МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Н. Э. БАУМАНА

ВЫЧИСЛЕНИЕ И ВЕРИФИКАЦИЯ ФОРМАТОВ ФУНКЦИЙ В РЕФАЛЕ

Иванов Георгий

МГТУ им. Н. Э. Баумана

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

11 ИЮНЯ 2019 ГОДА

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 1966 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, дана функция F:

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

Тогда:

```
F {
 A = B;
 e.0 C = D <F 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 {
 A = B;
 e.0 C = D <F e.0>;
}
```

Имеем уравнения:

$$\{e.0 : in(F)$$

Алгоритм вывода типов для базисного РЕФАЛа.

- Начальным форматом всех функций полагаем:

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

где ⊥ - некоторое специальное значение (функция, которая не возвращает никакого значения).

Обозначим:

- $Pat(f, i)$ - i-ая левая часть функции f,
- $Res(f, i)$ - 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 (для вызовов в правых частях). Если у уравнения нет решений - сообщаем об ошибке. Если система имеет единственное решение - берём для переменных соответствующее значение. Если несколько, то тогда учитываем все решения.

Алгоритм обобщённого сопоставления описан в работах Турчина [Эквивалентные преобразования рекурсивных функций, описанных на языке РЕФАЛ,1972]. Рассмотрим алгоритм, где не надо рассматривать повторные s-переменные, а также введением t-переменным.

Что значит решить уравнение?

$$< F\ args > \Rightarrow args : in(F) \Rightarrow\ \exists\ s\ args|_s\ \preccurlyeq in(F)$$

Алгоритм сопоставления

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

$$E : He$$

Жёсткое выражение имеет вид:

$$Ht_1' \dots Ht_n' \; e.XHt_M'' \dots Ht_1''$$

либо

$$Ht_1 \dots Ht_n$$

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 $e.1 \ldots e.N$, то получаем подстановки $e.1 \rightarrow \varepsilon, \ldots, e.N \rightarrow \varepsilon$. В противном случае — решений нет.
6. Жёсткое выражение состоит из e-переменной. Присваиваем этой переменной сопоставляемое выражение ($\mathrm{E} \leftarrow e.X$)

Алгоритм сопоставления

T/Ht	X	$Y \neq X$	$s.X$	$t.X$	$(e.X)$
X	тождество	нет решений	$X \leftarrow s.X$	$X \leftarrow t.X$	нет решений
$s.Y$	$s.Y \rightarrow X$	$s.Y \rightarrow Y$	$s.Y \leftarrow s.X$	$s.Y \leftarrow t.X$	нет решений
$t.Y$	$t.Y \rightarrow X$	$t.Y \rightarrow Y$	$t.Y \rightarrow s.X$	$t.Y \leftarrow t.X$	$\begin{cases} t.Y \rightarrow e.N \\ e.N : e.X \end{cases}$
$(e.Y)$	нет решений	нет решений	нет решений	нет решений	$e.Y : e.X$

Алгоритм сопоставления:

Решим систему уравнения для функций:

```
F {
 A = B;
 e.0 C = D <F e.0>;
}
```

Подставим решения:

e.0 : A

$$\left[\begin{array}{l} \begin{cases} e.0 \rightarrow \varepsilon \\ \varepsilon : A \end{cases} \\ \begin{cases} e.0 \rightarrow t.1\ e.2 \\ t.1 : A \\ e.2 : \varepsilon \end{cases} \end{array}\right. \Leftrightarrow \begin{cases} e.0 \rightarrow t.1\ e.2 \\ t.1 \rightarrow A \\ e.2 \rightarrow \varepsilon \end{cases}$$

Следовательно:

$$e.0 \equiv t.1\ e.2 \equiv A$$

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[1] больше,
- если больше слева — выписываем обобщение левых термов как очередной слева терм обобщения, у всех образцов срезается первый терм,
- если справа — симметрично;

1 – числовая характеристика объектного выражения, вычисляемое по формуле $C(P) = n_t + 2n_s + 3n_X + 3n_() - n_e + 1$

Алгоритм обобщения

- если слева у хотя бы одного из образцов есть e-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 e-переменная (симметрично предыдущему случаю).
- если слева есть хотя бы у одного образца e-переменная и справа есть хотя бы одна e-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.

Алгоритм вывода типов для полного РЕФАЛа.

Пусть дано предложение $Pat, R_1 : P_1 = Res;$

1. Решаем уравнение из R_1 (уравнения для вызовов функций из R_1). Получаем набор решений, подставляем в предложения.
$$Pat', R_1' : P_1' = Res';$$

2. Ожесточаем $P1'$ - делаем из него жёсткое выражение $H[P_1']$

$H[sym\ E] = sym\ H[E]$
$H[E\ sym] = H[E]\ sym$
$H[st.var\ E] = st.var\ H[E]$
$H[E\ st.var] = H[E]\ st.var$
$H[(E')\ E''] = (H[E'])\ H[E'']$
$H[E'(E'')] = H[E'](H[E''])$
$H[e.index] = e.index$
$H[\varepsilon] = \varepsilon$
$H[E] = e.new$

Случай с повторными переменными не рассматриваем.

Алгоритм вывода типов для полного РЕФАЛа.

Решаем уравнение

$$\overline{R_1}' : H[P_1']$$

Получаем набор сужений и присваиваний. Далее, подставляем в предложения.

$$Pat'', R_1'' : P_1'' = Res'';$$

После, используем алгоритм вывода для базисного РЕФАЛа.

Алгоритм вывода типов для полного РЕФАЛа.

Пусть дано предложение

$$Pat, R_1 : P_1, Res : \{ P_{11} = R_{11}; P_{21} = R_{21}; \}$$

Преобразуем его в предложения с условиями:

$$Pat, R_1 : P_1, Res : P_{11} = R_{11};$$
$$Pat, R_1 : P_1, Res : P_{21} = R_{21};$$

Далее, применяем алгоритм вывода для полного РЕФАЛа (условия).

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 40 тестах, написанных на Базисном РЕФАЛе и РЕФАЛе-5. В дальнейших планах исследования на тему:

- Модификации алгоритма обобщения
- Оптимизация производительности

Как установить?

```
pip3 install refalchecker
```