О суперкомпиляции

(к 80-летию со дня рождения В. Ф. Турчина)

Андрей П. Немытых
Институт программных систем РАН
г. Переславль-Залесский

“Современные проблемы математики, информатики и биоинформатики”
Конференция к 100-летию со дня рождения А. А. Ляпунова
11 - 14 октября 2011 г., Академгородок, Новосибирск

2011

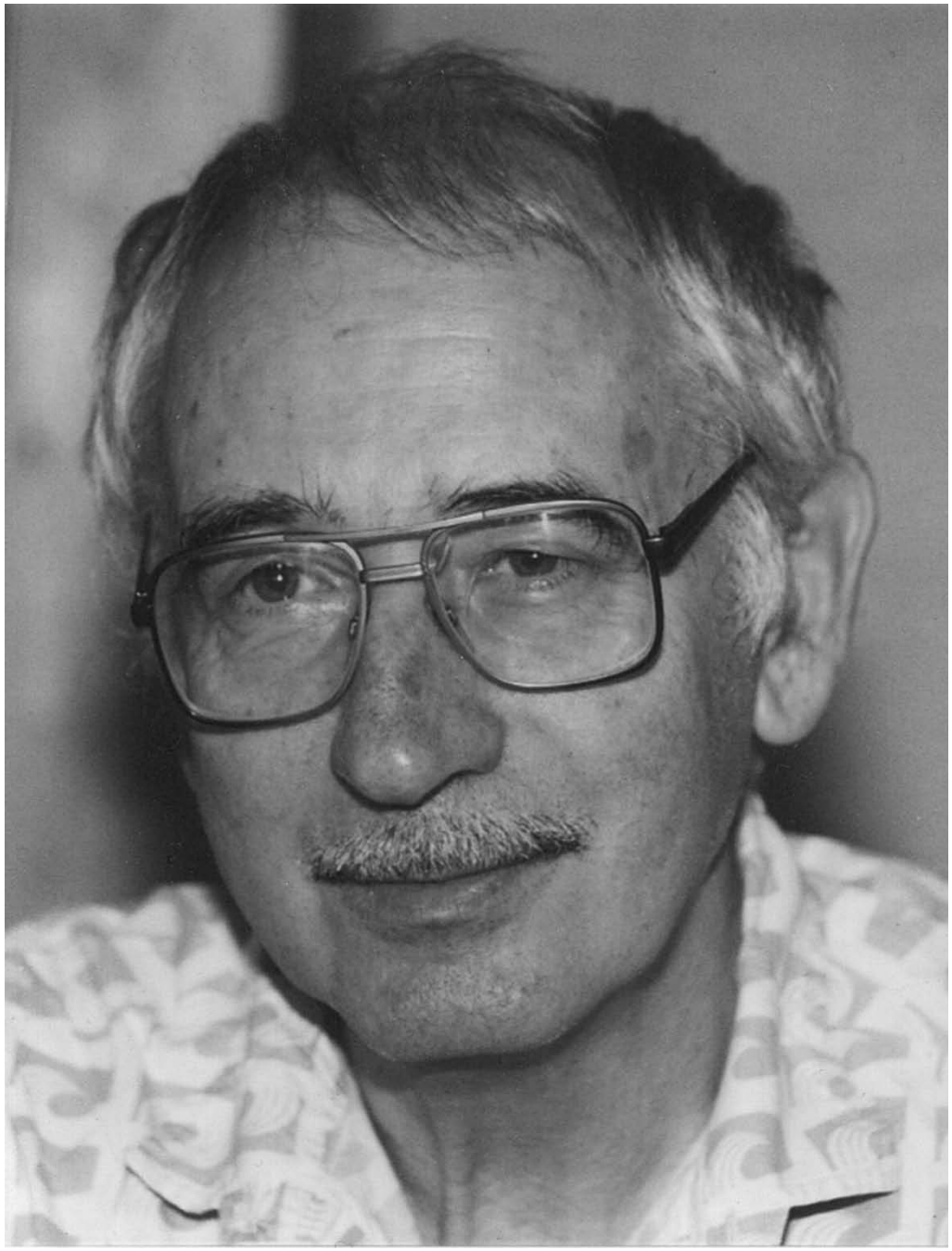
Мстислав Всеволодович Келдыш
(1911 - 1978)

и его сотрудники

Алексей Андреевич Ляпунов
(1911 - 1973)

Валентин Фёдорович Турчин
(1931 - 2010)

2011



Валентин Фёдорович Турчин
(1931-2010)

Исследования в области построения систематических методов специализации программ по отношению к фиксированным свойствам их аргументов, композиционной структуре этих программ и заданным инвариантам были начаты в 1970-х годах

- **А. П. Ершов, смешанные вычисления;**
- **В. Ф. Турчин, суперкомиляция;**
- **Ё. Футамура, generalized partial computation.**

В настоящее время, наряду с несколькими примитивными моделями суперкомпиляторов для простейших чисто теоретических языков, существует единственный

экспериментальный суперкомпилятор
SCP4

для реального языка программирования
РЕФАЛ-5

Пусть дана реализация функционального языка программирования $\langle$P,D,U,T$\rangle$, где D $= \bigcup_{n\in\mathbb{N}} M^n$.

Задача №1 $\hookrightarrow$

Программа p(x,y) $\in$ P, реализует *частично рекурсивную функцию* $F(x,y)$: D $\times$ D $\mapsto$ D. Для данного $x_0 \in$ D требуется построить программу q(y) $\in$ P :

$$\forall\ y{\in}D\ (q(y) = p(x_0,y)) \wedge (T(q,y) \leq T(p,x_0,y))$$

где $\exists$ q(y) $\Leftrightarrow$ $\exists$ p(x_0,y).

$$q(y) = p(x_0,y) = F(x_0,y)$$

Задача №2↪

Программа p(x,y) реализует *общерекурсивную функцию*

$$F(x, y) : X \times Y \mapsto \mathbf{D}, \text{ где } X \subset \mathbf{D}, Y \subset \mathbf{D}.$$

Для данного $x_0 \in X$ требуется построить программу q(y) $\in$ P :

$$\forall\, y \in Y.(q(y) = p(x_0, y)) \wedge (T(q, y) \leq T(p, x_0, y))$$

Программа q представляет некоторое *продолжение общерекурсивной функции* $F(x_0, y) : Y \mapsto$ D по второй компоненте.

Содержательная сторона задачи состоит в построении оптимальной q (по времени исполнения).

Методы суперкомпиляции ориентированы на решение второй задачи.

Н. Д. Джонс (Дания) предложил ослабить

- **первоначально поставленные цели за счёт упрощения методов специализации (“partial evaluation”);**

- **подходы к решению задачи.**

Эта упрощённая техника частичных вычислений наиболее разработана к данному моменту. Она решает первую задачу специализации.

Самоприменение частичного вычислителя mix (Н.Д. Джонс, П.Сестофт, Х.Сондергаад, 1983)

∃ **функция времени** T, делающая любую задачу специализации p(x_0,y) тривиальной. Получаем остаточную программу:

$$\text{q(y) } \{ = \text{p}(x_0,\text{y}); \}$$

Длина результата q(y) решения задачи самоприменения:

$$\text{mix}(\underline{\text{mix}(\text{p}_0,x,\text{y})})$$

mix по данной 3-строчной программе $\text{p}_0(x,\text{y})$, была 500 страниц текста.

Offline специализация **разделяет в отдельные стадии анализ (BTA) программы p и метаинтерпретацию локальных шагов p.**
↪

- **На входе BTA: p и указание – какие из её аргументов будут известны (s) на второй стадии и какие неизвестны (d).**

- **Выход: p^{ann}, где каждое элементарное действие помечено как s, если его можно вычислить без знания *конкретных* значений d части его входов. Аргументы действий размечаются как s и d.**

Группа Н. Д. Джонса, как и С. А. Романенко, решила не оригинальную классическую задачу самоприменения

$$\mathbf{mix}(\underline{\mathbf{mix}(\mathbf{p}_0,x,\mathbf{y})})$$

а задачу

$$\mathbf{mix}(\underline{\mathbf{mix}^{\mathrm{ann}}(\mathbf{p}_0^{\mathrm{ann}},x,\mathbf{y})})$$

Важно: входные данные (как статические, так и динамические) каждого шага программы просматриваются только один раз на этапе “специализации” (второй стадии).

Online специализация **производит метавычисление шагов p по ходу дела анализа её свойств, не ограничивая *a priori* себя ни в средствах, ни в количестве проходов по p.**

- **каждый проход даёт цикл: при самоприменении он в остаточной программе;**

- **проходы по данным, анализирующие являются ли они s или d, наблюдаются преобразующей копией специализатора.**

Алгоритмически неразрешимая и разрешимая части логики не разделены.

Резюме: разработка online специализации сложнее offline специализации; online специализация менее ограничена в методах и, потому, потенциально более сильная.

Важно: при offline происходят преобразования только исходной программы p; а при online *могут происходить также преобразования подпрограмм, построенных специализатором*, а не только подпрограмм программы p, написанных человеком.

Суперкомпиляция и “generalized partial computation”

– дают более сильные механизмы автоматического анализа и преобразования программ, чем частичные вычисления;

– ставят много более трудные задачи: познавательные, алгоритмические и технологические;

– позволяют понижать порядок временно́й сложности специализируемых программ;

SCP1: Простейшая модель суперкомпилятора рефала (В.Ф. Турчин, Б. Ниренберг, Д.В. Турчин, 1981 г.)

- работал в диалоговом режиме, запрашивая результат обобщения конфигураций;
- задача аппроксимации алгоритмически неразрешимой части логики осталась за скобками рассмотрения;
- шлифовка алгоритма прогонки – метавычисление вызовов по необходимости на этапе суперкомпиляции;
- решал *Задачу №2* специализации.

SCP2 разработан В.Ф. Турчиным в 1984 г.

- **“The concept of a supercompiler”, публикация 1986 г.;**

- **в языке параметризованных конфигураций появился связка отрицания, что позволило решить методами суперкомпиляции задачу преобразования наивного алгоритма поиска подстроки s в строке x в алгоритм KMP – специализация $p(s_0,x)$;**

- **обобщение работало *ad hoc*, для получения более-менее интересных преобразований требуется участие человека.**

- вторая половина 80-х гг.: А. Веденов;

- 1987 г.: препринты “Рефал-4 – расширение рефала-2, обеспечивающее выразимость прогонки” (С.А. Романенко), “Метавычислитель для языка рефал, основные понятия и примеры” (Анд. В. Климов, С.А. Романенко);

- 1988 г.: В.Ф. Турчин “The algorithm of generalization in the supercompiler” – алгоритм обобщения стека вызовов функций; и доказана теорема об его остановке;
  SCP2 был расширен этим алгоритмом.

В 1989 г. первая попытка самоприменения суперкомпилятора (публикация в 1990 г., В.Ф. Турчин).

Базисные конфигурации **(БК): обобщённые конфигурации программы p, в терминах которых можно описать результат её суперкомпиляции.**

- **в ручную построены конечные множества БК для нескольких простых задач самоприменения;**

- **эти БК обеспечивали конечность процесса суперкомпиляции на данных задачах без использования алгоритма обобщения.**

Удалив алгоритм обобщения и сообщая на входе SCP2 для каждой задачи множество базисных конфигураций (ей соответствующее) проведены удачные эксперименты с простыми задачами самоприменения.

Т.е. достигнуто самоприменение алгоритмически разрешимой части задачи специализации – без участия аппроксимирующего алгоритма обобщения.

– **1990 г.: Н.В. Кондратьев предпринял попытку реализации суперкомпилятора рефала, которая осталась незавершённой;**

– **1992 г.: аналогичная попытка С.М. Абрамова и Р.Ф. Гурина – они разрабатывали суперкомпилятор для модельного языка, работающего с данными LISPа; основная трудность, которая не была преодолена, состояла в разработке алгоритма построения выходных форматов функций.**

SCP3: В 1993 г. В.Ф. Турчин ограничил язык “плоским” алгоритмически полным фрагментом рефала: не допускаются явные синтаксические конструкции композиции вызовов функций и синтаксис образцов гарантирует, что время отождествления равномерно ограничено по размеру входных данных.

- основная цель – достижение полностью автоматического самоприменения;

- введение дополнительной типизации параметров, позволяющей более точно формулировать задачи на самоприменение;

- **1994 г.: достигнуто полностью автоматическое самоприменение SCP3 на нескольких простых задачах;**

- **публикация 1996 г.: В.Ф. Турчин, А.П. Немытых, В.А. Пинчук "A Self-Applicable Supercompiler";**

- **алгоритм обобщения плоских конфигураций не имел под собой прочной теоретической основы, хотя и работал *полностью автоматически*;**

- **1995 г.: М. Соренсен предложил использовать отношение Хигмана-Краскала для принятия решения: “Обобщать или не обобщать две данные конфигурации?”;**

- **1995 г.: С.М. Абрамов предложил алгоритм обобщения части конфигурации, описанной со связкой отрицания;**

- **1996 г.: М. Соренсен, Р. Глюк и Н.Д. Джонс – модельный суперкомпилятор для простейшего подмножества LISPа: не использует связки отрицания.**

SCP4: экспериментальный суперкомпилятор для языка рефал-5 (1999-2003, А.П. Немытых под руководством В.Ф. Турчина).

- **построение выходного формата функции F по ходу дела суперкомпиляции;**
- **SCP4 экспериментальный свободно распространяемый суперкомпилятор, который доступен также в режиме online;**
- **2007 г.: монография А.П. Немытых "Суперкомпилятор SCP4: Общая структура".**

- **2008-2010 гг.: реализованы простые суперкомпиляторы для модельных языков высшего порядка (H. Митчел и K. Рансайман (Англия), П.А. Джонсон и Й. Нордландер (Швеция), И.Г. Ключников);**

- **в них не реализован ключевой Обнинский алгоритм обобщения параметризованных стеков;**

- **некоторые понятия суперкомпиляции адоптированы к функциям высших порядков.**

Суперкомпиляция и метод частичных вычислений

Пусть дано некоторое конечное множество элементарных преобразований программ

$$\{q_1, q_2, \ldots, q_n\}$$

Суперкомпилятор жонглирует ими с целью оптимизации данной программы P.

Пусть:

- $\{q_1, q_2, \ldots, q_m\}$ **ответственны за обобщение параметризованных конфигураций программы P;**

- $\{q_{m+1}, q_{m+2}, \ldots, q_n\}$ **используется для метаинтерпретации шагов P;**

BTA-алгоритм есть аналог алгоритма обобщения. Сущность идеи Н. Д. Джонса:

- манипулировать преобразованиями $\{q_1, q_2, \ldots, q_m\}$ только посредством BTA-анализа;
- и результат этих манипуляций должен быть подан на вход второй стадии преобразований, которая использует только $\{q_{m+1}, q_{m+2}, \ldots, q_n\}$.

Такое разделение приводит к катастрофическим последствиям.

Применим идею Н. Д. Джонса к базисным операциям машины Тьюринга

$$\{t_1, \ldots, \text{move}_{to\_left}, \text{move}_{to\_right}, \ldots, t_k\} \quad \textbf{(I)}$$

и разобьём этот набор операций на два:

$$\{t_1, \ldots, \text{move}_{to\_left}\} \text{ и } \{\text{move}_{to\_right}, \ldots, t_k\} \quad \textbf{(II)}$$

Нужно сначала манипулировать только операциями (I) и только после этого можно использовать операции из (II).

Манипулируя всем набором можно построить любой алгоритм. Что можно запрограммировать, используя подход Н. Д. Джонса?

Благодарю за внимание!