'Name' Format '=' Format ';’

Format ::= Common ('e.Var' Common)?

Common ::= ('Name' | "chars" | '123' | 't.Var' | 's.Var' | '(' Format ')')*

Семантический анализ

- Все имена функций после < или встроенные, или описанные в текущем файле, или объявленные как $EXTERN.
- Для каждой переменной в результатном выражении должно быть её вхождение в образцовом выражении в предшествующей части предложения.
- Функция в исходном тексте должна быть определена один раз — или как функция, или как $EXTERN.
- В тексте спецификации типов семантический анализатор должен только проверять, что имена функций не повторяются — нет двух разных определений типов для одной функции.
- В паре файлов Рефал-5 и спецификация типов семантический анализ должен проверять, что все функции, для которых заданы спецификации типов, определены в исходном файле.

Алгоритм вывода типов для базисного РЕФАЛа.

- Выполняем сквозную нумерацию всех предложений. Ко всем индексам переменных приписываем номер предложения.
- Для каждой правой части строится набор уравнений. Извлекается один из вызовов функции, например F. Терм конкретизации заменяется на out(F), аргумент сопоставляется с in(F).

Например, для правой части: Для EA <F EB <G EC> ED <H EF> EG> EH строится система уравнений:

$$\begin{cases} \text{EB out(G) ED out(H) EG : in(F)} \\ \text{EC: in(G)} \\ \text{EF: in(H)} \end{cases}$$

- Начальным форматом всех функций полагаем in(F)=out(F)=⊥, где ⊥ - некоторое специальное значение (функция которая не возвращает никакого значения).

Обозначим *Pat(f, i)* - i-ая левая часть функции f, *Res(f,i)* - i-я правая часть f, $\overline{Res}(f, i)$- i-я правая часть f с заменой вызовов функций на out(g).

- Пытаемся решить систему уравнений. Если у уравнения нет решений - сообщаем об ошибке. Если система имеет единственное решение - берём для переменных соответствующее значение. Если несколько, то тогда берём первое попавшееся.
- Уточняем форматы. Строим обобщение с подстановками значений переменных.
- Повторяем решение системы уравнений до тех пор, пока in(f) и out(f) не перестанут меняться.

Алгоритм сопоставления (решения уравнения)

Жёсткое выражение имеет вид $Ht‘1…Ht‘N\ e.XHt“M$ …$Ht“1$, либо $Ht1…HtN$, где Ht — жесткие термы, e.X — e-переменная. Т.е. надо рассмотреть четыре случая:

- Жёсткое выражение начинается на жёсткий терм
- Жёсткое выражение кончается на жёсткий терм
- Жёсткое выражение состоит из одной e-переменной
- Жёсткое выражение пустое.

Алгоритм обобщения (термов)

P_1/P_2	X	$Y \neq X$	s	t	(P_2)
X	X	s	s	t	t
s	s	s	s	t	t
t	t	t	t	t	t
(P_1)	t	t	t	t	$GCG(P_1, P_2)$

Пример работы:

```
F {
    e.X = <G e.X> <H e.X>;
}
G { s.Y e.Z = s.Y }
H { e.Y (e.Z) = e.Z }
```

Форматы функций:

```
F s e (e) = s e;
G s e = s;
H e (e) = e;
```

```
F {
    A A A A e.X = <F A A e.X>;
    A = A;
}
```

Форматы функций:

невозможно вывести!

Выводы

В рамках данной курсовой работы был изучен язык Рефал-5. А также были реализованы алгоритмы сопоставления, обобщения подстановок, а также разработан и реализован алгоритм вывода типа функций. Данный верификатор был протестирован на более 30 программ, написанных на Базисном РЕФАЛе. В дальнейших планах исследования на тему:

- Модификации алгоритма обобщения
- Расширения алгоритма на более сложные функции