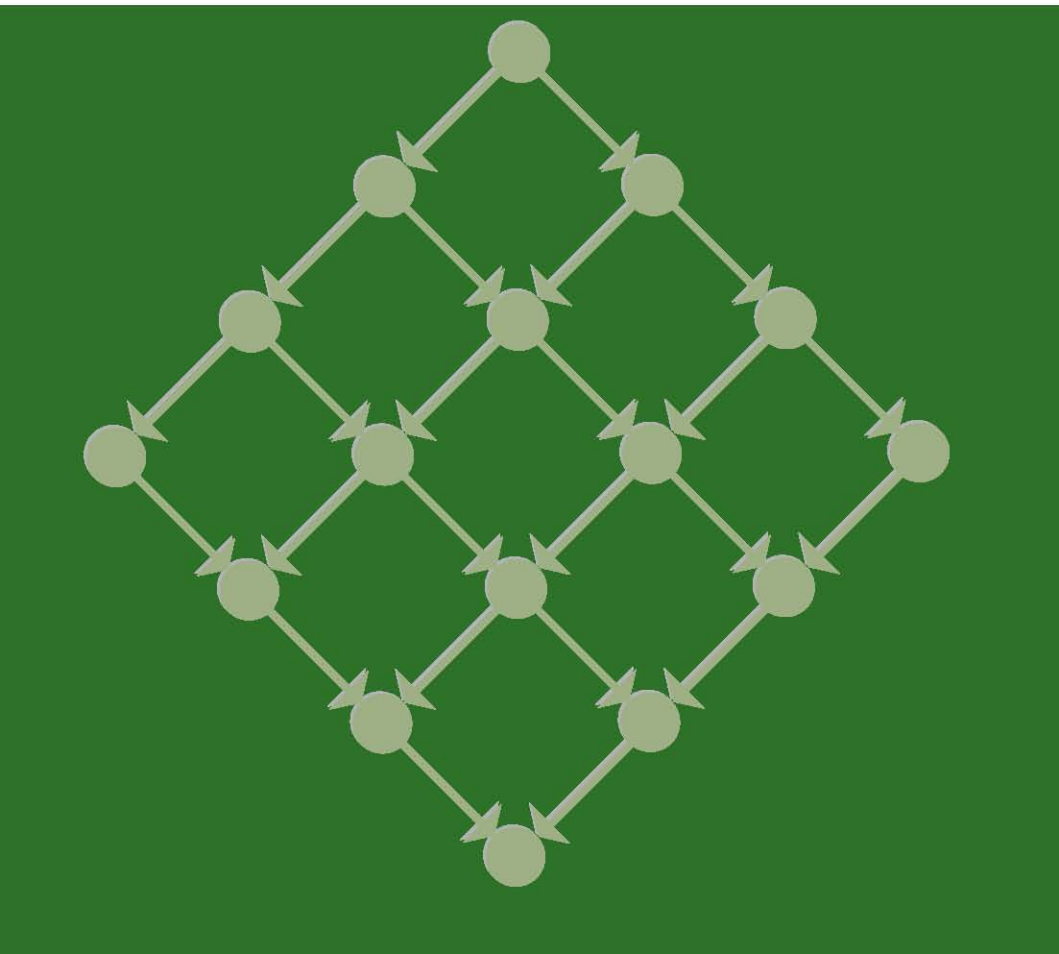
С. Д. Махортов

Математические основы искусственного интеллекта

теория LP-структур для построения и исследования моделей знаний продукционного типа



С. Д. Махортов

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА:
ТЕОРИЯ LP-СТРУКТУР ДЛЯ ПОСТРОЕНИЯ И ИССЛЕДОВАНИЯ МОДЕЛЕЙ ЗНАНИЙ ПРОДУКЦИОННОГО ТИПА

Под редакцией
В. А. Васенина

Москва
Издательство МЦНМО
2009

УДК 512.536.6:004.8
ББК 22.12+32.813
М36

Махортов С. Д.

М36 Математические основы искусственного интеллекта: теория LP-структур для построения и исследования моделей знаний продукционного типа / Под ред. В. А. Васенина. — М.: Издательство МЦНМО, 2009. — 304 с.

ISBN 987-5-94057-575-7

Излагается основанная на решетках алгебраическая теория, которая предназначена для моделирования и управления знаниями в интеллектуальных системах продукционного типа. Многие модели в информатике имеют продукционный характер, а структуры представления информации, как правило, являются иерархическими. Предложенная теория адекватно отражает вторичные продукционные связи в иерархических системах широкого спектра применения, а также обосновывает формальные исследования таких систем на предмет их эквивалентности, эквивалентных преобразований, верификации и оптимизации.

Описаны возможности применения теории LP-структур на примерах из различных областей информатики. Представлена интегрированная среда разработки продукционных экспертных систем, а также реализация в ее составе LP-структуры для верификации и оптимизации баз знаний. Приводятся результаты экспериментов, подтверждающие практическую значимость изложенной теории.

Книга предназначена для студентов, аспирантов и научных работников, занимающихся исследованиями в области алгебраических основ информатики и интеллектуальных систем.

УДК 512.536.6:004.8
ББК 22.12+32.813

ISBN 987-5-94057-575-7

ОТ РЕДАКТОРА

Информационные технологии на современном этапе развития общества проникают во все сферы деятельности человека, включая материальную и интеллектуальную, социально-культурную и политическую. Данный фактор порождает постоянно возрастающую зависимость общества от уровня надежности и эффективности программных решений, поддерживающих процессы его информатизации. К таковым в полной мере относятся сложно организованные и наукоемкие системы формального представления знаний, в первую очередь – широко распространенные на практике системы продукционного типа. Как следствие, оказывается весьма востребованным создание строгой математической базы, которая теоретически обосновывала бы корректность и надежность таких систем, а также возможности их оптимизации в автоматизированном режиме.

Настоящая книга посвящена решению указанной научной проблемы. В работе представлена созданная автором на основе выполненных им исследований теория LP-структур, описывающая новые эффективные методы для построения широкого спектра моделей знаний продукционного типа. Теория LP-структур предлагает общую методологию анализа продукционных систем в плане их эквивалентности, эквивалентных преобразований, верификации и оптимизации.

На основе рассмотрения особенностей продукционных систем в книге предложена обобщающая их новая алгебраическая модель. Данная модель базируется на иерархическом множестве (решетке) и содержит дополнительное бинарное отношение с набором специальных (продукционно-логических) свойств. Порождающий решетку частичный порядок отражает универсальные тавтологии и является фиксированным. Продукционное отношение определяется логическими связями конкретной предметной области и может подвергаться преобразованиям с целью её оптимизации по тем или иным критериям. Изучение такой модели показало, что аналогич-

ной семантикой обладают и другие системы в информатике, которые ранее никогда не относились непосредственно к продукционным. В результате построена новая математическая теория, имеющая широкую область приложений в информатике.

В целях практического применения теория LP-структур развита до эффективных способов компьютерных представлений бинарных отношений на решетках. На основе этой теории разработаны новые методы обратного вывода. Представленная в книге программная реализация интегрированной среды разработки экспертных систем, созданные с ее помощью базы знаний, результаты их верификации и оптимизации демонстрируют работоспособность теории LP-структур. Построена эффективная система логического программирования, обладающая новыми автоматизированными возможностями.

С учетом изложенного есть все основания полагать, что представленная в настоящей книге теория LP-структур вносит весомый вклад в решение крупной, национального уровня научной проблемы создания математической базы для верификации, оценки надежности и оптимизации программного обеспечения автоматизированных систем управления знаниями. Технологические решения на ее основе обладают несомненными инновационными перспективами и практической значимостью для экономики страны. Теория LP-структур имеет возможности дальнейшего развития, а ее применения могут охватить более широкие области информатики.

Доктор физико-математических наук,
профессор МГУ имени М. В. Ломоносова
В. А. Васенин

ПРЕДИСЛОВИЕ

Современное развитие информационных технологий напоминает рекурсивную систему – технологии, технологии для технологий и так далее. Соответственно и процесс программирования давно поддерживается собственными технологическими средствами, которые, в свою очередь, реализуются в виде специальных программных систем. Объемы и сложность решаемых на данном направлении задач быстро растут, и естественной в этой связи выглядит потребность в автоматизации и переносе как можно большей части процесса производства программ на ЭВМ. Однако, чем глубже «рекурсия», чем сложнее процесс, тем выше ответственность разработчиков, риски ошибок и следующих за ними неблагоприятных ситуаций и различного рода потерь. Особое значение отмеченные факторы имеют для систем информатизации объектов критически важных инфраструктур, где допущенные «промахи» могут привести к чрезвычайным ситуациям, экологическим катастрофам, материальным и другим потерям государственного масштаба. В результате национально значимой становится проблема создания математической базы, с помощью которой можно было бы теоретически обосновывать корректность и надежность программного обеспечения, а также поддерживать процессы его автоматической оптимизации.

Книга посвящена одному из направлений решения обозначенной проблемы. Излагается алгебраическая теория LP-структур, предназначенная для формального построения и исследования широко распространенных на практике логических систем продукционного типа. Многие модели в информатике имеют продукционный характер, а структуры представления информации, как правило, являются иерархическими. Предложенная теория адекватно отражает вторичные продукционные связи в иерархических системах широкого спектра применения, а также обосновывает формальные исследования таких систем на предмет их эквивалентности, эквивалентных преобразований, верификации и оптимизации.

Описаны возможности применения теории LP-структур на примерах из различных областей информатики. Это относится к стандартным и расширенным продукционным экспертным системам, системам компьютерной алгебры, автоматического доказательства теорем, автоматизированного рефакторинга, автоматизации программирования и так далее. Описана интегрированная среда разработки продукционных экспертных систем, а также реализация и применение в ее составе LP-структуры для верификации и оптимизации баз знаний. Приводятся результаты экспериментов, подтверждающие практическую значимость изложенной теории.

Представленные в работе исследования носят в основном теоретический характер. Ее назначение состоит в определении подхода и построении класса алгебраических структур, позволяющего формулировать и успешно решать задачи исследования и оптимизации продукционно-логических систем. Формулируя специальные свойства бинарных отношений в LP-структурах, можно аналогичными методами получать адекватные алгебраические модели для применения в различных областях информатики, в том числе и таких, которые остались за рамками исследований настоящей работы.

В то же время изложенные в монографии теоретические результаты имеют хорошие перспективы практического применения. Каждая из описанных возможностей применения LP-структур может быть доведена до программной реализации – решения задач автоматизации эквивалентных преобразований, верификации и минимизации продукционно-логических систем.

Автор выражает глубокую признательность своим учителям – доценту В. Е. Калечицу, благодаря школе которого с конца 70-х годов считает себя вправе называться программистом, а также профессору В. П. Глушко, в 80-е годы существенно повлиявшему на формирование математического мировоззрения автора.

Автор благодарен руководителям и участникам ряда научных семинаров г. Москвы, чьи замечания и обсуждения в последние годы оказались весьма полезными для написания данной работы. Это, в частности, член-корреспондент РАН Л. Н. Королев, профессор Р. И. Подловченко, профессор С. А. Абрамов, профессор О. П. Кузнецов, доктор физико-математических наук И. Б. Бурдонов, доцент В. А. Захаров.

Особую благодарность хотелось бы выразить научному редактору книги профессору В. А. Васенину, сделавшему многочисленные конструктивные замечания по ее тексту.

ВВЕДЕНИЕ

Эффективным средством формального построения и исследования компьютерных программ, основанных на самых различных парадигмах и технологиях, являются алгебраические системы ([112, 179, 185] и другие). Это положение в полной мере относится и к логическому программированию, особенно в части представления теорий и знаний. Алгебраическим методам представления знаний посвящены работы [66, 78], а также монографии [92, 194].

Математическую основу для создания и исследования моделей знаний предоставляет алгебраическая логика. Ее начала были заложены в работах А. Линденбаума, А. Тарского, систематическое изложение дано в монографиях [38, 187]. Теория Линденбаума–Тарского рассматривает логику нулевого порядка как универсальную алгебру, операции которой соответствуют логическим связкам пропозиционального языка. Примеры алгебраизации исчисления предикатов первого порядка представляют полиадические алгебры Халмоша [37] и цилиндрические алгебры Хенкина–Тарского [41]. Обзор методов алгебраизации кванторов содержится в монографии [63].

Однако общая алгебраическая логика, расширяя возможности исследования самих логических теорий, существенно не облегчает их практического применения. В силу своей универсальности она не решает ряда важных частных проблем, связанных с широко распространенными на практике логическими системами продукционного типа. На этот факт указывается в книгах [180, 197]. К проблемам такого рода могут быть отнесены вопросы эквивалентности, эквивалентных преобразований, верификации продукционных и подобных им систем, рассматривавшиеся частными методами в работах [1, 27, 48, 52, 64, 186] и других исследованиях. Обзоры имеющихся подходов к верификации знаний содержатся в [34, 191].

Перечисленные вопросы играют существенную роль при создании и исследовании формальных методов работы со знаниями, а

также определяют принципы построения программных средств автоматизации управления знаниями.

Особое место в ряду названных проблем занимает минимизация, поскольку в любой парадигме программирования действует золотое правило – избегать неоправданного дублирования кода или данных. В общей математической логике минимальная система аксиом называется базисом. Проблемы существования базисов допустимых правил для различных логик рассматривались В.В.Рыбаковым и его последователями [188-190]. Однако продукционные системы имеют особенности, дающие дополнительные возможности в плане минимизации.

Эквивалентные преобразования и минимизация множества *унифицированных* правил продукционных экспертных систем в некоторых работах изучались на основе пропозициональных хорновых функций (например, [7, 39, 40]). В статье [27] для исключения избыточности знаний используется логика первого порядка. В работе [30] рассмотрено несколько частных случаев упрощения множества правил на основе теории графов. Еще один путь минимизации продукционных систем может дать использование представления продукций сетями Петри [1] и решение задачи редукции сетей Петри [6, 47, 85].

В перечисленных работах нет строго обоснованного алгебраического подхода, универсального в рамках широкого спектра систем продукционного типа, который мог бы быть применен для решения задач эквивалентных преобразований, верификации и минимизации. Имеющиеся алгебраические исследования посвящены частным случаям продукционных систем либо другим их аспектам. В качестве примера можно привести связанную с теорией очевидности [76] алгебраическую теорию «демпстероидов» [35]. Она предназначена для вычислений результатов вывода в продукционных системах с функциями доверия [32].

В работе [74] расширенная продукционная система сводится к системе переписывания термов (СПТ). В семантике последней предлагается процедура обнаружения избыточных правил. Как это принято в системах переписывания термов, требуется «завершаемость» СПТ и, соответственно, исходного множества правил, иначе нет гарантии завершаемости процедуры. Этот интересный метод не охватывает продукционные системы, множества правил которых содержат циклы.

Возможности алгебраического исследования продукционно-логических систем содержит теория ультраоператоров А. В. Чечкина

[204]. В приложении к математической логике она предлагает рассматривать импликации в виде отдельного соответствия. Однако в открытой печати не представлены более подробные исследования ультраоператоров в данном направлении.

Интересная алгебраическая модель, позволяющая формализовать широкий круг логических задач, предложена Б. А. Куликом [128]. Введенная им алгебра кортежей представляет собой еще одну алгебраическую интерпретацию математической логики. В монографии [129] описываются также возможности применения изложенной теории к моделированию экспертных систем. Не вдаваясь в подробности, отметим, что опубликованные для алгебры кортежей результаты не связаны с решением перечисленных выше задач для продукционных систем.

Выше наряду с продукционными системами недаром были упомянуты также «подобные им» системы. В первую очередь, в контексте настоящей работы, конечно, имеются в виду собственно продукционные экспертные системы. Они остаются актуальными, что подтверждается значительным числом публикаций последних лет, посвященных как их теоретическим исследованиям, так и решению прикладных задач [5, 12–13, 44–45, 50–51, 55, 69, 80, 109]. База знаний одного из распространенных классов продукционных систем представляет собой совокупность правил вида «если ..., то ...», где в предпосылке и заключении фигурируют множества так называемых фактов. Эти множества независимо от правил также образуют иерархию как элементы булеана – множества подмножеств. Правила расширенной продукционной системы в предпосылке и заключении могут содержать пропозициональные формулы [15]. Такие формулы кроме правил связаны еще и иерархией в рамках соответствующей алгебры Линденбаума–Тарского. В подобном же виде могут храниться и математические знания – большинство теорем записывается в виде «дано ..., требуется доказать ...», где предпосылка и заключение являются формулами исчисления предикатов.

Широко применяемые в теории программирования условные системы переписывания термов [17, 46, 99] также основываются на правилах продукционного вида, связывающих пары элементов, которые принадлежат некоторой иерархии термов. Существуют и другие области информатики, на первый взгляд далекие от продукций, но интерпретируемые ими. В частности, элементы иерархии типов объектно-ориентированной программной системы [94] можно рассматривать как множество, на котором задано продукционное отношение агрегирования [199]. Даже в императивных программах

просматриваются продукционные связи между состояниями данных до и после выполнения каждой операции.

С учетом изложенного выше можно констатировать, что актуальной является проблема создания алгебраической теории, которая бы адекватно отражала вторичные продукционные связи в различных иерархических системах широкого спектра применения, а также обосновывала автоматизированные формальные исследования таких систем в плане их эквивалентности, эквивалентных преобразований, верификации и оптимизации.

Основной целью настоящей книги является изложение такой теории, а также описание и демонстрация возможностей ее применения на примерах из различных прикладных областей.

Научная оригинальность монографии заключается в следующих положениях.

• Для автоматизированной разработки и исследования систем продукционного типа предложен новый подход, выраженный в создании основанной на решетках алгебраической теории LP-структур (Lattice-Production Structure). Предполагается, что информация о некоторой предметной области может быть формально представлена в виде решетки.Описание методов использования решеток для представления знаний можно найти в [66, 78, 194]. Основная идея теории LP-структур состоит в моделировании продукционных связей (совокупности правил) дополнительным бинарным отношением с заданными свойствами (рефлексивность, транзитивность и некоторые другие свойства, зависящие от конкретной модели). При этом определяющее решетку исходное отношение частичного порядка отражает универсальные тавтологии и является фиксированным. Второе отношение порождается логическими связями конкретной предметной области и может подвергаться эквивалентным преобразованиям.

• Введено и обосновано понятие эквивалентности продукционно-логических систем на основе их логического замыкания. Доказаны возможности автоматических локально-эквивалентных преобразований LP-структур и соответственно моделируемых ими продукционных систем.

• Введено новое понятие продукционно-логического уравнения и обоснован способ его решения, что в применении соответствует *полному* обратному выводу. Концепция уравнений может быть также применена для верификации знаний. Ранее интересные классы логических уравнений рассматривались в монографиях [111, 133], однако представленные в них уравнения имеют отличную от систем

продукций природу. На нечетких бинарных отношениях основаны реляционные уравнения, рассматривавшиеся в [16] и ряде других работ. Основные трудности исследования здесь порождаются нечеткостью отношений, и поэтому процесс решения соответствует всего лишь единственному шагу обратного вывода.

• В монографии также доказано существование и получен эффективный способ построения логической редукции LP-структур. В приложениях он означает минимизацию соответствующих баз знаний, то есть построение эквивалентной продукционной системы с минимальным набором правил.

• Новым является распространение единого алгебраического подхода на достаточно широкий спектр различных систем: стандартные и расширенные продукционные экспертные системы; формальные системы математических знаний; условные системы переписывания термов; иерархии типов в объектно-ориентированном программировании. В частности, новым является введенный и использованный в работе тезис о продукционной семантике иерархии типов с отношением агрегирования. В результате на базе LP-структур обоснованы автоматизированные решения некоторых важных задач верификации типов и рефакторинга. Показана возможность применения продукционно-логических структур к новым исследованиям свойств императивных алгоритмов.

• В качестве примера для иллюстрации приложения LP-структур первого порядка в работе изложены элементы теории неклассических псевдодифференциальных операторов. Они содержат новые результаты в соответствующей области.

• Сформулирована новая концепция трехмерной структуры расширяемой программной системы, которая в дополнение к актуальной ранее двумерной модели [104, 200] содержит набор взаимосвязанных уровней программирования, от системного до пользовательского, завершающийся на верхнем уровне продукционной экспертной системой.

• Разработаны и реализованы оригинальные эффективные методы компьютерного представления основанных на решетках алгебраических структур.

• Предложены и реализованы новые методы обратного вывода и верификации для систем продукционного типа, базирующиеся на решении логических уравнений. Концепция LP-вывода направлена на минимизацию количества запросов к внешнему источнику. Запросы по возможности отправляются только для тех фактов, которые необходимы при выводе. Отрицательный ответ на единс-

твенный запрос исключает все последующие запросы об элементах целого множества. Кроме того, при LP-выводе предпочтение отдается тестированию множества фактов минимальной мощности.

Данные идеи не пересекаются с известными методами быстрого вывода, а дополняют их. Во-первых, алгоритмы RETE [25] и TREAT [58], базирующиеся на специальных сетевых представлениях множеств правил, изначально разрабатывались для стратегии *прямого* вывода и использовались в продукционных системах с прямым выводом (например, OPS5 [24], CLIPS [42]). Алгоритм LEAPS [59] осуществляет компиляцию в язык C множества правил той же продукционной системы OPS5. В последующих исследованиях наиболее популярный алгоритм RETE адаптировался для смешанного (двунаправленного) вывода [42, 49]. Изложенная в настоящей книге концепция уравнений предназначена для исключительно *обратного* вывода и не требует для своей реализации громоздких динамических структур данных, свойственных указанным выше алгоритмам, в случаях, когда нет никакой потребности в прямом (и соответственно смешанном) выводе. Во-вторых, ничто не мешает адаптировать имеющиеся быстрые алгоритмы обратного вывода для нахождения рассмотренных в работе решений продукционно-логических уравнений. Этот подход может оказаться интересным направлением развития теории LP-структур.

• Реализована интегрированная среда разработки и выполнения продукционно-логических систем, обладающая новыми возможностями исследования и оптимизации баз знаний.

Новая теория LP-структур занимает собственную нишу, однако при этом она соприкасается с некоторыми другими исследованиями.

Отметим в частности, что бирешетки [28] также предполагают наличие двух отношений на общем множестве, однако в прочих аспектах теория LP-структур имеет с ними мало общего, как с точки зрения формального определения, так и в плане возможных применений. В ряде работ (см. [14] и библиографию в ней) рассматриваются общетеоретические вопросы о свойствах бинарных отношений на частично упорядоченных множествах (в том числе и решетках), такие как монотонность, неподвижные (рефлексивные) точки, рекурсии. Однако эти исследования не учитывают специфики моделей продукционных систем.

Задача логического вывода на LP-структурах перекликается с проблемой нахождения функциональных зависимостей в реляционных базах данных (см., например, [100]). При выводе функци-

ональных зависимостей в базе данных используются дедуктивные правила Армстронга (они впервые введены в работе [4]), применяемые к элементам булеана. Однако в этой теории из «решеточных» операций используется лишь операция объединения, а набор правил Армстронга существенно беднее множества правил вывода в LP-структурах. Вполне возможно, что исследование реляционных баз данных может стать одной из новых областей применения теории LP-структур. Имеются также определенные перспективы использования подобных LP-структурам алгебраических систем для моделирования некоторых классов полуструктурированных данных (см., например, [88, 96]) с целью исследования и оптимизации запросов.

Близким к теории LP-структур может считаться FCA – формальный концептуальный анализ [26, 127], имеющий широкую область применения в исследованиях двумерных данных с семантикой «объекты-признаки». Он также основан на решетках и рассматривает бинарные отношения между множествами. Однако LP-структуры и FCA имеют существенные отличия. Результаты анализа публикаций свидетельствуют, что общих приложений у этих двух теорий практически нет, несмотря на иногда похожие формулировки решаемых в их рамках задач. Например, минимальный базис импликаций в FCA перекликается с логической редукцией LP-структуры. С помощью этих подходов ставятся и решаются задачи, соответствующие различным моделям в информатике. В частности, данный факт подтверждают представленные далее возможности применения LP-структур в объектно-ориентированном программировании.

Глава 1
ЗАДАЧИ, ПРИВОДЯЩИЕ К LP-СТРУКТУРАМ

Как отмечалось во введении, настоящая работа посвящена алгебраическим системам (LP-структурам), формализующим продукционно-логический вывод на основе теории решеток и бинарных отношений. Результаты исследований показали применимость таких теоретических положений к созданию программного обеспечения процессов представления и использования знаний на базе моделей продукционного типа. Цель настоящей главы – описать возможности таких приложений. Перечисляются задачи в различных областях информатики, описание которых может быть сведено к продукционным системам, моделируемым LP-структурами. К таковым относятся: верификация и оптимизация баз знаний экспертных продукционных систем; представление математических знаний; упрощение множеств правил условных систем переписывания термов; автоматизация некоторых методов рефакторинга; исследование свойств императивных алгоритмов. Соответственно решение перечисленных задач может осуществляться на основе LP-структур или их модификаций

Раздел 1.1 посвящен обсуждению сформулированной автором концепции многоуровневой разработки программных систем. Согласно этой концепции, верхний уровень разработки программ содержит интеллектуальную надстройку для пользовательского программирования на основе продукционных систем. Это положение стало отправной точкой для введения рассматриваемых в данной работе LP-структур. В последующих разделах главы анализируются возможные применения LP-структур для описания различных систем продукционного типа.

1.1. О многоуровневом программировании

В данном разделе формулируются концептуальные положения трехмерной структуры расширяемой программной системы. В качестве содержания третьего измерения предлагается классификация параллельных уровней разработки программы. Приводится

пример реализации программных средств поддержки указанной технологии.

Общеизвестно, что разработка любого более или менее серьезного программного комплекса связана с применением какой-либо одной или нескольких технологий программирования. Это обусловлено постоянно возрастающими сложностью и объемами решаемых в ходе такой разработки задач. При этом важнейшее место в процессе использования информационных технологий занимают характеризующие и формулирующие их концептуальные средства. Они определяют стиль, а также методы проектирования и разработки программного обеспечения (ПО). Важной особенностью настоящего времени является появление таких областей применения программных систем, для которых сопровождение ПО по длительности и ресурсозатратам сопоставимо с его разработкой. Этот факт означает, что никогда не наступает момент, когда «программа окончательно готова». К такого рода областям относится, например, автоматизация финансово-хозяйственной деятельности предприятий, где непрерывные и существенные изменения в законодательстве постоянно держат эксплуатируемое ПО в стадии частичной разработки. Для подобных программных систем особенное значение приобретает не только определение последовательности создания отдельных его частей, но и вопросы о том, разработчики какого уровня и на каких этапах занимаются их созданием. Аналогичные вопросы связаны также с разработкой ПО сложно организованных автоматизированных систем управления критически важными объектами [122–123].

Несмотря на эволюцию парадигм проектирования и написания кода от модульного к объектно-ориентированному программированию [94], по мнению автора, по-прежнему актуальным является представление о двумерности структуры развивающейся программы [104, 200]. Согласно такому представлению, каждое расширение функциональности программы можно рассматривать как добавление определенного компонента (вертикального слоя), относящегося к некоторым уже сформированным образованиям (горизонтальным слоям). Изъятие вертикального слоя при этом не приводит к «катастрофическим» последствиям для программы, а лишь обедняет ее функциональность. Каждый же горизонтальный слой программы является ее неотъемлемой, системообразующей частью. Даже, если допустить, что программу удалось спроектировать в виде единственного объекта некоторого класса, то и в этом случае его публичные свойства и методы, непосредственно экспортирующие функциональность объекта, можно считать вертикальными слоями, а

скрытые свойства и методы, опосредованно реализующие функциональность, – составляющими горизонтальные слои. Различие точек зрения на такое представление состоит лишь в том, какие слои являются первичными – горизонтальные или вертикальные.

Однако математик, привыкший к абстрактному обобщению, задался бы вопросом: «почему только два измерения?». В настоящем разделе делается попытка ввести третье измерение в модель структуры расширяемой программной системы.

В программировании давно существует понятие уровня разработки. Так множество существующих языков программирования всегда разделялось по их уровню. Представим на самом нижнем уровне такой иерархии машинный язык, а на самом верхнем – разговорный человеческий. Тогда какой-либо язык программирования относится к низкому или высокому уровню в соответствии с его относительным расположением в указанной иерархии (Ассемблер – язык низкого уровня, Паскаль – высокого). Заметим, что повышение уровня при этом, как правило, облегчает разработку программы, однако сужает возможности. Немного обобщим это понятие. Будем считать, что вся программная система разрабатывается сразу на нескольких уровнях, каждый из которых занимает свое место в представленной выше иерархии. Можно, например, выделить три таких уровня (в порядке повышения): системный, прикладной, пользовательский.

Системный уровень разработки программного комплекса (не путать с операционной системой, расположенной еще ниже, однако в иерархии это самый близкий к ОС уровень) – это базовый уровень, который содержит средства поддержки технологии разработки и создается наиболее квалифицированными программистами, возможно, незнакомыми с конечной областью применения всего программного комплекса. Программные объекты этого уровня могут использоваться для создания не только одной, но и целого класса систем, основанных на данной технологии.

Прикладной уровень занимает позицию выше и содержит в основном реализацию решения конкретных задач на уровне программиста. Специалисты, создающие этот уровень, могут быть менее квалифицированными в программировании, однако должны разбираться в предметной области, для которой создается программная система.

Пользовательский уровень – самый высокий в смысле приближения к уровню мышления обычного пользователя. Разработку на этом уровне часто называют настройкой, однако ее сложность и объемы порой не уступают разработке, особенно это касается упо-

мянутых выше систем автоматизации управления сложно организованными объектами. Специалисты этого уровня могут быть экспертами в предметной области, но иметь лишь общее представление о низкоуровневом программировании.

Приведенная классификация является наиболее общей, идеализированной и может уточняться по мере развития и реализации технологии. В частности, пользовательский уровень может состоять из нескольких подуровней, характеризующих степень «образованности» пользователя в вопросах программирования. В дальнейшем также оказывается, что уровни могут пересекаться. Кроме того, рассматриваемая классификация предполагает, что исходный язык программирования является компилируемым, по крайней мере, в близкий к машинному код. Конечно, режим интерпретации программ имеет свои существенные преимущества в плане верификации программ, возможностей отладки и выполнения других подобных функций. Это, в частности, показывает ряд разработок, в которых автор принимал непосредственное участие [116–119]. Однако в контексте обсуждаемых вопросов эффективность по крайней мере базовой части и наиболее «тонких» в смысле сложности используемых алгоритмов мест является определяющей.

Описываемая технология, как и почти любая другая технология программирования, нуждается в программных средствах поддержки. Рассмотрим такие средства в реализованных автором программных системах на основе операционной системы Windows с использованием среды разработки Delphi и сервера баз данных Interbase. Следует иметь в виду, что выбранные средства не являются единственно возможными в этом плане. Аналогичные идеи можно реализовать в C#/.Net или еще в какой-либо среде, поддерживающей компонентную модель программирования. Выбор возможных серверов баз данных (БД) еще более широк. В дальнейшем под термином «компонент» подразумевается компонент в смысле Delphi.

Итак, первоначально на упомянутом выше системном уровне разработки создается основа – базовое приложение, содержащее минимальный интерфейс пользователя и средства его формирования, а также механизм добавления/исключения компонентов. Этот механизм умеет загружать пакеты компонентов, регистрировать компоненты, настраивать их свойства, вызывать методы. Таким образом, уже на данном этапе программа без дальнейшей корректировки ее базового кода становится неограниченно расширяемой. Причина в том, что подключить к ней можно любой пакет компонентов Delphi,

разработкой которых занимаются сотни фирм и тысячи программистов всего мира. На следующем этапе разрабатывается и добавляется к базовому приложению визуальный дизайнер форм, позволяющий создавать новые формы из загруженных компонентов. При таком подходе одним из возможных путей, по которому пошел и автор, является моделирование в базовом приложении исходной среды разработки (IDE Delphi), в которой компоненты легко переводятся в режим дизайна. Основной метод здесь – изучение недокументированных интерфейсов среды и их реализация в специальном классе. Полезными источниками информации в этом плане явились книги [135, 181]. При загрузке базовым приложением пакетов Delphi кроме регистрации самих компонентов регистрируются также связанные с ними редакторы, что позволяет в режиме дизайна форм добиться полной визуальности, во многом не уступающей Delphi. Разрабатываемые пользователем формы можно сохранять как на клиентском компьютере («локальная настройка»), так и в общей базе данных («глобальная настройка»). В базовое приложение (на системном уровне) включаются также несколько классов наиболее часто используемых в данной области применения стандартных форм (форма-дерево, форма-таблица, форма-дерево/таблица), которые будут служить основой для других уровней разработки. На начальном этапе реализуется также как базовый стандартный набор операций с БД (соединение, сохранение/восстановление, выполнение SQL-команд). Вообще в базовое приложение рекомендуется включить возможности, которые могут потребоваться в дальнейшем в любой разрабатываемой системе проектируемого класса.

Следующим шагом является создание и распространение с базовым приложением *прикладных компонентов*. Среди них может содержаться, например, компонент «экспортер бухгалтерской проводки» или «расчет налога». Теперь, когда компоненты могут подключаться без программирования, пользователь сам сможет собрать нужную конфигурацию программы (например, для конкретного рабочего места бухгалтерии) и в дизайнере форм разработать, например, новую форму первичного бухгалтерского документа не с нуля (хотя и это возможно в нашей системе). Тираж распространения компонента соизмерим с тиражом разрабатываемой программы, причем компонент будет использоваться для разработки на пользовательском уровне. Создание же самих прикладных компонентов согласно рассматриваемой классификации осуществляется программистами на прикладном уровне. Прикладные компоненты можно создавать более адаптированными к пользователю, чем обычные компоненты (назовем их

системными). Разработанный дизайнер форм способен распознавать прикладные компоненты и показывать названия их свойств и методов в пользовательской интерпретации (в том числе на русском языке). Заметим, что системы пользовательского программирования путем сборки из заготовок разрабатывались В.Ф. Жоголевым и его учениками. В данных исследованиях они называются *системами обоснованного гиперпрограммирования* [110].

Далее выясняется, что кроме установки свойств и возможности вызова методов компонентов, необходимо обрабатывать связанные с ними события. Чтобы это было возможным без изменения исходных текстов базового приложения, разрабатывается (другой возможный вариант – используется готовый) скриптовый язык (например, подмножество Pascal), на котором можно создавать обработчики событий и другие программные части, сохраняемые в разрабатываемых формах. Конструкции языка в целях адаптации к потребностям пользователя должны допускать возможность переименовывания (в том числе по-русски). Реализуется двусторонняя связь опубликованных свойств и методов компонентов с переменными скриптового языка. Из соображений эффективности важно, чтобы этот язык допускал расширения в виде пользовательских функций (UDF), написанных на обычном языке программирования (например, Delphi) и подключаемых в виде DLL. Реализуется также редактор скриптовых программ с возможностью подсветки синтаксиса (с выбором языка – Pascal или SQL).

На данном этапе пользователь уже имеет неограниченные возможности для наращивания функциональных возможностей программ без изменения базового кода. Однако этому препятствует немаловажный фактор – пользователь должен быть в определенной мере программистом. Ему необходимо иметь представление о свойствах и методах используемых компонентов, знать основы несложного скриптового языка. И этого мало, поскольку нужны также определенные навыки программирования, которые невозможно получить без практики, даже если с точки зрения профессионального программиста решаемые на данном уровне задачи примитивны. Идея применения искусственного интеллекта для программирования не нова (см. [195] с библиографией), однако рассматриваемая ситуация имеет существенную особенность – в ней достаточно элементарного программирования при хорошем владении предметной областью. Эта особенность дает основания для оптимизма относительно разрешимости поставленной задачи. Если в [195] в качестве основы представления знаний о решаемой задаче декларируются

семантические сети и фреймы [178], то в случае несложного пользовательского программирования достаточно систем продукций. Соответственно для дальнейшего повышения пользовательского уровня разработки программной системы можно предложить создание и использование проблемно-ориентированной транслирующей экспертной продукционной системы. Такая система может интегрировать знания *среднего программиста*, необходимые для связи *квалифицированного пользователя* с «миром программирования». Одной из главных задач настоящей работы является создание математической базы для адекватного описания прикладных систем продукционного типа, которая позволяла бы контролировать в плане эквивалентности преобразований, верифицировать и оптимизировать процессы разработки и модификации ПО.

Важным аспектом многоуровневой технологии является предоставление, начиная с некоторого момента, возможности параллельной разработки системы на всех уровнях, возможности работы на более низком уровне для повышения эффективности разрабатываемой системы.

Почему же изложенную схему разделения на уровни можно считать третьим измерением в модели структуры разработки программ? Во-первых, потому, что на каждом из рассмотренных уровней может применяться упомянутое выше представление о слоях. Во-вторых, при должной технологической поддержке являются взаимозависимыми вдоль уровней (по крайней мере, в сторону повышения), соответствующие горизонтальные и вертикальные слои. Однако более глубокое исследование этого вопроса не является предметом данного раздела. Отметим, что предложенное разделение процесса разработки на иерархические уровни основано на личном опыте автора и его субъективном видении. Как следствие, возможны другие классификации, что, однако, не противоречит самому факту существования третьего измерения.

1.2. Основная терминология решаемых задач

Введем основные обозначения и определения, необходимые для дальнейшего изложения.

Пусть R – бинарное отношение на некотором множестве F. Для каждой упорядоченной пары $(a,b) \in R$ элемент a будем называть ее левой частью, а элемент b – правой частью.

Бинарное отношение R на произвольном множестве F называется:

• рефлексивным, если для любого $a \in F$ справедливо $(a,a) \in R$;

О Г Л А В Л Е Н И Е

Научное издание

Махортов Сергей Дмитриевич

Математические основы искусственного интеллекта: теория LP-структур для построения и исследования моделей знаний продукционного типа

Издательство Московского центра
непрерывного математического образования
119002, Москва, Большой Власьевский пер., 11. Тел. (499) 241-74-83

Подписано в печать 16.11.2009 г. Формат $60\times90\,{}^{1}\!/_{16}$. Бумага офсетная.
Печать офсетная. Печ. л. 19. Тираж 500. Заказ .

Отпечатано с готовых диапозитивов в ППП «Типография „Наука“».
121099, Москва, Шубинский пер., д. 6.