УДК 519.681, 519.683, 519.712
AMS MSC2020: 03B70, 68N15

Устранение рекурсии в полуинтерпретированных схемах программ

Шилов Н. В.

Университет Иннополис

Аннотация. В докладе представлен обзор результатов, полученных автором с начиная с 2010 г., по эффективному пребразванию («трансляции») паттернов рекурсивных программ (рекурсивных схем с неинтерпретированными или только частично интерпретированными функциональными и предикатными символами) в функционально эквивалентные стандартные схемы программ, то есть блок-схемы итеративных программ с теми же самыми неинтерпретированными или только частично интерпретированными функциональными и предикатными символами.

Ключевые слова: примитивно рекурсивные функции, рекурсивные функции, стандартные схемы программ, обогащенные схемы программ, рекурсивные схемы, функциональная эквивалентность схем программ.

Введение

Примитивно рекурсивные функции — это минимальный класс функций на натуральных числах, который получается[1] из функции-константы 0, одноместной функции следования +1 и проекции (выбора элемента кортежа) при помощи операторов суперпозиции (композиции функций) и примитивной рекурсии

$$\begin{aligned} h(x_1, \dots x_n, y) \ = \ & if\ y = 0\ then\ f(x_1, \dots x_n) \\ & else\ g\Big(x_1, \dots x_n, h\big(x_1, \dots x_n, (y-1)\big)\Big). \end{aligned}$$

[1] Внимание: стандартная нотация для базисных функций и операторов не соблюдена!

Класс частично рекурсивных функций определяется аналогично классу примитивно рекурсивных, только к двум операторам (суперпозиции и примитивной рекурсии) добавляется еще оператор минимизации аргумента: $h(x_1, \ldots x_n) = \arg\min\ y: f(x_1, \ldots x_n, y) = 0$.

Будем говорить, что один (синтаксически определенный) класс функций[2] транслируем в некоторый другой (синтаксически определенный) класс функций[3], если любая функция из первого класса (функционально) эквивалентна некоторой функции из второго класса[4].

Разумеется, задача распознания по описанию частично рекурсивной функции ее «транслируемость» в класс примитивно рекурсивных функций является неразрешимой. Однако, исследование синтетически определенных программных паттернов, которые часто возникают при задании частично рекурсивных функций, и которые транслируются в итеративные программы (соответствующие примитивно рекурсивным функциям) вызывала [4] и по-прежнему вызывает интерес [3] в теории программирования и практике оптимизирующих компиляторов [2].

1. Рекурсивное динамическоое программирование

Динамическое программирование было введено Ричардом Беллманом в 1950-х годах для решения задач оптимального планирования. Уравнение Беллмана — это название рекурсивного функционального уравнения для целевой функции, которое выражает оптимальное решение в «текущем» состоянии через оптимальные решения в «достижимых за один шаг» состояниях, оно формализует так называемое Принцип оптимальности Беллмана: оптимальная программа (или план) остается оптимальной на каждом этапе. Мы изучаем класс уравнений Беллмана, который соответству-

[2]Например, класс частично рекурсивных функций.

[3]Например, класс примитивно рекурсивных функций.

[4]Здесь эквивалентность означает, что обе синтаксически определенные функции вычисляют одну и ту же функцию.

ет следующему рекурсивному шаблону:

$$\begin{aligned} G(x) \; = \; & \mathit{if}\ p(x)\ \mathit{then}\ f(x) \\ & \qquad\qquad \mathit{else}\ g\Big(x,\ \Big\{h_i\big(x, G(t_i(x))\big),\ i \in [1..n(x)]\Big\}\Big) \end{aligned} \tag{1}$$

Мы рассматриваем этот шаблон как *рекурсивную программную схему* [1], то есть рекурсивную структура управления с *неинтерпретируемыми символами*:

- G — определяемый функциональный символ, представляющий (после интерпретации *базисных* функциональных и предикатных символов) целевую функцию $G : X \to Y$ для подходящих множеств X и Y;

- p — базисный предикатный символ, представляющий (после интерпретации) некоторый *известный*[5] предикат $p \subseteq X$;

- f — базисный функциональный символ, представляющий (после интерпретации) некоторую *известную*[5] функцию (операцию) $f : X \to Y$;

- g — базисный функциональный символ, представляющий (после интерпретации) некоторую *известную*[5] функцию (операцию) $g : X \times Z^* \to X$ для подходящего множества Z (но переменной местности[6] $n(x) : X \to \mathbb{N}$);

- все h_i и t_i ($i \in [1..n(x)]$) — базисные функциональные символы, представляющие (после интерпретации) некоторые *известные*[5] функции $h_i : X \times Y \to Z$, $t_i : X \to X$ ($i \in [1..n(x)]$).

В дальнейшем мы не будем делать явного различия в обозначениях для символов и интерпретируемых символов, а будем писать/говорить, например, *символ* g и/или *функция* g.

[5] То есть который/которую/которые уже умеем вычислять.
[6] То есть, фактически, от *списка* аргументов.

2. Основной результат

Теорема 1. *Предположим, что интерпретация базисных предикатных и функциональных символов в рекурсивной схеме (1) удовлетворяет следующим свойствам:*

- *количество аргументов $n : X \to \mathbb{N}$ — это некоторая константа $n \in \mathbb{N}$;*
- *все функции $t_1, \ldots\ t_n$ имеют обратные и $t_i = (t_1)^i$ для всех $i \in [1..n]$;*
- *предикат p является t_1-замкнутым, то есть из $p(u)$ следует $p(t_1(u))$ для всех $u \in X$.*

Пусть $m \in \mathbb{N}$ — число переменных достаточное для вычисления всех базисных предиката и функций p, f, h_i $(i \in [1..n])$, t_1 и t_1^-. Тогда целевая функция G может быть вычислена в этой же интерпретации некоторой итеративной программой (стандартной схемой), использующей $2n + m + 2$ переменных.

Доказательство. Подробности даны в [5], а здесь мы ограничимся только псевдокодом эквивалентной (полуинтерпретированной) схемы итеративной программы:

$$
\begin{array}{l}
var\ x, x_1, \ldots x_n :\ X;\\
var\ y, y_1, \ldots y_n :\ Y;\\
x := v;\\
if\ p(x)\ then\ y := f(x)\\
\quad else \left\{ \begin{array}{l}
do\ x := t_1(x)\ until\ p(x);\\
x_1 := x;\ x_2 := t_1(x_1);\ \ldots;\ x_n := t_1(x_{n-1});\\
y_1 := f(x_1);\ y_2 := f(x_2);\ \ldots;\ y_n := f(x_n);\\
do\ x := t_1^-(x);\\
\qquad \text{// Annotation: } x = t_1^-(x_1)\ \&\ bas(x) = \{x_1, \ldots\ x_n\}\ \&\\
\qquad\qquad \text{// } \&\ y_1 = G(x_1)\ \& \ldots \&\ y_n = G(x_n)\\
\qquad y :=\ g\Big(x,\ \big(h_1(x, y_1),\ \ldots,\ h_n(x, y_n)\big)\Big);\\
\qquad y_n := y_{n-1};\ \ldots;\ y_3 := y_2;\ y_2 := y_1;\\
\qquad y_1 := y;\\
\qquad x_1 := t_1^-(x_1);\ \ldots;\ x_n := t_1^-(x_n)\\
until\ x = v
\end{array} \right\}.
\end{array}
$$

□

Заключение

В работе [5] можно найти обзор исследований по устранению рекурсии в интерпретированных программах, использованию (однократно выделяемых в динамической памяти) массивов для устранения рекурсии в рекурсивной схеме (1), примеры олимпиадных задач по математике и программированию, основанные на рекурсивной схеме (1).

Некоторые вопросы и направления для дальнейших исследований представлены ниже.

- Доказать с использованием компьютерных инструментов автоматического доказательства Теорему 1 (и другие утверждения из работе [5]).
- Исследовать, как обобщить рекурсивный шаблон (1) и условия Теоремы 1 таким образом, чтобы сохранить устранение рекурсии.
- Разработать и реализовать плагин для некоторой IDE (интегрированной среды разработки), который анализирует программный код для поиска рекурсивных шаблонов, допускающих устранение рекурсии.

Список литературы

[1] *Котов, В. Е.* Теория схем программ / В. Е. Котов, В. К. Сабельфельд. — М. : Наука, 1991. — 274 с.

[2] *Легалов, А. И.* Преобразование хвостовых рекурсий в функционально-потоковых параллельных программах / А. И. Легалов, О. В. Непомнящий, И. В. Матковский, М. С. Кропачева // Моделирование и анализ информационных систем. — 2012. — Т. 19, № 4. — С. 48–58.

[3] *Шилов, Н. В.* Этюд об устранении рекурсии // Модел. и анализ информ. систем. — 2018. — Т. 25, № 5. — С. 549–560.

[4] *Knuth, D. E.* Textbook Examples of Recursion. — 1991. — 18 p. — URL: arXiv:cs/9301113. — Загл. с титул. экрана.

[5] *Shilov, N. V.* Teaching Efficient Recursive Programming and Recursion Elimination Using Olympiads and Contests Problems / N. V. Shilov, D. Danko // Frontiers in Software Engineering Education — First Int. Workshop, FISEE 2019. Lecture Notes in Computer Science. — V. 12271. — Springer, 2020. — P. 246–264.

Библиографическая ссылка

Шилов, Н. В. Устранение рекурсии в полуинтерпретированных схемах программ // Всероссийская научная конференция «Математические основы информатики и информационно-коммуникационных систем». Сборник трудов. — Тверь : ТвГУ, 2021. — С. 85–90.
https://doi.org/10.26456/mfcsics-21-13

Сведения об авторах

Шилов Николай Вячеславович
Университет Иннополис. Доцент

Россия, 420500, г. Иннополис, ул. Университетская, д. 1
E-mail: shiloviis@mail.ru