

На правах рукописи

Немытых Андрей Петрович

СПЕЦИАЛИЗАЦИЯ ФУНКЦИОНАЛЬНЫХ ПРОГРАММ МЕТОДАМИ
СУПЕРКОМПИЛЯЦИИ

05.13.17 — Теоретические основы информатики

Автореферат

диссертации на соискание учёной степени
кандидата физико-математических наук

Переславль-Залесский — 2007

Работа выполнена в Институте программных систем РАН.

Научный руководитель: доктор физико-математических наук,
член-корреспондент РАН
Абрамов Сергей Михайлович

Официальные оппоненты: доктор технических наук,
профессор
Хорошевский Владимир Фёдорович,
ВЦ РАН

кандидат физико-математических наук,

Романенко Сергей Анатольевич,
ИПМ им. М.В.Келдыша РАН

Ведущая организация: Институт проблем передачи информации
им. А. А. Харкевича РАН

Защита состоится “ ____ ” _____ 2007 в ____ ч. ____ мин. на заседании
Диссертационного совета Д 002.084.01 в Институте программных систем РАН
по адресу: 152020, г. Переславль-Залесский, Ярославской области, Институт
программных систем РАН.

С диссертацией можно ознакомиться в библиотеке Института программных
систем РАН.

Автореферат разослан “ ____ ” _____ 2007.

Учёный секретарь Диссертационного совета
кандидат технических наук

Пономарёва С.М.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. На начальных этапах развития технологии программирования программисту предлагалось мыслить в «аппаратных» понятиях, т.е. понятиях непосредственно связанных со структурой того устройства, на котором программа должна была исполняться. Однако, по мере развития технологии программирования, акцент стал смещаться в сторону понятий, связанных не с аппаратурой, а с сущностью задачи, которую должна решать программа. Это, в частности, выразилось в том, что начали появляться языки программирования высокого уровня, позволяющих адекватно отражать объектную область задачи. К таким языкам, например, относятся функциональные и логические языки (LISP, REFAL, PROLOG, HASKELL, ML, SCHEME и др.), а также различные языки, специализированные на конкретную область их применения. С другой стороны, аппаратная реализация современных широко используемых ЭВМ поддерживает фон-неймановскую модель вычислений; что приводит к неэффективной реализации таких языков – посредством интерпретации – более того, часто не прямой, а косвенной – через другую интерпретацию. К подобной неэффективности приводит и любое структурное программирование само по себе; ибо его целью является создание гибких, легко понимаемых и изменяемых программ. Всё чаще программы вычисляются другими программами, а потому естественно ожидать, что первые будут содержать простейшие структуры, ведущие к накладным расходам, которые никогда бы не допустил квалифицированный программист.

Методы автоматической оптимизации структурированных программ высокого уровня (а не программ отшлифованных профессиональными программистами на языках программирования низкого уровня) и призваны предоставить свободу развития новым технологиям программирования.

Одним из активно развивающихся здесь направлений является автоматическая специализация программ. Предположим, что вы купили дистрибутив операционной системы LINUX. В момент её установки на

вашем компьютере вы должны указать его аппаратные характеристики, то есть эти характеристики являются аргументами программы-установщика. Возникает желание максимально настроить LINUX на ваше железо, ибо в другом контексте он вам не понадобится. В этом и состоит задача специализации. Операционную систему вы устанавливаете однажды и, потому, стоит предложить поработать автоматическому специализатору, даже если его работа достаточно продолжительна во времени.

Суперкомпиляция есть набор методов автоматической специализации программ, написанных на функциональных языках.

Идеи лежащие в основе суперкомпиляции, с одной стороны, существенно мощней идей широко распространённой техники специализации, известной как частичные вычисления; с другой стороны, методы суперкомпиляции намного менее разработаны, чем методы частичных вычислений. И, до недавнего времени, не существовало ни одного экспериментального специализатора-суперкомпилятора, с которым могли бы экспериментировать не только его разработчики, но и иные пользователи. По существу, оставался открытым вопрос о принципиальной возможности построения полностью автоматического суперкомпилятора. Данная диссертационная работа даёт положительный ответ на этот вопрос.

Цели работы. Диссертационное исследование было направлено на решение следующих основных задач:

1. Построить и реализовать новые алгоритмы специализации функциональных программ, основанные на методах суперкомпиляции. Упростить и довести до алгоритмов полуавтоматические процедуры, представленные в работах В. Ф. Турчина [14], [16], [17], [19], и/или улучшить качественные характеристики этих процедур с точки зрения построения более эффективных остаточных программ.
2. Разработать и довести до алгоритмов методы построения выходных форматов функций-подпрограмм.

3. Построить экспериментальный автоматический суперкомпилятор, с которым могли бы экспериментировать посторонние пользователи. Изучить характеристики этого суперкомпилятора.

Общая методика работы. Большая часть идей, используемых в диссертации, хорошо известна в рамках суперкомпиляции. Основным инструментом анализа и преобразований программ является метаинтерпретация этих программ. Преобразования производятся по ходу дела метаинтерпретации (в режиме on-line) и остаточная программа строится исключительно на основе этой интерпретации, а не посредством пошаговой чистки исходной (преобразуемой) программы. Используемый параметрический язык описания конфигураций метадерева возможных вычислений является языком первого порядка.

Предметной областью нашего суперкомпилятора SCP4 является функциональный язык программирования РЕФАЛ-5. Этот же язык является языком реализации суперкомпилятора. Большинство алгоритмов работают в терминах внутреннего языка РЕФАЛ-графов, который ориентирован на адекватное описание временной эффективности. Это язык более низкого уровня по отношению к РЕФАЛу, но работает с теми же данными. Тем не менее, некоторые свойства преобразуемых алгоритмов формулируются в понятиях самого РЕФАЛа и соответствующие алгоритмы используют эти понятия.

Идеи методов построения выходных форматов компонент факторизации метадерева потенциальных вычислений, глобального анализа и распознавания мономов конкатенации (как и само понятие этих мономов) полностью оригинальны. После глобального анализа компоненты факторизации, производится повторная специализация этой компоненты по свойствам, выявленным глобальным анализом.

После основной стадии преобразований, отдельным проходом производится чистка входных и выходных формальных параметров, излишних вызовов функций.

Транслятор из языка РЕФАЛ-графов в язык С реализован А. П. Коньшевым [1]. Часть РЕФАЛ программ, которые используются в

дессертации в качестве тестовых примеров для суперкомпилятора SCP4, принадлежат А. В. Корлюкову [2], [4], [5].

Научная новизна. Все основные результаты являются новыми. Предложено несколько новых алгоритмов специализации. Для некоторых из ранее известных полуавтоматических/неформальных методов специализации разработаны алгоритмы, позволившие добиться их полной автоматизации. На основе разработанных алгоритмов реализован экспериментальный специализатор SCP4, который является *первым* полностью автоматическим специализатором программ, основанным на методах суперкомпиляции.

Практическая и теоретическая ценность. Представленные в диссертации алгоритмы распознавания синтаксической мономиальности программ и вычисления выходных форматов компонент факторизации дерева потенциальных вычислений могут быть полезны для решения классических задач самоприменения специализаторов, поставленных А. П. Ершовым [8], Ё. Футамурой [9] и В. Ф. Турчиным [14], [16]. В разделе 16.3 диссертационной работы подробно рассматриваются возможности алгоритма распознавания синтаксических мономов для понижения порядка временной сложности остаточных программ в задачах самоприменения. Данная диссертационная работа даёт положительный ответ на долго стоявший открытым вопрос о принципиальной возможности построения полностью автоматического суперкомпилятора; что является значительным шагом в направлении внедрения технологии суперкомпиляции в практику программного обеспечения современных компьютеров. В диссертации показано, что суперкомпилятор SCP4 может использоваться для автоматической верификации коммуникационных протоколов, посредством специализации их программных моделей. Например, были успешно верифицированы следующие cache coherence протоколы: IEEE Futurebus+, MOESI, MESI, MSI, The University of Illinois, Synapse N+1, DEC Firefly, Berkeley, Xerox PARC Dragon.

Апробация работы. Результаты работы докладывались и обсуждались на следующих конференциях:

- Международный Software Engineering симпозиум, Китай, 2001.
- Международный симпозиум Partial Evaluation and Semantics-Based Program Manipulation в Азии (Asia-PEPM), Япония, 2002.
- Международный симпозиум Computer Science in Russia, Екатеринбург, 2007.
- Международная конференция Perspectives of System Informatics посвященная памяти Андрея Ершова, Новосибирск, 2003.
- Международная конференция Program Understanding, Новосибирск-Алтай, 2003.
- Международная конференция Information Systems Technology and its Applications, Харьков, 2003.
- Международная конференция «Программные системы: теория и приложения», Переславль-Залесский, 2004.
- Международная конференция Reachability Problems, Финляндия, 2007.
- Российско-Французский коллоквиум Some mathematical problems of technological importance, Laboratoire Poncelet, Московский Независимый Университет, 2005.
- Научные семинары ИПС РАН, ИПМ РАН, ИСП РАН, ИППИ РАН, Institute of Software Китайской Академии Наук, университетов г. Ухань (Wuhan University)(Китай), г. Токио (Waseda University), г. Ливерпуль (The University of Liverpool).

Публикации. Основные результаты диссертации опубликованы в 15 работах [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], перечисленных в конце списка литературы.

Работа [25] опубликована в издании, входившем в перечень ВАК на момент публикации и имеющемся в перечне ВАК на данный момент. Работа [26] является монографией автора диссертации.

Личный вклад. Все результаты исследований, составляющие основное содержание диссертации, получены автором самостоятельно.

Структура и объём работы. Диссертация объёмом 322 страницы состоит из введения, двадцати одной основной главы, которые разбиты на части и разделы, заключения и приложения. Каждая глава и каждая часть начинаются с кратких введений, выделенных *курсивом*. Каждая глава заканчивается заключающей частью, в которой кратко сформулированы результаты данной главы. В главе «Результаты» сформулированы основные результаты диссертационной работы. Список цитируемой литературы состоит из 91 наименования.

СОДЕРЖАНИЕ РАБОТЫ

Во **Введении** приводится обоснование актуальности темы диссертации, формулируется постановка задачи, перечисляются основные полученные результаты, поясняется их положение в контексте текущих исследований, а также описывается структура диссертации.

Определение 1 *Реализацией функционального языка программирования \mathcal{R} назовём четвёрку $\langle P, D, U, T \rangle$, где множество P называется множеством \mathcal{R} -программ, множество D называется множеством \mathcal{R} -данных; частично рекурсивные функции $U: P \times D \mapsto D$ и $T: P \times D \mapsto \mathbb{N}$ называются соответственно универсальной функцией (или семантикой) и сигнализирующей функцией времени языка \mathcal{R} . Здесь \mathbb{N} – множество натуральных чисел.*

Ниже мы будем использовать обозначение $p(x)$ как сокращение для $U(p, x)$.

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

программа $p(x, y)$ из P , реализует функцию¹ $F(x, y) : X \times Y \mapsto D$, где $X \subset D, Y \subset D$. Зафиксируем значение первого аргумента этой функции $x_0 \in X$. В задаче специализации требуется построить другую программу $q(y) \in P$ такую, что

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

Программу q называют остаточной программой. Содержательная задача состоит в построении *оптимальной* q (по времени исполнения). Таким образом, программа q представляет некоторое продолжение функции $F(x_0, y)$ по второй компоненте.

Выше приведена частная формулировка общей задачи специализации программ. В общей постановке требуется проспециализировать программу по данному контексту применения самой программы или, более широко, её подпрограмм. Разные уточнения понятия оптимальности определяют конкретные *аппроксимации* задачи специализации как таковой.

В первой главе проведен анализ современного состояния проблемы автоматической специализации программ, сделан обзор научных работ, проводимых в этой области. Рассмотрены разные подходы к постановке задачи специализации как таковой; анализируются принципиальные отличия суперкомпиляции от других существующих методов; делается исторический обзор попыток построения суперкомпиляторов. Поясняется положение методов, разработанных в диссертации, в контексте текущих исследований.

Во второй главе дан набросок структуры суперкомпилятора SCP4, приведена его блок-схема. Дано неформальное введение в функциональный язык программирования РЕФАЛ-5, в терминах которого в диссертационной работе исследуются методы суперкомпиляции.

В третьей главе определён язык параметров, описывающий состояния РЕФАЛ машины; в частности, – её входную точку.

В четвёртой главе определён язык РЕФАЛ-графов. Под языком РЕФАЛ-графов понимается язык, позволяющий показать структуру промежуточного состояния программы в процессе ее преобразований.

¹То есть именно функцию, а не частичную функцию.

В пятой главе описывается один из основных инструментов преобразований, который называется прогонкой. Прогонка есть метаобобщение одного шага РЕФАЛ машины. РЕФАЛ машина работает с полем зрения (рабочей памятью), описывающей конкретное состояние машины. Прогонка работает с параметризованным полем зрения.

В шестой главе описываются алгоритмы свёртки метадрева возможных вычислений преобразуемой программы p . Подробно рассматриваются принципы стратегий, на которых основаны эти алгоритмы. Алгоритмы свёртки являются критическими алгоритмами, суперкомпиляции с точки зрения успешной оптимизации по времени программы p : именно они (вместе с алгоритмами глобального анализа) аппроксимируют решение задачи специализации как таковой. Опорными узлами в метадреве возможных вычислений программы p называются узлы, которые могут быть объявлены входными точками функций-подпрограмм остаточной программы.

Алгоритмы свёртки, в частности, включают в себя:

- Механизм выделения функции-цикла в результате анализа последовательности параметризованных стеков, находящихся в опорных узлах метадрева развёртки вдоль пути от корня этого дерева к текущему (анализируемому) узлу (Глава 6).
- Алгоритмы вложения (раздел 6.1) и обобщения параметризованных конфигураций (раздел 6.1).
- Обнинский алгоритм расщепления стека функций, принадлежащий В. Ф. Турчину (раздел 6.3.1.1).
- Алгоритм расщепления данной задачи на специализацию на подзадачи (раздел 6.1).

Введено отношение «похожести» (раздел 6.3.1), связывающее семантическое проявление цикла в метадреве с синтаксическими структурами в этом дереве. Это отношение представлено, как пересечение отношения «похожести» чисто функциональной структуры стеков (последовательности имён функций) и «похожести» параметрического

описания аргументов разных вызовов одной функции. Отношение «похожести» аргументов вызовов функций является модификацией отношения Хигмана-Крускала, которое адаптировано на множество РЕФАЛ-термов. Сформулированы свойства этих отношений и непосредственно алгоритма обобщения обеспечивающие конечность процедуры факторизации метадрева вычислений, часть из которых доказана.

Введено понятие последовательности временного развития стека (раздел 6.3.1.1).

Определение 2 Пусть дан конечный алфавит $A ::= \{a_j\}$. Пусть W – множество конечных слов над A , включая пустое слово. Пусть даны функция $G: A \mapsto W$ и некоторое слово $s \in W$. Определим последовательность (быть может, конечную) с записью времён рождения $stack_n$:

1. $stack_0 ::= (s, 0)$

2. пусть определен $stack_i ::= (a_i, t) u_i$, где $a_i \in A$, $t \in \mathbb{N}$, а u_i – оставшаяся часть последовательности $stack_i$. Тогда

$stack_{i+1} ::= Time(G(a_i), MaxTime) u_i$,

где $MaxTime$ есть максимум по вторым компонентам пар из $stack_i$.

$Time(a\ w, time) ::= (a, time+1) Time(w, time+1)$;

$Time(, time) ::= ;$, где $a \in A$, $w \in W$.

Последовательность $stack_n$ назовём временным развитием стека (G, s) .

Описаны стратегии алгоритмов свёртки:

- Алгоритм вложения просматривает опорные узлы, лежащие на пути от корня метадрева до текущего узла, в порядке времени создания этих узлов (раздел 6.2).
- Обобщение просматривает опорные узлы вдоль пути от текущего узла до корня мета-древа, то есть в направлении противоположном ориентации (противоположном просмотру этих узлов алгоритмом вложения) (раздел 6.3.5).

Доказано, что местность сред двух префиксов, выделяемых обнинским алгоритмом Турчина, совпадают (раздел 6.3.1.1, Утверждение 2).

Доказаны теоремы:

- Частично рекурсивная функция, определяемая обобщённым параметризованным стеком, является расширением частично рекурсивной функции, которая определена стеком заменяемого предыдущего опорного узла (раздел 6.3.2, Теорема 1).
- Результат подстановки параметров, построенной обобщением предыдущего стека, в параметризованные вызовы функций в обобщённом стеке определяет частично рекурсивную функцию заменяемого предыдущего опорного узла (раздел 6.3.2, Теорема 2).

Рассмотрены свойства результатов вложения и обобщения, с точки зрения временной эффективности последующего выполнения результата преобразований (раздел 6.4). В частности, показано, что в процессе обобщения информация о параметризованных средах может быть потеряна тремя принципиально разными способами:

- разбиением задачи на подзадачи и описанием связей между ними;
- обобщением параметризованных выражений с построением нерекурсивных (без функциональных вызовов) сводящих подстановок, которые остаются в метадереве возможных вычислений;
- обобщением параметризованных выражений с построением рекурсивных сводящих подстановок, которые остаются в метадереве возможных вычислений.

В седьмой главе даны принципы развёртки специализируемой программы. Развёртка перестраивает структуру параметризованных стеков функций в листьях грозди прогонки, согласно некоторой стратегии; выбирает в поле зрения суперкомпилятора лист, стек которого будет предметом преобразования следующего вызова прогонки.

В восьмой главе вводится понятие компоненты факторизации метадрева возможных вычислений. Дана классификация компонент факторизации с точки зрения их глобального анализа.

В процессе свёртки могут быть построены три типа компонент:

1. **самодостаточные** – все рёбра исходящие из вершин таких подграфов заканчиваются только в вершинах данного подграфа;
2. **компоненты** содержащие только вершины, исходящие рёбра из которых заканчиваются либо в вершинах данного подграфа, либо в вершинах компонент факторизации, построенных ранее (далее мы будем говорить, что подграф ссылается на построенные подграфы);
3. **компоненты**, в которых существуют вершины с исходящими из них рёбрами, которые заканчиваются на базисных узлах являющихся корнями поддрева, факторизация которого ещё не завершена (далее в этом случае мы будем говорить, что подграф ссылается на непрофакторизованную часть мета-древа).

В девятой главе описываются основные свойства алгоритма чистки семантически недостижимых ветвей компонент факторизации метадрева возможных вычислений.

В десятой главе описываются алгоритмы глобального анализа компоненты факторизации F , ключевыми из которых являются алгоритм автоматического построения выходного формата функции F по ходу дела суперкомпиляции (раздел 10.1.2) и алгоритм распознавания мономов конкатенации (раздел 10.2). Вводится понятие частично рекурсивного монома конкатенации (раздел 10.2.2).

Определение 3 Пусть дана n -местная частичная функция

$$F(x_1, x_2, \dots, x_n) : DATA^n \mapsto DATA_{\perp}$$

где $DATA$ – множество данных РЕФАЛа. Скажем, что F частичный моном приписывания, если существует такая конечная последовательность натуральных чисел k_1, k_2, \dots, k_m , что на области

определения функции верно тождество

$$F(x_1, x_2, \dots, x_n) = x_{j_1}^{k_1} x_{j_2}^{k_2} \dots x_{j_m}^{k_m}$$

где $\forall s : (0 < s \leq m) 0 < j_s \leq n$. Здесь операция умножения – приписывание, возведение в степень относится к этой же операции.

Пусть $Refal_n$ есть РЕФАЛ-подобный язык, в котором разрешены только n -местные функции-определения (точное определение см. в разделе 10.2.2).

Определение 4 Пусть дана $Refal_n$ программа P такая, что местности всех определений из P совпадают и равны n . Пусть $F_i(x_1, x_2, \dots, x_n)$ входные форматы определений из P . И пусть дана конечная последовательность натуральных чисел k_1, k_2, \dots, k_n , где $k_j > 0$ для всех j . Пусть $[y]^j$ обозначает y, \dots, y , где y повторено j раз. Формальным повышением местности программы P по отношению к n -ке k_1, k_2, \dots, k_n назовём её следующее преобразование:
для всех i

- каждую левую часть $left-part = p_1, \dots, p_n$ каждого предложения из P преобразуем к виду $[p_1]^{k_1}, \dots, [p_n]^{k_n}$;
- каждый функциональный вызов $\langle F_i \ arg_1, \dots, arg_n \rangle$ программы P преобразуем к виду $\langle F_i \ [arg_1]^{k_1}, \dots, [arg_n]^{k_n} \rangle$.

Построенную программу обозначим P^{k_1, k_2, \dots, k_n} и назовём k_1, k_2, \dots, k_n местной по отношению к P . (Определения из P^{k_1, k_2, \dots, k_n} будем обозначать аналогично: $F_i^{k_1, k_2, \dots, k_n}$.)

Определение 5 Пусть дана $Refal_n$ программа P такая, что местности всех определений из P совпадают и равны n . Пусть $F_i(x_1, x_2, \dots, x_n)$ входные форматы определений из P . И пусть дана перестановка σ элементов последовательности $1, \dots, n$. Формальной перестановкой параметров программы P посредством σ , назовём её следующее преобразование:

для всех i

- каждую левую часть $left-part = p_1, \dots, p_n$ каждого предложения из P преобразуем к виду $p_{\sigma(1)}, \dots, p_{\sigma(n)}$;
- каждый функциональный вызов $\langle F_i \ arg_1, \dots, arg_n \rangle$ программы P преобразуем к виду $\langle F_i \ arg_{\sigma(1)}, \dots, arg_{\sigma(n)} \rangle$.

Построенную программу обозначим ${}^\sigma P$ и назовём σ -переставленной по отношению к P . (Определения из ${}^\sigma P$ будем обозначать аналогично: ${}^\sigma F_i$.)

Определение 6 Назовём $Refal_n$ программу P синтаксическим мономом, если существует конечная последовательность натуральных чисел k_1, \dots, k_n (где $k_j > 0$ для всех j), для которой определена программа P^{k_1, k_2, \dots, k_n} , и существует такая перестановка σ последовательности $1, \dots, m$ ($m = k_1 + \dots + k_n$), что после формальной замены в программе ${}^\sigma P^{k_1, k_2, \dots, k_n}$

- каждого функционального вызова

$$\langle \text{function-name } arg_{\sigma(1)}, \dots, arg_{\sigma(m)} \rangle$$

конкатенацией его аргументов $arg_{\sigma(1)} \dots arg_{\sigma(m)}$,

- каждой левой части

$$pattern_{\sigma(1)}, \dots, pattern_{\sigma(m)}$$

каждого предложения конкатенацией его образов

$$pattern_{\sigma(1)} \dots pattern_{\sigma(m)},$$

левая часть *left-side* каждого предложения *sentence* программы $\sigma P^{k_1, k_2, \dots, k_n}$ будет графически совпадать с правой частью этого предложения.

Доказана теорема о достаточном условии частичного монома конкатенации (раздел 10.2.2, Теорема 2).

Теорема 1 (достаточное условие мономиальности)

В обозначениях определения 6. Пусть $F_j(x_1, \dots, x_n)$ есть входной формат определения *definition* F_j . Пусть y_i обозначает i -тый член последовательности $[x_1]^{k_1}, \dots, [x_n]^{k_n}$, тогда для всех $j - F_j$ из синтаксического монома P определяет частичную функцию F_j , являющуюся частичным мономом $y_{\sigma(1)} \dots y_{\sigma(m)}$:

$$F_j(x_1, \dots, x_n) = y_{\sigma(1)} \dots y_{\sigma(m)}.$$

Описана стратегия выбора гипотезы мономиальности (раздел 10.2.3).

Пусть программа P содержит ровно одно определение F и пусть $\langle F \ x_1, \dots, x_n \rangle$ есть его входной формат. Рассмотрим все пассивные предложения² из F : $p_{1_i}, \dots, p_{n_i} = \text{right-side}_i$. Пусть d_{j_i} обозначают некоторые фиксированные данные РЕФАЛа. Предположим, что для каждого i существует хотя бы одна точка d_{1_i}, \dots, d_{n_i} такая, что определение F описывает частичную функцию, при вызове которой $\langle F \ d_{1_i}, \dots, d_{n_i} \rangle$ на первом же шаге машины, отождествление произойдёт в предложении $p_{1_i}, \dots, p_{n_i} = \text{right-side}_i$. То есть набор данных d_{1_i}, \dots, d_{n_i} представляет собой один из базисных случаев рекурсивного определения F . Правая часть right-side_i , по определению, не будет изменена в предполагаемом процессе анализа на мономиальность. Следовательно, если P частичный синтаксический моном, тогда существует моном M_i от формальных переменных x_1, \dots, x_n такой, что результат подстановки вместо этих переменных соответственно p_{1_i}, \dots, p_{n_i} есть right-side_i . Для всех i и j должно выполняться графическое равенство $M_i(x_1, \dots, x_n) = M_j(x_1, \dots, x_n)$.

Описанная выше стратегия порождения гипотезы обладает одним существенным недостатком: построение гипотезы неоднозначно.

²Правые части которых не содержат вызовов функций.

Теорема 2 В обозначениях данного раздела (см. выше), пусть в программе P существует хотя бы одно предложение $p_{1_i}, \dots, p_{n_i} = \text{right-side}_i$ такое, что right-side_i не содержит вызовов функций и для всех k образцы p_{k_i} отличны от пустых выражений. Тогда может существовать лишь конечное число гипотез, которые можно построить в рамках описанной выше стратегии.

Показано, что на некоторых примерах эти алгоритмы способны понижать временную сложность программы с экспоненциальной до константной $O(1)$.

В одиннадцатой главе описываются принципы использования результатов глобального анализа компоненты факторизации. Повторная специализация подграфа специализирует компоненту факторизации по структуре выходных форматов подграфов F_i , на которые эта компонента ссылается (то есть существует ребро с началом в некотором узле этой компоненты и концом во входной точке одного из F_i), и по инвариантам, обнаруженным глобальным анализом. Повторная специализация проводится без дополнительных разверток.

В двенадцатой главе описывается алгоритм уменьшения местности (арности) функций.

В тринадцатой главе описывается стратегия глобальности использования заголовков (входных форматов) подфункций. Глобальность заголовков компонент факторизации может сократить число этих компонент в остаточной программе.

В четырнадцатой главе анализируются семантические понятия временной эффективности представленные в остаточной программе на языке РЕФАЛ-графов (внутреннем языке преобразований SCP4), которые не могут быть адекватно описаны терминами РЕФАЛа-5 – в его конкретной модели вычислений.

В пятнадцатой главе анализируются принципы анализа остаточной программы, позволяющие отобразить ее на языке Си достаточно эффективно.

В шестнадцатой главе описывается формальный язык постановки задач на суперкомпиляцию и связи его с языком параметров описаний

конфигураций в момент преобразований. Последний был описан выше и в данной главе уточняется: вводится подтипизация параметров (раздел 16.2), уточняются алгоритмы прогонки (раздел 16.2.1) и свёртки (раздел 16.2.2). Рассматривается приложение алгоритма распознавания мономов конкатенации в задаче самоприменения (раздел 16.3).

В семнадцатой главе описано большое число преобразований простых программ, демонстрирующих возможности суперкомпилятора SCP4. Даны примеры классического использования специализатора в качестве компилятора из некоторого языка L в объектный язык специализатора, в нашем случае РЕФАЛ. В качестве языка L используются язык Машины Тьюринга и язык Регистровой Машины. Указывается на удачное приложение методов суперкомпиляции для верификации параметризованных коммуникационных протоколов.

В восемнадцатой главе формулируется модель вычислений РЕФАЛ программ. В рамках этой модели даётся оценка ускорения, получаемого в результате суперкомпиляции рассмотренного ранее интерпретатора Машины Тьюринга Turing (раздел 17.1, пример 10). Рассматривается вопрос о соотношении логического и физического времени исполнения программы. Исследуются характеристики суперкомпилятора SCP4; в частности, доказана следующая теорема (раздел 18.1, Теорема 1):

Теорема 3 *Для любой программы Prog на языке Машины Тьюринга (MT) результат специализации суперкомпилятором SCP4 интерпретатора Turing по Prog не содержит терминологии программы Prog. То есть включает в себя только синтаксические единицы Prog, описывающие данные на ленте: текущие символы в ячейках и записываемые в ячейки символы. Следовательно, происходит «эффективная» компиляция программы Prog из языка MT в РЕФАЛ.*

В девятнадцатой главе рассмотрен язык разметки программы, посредством которого программист может объявить суперкомпилятору SCP4 стратегии преобразований этой программы или подсказать ему правильный выбор конкретных локальных действий.

В двадцатой главе описаны некоторые свойства рассматриваемой нами реализации РЕФАЛа-5, которые являются принципиальными с точки зрения простоты построения суперкомпилятора.

В главе **Общее заключение** автор диссертации формулирует нерешённые задачи, решение которых, могло бы внести существенный вклад в дальнейшее понимание природы задачи автоматической специализации программ как таковой.

В двадцать первой главе перечислены основные результаты диссертационной работы, сведения об апробации и приложении этих результатов.

В **Приложении** даётся и анализируется результат специализации интерпретатора Машины Тьюринга по программе умножения натуральных чисел посредством суперкомпилятора SCP4.

Основные результаты.

1. Разработаны и реализованы аппроксимирующие алгоритмы распознавания частично рекурсивных константных функций, частично рекурсивных функций проекции. Показано, что на некоторых примерах эти алгоритмы способны понижать временную сложность программы.
2. На основе полуавтоматических процедур обобщения параметризованных конфигураций, представленных в работах В. Ф. Турчина [14], [16], [17], [19], разработаны и реализованы алгоритмы обобщения параметризованных конфигураций. Улучшены качественные характеристики этих алгоритмов с точки зрения построения более эффективных остаточных программ.
3. Разработан и реализован алгоритм построения выходных форматов компоненты факторизации F метадрева возможных вычислений в режиме online (по ходу дела суперкомпиляции); что позволяет сразу использовать построенный формат для специализации по нему как других компонент-функций, вызывающих функцию F , так и самой

функции F . Показано, что на некоторых примерах этот алгоритм способен понижать временную сложность программы.

4. Предложено понятие частично рекурсивного монома конкатенации. Доказана теорема о достаточном условии частично рекурсивного монома конкатенации. Разработана стратегия выбора гипотезы мономиальности. На основании этих теоремы и стратегии разработан и реализован алгоритм, распознающий частично рекурсивные синтаксические мономы конкатенации. Показано, что на некоторых примерах этот алгоритм способен понижать временную сложность программы с экспоненциальной до константной $O(1)$. Показано, что этот алгоритм может быть использован в решении задачи самоприменения для понижения порядка временной сложности результата самоприменения.
5. Разработан и реализован экспериментальный *автоматический* суперкомпилятор SCP4, предметной областью которого является функциональный язык программирования РЕФАЛ-5. Демонстрация суперкомпилятора доступна на Web-странице в режиме on-line [37].
6. Исследованы характеристики суперкомпилятора SCP4.

СПИСОК ЛИТЕРАТУРЫ

- [1] *Коньшев А. П.* Компилятор из языка Рефал-графов в язык С: исходные тексты и демонстрация. 2000 [Электрон. ресурс]// <http://www.botik.ru/pub/local/scp/refal5/>.
- [2] *Корлюков А. В.*, Пособие по суперкомпилятору SCP4, 1999 [Электрон. ресурс]// <http://www.refal.net/supercom.htm>.
- [3] *Корлюков А. В.* Получение формул в математике. // Тезисы докладов VIII Белорусской математической конференции, стр.178, 2000 [Электрон. ресурс]// <http://www.bsu.by/Converences/konf8/Sections/Abstr14/Korlukov/Korlukov.htm>.
- [4] *Корлюков А. В.* Несколько примеров преобразований программ суперкомпилятором SCP4. 2001 [Электрон. ресурс]// <http://www.refal.net/~korlukov/pearls/>.
- [5] *Корлюков А. В.* Суперкомпилируем суперагентов // Альфа, Т. 1, стр.89-98. Гродно: Гродненский государственный университет, 2001.
- [6] *Chmutov S. V., Gaydar E. A., Ignatovich I. M., Kozadov V. F., Nemytykh A. P., Pinchuk V. A.* Implementation of the symbol analytic transformation language FLAC // LNCS, Vol. 429, pp.276 / The Proc. of DISCO'90. Springer-Verlag, 1990.
- [7] *Chmutov S. V., Nemytykh A. P., Gaydar E. A., Ignatovich, I. M., Kozadov V. F., Pinchuk V. A.* The symbol analytic transformation language FLAC: sources, executable modules, 1991. [Электрон. ресурс]// <ftp://www.botik.ru/pub/local/scp/flac/flac386.zip>.
- [8] *Ershov A. P.* Mixed computation: potential applications and problems for study // Theoretical Computer Science, Vol. 18, pp.41-67.
- [9] *Futamara Y.* Partial Evaluation of Computation Process – An Approach to a Compiler-Compiler // Systems. Computers. Controls., Vol. 2(5), pp.45-50, 1971. (An updated version of the paper was republished in J. Higher-Order and Symbolic Computation, Vol. 12, pp.381-391, 1999.)

- [10] *Glück R., Turchin V. F.* Application of metasystem transition to function inversion and transformation // The Proc. of the ISSAC'90, pp.286-287. ACM Press, 1990.
- [11] *Nemytykh A. P., Pinchuk V. A.* Program Transformation with Meta-system Transitions: Experiments with a Supercompiler // LNCS, Vol. 1181, pp.249-260 / The Proc. of the Perspectives of System Informatics, Springer-Verlag, 1996. <ftp://ftp.botik.ru/pub/local/APP/meta-trans.ps.gz>.
- [12] *Nemytykh A. P., Pinchuk V. A., Turchin, V. F.* A Self-Applicable Supercompiler // LNCS, 1110, pp.322-337 / The Proc. of Partial Evaluation, International Seminar. *Danvy o., Glück R., Thiemann P. (Eds.)* Springer-Verlag, 1996. (<ftp://ftp.botik.ru/pub/local/APP/self-appl.ps.gz>).
- [13] *Turchin V. F.* A supercompiler system based on the language REFAL // SIGPLAN Notices, Vol. 14(2), pp.46-54. 1979.
- [14] *Turchin V. F.* The Language Refal, the Theory of Compilation and Meta-system Analysis // Courant Computer Science Report #20, New York University, 1980.
- [15] *Turchin V. F., Nireberg R., Turchin D. V.* Experiments with a supercompiler // Conference Record of the ACM Symposium on LISP and Functional Programming, pp.47-55. ACM Press, 1982.
- [16] *Turchin V. F.* The concept of a supercompiler // ACM Transactions on Programming Languages and Systems, Vol. 8, pp.292-325. ACM Press, 1986.
- [17] *Turchin V. F.* The algorithm of generalization in the supercompiler // The Proc. of the IFIP TC2 Workshop, Partial Evaluation and Mixed Computation, 531-549. North-Holland Publishing Co., 1988.
- [18] *Turchin V. F.* Refal-5, Programming Guide and Reference Manual. Holyoke, Massachusetts: New England Publishing Co., 1989. (electronic version: <http://www.botik.ru/pub/local/scp/refal5/>, 2000).

- [19] *Turchin V. F.* Metacomputation in the language Refal. 1990. (unpublished, private communication)
- [20] *Turchin V. F.* Program Transformation with Metasystem Transitions // J. of Functional Programming, Vol. 3(3), pp.283-313. 1993.
- [21] *Turchin V. F.* Metacomputation: Metasystem transition plus supercompilation // LNCS, Vol. 1110, pp.481-509 / The Proc. of the PEPM'96. Springer-Verlag, 1996.
- [22] *Turchin V. F.* Supercompilation: Techniques and results // LNCS, Vol. 1181, pp.227-248 / The Proc. of Perspectives of System Informatics. Springer-Verlag, 1996.
- [23] *Turchin V.F., Turchin D.V., Konyshov A.P., Nemytykh A.P.* Refal-5: sources, executable modules. [Электрон. ресурс]// <http://www.botik.ru/pub/local/scp/refal5/>, 2000.

ПУБЛИКАЦИИ АВТОРА ПО ТЕМЕ ДИССЕРТАЦИИ

- [24] *Лисица А. П., Немытых А. П.* Работа над ошибками // Программные системы: теория и приложения. Т. 1, стр. 195-232, М: Физматлит, 2006. (доступна на ftp://www.botik.ru/pub/local/scp/refal5/psta_errors2006.pdf).
- [25] *Лисица А. П., Немытых А. П.* Верификация как параметризованное тестирование (эксперименты с суперкомпилятором SCP4) // Программирование. Т. 33(1), стр. 22-43, М., 2007.
- [26] *Немытых А.П.* Суперкомпилятор SCP4: Общая структура. М.: УРСС, 2007. 152 с.
- [27] *Korlyukov A. V., Nemytykh A. P.* Supercompilation of Double Interpretation. (How One Hour of the Machine's Time Can Be Turned to One Second) // Вестник национального технического университета „Харьковского политехнического института“, Т. 1, pp.123-150. Харьков, 2004. (электронная версия:

http://www.refal.net/~korlukov/scp2int/Karliukou_Nemytykh.pdf,
исходные тексты и демонстрация:
http://www.refal.net/~korlukov/demo_scp4xslt.zip)

- [28] *Lisitsa A., Nemytykh A. P.* Verification of parameterized systems using supercompilation. A case study // The Proc. of the Third Workshop on Applied Semantics (APPSEM05) / Hofmann M., Loidl H.W. (Eds.), pp.12-15. Fraunheimsee, Germany. Ludwig Maximillians Universitat Munchen. 2005, September. Accessible via:
ftp://www.botik.ru/pub/local/scp/refal5/appsem_verification2005.ps.
- [29] *Lisitsa A., Nemytykh A. P.* Towards verification via supercompilation // The Proc. of the 29th Annual International Computer Software and Applications Conference COMPSAC 2005 / Workshops and Fast Abstracts Volume, pp.9-10. IEEE 2005.
- [30] *Lisitsa A., Nemytykh A. P.* Reachability Analysis in Verification via Supercompilation // The Proc. of the Satellite Workshops of DTL 2007 / TUCS General Publication, No. 45, Part 2, pp.53-67. Turku, Finland. June 2007.
- [31] *Lisitsa A., Nemytykh A. P.* A Note on Specialization of Interpreters // LNCS, Vol. 4649, pp.237-248 / The Proc. of The 2-nd International Symposium on Computer Science in Russia. Springer, 2007.
- [32] *Nemytykh A. P.* Supercompiler SCP4: Use of quasi-distributive laws in program transformation // Wuhan University Journal of Natural Sciences. , Vol. 95(1-2), pp.375-382 / The Proc. of International Software Engineering Symposium. Wuhan, China. 2001, March.
- [33] *Nemytykh A. P.* A Note on Elimination of Simplest Recursions // The Proc. of the ACM SIGPLAN Asian Symposium on Partial Evaluation and Semantics-Based Program Manipulation, pp.138-146. ACM Press. Aizu, Japan. 2002.
- [34] *Nemytykh A. P.* The Supercompiler SCP4: General Structure (extended abstract) // LNCS, Vol. 2890, pp.162-170 / The Proc.

of the Perspectives of System Informatics. Springer-Verlag, 2003. (ftp://www.botik.ru/pub/local/scp/refal5/nemytykh_PSI03.ps.gz).

- [35] *Nemytykh A. P.* Playing on REFAL // The Proc. of the International Workshop on Program Understanding, pp.29-39. Novosibirsk – Altai Mountains. July 2003. (<http://www.botik.ru/pub/local/scp/refal5/korlyukov.html>).
- [36] *Nemytykh A. P.* The Supercompiler SCP4: General Structure // Программные системы: теория и приложения Т. 1, стр.448-485. М.: Физматлит, 2004. (<ftp://ftp.botik.ru/pub/local/scp/refal5/GenStruct.ps.gz>).
- [37] *Nemytykh A. P., Turchin, V. F.* The Supercompiler SCP4: sources, on-line demonstration. 2000. [Электрон. ресурс]// <http://www.botik.ru/pub/local/scp/refal5/>.
- [38] *Turchin V. F., Nemytykh A. P.* Metavariables: Their implementation and use in Program Transformation // Technical Report CSc. TR 95-012. City College of the City University of New York, 1995.

Работа [25] опубликована в издании, входившем в перечень ВАК на момент публикации и имеющемся в перечне ВАК на данный момент. В этой статье автором разработан метод верификации параметризованных коммуникационных протоколов посредством суперкомпиляции интерпретаторов этих протоколов. Используя суперкомпилятор SCP4, автор успешно применил этот метод для верификации серии cache coherence протоколов.

В статье [24] автор показал, что разработанный им метод верификации параметризованных протоколов посредством суперкомпиляции может быть применён для поиска тестовых примеров, указывающих на ошибки в спецификациях.

В работах [28], [29] автором были заложены основы метода верификации, отмеченного выше.

Работа [26] является монографией автора диссертации.

В работе [27] автор показал, что использование суперкомпиляции для специализации двойной интерпретации может улучшить

мультипликативную константу во временной сложности на три порядка.

В статье [30] автор построил формальную модель суперкопиляции, достаточную для успешной верификации некоторого класса параметризованных cache coherence протоколов.

В работе [31] автор показал, что верификацию коммуникационных протоколов можно рассматривать как специализацию их интерпретаторов по фиксированной части начальной конфигурации этих протоколов.

Здесь [37] представлены исходные тесты суперкомпилятора SCP4, реализованного автором.

В работе [38] автором был построен язык параметров, описания параметризованных конфигураций, позволяющий уточнять постановки задач на самоприменение.

Все остальные указанные работы были выполнены автором диссертации без соавторов.