Р О С С И Й С К А Я А К А Д Е М И Я Н А У К
Институт систем информатики им. А. П. Ершова

На правах рукописи

Петров Евгений Сергеевич

МЕТОДЫ ИНТЕГРАЦИИ ЛОГИЧЕСКОГО ПРОГРАММИРОВАНИЯ И ПРОГРАММИРОВАНИЯ В ОГРАНИЧЕНИЯХ

05.13.11 — математическое и программное обеспечение вычислительных машин, комплексов и сетей

Автореферат диссертации на соискание ученой степени кандидата физико-математических наук

Новосибирск 1999

Работа выполнена в Институте систем информатики имени А. П. Ершова Сибирского отделения Российской академии наук (ИСИ СО РАН).

Научный руководитель — *доктор физико-математических наук, доцент Ягно Т. М.*

Официальные оппоненты: *доктор физико-математических наук, профессор Евстигнеев В. А.*

доктор физико-математических наук, профессор Пальчунов Д. Е.

Ведущая организация: *Иркутский государственный университет*

Защита состоится 16 апреля 1999 в 14 час. 30 мин. на заседании диссертационного совета К0003.93.01 в Институте систем информатики Сибирского отделения РАН по адресу:

630090, Новосибирск, пр. акад. Лаврентьева, 6.

С диссертацией можно ознакомиться в читальном зале библиотеки ВЦ СО РАН (пр. акад. Лаврентьева, 6)

Автореферат разослан

Ученый секретарь
диссертационного совета
К0003.93.01
к.ф.-м.н. М. А. Бульонков

Общая характеристика работы

Актуальность темы. Современные логическое программирование и программирование в ограничениях представляют декларативный взгляд на программирование и рассматривают решение задачи как объект, а не как процесс (чем характеризуется императивное программирование). В логическом программировании выполнение программы соответствует конструктивному выводу логической формулы из заданных аксиом. Для программирования в ограничениях характерно систематическое использование инвариантов, математических свойств и законов, которые присутствуют в решаемой задаче.

В программировании в ограничениях естественным образом возникает большое число классических трудно решаемых и алгоритмически неразрешимых задач. Целью программирования в ограничениях является разработка эффективных алгоритмов, полностью решающих задачи некоторого выделенного класса и вырабатывающих корректную, но неполную информацию о решении задачи в остальных случаях.

Такая неполнота является частным видом "недоопределенности", изучаемой в рамках недоопределенных вычислительных моделей, предложенных А. С. Нариньяни в начале 80-х годов в качестве средства представления и обработки знаний. С начала 90-х годов недоопределенные вычислительные модели с успехом используются в программировании в ограничениях. Среди достоинств недоопределенных вычислительных моделей как метода программирования в ограничениях выделяются два: 1) применимость к решению задач со смесью данных различных типов и 2) возможность настройки на предметную область решаемой задачи.

В рамках программирования в ограничениях разработан ряд специальных методов и на их основе создано большое число систем для решения прикладных задач в области автоматического проектирования, ресурсного планирования, задач моделирования в различных естественнонаучных дисциплинах.

Для решения задач численного моделирования, возникающих в химии, роботике, механике, финансовой математике в программировании в ограничениях и недоопределенных моделях широко используется интервальное представление неизвестных значений в сочетании с потоковым принципом управления. Актуальным является изучение теоретических свойств таких методов, в частности, их сходимости к точному решению, что позволит точнее определить класс эффективно решаемых

задач и дальнейшее направление совершенствования.

В настоящее время методы программирования в ограничениях активно используются в рамках двух парадигм: объектно-ориентированной и логической. Более органичным сочетанием, в силу декларативности обеих его составляющих, представляется логическое программирование в ограничениях. Однако все известные системы логического программирования в ограничениях изначально ориентированы на строго определенные классы задач. Актуальным является создание систем логического программирования в ограничениях, настраиваемых на предметную область решаемой задачи, и разработка соответствующих методов.

Богатый класс задач из области программирования в ограничениях образуют так называемые задачи с конечными областями значений переменных — хорошо формализованные фрагменты задач распределения ресурсов и планирования. Важными объектами в таких задачах являются конечные множества и графы (отображения конечных множеств). В настоящее время задачи с конечными областями значений переменных принято сводить к задаче целочисленного (линейного) программирования, которая затем решается путем организованного в потоковом стиле сжатия областей значений переменных. В результате такой трансформации, как правило, увеличивается размерность задачи и на втором плане оказывается ее комбинаторная сущность. Актуальным подходом к решению задач с конечными областями значений переменных является создание средств для представления и обработки данных комбинаторной природы.

Целью диссертационной работы является теоретическое исследование вопросов сходимости в недоопределенных вычислительных моделях, развитие таких моделей за счет интеграции с логическим программированием, за счет создания эффективных средств представления обработки множеств; разработка инструментальных систем на основе недоопределенных вычислительных моделей.

Научная новизна работы состоит в следующем:

- разработан метод интеграции недоопределенных вычислительных моделей и логического программирования, который повышает эффективность логических программ путем автоматического учета в логическом выводе неполной информации; для случая интервальной недоопределенности установлено необходимое и достаточное условие сходимости вычислений в таких моделях при решении линейных уравнений, установлена связь со сходимостью классических методов Зейделя,

Якоби;

- разработаны средства представления и обработки множеств в недоопределенных вычислительных моделях, которые позволяют эффективно использовать интенсиональные и экстенсиональные спецификации множеств для решения комбинаторных задач; разработаны язык для описания таких задач и его интерпретатор LogiCalc, использующий предложенные средства представления и обработки множеств;
- разработана Интервальная библиотека для системы логического программирования ECL^{i}PS^e, предназначенная для автоматического учета в логическом выводе нелинейных ограничений над вещественными значениями; библиотека включает средства для спецификации массовых ограничений, символьных преобразований, управления процессом вычислений; на основе Интервальной библиотеки разработаны приложения для решения задач автоматического проектирования и моделирования финансовых операций.

Практическая ценность работы состоит в разработке нового метода представления и обработки неполных (неточных) данных в логических программах, легко адаптируемого к данным различных типов и различным классам задач, в создании Интервальной библиотеки, позволяющей учитывать при логическом выводе нелинейные ограничения.

Апробация работы и публикации. Исследования по теме диссертации были поддержаны грантом 96–01–01608 Российского фонда фундаментальных исследований и грантом 98–06 Франко-русского центра по прикладной математике и информатике им. А. М. Ляпунова.

Результаты работы неоднократно докладывались на семинаре лаборатории Искусственного интеллекта, на 8-й Европейской летней школе “Логика, язык, информация” (Прага, Чехия, 1996), 3-й конференции “Перспективы системной информатики” (Новосибирск, 1997), 6-й Скандинавской конференции по искусственному интеллекту (Хельсинки, Финляндия, 1997), 3-м Сибирском конгрессе по индустриальной и прикладной математике (Новосибирск, 1998), 3-й Объединенной конференции по разработке систем, основанных на знаниях (Смоленице, Словакия, 1998), на объединенном семинаре лаборатории Системного программирования и лаборатории Искусственного интеллекта ИСИ СО РАН.

Структура и объем работы. Диссертационная работа состоит из введения, четырех глав, заключения, списка литературы из 69 наименований. Объем работы составляет 122 страницы. Работа включает 4 таблицы и 9 рисунков.

Содержание работы

Во введении обосновывается актуальность темы диссертации, кратко характеризуются достижения в области логического программирования и программирования в ограничениях; дается обзор работы по главам.

Глава 1 посвящена недоопределенным вычислительным моделям, их свойствам и рассмотрению таких моделей в контексте программирования в ограничениях.

В **разделе 1.1** вводятся термины, используемые на протяжении всей работы, дается краткий обзор методов решения задач удовлетворения ограничениям (**1.2**), устанавливается связь между задачами удовлетворения ограничениям и недоопределенными вычислительными моделями (**1.3**), приводится общий алгоритм вычислений в таких моделях (**1.4**). В **разделе 1.6** исследуется сходимость вычислений в моделях с интервальной недоопределенностью при решении линейных уравнений.

Принято считать, что основы программирования в ограничениях были заложены в середине 70-х годов А. Макверсом, У. Монтанари, Е. Фредером, хотя ключевые идеи высказывались еще Ю.И. Журавлевым, Э. Тыугу, Д. Уолцем с начала 60-х годов.

В программировании в ограничениях решаются так называемые задачи удовлетворения ограничениям (CSP, "Constraint Satisfaction Problem"), в которых по заданной системе ограничений (набору формул) некоторой сигнатуры и модели этой сигнатуры требуется отыскать набор значений, удовлетворяющий в этой модели одновременно всей системе. Сигнатура перечисляет средства, доступные для описания предметной области, а ее модель фиксирует способ понимания этих средств, принятый в этой предметной области. Выберем некоторую сигнатуру и ее модель и будем ссылаться на эту модель как на "стандартную модель".

Большое число классических трудно решаемых и алгоритмически неразрешимых задач является задачами удовлетворения ограничениям. Поэтому для решения CSP вместо точных используются приближенные эффективные алгоритмы, полностью решающие задачи некоторого выделенного класса, а в остальных случаях вырабатывающие корректную, но неполную информацию о решении.

Такая неполнота является частным видом "недоопределенности", описанной в классической работе А.С. Нариньяни "Недоопределенность в системе представления и обработки знаний" и большом числе других публикаций по данной тематике, начинающихся с 1980 г.

Для исследования и использования на практике феномена недоопределенности в работе используется понятие “недоопределенного расширения стандартной модели” (н-расширения). Н-расширение является моделью той же сигнатуры, что и стандартная модель. Ее носитель образован недоопределенными значениями (н-значениями) — наборами обычных значений; интерпретация аппроксимирует стандартную интерпретацию. На практике н-значение компактно представляет диапазон возможных стандартных значений переменной.

Интерпретация отношения в н-расширении задается при помощи набора вычислительных функций, каждая из которых, исходя из стандартной интерпретации данного отношения, оценивает н-значение одного аргумента через н-значения остальных.

В н-расширении каждая система ограничений обладает таким решением $\gamma_\top$, что для каждого другого ее решения ${}^\star\gamma$ и каждой переменной x выполняется ${}^\star\gamma(x) \subseteq \gamma_\top(x)$. Решение $\gamma_\top$ называется недоопределенным решением системы ограничений.

Н-решение системы ограничений находится методом недоопределенных вычислений (н-вычислений). Метод работает с некоторым “текущим” означиванием, постепенно приближая его к н-решению. Как метод программирования в ограничениях н-вычисления относятся к методам обеспечения локальной совместности. Н-вычисления складываются по принципу управления по данным из исполнения отдельных вычислительных функций. Целью этих действий является согласование текущего решения со всей системой ограничений.

Систему ограничений можно изобразить в виде двудольного графа, вершинами которого являются переменные и ограничения, а ребра обозначают вхождения переменных в ограничения. Такой двудольный граф называется сетью ограничений. Сеть ограничений состоит из звезд. Центром каждой звезды является ограничение, лучи расходятся к переменным и константам, связанным этим ограничением. Н-вычисления можно представлять как распространение по сети ограничений волны, состоящей из доопределений н-значений. Н-вычисления завершаются, когда такая волна иссякает.

Н-вычисления и другие методы программирования в ограничениях с успехом используют интервальное представление неизвестных значений и потоковый принцип управления для решения задач численного моделирования в химии, роботике, механике, финансовой математике. Несмотря на популярность такого подхода, число работ, в которых ис-

следуется его сходимость к точному решению и другие теоретические свойства, невелико. Среди таких работ необходимо отметить выполненные коллективом, который возглавляет Ю.И. Шокин.

В **разделе 1.6** автор исследует сходимость в недоопределенных вычислительных моделях при решении линейных уравнений. Пусть дана система n линейных уравнений $x = Ax + f$. Будем рассматривать ее как систему n ограничений. Хотя исследуемая система ограничений содержит n^2 вхождений переменных, для простоты будем считать, что н-значение переменной x_i доопределяется на основе лишь уравнения i, т.е. требуется не более n вычислительных функций.

Предположим, что x^* — решение системы $x = Ax + f$. Будем говорить, что н-вычисления сходятся, если n-мерный интервал, даваемый н-вычислениями, при увеличении точности вычислений стремится к $\{x^*\}$. Сходимость н-вычислений влечет единственность решения системы $x = Ax+f$. Относительно сходимости н-вычислений при решении системы $x = Ax + f$ в работе доказываются следующие теоремы.

Теорема 1.1 (необходимое условие сходимости). Пусть для системы линейных уравнений $x = Ax + f$ сходятся н-вычисления.

Тогда найдется такой вектор-строка $a \geq \mathbf{0}$, что

$$a(E - |A|) \geq \mathbf{1}.$$

В комментарии к Т. 1.1 показано, что сформулированное условие является также и достаточным.

Теорема 1.2 (связь методов н-вычислений, Зейделя, Якоби). Пусть для системы линейных уравнений $x = Ax+f$ сходятся н-вычисления. Тогда для системы линейных уравнений $x = Ax+f$ сходятся метод Зейделя и метод Якоби.

Теорема 1.2 предваряется примером, показывающим, что обратное следствие не имеет места.

Теорема 1.3 (связь методов н-вычислений, Якоби). Пусть элементы матрицы A неотрицательны.

Тогда (н-вычисления сходятся для системы $x = Ax+f$) $\iff$ (метод Якоби сходится для системы $x = Ax + f$).

Глава 2 посвящена подходам к интеграции недоопределенных вычислительных моделей и системы логического программирования $\mathrm{ECL}^i\mathrm{PS}^e$.

В **разделе 2.1** кратко описаны способы формализации современного логического программирования посредством систем переписывания термов (**2.1.1**) и посредством сведения к семантическому программированию (**2.1.2**). Далее приводятся необходимые сведения о системе

ECL^{i}PS^e (**2.2**) и описывается представление системы ограничений в виде вычислительной сети, предназначенное для организации н-вычислений в системе ECL^{i}PS^e (**2.4.1**), и описывается организация управления (**2.4.2**).

Можно выделить три парадигмы в программировании, существенным образом использующие математическую логику в процессе решения задач: логическое программирование, логическое программирование в ограничениях и так называемое Σ-программирование.

Основы собственно логического программирования были заложены Р. Ковальским (теоретические вопросы), А. Колмероэ (первая реализация), Д. Уорреном (эффективная реализация интерпретатора языка Prolog) в середине 70-х годов. На конец 70-х — начало 80-х годов пришелся первый пик в развитии систем логического программирования.

Второй этап развития связан с применением в логическом программировании методов программирования в ограничениях. Первой промышленной системой логического программирования в ограничениях считается система CHIP (Г. Симонис, П. ван Хентенрик и др., 1988). За ней последовали Prolog-III (А. Колмероэ, 1990), CLP(BNR) (У. Олдер, Ф. Бенаму, 1990), Флэнг (А.В. Манцивода, 1991), ECL^{i}PS^e (М. Майер, И. Шимф и др., 1992).

К началу 90-х годов Ж. Коэном были сформулированы следующие принципы логического программирования в ограничениях. Логическая программа с ограничениями представляет собой конечный набор предложений вида "$g_0 \leftarrow g_1, \ldots, g_n, C$", где g_i — цели (атомарные формулы), не являющиеся ограничениями, C — множество ограничений. Вычисление логической программы с ограничениями моделируется работой недетерминированной машины. Состояние машины определяется цепочкой целей t_j, накопленной системой ограничений S и подстановкой W, хранящей вычисленные значения переменных.

На каждом шаге машина заменяет в цепочке первую цель t_0 на $g_1, \ldots, g_n$, расширяет систему ограничений до $C \cup S \cup \{g_0 = t_0\}$). Если она совместна, то подстановка W меняется на $W\theta$, задающую значения переменных, которые полностью определены новой системой ограничений. Для вычисления $W\theta$ применяются методы программирования в ограничениях, математического программирования и т.п. Новым состоянием становится $(W\theta, [g_1\theta, \ldots, g_n\theta, \ldots, t_j, \ldots], C \cup S \cup \{g_0 = t_0\})$. Заключительным является любое состояние с пустой цепочкой целей. Ответ на запрос пользователя совместно дают получившаяся подстановка и

остаточная система ограничений.

Существенным недостатком созданных по такой схеме систем (Prolog-III, CLP(BNR)) является "замкнутость", отсутствие возможности дополнить готовую систему модулем для решения систем ограничений какого-либо нового класса; имеется лишь возможность создать полностью новую систему с более широкими возможностями.

Семантическое (Σ-) программирование, предложенное в начале 80-х годов Ю.Л. Ершовым, С.С. Гончаровым, Д.И. Свириденко, синтезирует в себе большую часть логико-математических подходов в программировании: логическое, функциональное, трансформационное программирование, концепции "абстрактных типов данных", синтез программ.

В Σ-программировании выделяются язык спецификации задач и Σ-машина — абстрактное устройство, решающее эти спецификации в терминах неподвижных точек. Спецификации задач также называются Σ-программами. Σ-программа "собирается" из отдельных Σ-схем.

Каждая Σ-схема, по сути, является системой "уравнений" относительно предикатных (функциональных) переменных и базовых предикатов и констант. Отдельное такое "уравнение" выражается Σ-формулой, вид которой ограничивается так, чтобы каждая Σ-программа определяла монотонный оператор. Денотационная семантика данной Σ-программы определяется как наименьшая неподвижная точка такого оператора.

Одной из практических реализаций принципов Σ-программирования является язык Флэнг, разработанный под руководством А.В. Манциводы в начале 90-х г. и объединяющий парадигмы функционального и логического программирования. Флэнг предназначен для обработки символьной информации и решения сложных комбинаторных задач, специфицированных при помощи ограничений над переменными с конечными областями значений (конечными доменами).

Программы на Флэнге являются разновидностью Σ-программ; их поведение и денотационная семантика хорошо описываются в рамках Σ-программирования. Например, конечные домены во Флэнге моделируются переменными, связанными в Σ-формулах ограниченными кванторами.

Система ECL^{i}PS^e более тяготеет к схеме Ж. Коэна, однако новые типы данных "метатерм" и "отложенная цель", поддерживаемые этой системой, позволяют расширять возможности системы ее же средствами.

Метатерм состоит из обычного терма и термов, являющихся значе-

ниями его атрибутов. С точки зрения стандарта языка Пролог метатермы являются термами. Метатерм, состоящий из обычного терма t и термов v_i, которые являются значениями атрибутов a_i, записывается в виде t{a_1:v_1,a_2:v_2,...}. На уровне языка поддерживается создание метатермов и модификация значений их атрибутов. Способ унификации метатермов должен быть описан пользователем. Атрибуты увеличивают объем информации, участвующий в унификации, позволяют сделать эту операцию логически более сильной.

Отложенная цель состоит из прологовской цели и идентификатора, предохраняющего ее от унификации. Над отложенными целями определены три основных действия: создание, активация для исполнения и исполнение всех активных целей. В момент создания отложенная цель автоматически помещается в системный список отложенных целей. Прагматическим назначением отложенных целей является управление будущим поведением программы. Отложенная цель записывается в виде `'GOAL'(G)`.

Пусть 1) каждое н-значение $^\star a$ взаимно-однозначно кодируется некоторым константным термом $t_{^\star a}$; 2) каждая из n вычислительных функций отношения $\boldsymbol{p}/n$ вычисляется $(n+1)$-местным предикатом lib_p_k. Первые n аргументов такого предиката соответствуют аргументам вычислительной функции, а $(n+1)$-й — ее значению.

Для описания н-вычислений в терминах метатермов и отложенных целей вводится понятие направленного ограничения. Направленным ограничением называется пара (p, k), где $p = \boldsymbol{c}(x_1, \ldots, x_n)$ — обычное ограничение, k — номер одного из аргументов p. Будем говорить, что направленное ограничение (p, k) "считывает" переменные x_i $(i \neq k)$ и "записывает" переменную x_k (точнее, соответствующие н-значения).

Сеть ограничений, введенная в главе 1, преобразуется в вычислительную сеть. В отличие от сети ограничений, вычислительная сеть является направленным двудольным графом. Вершинами вычислительной сети являются направленные ограничения и переменные. Дуга от переменной к направленному ограничению показывает, что ограничение считывает переменную. Дуга в обратном направлении показывает, что ограничение записывает переменную.

Вычислительная сеть задается набором значений типов метатерм и отложенная цель. Ниже $\mathrm{Rep}(x)$ и $\mathrm{Rep}((p, k))$ обозначают представление переменной x и направленного ограничения (p, k).

Пусть переменная x с н-значением $^\star x$ считывается направленными

ограничениями (p,k), (q,l), (r,m), ... Метатерм $\mathrm{Rep}(x)$ имеет один атрибут `sd`. Первая компонента атрибута хранит представление $t_{\star x}$ значения переменной x. Вторая компонента атрибута хранит список $[\mathrm{Rep}((p,k)),\ \mathrm{Rep}((q,l)),\ \mathrm{Rep}((r,m)),\ \ldots]$ представлений направленных ограничений, считывающих переменную x.

Если направленное ограничение (p,k) считывает переменные x_i $(i \neq k)$ и записывает переменную x_k, то

$$\mathrm{Rep}((p,k)) = \texttt{'GOAL'(}lib\_p_k\texttt{(}\mathrm{Rep}(x_1)\texttt{,}\ldots\texttt{,}\mathrm{Rep}(x_n)\texttt{,}\mathrm{Rep}(x_k)\texttt{))}.$$

Вычислительная сеть, таким образом, распределяется по переменным, фрагменты сети отождествляются с отложенными целями. При срабатывании отложенная цель разрушается и исчезает соответствующая часть вычислительной сети. Чтобы поддерживать неизменным вид вычислительной сети в течение всего времени вычислений, каждый предикат lib_p_k добавляет в системный список новую отложенную цель, представляющую только что исчезнувший участок сети. Таким образом, вычислительная сеть сохраняет свой исходный вид. В заключение главы 2 разобран пример, иллюстрирующий основные положения предложенного метода.

В **главе 3** описываются две инструментальные разработки соискателя: интерпретатор LogiCalc и Интервальная библиотека для системы логического программирования $\mathrm{ECL}^i\mathrm{PS}^e$.

Разделы 3.1–3.6 посвящены средствам представления и обработки конечных множеств в интерпретаторе LogiCalc.

Средства представления и обработки множеств встраиваются в языки императивной и декларативной ориентации. Среди всех императивных языков наиболее существенным образом тип данных “множество” поддерживается в языке SETL (Дж. Шварц и др., начало 80-х). Язык SETL позволяет работать с произвольными конечными множествами (в том числе составных объектов). Конечное множество является полноправным членом иерархии типов данных языка SETL.

Язык SETL включает так называемые формирователи множеств. Стандартной математической записью этой конструкции является $\{x|C\}$, “множество всех таких x, что C”. По правилам языка SETL формирователь множества записывается в виде $\{x \in s|C\}$ или $\{e : x \in s|C\}$, где x — переменная, e — выражение, универсум s — ранее построенное множество. Формирователи множеств являются очень выразительной конструкцией.

Интерпретатор языка SETL, исполняя формирователь множества, по очереди перебирает элементы универсума и включает в множество те из них, которые обладают выделяющим свойством. Такой подход "generate and test" теряет привлекательность, если широкий универсум сочетается с жестким выделяющим условием.

Попытки использования множеств неоднократно делались и в логическом программировании. В одном из подходов (А. Довьер, Дж. Росси и др., интерпретатор {*log*}, начало 90-х) язык Пролог расширяется интерпретированным функциональным символом (`with`/2), служащим для задания множеств. Поскольку множество определяется только своими элементами, а не порядком их перечисления и кратностью их повторов, одно множество может быть задано разными способами. Это обстоятельство создает вокруг унификации множеств проблемы: 1) определение унифицируемости двух множеств является NP-полной задачей; 2) если два множества унифицируемы, то их наиболее общий унификатор не отражает информации, полученной в ходе унификации.

Создатели системы {*log*} решили последнюю проблему, отказавшись от детерминированности унификации. Например, унификация `{X,Y}` и `{0,1}` дает два частных решения: `X=0`, `Y=1` и `X=1`, `Y=0`. Однако с расширением множества число таких частных решений может стремительно расти, что порождает другую проблему — увеличение дерева поиска.

Другая разработка — это основанная на распространении ограничений библиотека Conjunto в системе логического программирования в ограничениях ECL^{i}PS^e. В библиотеке Conjunto каждое множество задается своей верхней и нижней границей относительно теоретико-множественного включения. Вычисления организуются так, что различие между этими границами может только уменьшиться. Переменные, обозначающие искомые множества, связываются отношениями $=$, $\in$, $\subseteq$ и т.п. Для задач на разбиение и упаковку имеются отношения для работы с целочисленными и вещественнозначными отображениями.

В данном случае унификация детерминирована и эффективна (относительно размера верхней границы множества), однако существенно ограничены выразительные возможности. Запрещается задавать множества интенсионально, создавать множества множеств, обозначать элементы множества переменными и затем уточнять их значения.

В **разделах 3.2–3.6** описывается система LogiCalc: входной язык интерпретатора LogiCalc (**3.2**), основные механизмы этой системы и их улучшения (**3.3**, **3.4**), средства обработки конечных интенсиональных и

экстенсиональных множеств на основе недоопределенных вычислительных моделей (**3.5**, **3.6**).

Множеством значений переменных и констант в системе LogiCalc является наименьшее множество, содержащее отрезок целых чисел [−MAX, MAX], атомы и замкнутое относительно операций построения конечных кортежей и множеств. В системе LogiCalc задача записывается в виде системы уравнений, неравенств, теоретико-множественных отношений, связывающих какие-либо выражения. Выражения состоят из констант, переменных и операций (функциональных символов).

Основой системы является метод н-вычислений, адаптированный к классу задач удовлетворения ограничениям с конечными множествами. Запись задачи на языке системы LogiCalc переводится в систему ограничений и затем для этой системы ограничений отыскивается н-решение.

От интерпретатора языка SETL систему LogiCalc отличает использование принципа "test and generate", при котором универсум множества играет второстепенную роль, а на первом плане находятся условия, определяющие принадлежность элементов множеству. При формировании множеств с широким универсумом система LogiCalc более эффективна, чем интерпретатор языка SETL.

По сравнению с системами логического программирования LogiCalc обладает более эффективными средствами обработки множеств, обеспечивающими приемлемый компромисс между точностью и временем вычислений, и более естественным и выразительным языком спецификации задач.

Далее в главе 3 описывается Интервальная библиотека для решения нелинейных ограничений в системе ECL^{i}PS^e (**3.7**–**3.11**).

Решение нелинейных систем ограничений и оптимизация при нелинейных ограничениях необходимы во многих инженерных задачах, а также задачах, возникающих в механике, экономике, химии, роботике и т.д. Созданием и исследованием необходимых для этого методов занимается нелинейное программирование. Возможна следующая типизация этих методов: численные *vs.* интервальные, оптимизация *vs.* поиск решения, ограниченная оптимизация *vs.* неограниченная оптимизация.

Примером интервального метода является интервальный метод Ньютона поиска решений, вычисляющий интервалы, гарантированно содержащие приближения по классическому методу Ньютона. Этот метод устойчив вблизи решения, способен разделять близко лежащие ре-

шения. В комбинации с другими техниками используется в системах `Helios`, NUMERICA, разработанных под руководством П. ван Хентенрика во второй половине 90-х годов.

Система CLP(BNR) также использует интервальный метод поиска решения, состоящий в нахождении интервального решения, являющегося неподвижной точкой для системы преобразований многомерных интервалов. Вычислительные модели с интервальной недоопределенностью, используемые в системе Unicalc (А.Л. Семенов, около 1990), также могут считаться интервальным методом поиска решения.

Разделы 3.8–3.11 описывают Интервальную библиотеку: способы спецификации различных типов ограничений (**3.9**), предоставляемые библиотекой средства символьных преобразований (**3.10**), специальные возможности, рекомендации по использованию, отличия от классического метода н-вычислений (**3.11**).

Интервальная библиотека является результатом интеграции недоопределенных вычислительных моделей и логического программирования и предназначается для повышения эффективности пролог-программ посредством использования нелинейных ограничений. Интервальная библиотека качественно упрощает создание приложения за счет частичной замены процесса его отладки на исследование его математической модели.

Библиотека принимает уравнения и неравенства (ограничения), записанные с помощью арифметических операций, тригонометрических, показательных и обратных к ним функций, и для каждой встречающейся в них переменной вычисляет интервал, содержащий все ее возможные значения. Вычисления возможны в вещественном и смешанном режиме. Библиотека предоставляет пользователям возможность определять собственные функции, управлять точностью вычислений, использовать массивы и массовые ограничения при задании больших систем ограничений, имеющих регулярную структуру. Использование перечисленных возможностей демонстрируется на типовых задачах, даются общие рекомендации по разделению решений, оптимальному выбору управляющих параметров.

В отличие от систем `Newton`, NUMERICA Интервальная библиотека используется в рамках промышленной системы программирования, что позволяет создавать на базе системы ECL^{i}PS^e с Интервальной библиотекой полноценные приложения. По сравнению с системой CLP(BNR) библиотека предоставляет более выразительный язык спецификации за-

дач и более широкий набор операций, используемых в ограничениях.

В библиотеке использована одна из первых версий вычислительного ядра системы Unicalc в доработанном виде. Целью создания библиотеки были взаимное расширение возможностей системы ECL^{i}PS^e и недоопределенных вычислительных моделей и демонстрация практической применимости предложенного в главе 2 метода использования недоопределенных вычислительных моделей в логическом программировании.

В **главе 4** рассказывается о результатах тестирования Интервальной библиотеки и созданных на ее основе приложениях.

Интервальная библиотека тестировалась на наборе задач для системы `Newton`, более поздней версией которой является система NUMERICA. Разработчики системы `Newton` приводят также результаты сравнения своей системы с традиционным интервальным методом, использующим для сжатия интервалов так называемый оператор Хансена–Сегупты и поиск в глубину с отсечением.

Набор задач включает тесты фиксированной размерности и масштабируемые задачи. Интервальная библиотека с успехом конкурирует с названным выше традиционным интервальным методом. На некоторых тестах фиксированной размерности Интервальная библиотека уступает системе `Newton` по быстродействию в 1.5–2.5, что можно объяснить особенностями реализации Интервальной библиотеки и системы `Newton`. На ряде масштабируемых задач библиотека либо показывает меньшие темпы роста затрат времени с ростом размерности задачи, либо абсолютно превосходит систему `Newton`.

Раздел 4.2 описывает разработанное на основе Интервальной библиотеки приложение для построения наборов оптимальных сетей (коммуникационных, электрических и т.п.). Построение набора из одной сети эквивалентно классической NP-полной задаче о кратчайшем штейнеровском дереве. Для наборов из нескольких сетей задача осложняется условиями несовместности сетей разных типов. Созданное приложение позволяет за приемлемое время доказывать оптимальность получаемых решений для задач с 40–50 вершинами. Для задач со 100–400 вершинами приложение позволяет получать несколько улучшающих друг друга приближений.

Раздел 4.5 описывает приложение для расчета инвестиционных портфелей с наименьшим риском. Исходя из статистических данных о доходности отдельных составляющих портфеля, приложение по из-

вестным методикам рассчитывает среднюю доходность, риск, коэффициенты корреляции между доходностями различных составляющих. На основе полученных данных строится модель, описывающая оптимальный портфель. Пользователю предоставлена возможность изучать эту модель в интерактивном режиме и в полуавтоматическом режиме конструировать оптимальный инвестиционный портфель. Использование интервального представления числовых данных гарантирует достоверность результатов, позволяет учесть расхождения между различными методиками расчетов средней доходности, риска, коэффициентов корреляции.

Основные результаты

1. Разработан метод интеграции недоопределенных вычислительных моделей и логического программирования. Разработанный метод повышает эффективность логических программ путем автоматического учета в логическом выводе неполной информации. Для случая интервальной недоопределенности теоретически исследованы свойства недоопределенных вычислительных моделей. Установлено необходимое и достаточное условие сходимости вычислений в таких моделях при решении линейных уравнений, установлена связь со сходимостью классических методов Зейделя, Якоби.
2. Разработаны средства представления и обработки множеств в недоопределенных вычислительных моделях. Разработанные средства позволяют использовать интенсиональные и экстенсиональные спецификации множеств для эффективного решения комбинаторных задач. Разработаны язык для описания таких задач и его интерпретатор LogiCalc. Предложенные средства представления и обработки множеств использованы в интерпретаторе LogiCalc.
3. Разработана Интервальная библиотека для системы логического программирования ECL^{i}PS^e. Библиотека предназначена для автоматического учета при логическом выводе неполной информации о вещественных решениях нелинейных ограничений. Библиотека включает средства для спецификации массовых ограничений, символьных преобразований, управления процессом вычислений. На основе Интервальной библиотеки разработаны приложения для решения задач автоматического проектирования и моделирования финансовых операций.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1. Petrov E. S. LogiCalc: integrating constraint programming and subdefinite models // Proc. 2nd Workshop on Non-Standard Logics and Logical Aspects of Computer Science. — Irkutsk: Irkutsk State Univ., 1996. — P.97–98.
2. Yakhno T. M., Petrov E. S. Application of subdefinite models for constraint satisfaction problems // Proc. 2nd Int. Conf. “Perspective of System Informatics”. — Berlin: Springer, 1996. — P.27–34. — (Lect. Notes Comput. Sci.; Vol. 1181).

3. Yakhno T. M., Petrov E. S. LogiCalc: integrating constraint programming and subdefinite models // Proc. 2nd Int. Conf. “Practical Application of Constraint Technology”. — London, 1996. — P.357–372.
4. Яхно Т. М., Петров Е. С. LogiCalc: интеграция методов программирования в ограничениях и недоопределенных моделей // Программирование. — 1996. — N.3. — С.70–79.
5. Петров Е. С. Сходимость метода недоопределенных вычислений при решении линейных уравнений // Проблемы представления и обработки неполностью определенных знаний. — Москва–Новосибирск: Рос. НИИ Искусственного интеллекта, 1996. — С.41–51.
6. Petrov E. S. LogiCalc: solving set constraints by subdefinite models // Proc. 8th European Summer School in Logic, Language and Information (student session). — Prague: Czech Tech. Univ., 1996.
7. Yakhno T. M., Zilberfaine V. Z., Petrov E. S. Application of ECL^{i}PS^e: Interval Domain library // Proc. 3rd Int. Conf. “Practical Application of Constraint Technology”. — London, 1997. — P.339–358.
8. Petrov E. S. Applications of Interval Domain library: expressing connectivity via non-linear constraints // Proc. 6th Scandinavian Conf. on Artificial Intelligence (Helsinki, Finland). — Amsterdam: IOSPress, 1997. — P.143–150.
9. Yakhno T. M., Zilberfaine V. Z., Petrov E. S. Interval Domain library for ECL^{i}PS^e and its applications // ICL Systems J. — 1997. — November. — P.35–50.
10. Петров Е. С. Интервальная библиотека: опыт интеграции логического программирования и программирования в ограничениях // Программирование. — 1998. — N.3. — С.40–49.
11. Yakhno T. M., Petrov E. S. Constraint programming for knowledge representation // Proc. 3rd Joint Conf. on Knowledge–Based Software Engineering (Smolenice, Slovakia). — Amsterdam: IOSPress, 1998. — P.116–123.
12. Петров Е.С. Интеграция недоопределенных моделей в систему ECL^{i}PS^e // Тр. 6–й национальной конф. по искусственному интеллекту с междунар. участием. — Пущино,1998. — Т.1. — С.355–360.

Подписано в печать 12.02.99 Объем 1.1 уч.-изд.л.
Формат бумаги 60×90 1/16 Тираж 100 экз.

Отпечатано на ризографе “AL-Group”.
630090, Новосибирск–90, пр. акад. Лаврентьева, 6.