МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Н. Э. БАУМАНА

ВЫЧИСЛЕНИЕ И ВЕРИФИКАЦИЯ ФОРМАТОВ ФУНКЦИЙ В РЕФАЛЕ

Выполнил:
Иванов Георгий

Научный руководитель:
Коновалов Александр Владимирович

МОСКВА, 2019

Цель и задачи

- Целью данной 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 – построить аппроксимации областей определения и областей значений функций в виде жёстких выражений. А потом проверить соответствие вызовов функций их областям определения.

РЕФАЛ

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.

В МГТУ имени Н.Э. Баумана на кафедре ИУ9 компилятор этого языка используется в качестве учебного полигона для курсовых и дипломных работ.

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:

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

Например,

$$Add\ t.X\ s.Y1\ e.Y2 = s.Z1\ e.Z2$$

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.

Например, даны функции F, G, H:

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

Тогда:

```
F { e.1 s.2= ;}
G { (e.3) e.4= ; }
H { e.0= <F e.0> <G e.0>; }
```

Алгоритм вывода типов для базисного РЕФАЛа.

- 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).

К функциям F,G,H:

```
F { e.1 s.2= ;}
G { (e.3) e.4= ; }
H { e.0= <F e.0> <G e.0>; }
```

Имеем уравнения:

$$\begin{cases} \mathrm{e.0} : \mathrm{in(G)} \\ \mathrm{e.0} : \mathrm{in(H)} \end{cases}$$

Алгоритм вывода типов для базисного РЕФАЛа.

- Начальным форматом всех функций полагаем:

$$\mathrm{in(F)} = \mathrm{out(F)} = \perp$$

где $\perp$ - некоторое специальное значение (функция, которая не возвращает никакого значения).

Обозначим:

$\mathrm{P}at(\mathrm{f}, i)$ - i-ая левая часть функции f,

$\mathrm{R}es(\mathrm{f}, i)$ - i-ая правая часть f,

$\overline{Res}(f, i)$- i-ая правая часть f с заменой вызовов функций на out(g).

Алгоритм вывода типов для базисного РЕФАЛа.

- 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.

Полный алгоритм обобщённого сопоставления описан в работах Турчина [Эквивалентные преобразования рекурсивных функций, описанных на языке РЕФАЛ,1972]. Но неполный алгоритм является упрощением полного (из-за того что мы не рассматриваем присваивания мы отбрасываем) - в нём не надо рассматривать повторные s-переменные, которые дают «чужие» переменные, а также введением t-переменным.

Алгоритм сопоставления

Дано сопоставление выражения общего вида E (которое может включать даже вызовы функций) и жёсткого выражения He:

$$E : He$$

Жёсткое выражение имеет вид:

$$Ht_1{}' \ldots Ht_n{}' \; e.X Ht_M{}'' \ldots Ht_1{}''$$

либо

$$Ht_1 \ldots Ht_n$$

где Ht — жесткие термы, $e.X$ — e-переменная. Тогда 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 сопоставления

1. Жёсткое выражение начинается на жёсткий терм ($HtHe'$), где Ht - жесткий терм.
   Если сопоставляемое выражение имеет вид: TE', где T - некий терм (символ, выражение в скобках, переменные s или t) то выполняем сопоставления соответственно: $T : Ht$ и $E' : He'$

2. Жёсткое выражение оканчивается на жёсткий терм ($He'Ht$), где Ht - жесткий терм.
   Аналогично предыдущему случаю

3. Если уравнение имеет вид $e.X\ E: Ht\ He$, где Ht - не e-переменная, то решаем две системы. В первом делаем подстановку $e.X \rightarrow \varepsilon$ (и уравнение обращается в $E: Ht\ He$), во втором - $e.X \rightarrow t.i\ e.j$ (т.е. уравнение обращается в $t.i\ e.j\ E: Ht\ He$, откуда $t.i : Ht$ и $e.j\ E: He$). Здесь $t.i$ и $e.j$ - переменные с новыми индексами.

4. Случай $E\ e.X: Ht\ He$ решается аналогично предыдущему (две системы с подстановками $e.X \rightarrow \varepsilon$ и $e.X \rightarrow e.i\ t.j$

5. В случае $E : \varepsilon$ если E имеет вид $e.1\ \dots e.N$, то получаем подстановки $e.1 \rightarrow \varepsilon, \dots, e.N \rightarrow \varepsilon$ \$. В противном случае — решений нет.

6. Жёсткое выражение состоит из e-переменной. Присваиваем этой переменной сопоставляемое выражение ($\mathrm{E} \leftarrow e.X$)

Алгоритм сопоставления

Дано сопоставление $T : Ht$. Рассматриваем 4 случая:

- Если Ht является t-переменной $t.X$, то строим присваивание $(\mathrm{T} \leftarrow t.X)$
- Если Ht является s-переменной, то T может быть только символом, s-переменной.
- Если Ht является символом (идентификатором, именем функции, числом или литерой), то T может быть тем же символом, t- или s-переменной.
- Если Ht является выражением в круглых скобках (He'), то T может быть только выражением в круглых скобках (E'). Соответственно, сопоставление выполняется рекурсивно
$$E' : He'$$

Алгоритм сопоставления:

Решим систему уравнения для функций:

F { e.1 s.2= ;}
G { (e.3) e.4= ; }
H { e.0= <F e.0> <G e.0>; }

Подставим решения:

e.0 : e.1 s.2
e.0 : (e.3) e.4

$$\left[\begin{array}{l}\begin{cases} e.0 \rightarrow \varepsilon \\ \varepsilon : e.1\ s.2 \\ \varepsilon : (e.3)\ e.4 \end{cases} \\ \begin{cases} e.0 \rightarrow e.5\ t.6 \\ t.6 : s.2 \\ e.5 : e.1 \\ e.5\ t.6 : (e.3)\ e.4 \end{cases}\end{array}\right.$$

Алгоритм сопоставления:

$$
\begin{cases}
e.0 \ \rightarrow e.5\, t.6 \\
t.6 \rightarrow s.7 \\
s.7 : \ e.2 \\
e.5 \ \leftarrow e.1 \\
e.5\, s.7 : (e.3)\, e.4
\end{cases}
\Leftrightarrow
\left[
\begin{array}{l}
\begin{cases}
e.0 \ \rightarrow e.5\, t.6 \\
t.6 \rightarrow s.7 \\
s.7 \leftarrow e.2 \\
e.5 \ \leftarrow e.1 \\
e.5 \rightarrow \varepsilon \\
s.7 : (e.3)\, e.4
\end{cases} \\
\begin{cases}
e.0 \ \rightarrow e.5\, t.6 \\
t.6 \rightarrow s.7 \\
s.7 \leftarrow e.2 \\
e.5 \ \leftarrow e.1 \\
e.5 \rightarrow t.8\, e.9 \\
t.8 : (e.3) \\
e.9\, s.7 : e.4
\end{cases}
\end{array}
\right.
\Leftrightarrow
\begin{cases}
e.0 \ \rightarrow e.5\, t.6 \\
t.6 \rightarrow s.7 \\
s.7 \leftarrow e.2 \\
e.5 \ \leftarrow e.1 \\
e.5 \rightarrow t.8\, e.9 \\
t.8 \rightarrow (e.10) \\
e.10 \ \leftarrow e.3 \\
e.9\, s.7 \leftarrow e.4
\end{cases}
$$

Следовательно:

$$e.0 \equiv e.5\, t.6 \equiv e.5\, s.7 \equiv t.8\, e.9\, s.7 \ \equiv (e.10)\, e.9\, s.7$$

Алгоритм вывода типов для базисного РЕФАЛа.

- Уточняем форматы. Строим обобщение с подстановками значений переменных.

Разберём алгоритм обобщения. Определяется образец общего вида, анализируя внешний вид отдельных образцов. К этому способу можно отнести стратегию, примененную в суперкомпиляторе SCP4 [Суперкомпилятор SCP4: Общая структура, А.П. Немытых.: 2007] и стратегию построения ГСО, реализованную ранее в Рефале-5λ.

Алгоритм обобщения

Если у нас несколько (жёстких) образцов, то смотрим на левый и правый край каждого из них:

- если все левые края описывают термы (т.е. являются символами, скобками, s- или t-переменными) и все правые края описывают термы:

- обобщаются все первые термы, обобщаются все последние термы,
- выбирается, с какой стороны сложность больше,
- если больше слева — выписываем обобщение левых термов как очередной слева терм обобщения, у всех образцов срезается первый терм,
- если справа — симметрично;

Алгоритм обобщения

- если слева у хотя бы одного из образцов есть е-переменная, а справа все края описывают термы:
- обобщаем все последние термы
- результат обобщения пишем как правый край результата,
- обрезаем у всех образцов правый терм;
- если все левые края описывают термы, а справа хотя бы у одного из образцов есть е-переменная (симметрично предыдущему случаю).
- если слева есть хотя бы у одного образца е-переменная и справа есть хотя бы одна е-переменная: результат обобщения — e;
- если все образцы пустые: результат обобщения — ε
- если есть хотя бы один пустой: результат обобщения — e

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	$Gen(P_1, P_2)$

Алгоритм вывода типов для базисного РЕФАЛа.

- Повторяем решение системы уравнений до тех пор, пока in(f) и out(f) не перестанут меняться.

Выводы

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

- Модификации алгоритма обобщения
- Расширение алгоритма на полный Рефал-5.