Типизация функций для Рефала-5

Дмитрий П. Сырбу

МГТУ имени Н. Э. Баумана

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

8 Июня 2021 года

Постановка задачи

Целью данной работы является описания подмножества КС-грамматик замкнутое относительно теоретико-множественных операций, в рамках которого выразимо описание типов функций языка Refal-5 и сопоставление с образцом.

Будет предложена процедура нормализации грамматики, а также описан процесс верификации типов.

Нотация типов

Типы описываются регулярными языками.

Правила грамматики имеют вид:

- *< ⟨имя-функции⟩ ⟨типовое-выражение⟩> ::= ⟨типовое-выражение⟩*
- *⟨имя-типа⟩ ::= ⟨типовое-выражение⟩*

где *⟨имя-типа⟩* — переменная вида s-, t- или e-,

а *⟨типовое-выражение⟩* — выражение РБНФ, построенное из символов Рефала, переменных, круглых скобок, фигурных скобок {, }, знака | и знаков *, +, ?.

e.LongNumber ::= s.Sign? s.NUMBER+

s.Sign ::= '+' | '-'

Нотация типов

Типы, описанные при помощи e-, t- и s-переменных, мы будем называть, соответственно, e-типами, t-типами и s-типами.

Правая часть в правиле должна описывать синтаксическое подмножество значений типа слева.

- У e-типов справа может быть записано произвольное выражение.
- Правило для t-типов должно иметь вид:
  t.Name ::= ⟨терм⟩ | ⟨терм⟩ ... ; где *⟨терм⟩* — символ, произвольное выражение в скобках, s- или t-тип.
- Для s-типов в правой части правила должен быть набор альтернатив, включающий только символы и другие s-типы.

Нормальная форма грамматики

- Все типы грамматики объединены в тройки: для каждого типа, описываемого переменной с некоторым индексом, существуют связанные с ним два типа других видов с тем же индексом. Тройка типов с одинаковым индексом объединена следующим соотношением:

$$t.SomeType ::= s.SomeType \mid (e.SomeType)$$

- Типовое выражение для s-типа может быть либо @, либо перечислением константных символов (не может включать s-переменные), символы в правиле не должны повторяться.
- Типовое выражение для e-типа может быть либо @, либо состоять только из t-переменных, фигурных скобок, знаков | и *.

Преобразование грамматики к нормальной форме

- Устранение квантификаторов ? и +.
- Нормализация правил для s-типов и упрощение правил для t-типов.
- Построение троек для s- и e-типов.
- Нормализация и построение троек для t-типов.
- Нормализация для e-типов.

Пример нормализации

$e.A ::= '+'\ s.CHAR*$

$t.A ::= s.A \mid (e.A)$

$s.A ::= @$

$e.A ::= t.['+'\ ]\ t.CHAR*$

$t.['+'\ ] ::= s.['+'\ ] \mid (e.['+'\ ])$

$s.['+'\ ] ::= '+'$

$e.['+'\ ] ::= @$

Ортогональная нормальная форма грамматики типов (ОНФ)

Типы в ОНФ делятся на две группы — набор исходных типов, (*SRC*) т.е. типы, которые были в исходной грамматике до нормализации, и набор ортогональных типов, (*ORTHO*) построенный в ходе процедуры нормализации.

Типы из *ORTHO* записаны в НФ — s-типы представляют собой множества символов, e-типы — регулярные языки в алфавите t-типов из *ORTHO*, t-типы — как объединение s-типа и скобочного терма с e-типом внутри.

Тройки типов из *ORTHO* попарно ортогональны.

Типы из *SRC* описаны как объединения типов соответствующего вида из *ORTHO*.

Ортогональные типы

Генерируются 2^N различных строк из знаков 0 и 1 длиной N. Это будут индексы типов из набора *ORTHO*. Набор *ORTHO* должен быть описан в нормальной форме, т.е. быть набором троек. Поэтому каждый из этих новых индексов будет индексом и s-, и t-, и e-типа. Ортогональные t-типы записываются как

$$t.\langle ortho\rangle ::= s.\langle ortho\rangle \mid (e.\langle ortho\rangle)$$

Каждый тип из *ORTHO* представляет собой пересечение N множеств, каждое из этих множеств — либо множество значений типа данного вида из некоторой исходной тройки, либо дополнение типа исходной тройки до универсума. Знак 1 в i-й позиции индекса означает, что i-е множество совпадает с типом соответствующего вида исходной тройки, знак 0 — что берётся дополнение этого типа тройки до универсума соответствующего вида.

Ортогональные типы

Каждой тройке в *SRC* назначаем номер, начиная с 1. Будем считать, что у нас получилось N троек.

Получив пространство имён *ORTHO,* типы из *SRC* представляются объединение типов соответствующего вида из ORTHO с 1 в i-й позиции, где i — номер тройки данного типа:

$$t.\langle i\rangle ::= \ldots \mid t.\langle \ldots\rangle 1 \langle \ldots\rangle \mid \ldots$$

$$s.\langle i\rangle ::= \ldots \mid s.\langle \ldots\rangle 1 \langle \ldots\rangle \mid \ldots$$

$$e.\langle i\rangle ::= \ldots \mid e.\langle \ldots\rangle 1 \langle \ldots\rangle \mid \ldots$$

Процедура ортогонализации регулярной грамматики типов

Построение нормальной формы и нумерация троек.

1. Построение набора имён ортогональных типов, запись ортогональных t-типов.
2. Выражение исходных t-типов через ортогональные.
3. Вычисление значений ортогональных s-типов.
4. Запись исходных e-типов в алфавите ортогональных типов.
5. Вычисление значений ортогональных e-типов.
6. Выражение исходных s- и e-типов через ортогональные.

Теорема о расширении ОНФ

Пусть нам дана грамматика в нормальной форме и пусть мы можем разбить её тройки на два множества: U и V, причём регулярные языки, описываемые e-типами из U, не содержат строк, содержащих t-типы из V.

Тогда построение ОНФ для грамматики $U \cup V$ можно разбить на несколько этапов:

1. Построение ОНФ для U, обозначим компоненты этой ОНФ как SRC-U и ORTHO-U.
2. Выражение e-типов из V в терминах t-типов из V и ORTHO-U.
3. Построение ОНФ для $U \cup V$ путём расщепления типов из ORTHO-U.

Инкрементное построение ОНФ

Теорема о расширении ОНФ позволяет нам построить эффективный алгоритм для построения ОНФ.

Для этого нужно построить НФ для исходной грамматики, множество троек разбить на подмножества U1, U2, …, Uk, таким образом, чтобы для $i < j$ типы из Ui не зависели от типов Uj.

Затем построить ОНФ для U1 и последовательно её расширять процедурой из теоремы.

Постановка задачи простой верификации

Дана программа на ограниченном Рефале, дана грамматика типов и для каждой функции программы описан её тип в терминах типов грамматики. Алфавит допустимых символов в поле зрения конечен и образован теми символами, которые явно представлены константами либо в программе, либо в грамматике типов, либо в типах аргументов и результатов функций.

Требуется для каждого вызова функции проверить, что фактический тип её аргумента является подмножеством её формального типа аргумента и что фактический тип каждой правой части функции является подмножеством формального типа результата этой функции.

Решение простой задачи верификации типов

Общий вид алгоритма верификации можно описать в следующем виде:

1. Исходные данные подвергаем предварительной нормализации.

   $<\langle \text{Func} \rangle\ \text{e.Arg}[\langle \text{Func} \rangle] > == \text{e.Res}[\langle \text{Func} \rangle]$

2. В результате аргумент каждой функции будет представлен как объединение нескольких непересекающихся типов (по определению ОНФ).

   $\text{e.Arg}[\langle \text{Func} \rangle] ::= \text{e.}\langle \text{ortho1} \rangle \mid \ldots \mid \text{e.}\langle \text{orthoN} \rangle$

3. Решение задачи сводится к сопоставлению регулярного языка $\text{e.}\langle \text{orthoi} \rangle$ с образцом предложения и проверки вызовов в результатной части.

Сопоставление e-типа из ORTHO с образцом Рефала

В процессе сопоставления мы будем рассматривать системы уравнений вида:

$$RegLang : Pat,$$

где *RegLang* является регулярным языком, *Pat* — L-выражением.

Решение в процессе будет разветвляться. Если некоторая ветвь решений не имеет, то она отсекается. На концах ветвей мы будем иметь решения.

Решение представляет собой отображение переменных образца в типы ОНФ.

Проверка правой части для некоторого решения

Входными данными для процедуры проверки являются грамматика типов, отображение переменных из образца на их значения и результатное выражение — правая часть предложения.

Требуется построить фактические типы аргументов функций и фактический тип правой части и проверить вложение этих типов в формальные типы — соответственно, типы аргументов вызова и тип результата самой функции.

Проверять вложение мы можем только для типов одной грамматики, представленной в ОНФ. Поэтому нам нужно типы аргументов и тип правой части добавить к исходной грамматике и повторно выполнить процедуру нормализации.

Правила верификации

$\langle Q, qs, \delta, F \rangle : T\ E \longrightarrow \begin{cases} t.X : T \\ \langle Q, q', \delta, F \rangle : E \end{cases}$

$\langle Q, qs, \delta, F \rangle : E\ T \longrightarrow \begin{cases} \langle Q, qs, \delta, \{ q'' \} \rangle : E \\ t.X : T \end{cases}$

$t.X : t.Y \longrightarrow t.Y \rightarrow t.X$

$t.X : s.Y \longrightarrow s.Y \rightarrow s.X$

$t.X : (E) \longrightarrow e.X \rightarrow E$

$\langle Q, qs, \delta, F \rangle : e.Y \longrightarrow e.Y \rightarrow \langle Q, qs, \delta, F \rangle$

Заключение