На правах рукописи

Пинжин Алексей Евгеньевич

АЛГОРИТМЫ ИНТЕЛЛЕКТУАЛЬНОЙ ОБРАБОТКИ ИНФОРМАЦИИ И СТРУКТУРНОГО СИНТЕЗА ПРОГРАММ

Специальность 05.13.01 - Системный анализ, управление и обработка информации (отрасль: промышленность)

А в т о р е ф е р а т
диссертации на соискание ученой степени
кандидата технических наук

Томск – 2009

Работа выполнена в Томском политехническом университете

Научный руководитель – доктор физико-математических наук, доцент
Новосельцев Виталий Борисович

Официальные оппоненты: доктор технических наук, профессор
Матросова Анжела Юрьевна

доктор технических наук, профессор
Спицын Владимир Григорьевич

Ведущая организация – Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск

Защита состоится «21» января 2010 г. в 15 ч. 00 м.
на заседании совета по защите докторских и кандидатских диссертаций Д 212.269.06 в Томском политехническом университете по адресу: 634034, г. Томск, ул. Советская, 84/3, Институт «Кибернетический центр».

С диссертацией можно ознакомиться в библиотеке Томского политехнического университета по адресу: 634034, г. Томск, ул. Белинского, 55.

Автореферат разослан «___» декабря 2009 г.

Ученый секретарь совета по защите докторских и кандидатских диссертаций Д 212.269.06, кандидат технических наук, доцент  Сонькин М.А.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность. В современном мире информационных технологий наблюдается постоянный рост интереса к методам интеллектуальной обработки информации, который определяется, с одной стороны, растущими объемами и интеграцией хранилищ данных, а с другой – постоянным ростом спроса на информационные услуги. Указанные тенденции проявляются как на уровне корпоративных информационных систем в области медицины, экономики, прогнозирования и др., так и на глобальном уровне, где одним из следствий возросших потребностей в «интеллектуализации» всемирной сети стало возникновение парадигмы Semantic Web. Основой многих современных систем интеллектуальной обработки информации являются машины логического вывода, реализующие те или иные методы доказательства логических теорем. Примерами таких систем являются экспертные системы, системы поддержки принятия решений, интеллектуального поиска и др.

Другой важной проблемой теории искусственного интеллекта является автоматический синтез программ. По отношению к общей проблеме логического вывода, задача синтеза программ в традиционной постановке является более узкой. Алгоритмы синтеза применяются в CASE-системах, при проектировании аппаратного обеспечения, а также в научно-практических разработках, связанных со сложными математическими вычислениями и построением сложных алгоритмов. Примерами последних являются математические расчеты в астрономии, авиационно-космической промышленности и др. Однако, интерпретация задачи синтеза как извлечения (трассировки) последовательности шагов доказательства, приводящих к заданному следствию, значительно расширяет круг применения подобных алгоритмов.

Диссертация посвящена исследованию алгоритмов логического вывода в специальных исчислениях и синтеза программ с условиями, подпрограммами и рекурсией. Область исследования ограничена дедуктивным подходом, где программа извлекается из доказательства соответствующей теоремы. Появление первых работ в области синтеза программ на основе дедуктивного подхода приходится на 60-е гг. прошлого столетия. В работах Р. Уолдингера, Ц. Ченя, Р. Ли, З. Манны при доказательстве теоремы использовался метод резолюций Робинсона и различные вариации, направленные на повышение его эффективности. В 70-ее годы появился язык логического программирования Пролог, который в последующие десятилетия получил широкое распространение и был использован при решении множества практических задач, в частности при построении экспертных систем. На базе Пролога появилось множество реализаций, решающих задачу синтеза программ (AutoBayes, NuPrl, Oyster и др.). Однако применяемый в Прологе метод вывода от целей к посылкам приводит к издержкам, связанным с необходимостью возврата при

установлении недостижимости очередной подцели. В результате, решение сложных задач может привести к экспоненциальному росту вычислительных затрат, что ограничивает использование Пролога для построения эффективных систем интеллектуальной обработки данных.

В последние десятилетия направление исследований в области автоматического синтеза программ сместилось от идеи полной замены человека к задаче поддержки программиста при разработке алгоритмов, выполнении верификации программ и т.п. Основная критика традиционного подхода заключается в низкой производительности алгоритмов, наличии проблемы полноты вывода, а также несовместимости конструкций логических языков и языков программирования.

Важный вклад в развитие данного направления внесли работы отечественных исследователей. В частности, идея альтернативного метода вывода от аксиом к целям, впервые высказанная Г. Генценым, получила развитие в исследованиях С. Ю. Маслова и Г. Е. Минца, где приобрела название «обратного метода». В работах А. Я. Диковского, Н. Н. Непейводы, Э. Х. Тыугу предлагаются вычислительные модели, ориентированные на описание конструкций структурного программирования, и исследуются теоретические аспекты синтеза рекурсивных программ. Теория структурных функциональных моделей (С-моделей) В. Б. Новосельцева является развитием теории вычислительных моделей и содержит теоретическое доказательство возможности построения алгоритмов, обладающих свойством полноты и решающих задачу вывода за полиномиальное время.

В настоящей работе предлагается вариант реализации алгоритма вывода для задачи автоматического логического вывода и синтеза программ с использованием теории С-моделей в качестве базового аппарата для формализации знаний о предметной области. Для ограничения пространства вывода и снижения вычислительных затрат разработаны специальные структуры данных внутреннего представления исходной спецификации. Предлагаемый подход особенно эффективен для устойчивых баз знаний, содержимое которых меняется достаточно редко. Особое место в диссертации отводится планированию программ с подпрограммами и рекурсией, где предлагается реализация метода «динамической развертки», позволяющего избежать дополнительных затрат на хранение информации о вложенности подпрограмм и решить проблему их поиска без снижения производительности алгоритма.

Цель работы и задачи исследования. Целью исследования является разработка высокопроизводительных алгоритмов логического вывода и синтеза программ на структурных функциональных моделях, а также адаптация этих алгоритмов для решения ряда задач, выраженных в языках традиционных логических исчислений.

Для достижения указанной цели в работе решаются следующие задачи:

1. Формальное определение модели исходных данных и формулировка задачи планирования и синтеза на базе теории С-моделей.
2. Определение общих требований и архитектуры машины вывода.
3. Формулировка правил вывода и разработка структур данных компиляции исходной спецификации с целью снижения вычислительных затрат при выполнении задачи планирования.
4. Разработка и программная реализация алгоритмов планирования и синтеза программ.
5. Расчет вычислительных затрат и экспериментальная оценка производительности разработанного алгоритма на тестовых выборках данных.
6. Сопоставление выразительных возможностей и определение правил трансформации элементов спецификации языка Пролог в конструкции теории структурных функциональных моделей.
7. Адаптация разработанной машины вывода для решения задач вывода на Хорновских выражениях логики высказываний, логики первого порядка, дескриптивной логики.
8. Экспериментальное сравнение производительности разработанной машины вывода с существующими аналогами.

Объектом исследования является область теории искусственного интеллекта, связанная с проблемами автоматического доказательства логических теорем и синтеза программ.

Предметом исследования являются вычислительные модели и алгоритмы логического вывода и синтеза программ.

Методы исследований. В работе использованы элементы теории структурных функциональных моделей, теории множеств, логики высказываний, первого порядка и дескриптивной логики, методы структурного программирования.

Научная новизна работы заключается в следующем:

1. На базе теории структурных функциональных моделей разработаны алгоритмы логического вывода и синтеза программ, обладающие высокими показателями скорости вычислений за счет представления исходной спецификации в виде специальных структур данных и использования метода динамической развертки, позволяющего сократить пространство вывода при помощи организации стека подпрограмм и рекурсивного планирования на иерархии подсхем.

2. На основании анализа элементов спецификации языка Пролог и сопоставления их с конструкциями теории структурных функциональных моделей введены новые формальные правила и ограничения трансформации задач вывода на логике первого порядка в задачи планирования на структурных функциональных моделях.

3. Произведена адаптация разработанной машины вывода для задач вывода, выраженных в языках традиционных логических исчислений (логики высказываний, логики первого порядка и дескриптивной логики) и экспериментально подтверждена более высокая скорость вычислений по сравнению с существующими аналогами.

Практическая ценность и реализация результатов работы.

Разработанные структуры данных и алгоритмы планирования реализованы в виде прикладного программного пакета, который может быть использован для решения широкого круга задач, связанного как непосредственно с исследованиями в области синтеза программ, так и с решением проблем прикладного характера, таких как прогнозирование, интеллектуальный поиск, построение экспертных систем, систем поддержки принятия решений и др. Предлагаемый подход хорошо масштабируется и особенно эффективен при обработке баз знаний большого объема.

Предложенные модели данных компиляции и алгоритмы вывода были внедрены в ООО «ФлексСофт» при разработке медицинской информационной системы для обработки экспертной базы знаний ExpertDoc. Применение предлагаемых подходов позволило многократно повысить скорость поиска противопоказаний лекарственных препаратов и реализовать возможность трассировки найденного решения.

Результаты работы также были использованы в проекте единой информационной среды Томского политехнического университета для решения задачи определения прав доступа пользователей (сотрудников университета) к информации о студентах, хранимой в единой базе данных. Применение предложенной стратегии вывода позволило реализовать эффективный алгоритм вычисления путем обхода дерева решений.

Предложенный подход логического вывода и синтеза программ используется в учебном курсе теории языков программирования и методов трансляции, который преподается студентам Томского политехнического университета.

Основные положения, выносимые на защиту:

1. Разработанный алгоритм вывода и синтеза программ, использующий метод динамической развертки и стратегию предварительной подготовки данных, характеризуется следующими показателями количества затрачиваемых операций относительно длины описания модели предметной области:

а) линейная зависимость для программ с условиями и подпрограммами;

б) полиномиальная зависимость не выше третьей степени для программ с рекурсией.

2. Предложенный метод сопоставления элементов спецификации языка Пролог и конструкций теории структурных функциональных

моделей определяет принципиальную возможность, правила и ограничения трансформации задач вывода, выраженных на языке логики первого порядка в задачи планирования на структурных функциональных моделях.

3. Разработанный алгоритм вывода на структурных функциональных моделях может быть использован для доказательства ряда логических теорем, сформулированных на языках логики высказываний, логики первого порядка и дескриптивной логики.

4. Результаты экспериментов подтверждают более высокие показатели производительности разработанной машины вывода по сравнению с существующими аналогами.

Достоверность полученных результатов подтверждается полнотой и корректностью теоретических обоснований и результатами экспериментов, проведенных с помощью разработанной программной реализации.

Апробация работы.

Основные результаты работы представлены в виде:

3-х статей в журнале «Известия ТПУ», Томский политехнический университет, г. Томск,

а также докладов на ряде научных форумов:

«IX Всероссийская конференция молодых ученых по математическому моделированию и информационным технологиям (YM-2008)» (г. Кемерово, 28-30 октября 2008 г.);

Международная конференция «Программные системы: теория и приложения (PSTA-2009)» (г. Переславль-Залесский, ИПС РАН, 13-15 мая 2009 г.);

Международная конференция «Теоретические и прикладные вопросы современных информационных технологий (ТиПВСИТ’2009)» (г. Улан-Удэ, 20-26 июля 2009 г.)

Всего по теме диссертации опубликовано 6 работ, из них 3 статьи в журнале, входящем в перечень ВАК, 2 статьи по материалам конференций и 1 работа в качестве тезисов доклада на конференции.

Личный вклад:

1. Формальное описание модели данных и постановки задачи построены автором на основе теории структурных функциональных моделей В. Б. Новосельцева.

2. Модели данных и алгоритмы синтеза программ, включая оригинальный алгоритм динамической развертки, разработаны автором и являются реализацией задач, поставленных научным руководителем аспирантского исследования В. Б. Новосельцевым.

3. Формальное обоснование применения разработанной машины вывода для решения задач, сформулированных на языках логики высказываний, логики первого порядка и дескриптивной логики, предложено лично автором.

4. Реализация машины вывода и экспериментальные оценки производительности выполнены лично автором.

Объём и структура работы. Диссертация состоит из введения, четырёх глав, заключения, списка источников из 62 наименований и 9 приложений. Основная часть работы изложена на 85 страницах, содержит 21 рисунок и 1 таблицу.

СОДЕРЖАНИЕ РАБОТЫ

Во введении рассматривается актуальность диссертационного исследования, формулируются цели и задачи работы, её научная новизна и практическая ценность, перечисляются выносимые на защиту тезисы. Приводится краткое содержание основных разделов работы.

В первой главе вводятся основные определения и соглашения, предлагается формальная модель предметной области и постановки задачи на базе теории структурных функциональных моделей.

Модель предметной области Σ рассматривается как совокупность множеств (A, F, P, D), представляющих собой наборы атрибутов, функциональных связей, селекторов и схем соответственно. Выделено непустое подмножество примитивных схем $E \subset D$.

Атрибут – характеристика объекта предметной области, связанная с некоторой схемой T из множества D. Если $T \in D \setminus E$, то связь атрибута a со схемой T выражается записью $a{:}T$ и атрибут называется *непервичным*. Если $T \in E$, то атрибут является *примитивным* (например, целое число).

Функциональная связь (ФС) – отображение множества исходных атрибутов в целевой атрибут: f: $a_1,\ldots, a_i \rightarrow a_0$.

Селектор (предикат) – условие, при котором определенное подмножество атрибутов и ФС имеет смысл.

Схема – совокупность атрибутов, ФС и селекторов. Схема T атрибута t определяется выражением вида:

$t{:}T$=($a_0{:}S_{00}$, $a_1{:}S_{01}$, …, $a_n{:}S_{0n}$

if $p_1(\ldots) \supset a_{10}{:}S_{10}$, $a_{11}{:}S_{11}\ldots | f\_set_1$

☐...☐

$p_k(\ldots) \supset a_{k0}{:}S_{k0}$, $a_{k1}{:}S_{k1}\ldots | f\_set_k$ *endif*

$| f\_set$),

где $T \in D \setminus E$ – имя схемы, $S_{ij} \in D$ – собственные подсхемы, а $a_{ij} \in A$ – атрибуты схемы T. Обозначение *f_set* скрывает список ФС схемы T. Часть определения между обозначениями *if…endif* определяет *вариантную часть* схемы, которая состоит из *условных ветвей*, разделенных символом ☐, и соответствует программной конструкции *if…elseif...else*. Символы $p_i \in P$ называются *выбирающими селекторами* и представляют собой логическую функцию вида $p_i(a_{n0},\ldots)$, где $a_{n0},\ldots$ – атрибуты заголовка схемы. Следом за описанием селектора указываются *условные атрибуты*

a_{i0}:S_{i0}, a_{i1}:S_{i1}…, а следующее за ними выражение f_set_i скрывает множество *условных ФС*. Условная ФС может включать в себя атрибуты заголовка и условные атрибуты соответствующей ей ветви. Условие p_i определяет *допустимость* атрибутов и ФС, входящих в i ветвь. Иными словами, условные атрибуты и ФС имеют смысл только в соответствующей им ветви вариантной части схемы.

Непервичные атрибуты порождают следующие имена. Пусть t – имя непервичного атрибута и t:T. Пусть $a_0,\ldots,a_n$ – имена собственных атрибутов, определенных в схеме T. Тогда, $t.a_0$, …, $t.a_n$ – также являются именами атрибутов. Атрибуты $a_0,\ldots,a_n$ называются суб-атрибутами по отношению к атрибуту t.

Описание предметной области поставляется в виде *структурной функциональной модели* (С-модель), которая представляет собой конечную совокупность схем $M = (T_1,\ldots,T_n)$.

Функциональная связь, входящая в схему T, является *допустимой* если ее аргументами и результатом являются примитивные атрибуты схемы T, либо суб-атрибуты непервичного атрибута, принадлежащего схеме T. Предметом исследования в настоящей работе являются *синтаксически замкнутые схемы*, содержащие только допустимые ФС.

Кроме того, указываются некоторые дополнительные ограничения для ФС схем С-модели:

- ✓ ФС содержит по крайней мере один собственный атрибут схемы;
- ✓ в *f_set* нет ФС, в которые вовлечены атрибуты из разных вариантных ветвей схемы или из вариантных частей ее подсхемы;
- ✓ длина любого атрибута, связанного ФС, не превышает двух.

Далее, в главе рассматриваются понятия НДС- и РДС-моделей, представляющие основной предмет исследования в предлагаемой работе.

С-модель M=($T_1,\ldots,T_s$) является *нерекурсивной детерминированной (НД-) С-моделью*, если она удовлетворяет следующим условиям:

- ✓ модель M является *синтаксически замкнутой*, т.е. подсхемы, встречающиеся в определении каждой схемы $T_i \in M$, являются элементарными либо входят в состав модели M;
- ✓ схемы $T_1,\ldots,T_s$ являются *детерминированными*, т.е. не содержат вариантной части, либо вариантная часть содержит две взаимоисключающие условные ветви;
- ✓ любая схема T_i модели M определяется через элементарные атрибуты, либо через схемы, предшествующие T_i в модели M.

Введение рекурсивных вхождений порождает конструкции, когда атрибут, связанный с некоторой схемой, явно или посредством ряда непервичных атрибутов входит в описание этой же схемы. Вводится следующее ограничение: для каждой схемы $T_i \in M$ рекурсивные вхождения подсхем вида t:T_i могут появляться только в одной из альтернативных ветвей схемы T_i. С-модели, удовлетворяющие этому ограничению, а также

первым двум ограничениям НДС-моделей, называются *рекурсивными детерминированными (РД-) С-моделями*.

Задача планирования и синтеза S определяется тройкой (A_0,X_0,T), где A_0 и X_0 – наборы имен исходных и искомых атрибутов, а T – схема, в которой определены эти имена. Задача планирования состоит в поиске решения $S=(A_0,X_0,T)$ на базе С-модели M, а задача синтеза – в генерации программы, принимающей на вход набор A_0 и возвращающей X_0.

Во второй главе предлагается общая архитектура машины вывода, перечисляются правила вывода, структуры данных компиляции исходной модели и алгоритмы планирования и синтеза.

В первом параграфе приводится ссылка на обоснование возможности построения алгоритмов вывода на НДС- и РДС-моделях, решающих задачу за полиномиальное время. Указывается, что многих издержек при выводе на С-моделях можно избежать за счет подбора структур данных и динамической развертки при планировании на схемах с подсхемами.

Излагается общая логика работы и состав основных модулей программной реализации машины вывода (рис. 1).

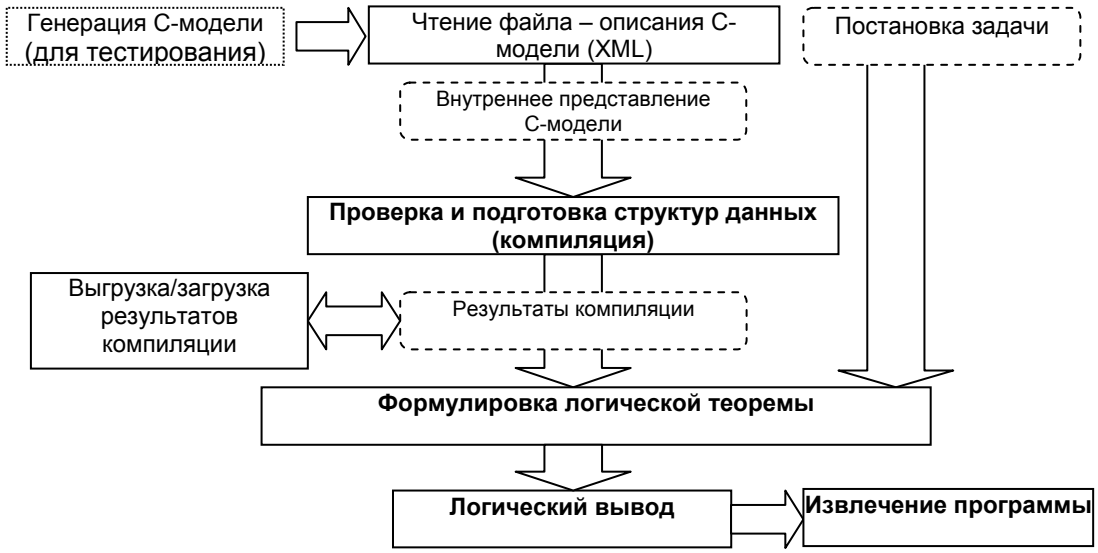


Рис. 1. Структура машины вывода на С-модели

Далее, предлагаются четыре формальных правила вывода:

1) $\dfrac{f:A\to x,A}{x}$, утверждает, что при выявлении *достижимости* аргументов A, утверждается *достижимость* результата x ФС f.

2) $\dfrac{t:T=(A,t_1:T_1\mid f:A\to t_1.x),t.A}{t_1:T_1=(x\mid ...),t_1.x}$ – расширяет первое правило для случая динамической развертки при планировании на схемах с подсхемами.

3) $\dfrac{f : A \to x, f / p, A / p}{x / p}$ – определяет условие *допустимости p* атрибута *x* при наличии условия допустимости *p* у функциональной связи *f*.

4) $\dfrac{f_1 : A \to x / p, f_2 : A \to x / \sim p}{x}$ – утверждает, что достижимость атрибута в каждой условной ветви определяет его достижимость в основной части схемы.

Далее рассматриваются модели преобразования (компиляции) НДС-модели в структуры данных, оптимизированные для реализации перечисленных правил. Особо отмечается, что результаты компиляции не зависят от постановки задачи и нотации описания исходных данных, что позволяет использовать эти результаты повторно для различных задач планирования.

Предложенная модель данных показана на рис. 2.

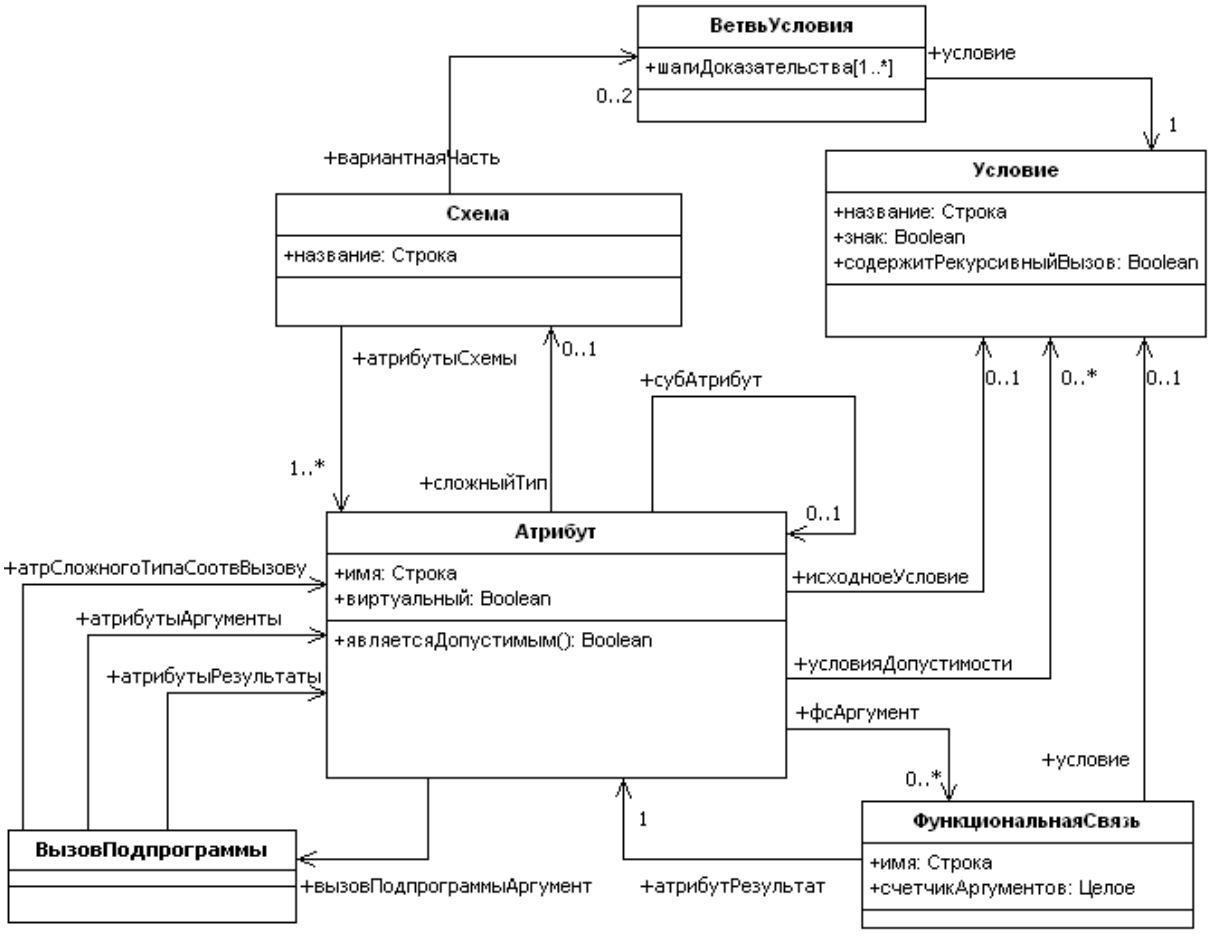


Рис. 2. Модель структур данных компиляции С-модели

Как видно из схемы, на этапе компиляции выполняется «инверсия» ассоциации «ФС–аргументы» с помощью ссылки фсАргумент. При применении правил вывода (1–4) инверсная связь позволяет реализовать поиск ФС, где атрибут участвует в качестве аргумента, за линейное

количество операций относительно числа аргументов. Это относится и к связи между подсхемами и доставляющими атрибутами этих подсхем. Ссылка вызовПодпрограммыАргумент обеспечивает поиск непервичного атрибута и соответствующей ему подсхемы при выполнении правила вывода (2).

Следующим важным элементом является использование счетчика при проверке достижимости всех аргументов ФС. С помощью счетчика реализуется эффективный метод контроля, где обнаружение достижимого атрибута-аргумента не требует его сопоставления с другими аргументами.

Хранение информации об условиях реализуется ассоциациями исходноеУсловие и условие объектов Атрибут и ФункциональнаяСвязь. Благодаря этим ассоциациям в совокупности со связью атрибутРезультат, перенос условия на атрибут выполняется с помощью элементарной операции копирования ссылки условие в список условияДопустимости. Проверка достижимости согласно правилу вывода (4) обеспечивается методом являетсяДопустимым класса Атрибут, определяющего наличие прямого и обратного условия с помощью простой логической операции.

В процессе компиляции также осуществляется трансформация условий, обеспечивающая доказательство достижимости атрибутов, входящих в логическую функцию селектора. Для каждой ветви условия создается *виртуальный атрибут*, являющийся результатом функции селектора. Функция селектора вводится в набор ФС основной части схемы, а виртуальный атрибут – в заголовок схемы и в список аргументов ФС, доставляющих атрибуты в соответствующую условную ветвь.

Постановка задачи включает в себя создание атрибута непервичного типа и объявление объекта ВызовПодпрограммы со ссылками на исходные и целевые атрибуты (списки атрибутыАргументы и аргументыРезультаты). В результате планирования исходный вызов подпрограммы связывается со объектом Подпрограмма (рис. 3), содержащим шаги доказательства (рис. 4).

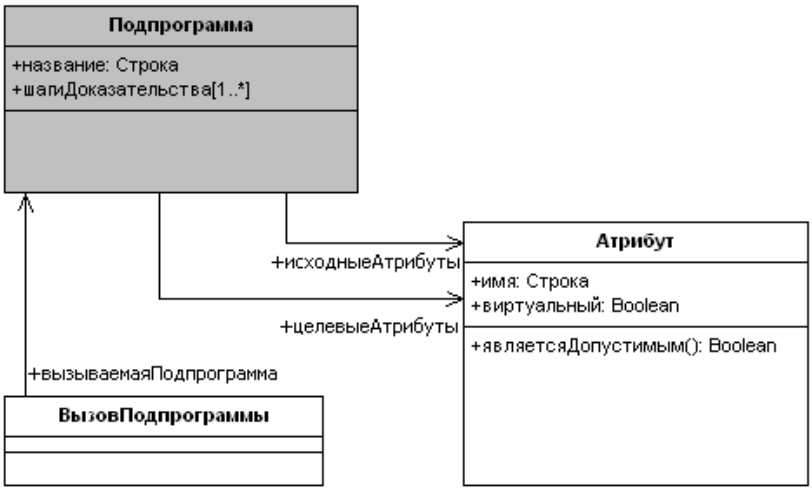


Рис. 3. Модель данных подпрограммы

Шаги доказательства, формируемые в процессе вывода, представляют собой последовательность объектов, каждый из которых содержит ссылки на достижимые атрибуты и элемент программы – *программный терм*, применение которого обеспечивает их достижимость. Выделяются три вида программных термов:

- ✓ Функциональная связь.
- ✓ Вызов подпрограммы.
- ✓ Вариантная часть (в виде пары ветвей условия).

Функциональные связи и вызовы подпрограмм называются *предложениями вычислимости* (ПВ).

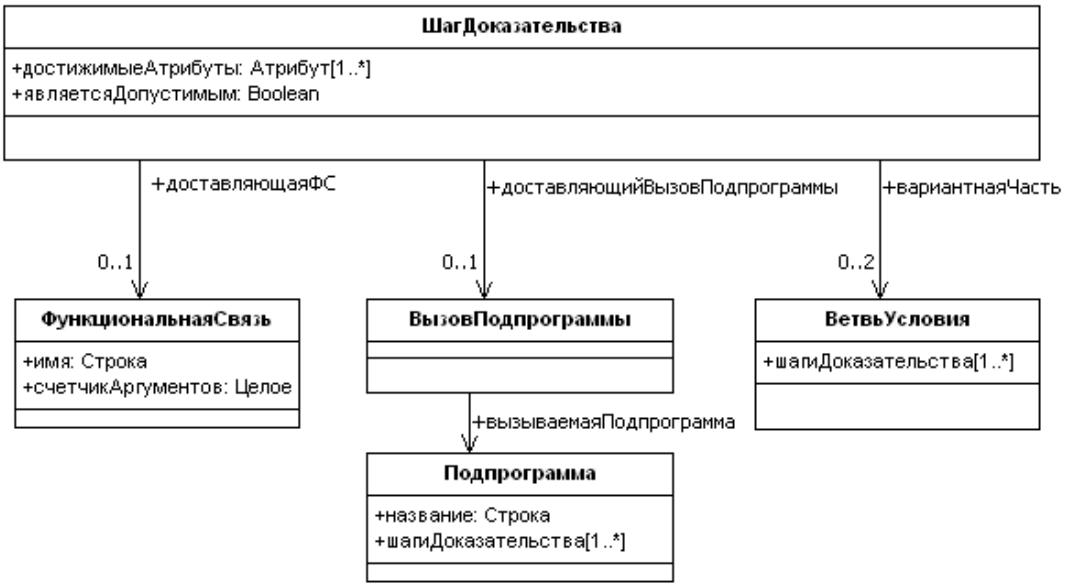


Рис. 4. Модель данных результатов вывода

Алгоритм планирования принимает на вход описанные выше структуры данных и постановку задачи. При осуществлении доказательства используются счетчики – если при обработке очередного атрибута счетчик аргументов ФС равен нулю (т.е. достижим атрибут – результат ФС), то формируется очередной шаг доказательства. Если на ФС наложено условие, то шаг помещается в список шагов доказательства текущей подпрограммы и соответствующей ветви условия. Условие ФС добавляется в список условияДопустимости и с помощью метода являетсяДопустимым, проверяется достижимость атрибута на всех ветвях условия. При получении утвердительного ответа он помещается в список достижимыеАтрибуты шага доказательства, содержащего ветви условия.

При осуществлении вывода на схемах с подсхемами используется стратегия *динамической развертки* – вывод всех возможных аргументов вызова внутренних подпрограмм в текущей подпрограмме, рекурсивный запуск планирования на очередной внутренней подпрограмме и продолжение работы во внешней подпрограмме.

Процесс *синтеза программы* осуществляется на основе списка шагов доказательства, полученного в результате планирования. Предлагаемый метод синтеза содержит два этапа: очистка и сборка программы. Очистка включает в себя обработку списка шагов доказательства «снизу-вверх» и удаление элементов, не участвующих в получении целей. Набор шагов, полученный в результате очистки, является абстрактной схемой программы. Процесс сборки реализует конечные потребности пользователя, например составление текста программы с использованием синтаксиса того или иного языка программирования.

Далее, алгоритмы расширяются возможностями **планирования и синтеза программ с явной и неявной рекурсией**. Представлена формальная модель доказательства в классе РДС-моделей (рис. 5).

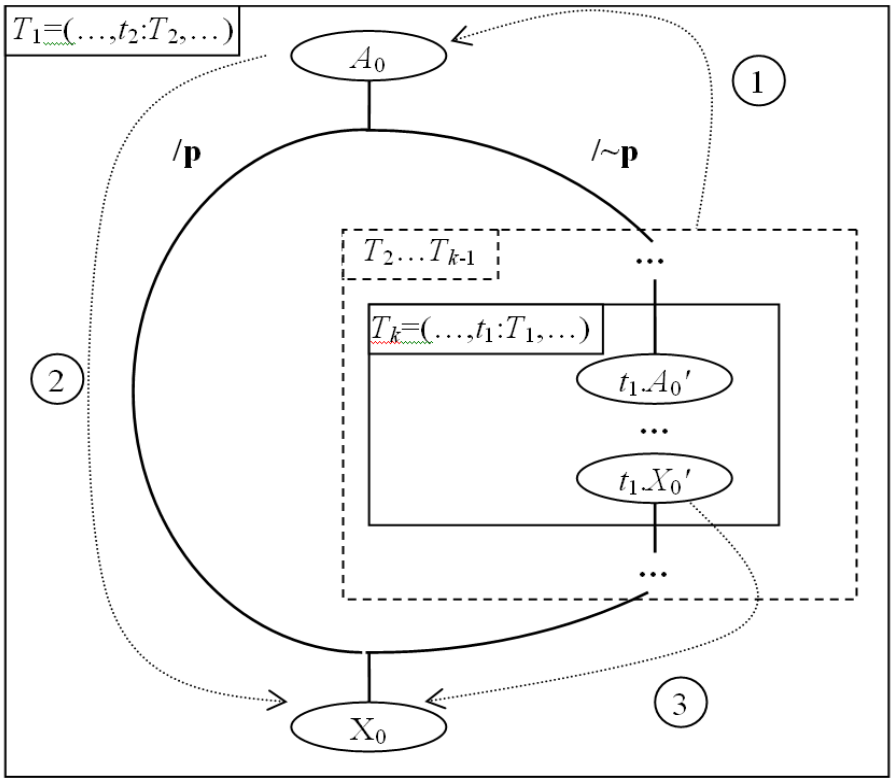


Рис. 5. Структура РДС-модели

На диаграмме прямоугольниками изображены схемы РДС-модели. В левом верхнем углу каждой схемы отмечено ее имя и имя атрибута, порождающего рекурсивный вызов. Эллипсами отображены исходные (A_0) и целевые (X_0) атрибуты согласно постановке задачи на схеме. Жирными линиями отображается последовательность ФС схемы (*f_set*), дуга в левой части отображает условные ФС альтернативной ветви *p*, в правой – ФС ветви *~p*. В ветви *~p* содержится ряд вызывающих друг друга промежуточных подсхем ($T_2 \ldots T_{k-1}$) и, наконец, схема T_k, содержащая

обращение $t_1{:}T_1$ к головной схеме T_1. Стрелками (1), (2), (3) обозначена предлагаемая последовательность этапов доказательства.

Первый этап включает в себя *идентификацию рекурсии* – выявление рекурсивного характера подсхемы, а также построение *транзитивного замыкания* – определение аргументов рекурсивной процедуры. Будем называть вызов головной подпрограммы T_1 *внешним*, а вызов $t_1{:}T_1$ – *внутренним* вызовом. Если на вход внутреннего вызова подается множество аргументов, не пересекающееся с аргументами внешней процедуры, то внутренний вызов не является рекурсивным. Если аргументы внутреннего вызова включают все аргументы внешней процедуры – вызов является рекурсивным. Если же на внутренний вызов подается лишь часть аргументов внешнего вызова, выполняется попытка построения замыкания – итеративно определяются аргументы внутреннего вызова при сокращенном множестве аргументов внешнего вызова пока не достигается устойчивое, пустое или непересекающееся множество аргументов внутреннего рекурсивного вызова.

Второй и третий этапы вывода осуществляются на рекурсивной подсхеме и выполняются согласно принципу структурной индукции – устанавливается выводимость $(t{:}T_1, A_0, X_0/p)$ (база индукции), затем, исходя из предположения, что $(t_k{:}T_k, t_1.A_0', t_1.X_0')$ доказуемо (гипотеза индукции), выполняется доказательство теоремы $(t_1{:}T_1, t_1.X_0', X_0/{\sim}p)$. Предлагаемый метод позволяет синтезировать программы, в которых предусмотрен выход из рекурсии по одной из ветвей условия.

Компиляция РДС-модели не требует каких-либо дополнительных действий, т.к. выявление и обработка рекурсивных вхождений выполняется на стадии вывода. Для этих целей в структуры данных доказательства добавляются новые элементы (рис. 6):

- ✓ ссылка внешнийВызовПодпрограммы – указывает на вызов подпрограммы, который инициирует текущий вызов;
- ✓ свойство логического типа содержитРекурсивныйВызов объекта Условие – используется для выделения рекурсивной ветви вариантной части.

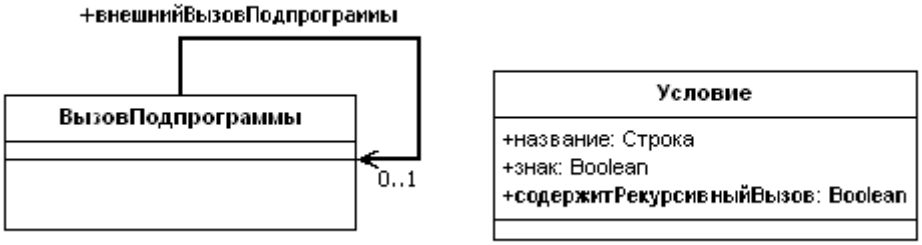


Рис. 6. Модификация модели данных для вывода на РДС-модели

Алгоритм вывода модифицируется следующим образом: при получении очередного вызова подпрограммы из стека, свойству

внешнийВызовПодпрограммы назначается текущий вызов. Далее, производится восходящее перечисление всех внешних вызовов и проверяется идентичность их схем той схеме, которая соответствует текущему вызову. При совпадении схем и обнаружении сужения списка аргументов внешний и внутренний вызовы предварительно передаются в процедуру построения замыкания. После этого делается окончательный вывод о наличии рекурсивного вызова и выполняется его обработка.

Процесс извлечения рекурсивной программы из доказательства происходит без изменений, согласно алгоритму синтеза для НДС-моделей.

Алгоритмы вывода и синтеза программ в работе представлены в виде блок-схем, псевдокода и описания программной реализации на языке Java. Приведены примеры исходной спецификации в формате RDF/XML и результаты синтеза.

В третьей главе предлагаются методы и результаты теоретической и экспериментальной оценки производительности алгоритма.

Целью расчетов, приведенных в первом параграфе главы, является математическое обоснование показателей скорости работы алгоритма относительно объема НДС-модели. Для выявления зависимости принимаются некоторые допущения:

- ✓ пусть r обозначает количество схем в модели;
- ✓ количество ФС каждой схемы одинаково и равно s;
- ✓ количество аргументов каждой ФС одинаково и равно a;
- ✓ количество непервичных атрибутов в каждой схеме равно u;
- ✓ количество доставляющих ФС каждой подсхемы равно b;

Для предельного случая, когда все ФС являются условными и в результатах планирования участвуют все подсхемы, общее время планирования определяется следующей формулой:

$$T = r(T^{нач} + s(at^{арг.ФС} + t^{ш/д} + t^{усл.ФС} + t^{пров.доп} + t^{усл.ветви}) + u(bt^{арг.п/п} + t^{стекп/п})) \quad (*)$$

В расчет включены следующие вычислительные затраты:

$T^{нач}$ – начальный этап: создание подпрограммы, сброс счетчиков и др. При больших объемах схем затраты являются пренебрежительно малыми;

$t^{арг.ФС}$ – время получения ФС, в которую атрибут входит в качестве аргумента и уменьшение счетчика аргументов ФС;

$t^{ш/д}$ – время создание шага доказательства и помещение его в список;

$t^{усл.ФС}$ – время проверки наличия условия у ФС;

$t^{пров.доп}$ – время проверки допустимости ФС;

$t^{усл.ветви}$ – получение ветви условия, соответствующей условию ФС;

$t^{арг.\ п/п}$ – время добавления аргумента вызова подпрограммы;

$t^{стек\ п/п}$ – время добавления/получения вызова подпрограммы из стека.

Таким образом, в случае вывода на НДС-моделях, время работы алгоритма сохраняет линейную зависимость от количества ПВ и выражается полиномом первой степени T=O(r, s, a, u, b)

Целью расчетов приведенных во втором параграфе является *получение вида зависимости вычислительных затрат от объема данных РДС-модели*. За исключением дополнительных затрат на поиск рекурсивных вхождений и построение замыканий, введение рекурсивных конструкций не ухудшает общей оценки скорости работы алгоритма, т. к. вывод на подзадачах (t:T_1, A_0, X_0/p) и (t_1:T_1, t_1.X_0', X_0/~p) (2 и 3 этапы на рис. 5) полностью включается в основной алгоритм поиска решения.

В параграфе рассматривается предельно сложный случай РДС-модели. Пусть n обозначает общее количество атрибутов схемы. Предположим, что n для всех схем одинаково. Так как $s \le n$, примем $s=n$. Предположим, что каждая схема включает в себя вызовы всех подсхем модели. Тогда $u=r$ и $r \le n$. Примем $r=n$, следовательно $u=n$. С учетом принятых допущений оценка сложности алгоритма поиска рекурсивного вхождения определяется $T_{поиск\ рекурсии} \le O(n^3)$.

Далее рассматривается случай, когда рекурсивное вхождение встречается в каждой схеме, опосредовано всеми подсхемами и затрагивает все ФС этих схем. При допущении, что построение замыкания требует максимального количества итераций, равного количеству доставляющих ФС каждой подсхемы (т.е. равного b),), сложность алгоритма построения всех замыканий определяется $T_{замыкание} \le O(b \cdot n^3)$.

Полученные результаты позволяют сделать вывод о том, что скорость работы алгоритма вывода на РДС-модели не выходит за рамки полиномиальной функции третьей степени. Отмечается, что предельный случай имеет мало общего с реальными практическими задачами, где оценки сложности алгоритмов поиска рекурсии и построения замыкания обычно не превышают $O(n^2)$ и $O(b \cdot n^2)$ соответственно.

В третьем параграфе описаны результаты *экспериментальной оценки* скорости алгоритма для пяти типов задач: линейные программы, программы с условиями, подпрограммами, рекурсией с построением и без построения замыкания. Результаты тестов приведены на рис. 7.

На вход машины вывода подавались сгенерированные схемы с возрастающим числом ФС, связанных таким образом, чтобы обеспечить необходимость обработки всех ПВ. Генерация и вывод проводились многократно для сглаживания побочных эффектов. При создании схем с условиями, из n ФС схемы в каждую условную ветвь помещалось (n–6)/2 ФС, чтобы обеспечить затраты, близкие к максимальным согласно (*). Оценка для программ с рекурсией выполнялась на моделях, содержащих один рекурсивный вызов, опосредованный всеми подсхемами модели и требующий их двукратного прохождения для построения замыкания.

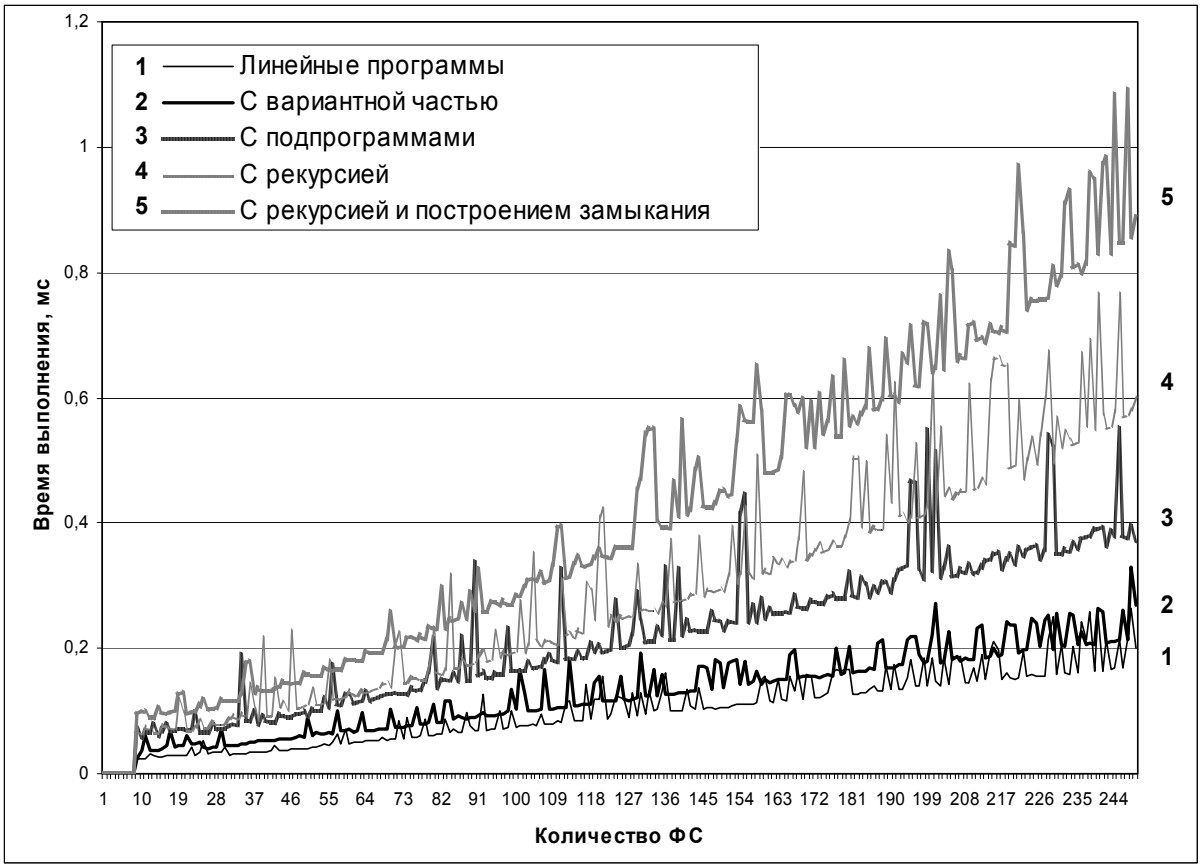


Рис. 7. Результаты экспериментальной оценки эффективности алгоритма

Из представленных результатов следует, что затраты на обработку условных ФС замедлили скорость примерно на ¼ относительно линейного случая. Затраты на обработку подсхем увеличили время работы примерно в два раза. Для случаев планирования программ с рекурсией и построением замыкания было выполнено сглаживание побочных эффектов (выбросов) и выполнена полиномиальная и экспоненциальная аппроксимация. Анализ полученных трендов подтвердил соответствие экспериментальных результатов заявленным полиномиальным оценкам.

В четвертой главе предлагаются методы преобразования спецификаций языка Пролог к конструкциям теории С-моделей. На основе полученных результатов делается вывод о классе задач логики первого порядка, которые могут быть решены разработанной машиной вывода.

Предлагается следующий метод трансформации:

1. Множество предикатов P представляется в виде совокупности схем С-модели:

$M = (T_0,\ldots,T_n)$, где $T_i \in M \leftrightarrow P_i \in P$. (символ "$\leftrightarrow$" здесь и далее интерпретируется как «взаимно-однозначное соответствие»)

2. Переменные, входящие в предикат преобразуются в собственные элементарные атрибуты заголовка схемы:

$P_i(r_0,\ldots,r_j) \leftrightarrow T_i = (\ldots,r_0,\ldots,r_j,\ldots)$, $t_i \in E$, $i=1\ldots j$

3. Создается набор ФС *f_set*, реализующих логику предиката – поиск значений переменных и вычисление истинностных значений путем подстановки индивидных символов:

$P_i(r_0,\ldots,r_j) \leftrightarrow T_i = (\ldots,r_0,\ldots,r_j,\ldots \mid f\_set)$, $f\_set = (\ldots, f^{l}_{nm}: r_n,\ldots,r_m \rightarrow r_l, \ldots)$

Отмечается, что реализация предикатов в виде ФС требует дополнительного исследования. Одним из вариантов реализации может быть язык запросов к базе фактов, подобный SQL.

4. Предикаты-факты отражают множество кортежей, удовлетворяющих соответствующей схеме, и определяют реализацию ФС, объявленных в предыдущем пункте.

5. Каждое вхождение предиката-посылки в правую часть Хорновского правила, определяется атрибутом непервичного типа в заголовке схемы, соответствующей предикату-следствию этого правила:

$P_0 :\!- P_1,\ldots,P_n \leftrightarrow T_0 = (\ldots, t_1:T_1,\ldots, t_n:T_n,\ldots)$

6. Переменные, входящие в предикаты-посылки правой части Хорновского правила преобразуются в атрибуты заголовка схемы, соответствующей предикату-следствию и служат доставляющими атрибутами подсхем, соответствующих предикатам правой части правила.

$P_0 :\!- \ldots, P_i(r_0,\ldots,r_j),\ldots \leftrightarrow T_0 = (\ldots, r_0,\ldots, r_i,\ldots, t_i:T_i \mid$

$f^{ti}_{r0}: r_0 \rightarrow t_i.r_0,\ldots, f^{ti}_{rj}: r_j \rightarrow t_i.r_j,\ldots, f^{r0}_{ti}: t_i.r_0 \rightarrow r_0,\ldots, f^{rj}_{ti}: t_i.r_j \rightarrow r_j,\ldots)$

7. Функции, входящие в исходную спецификацию (если таковые допускаются) преобразуются в ФС соответствующих схем.

Приведенные правила дают возможность преобразования большинства стандартных спецификаций языка Пролог к конструкциям С-моделей. Кроме того, приведенный метод допускает условные и рекурсивные конструкции, которые являются важной составляющей Пролога. При этом на исходные данные накладываются ограничения НДС- и РДС-моделей, что несколько сужает круг решаемых задач для исчисления первого порядка.

Далее, на основе предложенной методики предлагается метод адаптации разработанной машины вывода для решения ряда задач на Хорновских выражениях в традиционных логических исчислениях. Рассматривается случай С-модели без подсхем и вариантной части, состоящей из одной *простой схемы* $M=(T)$, представляющей собой множество ФС, вовлекающих только примитивные атрибуты. Переход к примитивным типам позволил задать следующие интерпретации конструкций С-моделей:

Логика высказываний (ЛВ). Каждой величине a_{ij} ставится в соответствие высказывание A_{ij}, а функциональный символ интерпретируется как знак логического следствия. Тогда множество ФС схемы может быть представлено в виде набора Хорновских правил:

$A_0 \leftarrow A_{01} \wedge \ldots \wedge A_{0n}, \ldots, A_k \leftarrow A_{k1} \wedge \ldots \wedge A_{kn}.$ (1)

Задача планирования $S=(A_0,X_0,T)$, при $A_0 = \{a_1, \ldots, a_i\}$, $X_0 = \{a_{i+1}, \ldots, a_j\}$, для постановки задачи проверки выполнимости набора высказываний интерпретируется как $X_0 \leftarrow A_0$ и соответствует:

$$A_1 \ldots A_i \neg A_{i+1} \ldots \neg A_j \qquad (2)$$

Логика первого порядка (ЛПП). Используя метод обратной пропозиционализации, высказывания (1,2) приводятся к выражениям ЛПП:

$$\forall z(A_0(z) \leftarrow A_{01}(z) \wedge \ldots \wedge A_{0n}(z)_{, \ldots,} A_k(z) \leftarrow A_{k1}(z) \wedge \ldots \wedge A_{kn}(z) \qquad (3)$$

$$A_1(z) \ldots A_i(z) \ \neg A_{i+1}(z) \ \ldots \neg A_j(z))$$

Дескриптивная логика (ДЛ). Известно, что выражения ДЛ могут быть приведены к высказываниям ЛПП, если они не вовлекают в предикаты более двух переменных. Согласно правилам трансформации Хорновские правила в ДЛ записываются в виде аксиом вида: $A_0 \subseteq A_{01} \cap \ldots \cap A_{0n}$.

Задача категоризации в ДЛ: $A_0 \subseteq A_{ij}$ – ? аналогичным образом трансформируется в выражение ЛПП. Все это в совокупности позволило сформулировать следующую постановку проблемы (3) в виде правил ДЛ:

$$A_0 \subseteq A_{01} \cap \ldots \cap A_{0n}, \ldots, A_k \subseteq A_{k1} \cap \ldots \cap A_{kn}$$

$$A_1(z) \equiv \top \ \ldots \ A_i(z) \equiv \top \qquad A_{i+1}(z) \equiv \bot \ \ldots \ A_j(z) \equiv \bot.$$

Практическая реализация изложенных методов заключалась в создании конвертеров для форматов данных, используемых известными алгоритмами вывода (табл. 1).

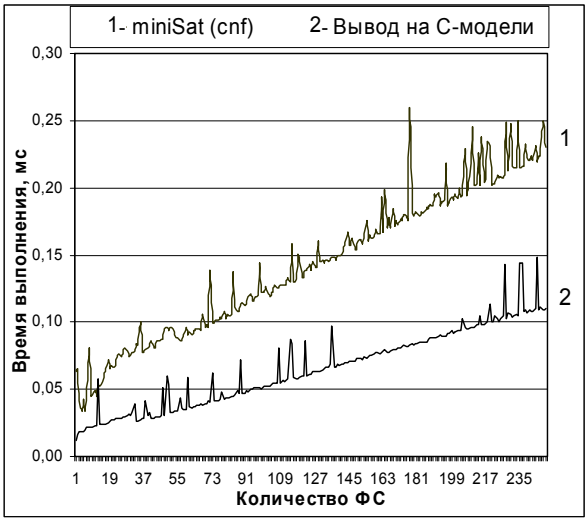
Табл. 1 Форматы данных и соответствующее логическое исчисление

Машина вывода	**Логическое исчисление**	**Формат данных**
miniSat (sat4j)	ЛВ	Cnf
Darwin 1.4.1	ЛПП	Tme
paradox2.2, iProver0.2	ЛПП	Tptp-fof
Pellet	ДЛ	Owl-rdf

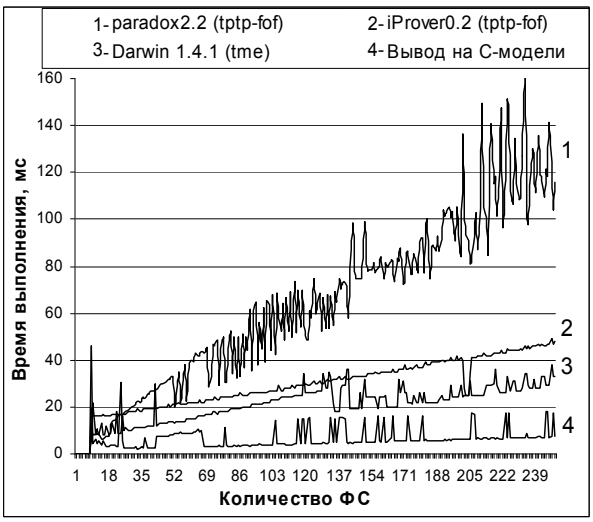
Выбор реализаций машин вывода объясняется следущими соображениями. Пакет Sat4j содержит реализацию алгоритма miniSat, показавшего максимальную скорость по итогам конкурса решения SAT-проблемы в 2006 г. Darwin, Paradox, iProver – номинанты конкурса CADE ATP System Competition (CASC) 2007, проводимого в рамках ежегодной конференции по автоматизированному логическому выводу. Пакет Pellet считается одной из наиболее высокопроизводительных реализаций среди свободно распространяемых машин вывода на ДЛ.

Результаты испытаний приведены на рис. 8. Отметим, что при измерении времени работы алгоритмов на ЛПП и ДЛ учитывается время загрузки исходных данных (согласно требованиям CASC 2007), поэтому на графике (а) результаты работы предложенного алгоритма отличаются от аналогичных результатов (б) и (в).

а)



б)



в)

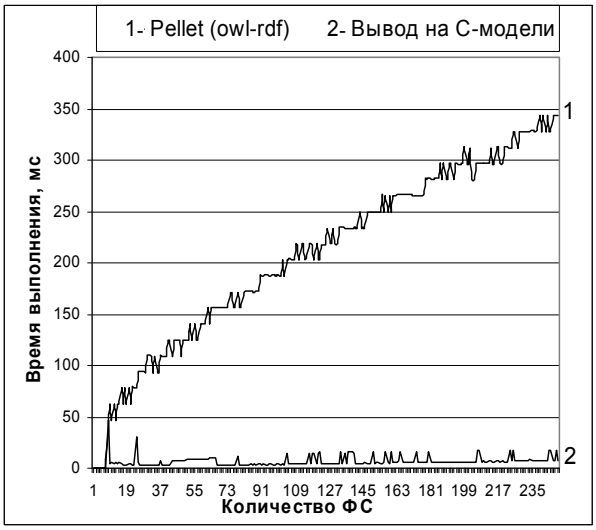


Рис. 8. Результаты сравнения производительности машин вывода (а – на ЛВ, б – на ЛПП, в – на ДЛ)

Приведенные результаты экспериментов не оцениваются как абсолютные показатели, однако демонстрируют сопоставимость производительности предлагаемой реализации по отношению к аналогам.

В заключении приведено обобщение основных результатов диссертационной работы.

В приложения вынесены: полная модель данных, формат файла описания С-модели, Java-код алгоритмов планирования и очистки, пример спецификации РДС-модели для проведения тестов, примеры трансформации С-модели в форматы различных логических исчислений.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

В ходе исследования получены следующие результаты:

1. Разработан алгоритм планирования и синтеза, демонстрирующий линейную зависимость количества вычислительных операций относительно объема спецификации при планировании программ с условиями и подпрограммами и полиномиальную зависимость не выше третьей степени при планировании программ с рекурсией.
2. Предложена формальная модель трансформации структурной вычислительной модели в выражения языка Пролог. Тем самым заложена основа для применения разработанных алгоритмов в различных системах интеллектуальной обработки информации.
3. Предложены методы интерпретации ряда задач, выраженных на языках традиционных логических исчислений в формат задач для разработанной машины вывода. Реализованы преобразователи исходных спецификаций, выполнено экспериментальное сравнение скорости вычислений и показана высокая эффективность разработанной машины вывода по сравнению с аналогами.

ОСНОВНЫЕ ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1. Новосельцев В. Б., Пинжин А. Е. Реализация эффективного алгоритма синтеза линейных функциональных программ // Известия Томского политехнического университета. – Томск : изд-во ТПУ, 2008. – Т. 312. – № 5. – С. 32–35.

2. Пинжин А. Е., Новосельцев В. Б. Эффективный алгоритм синтеза программ с условиями и подпрограммами // Известия Томского политехнического университета. – Томск : изд-во ТПУ, 2008. – Т. 313. – № 5. – С. 77–84.

3. Пинжин А. Е. Алгоритм синтеза программ с явной и неявной рекурсией // Известия Томского политехнического университета. – Томск : изд-во ТПУ, 2008. – Т. 313. – № 5. – С. 84-88.

4. Пинжин А. Е. Реализация системы логического вывода для логики первого порядка на базе структурных функциональных моделей // Материалы 9-й Всероссийской конференции молодых ученых по математическому моделированию и информационным технологиям (YM-2008). – Кемерово, 2008. – С. 60–61;

5. Пинжин А. Е. Реализация системы логического вывода на основе структурных функциональных моделей для ряда логических исчислений // Материалы международной конференции «Программные системы: теория и приложения (PSTA-2009)». – Переславль-Залесский, 2009. – С. 38–44;

6. Пинжин А. Е. Построение эффективной машины логического вывода для ряда логических исчислений на основе предварительной подготовки исходных данных // Материалы международной конференции «Теоретические и прикладные вопросы современных информационных технологий (ТиПВСИТ’2009)». – Улан-Удэ, 2009. С. 363–368;