Комбинаторная логика в программировании *

Вольфенгаген В.Э.

Кафедра ПКИ и ИТ, Институт “ЮрИнфоР-МГУ”,
Москва, 127006 РФ,
vew@jmsuice.msk.ru,
WWW: http://www.wolfengagen.mephi.ru

Аннотация На ранних стадиях программирование представляло собой вид искусства, когда программист писал программу для решения определенной задачи и сопровождал ее более или менее подробно составленной документацией, то теперь создана мощная индустрия программирования с сопутствующей ей инженерией программирования. В настоящее время в исследованиях по программированию или в сфере компьютерных наук, как правило, поддерживаются работы, в которых вносится некоторое небольшое улучшение в решение уже хорошо известной проблемы. Вместе с тем из виду упускаются действительно важные и фундаментальные исследования, ведущие к поиску новых концепций вычислений на компьютере и недостаточное внимание уделяется накоплению знаний в области программирования.
В настоящей работе основное внимание уделено вычислениям с объектами, удельный вес и роль которых в данной области все более возрастает, превращаясь в доминирующую тенденцию.

* Настоящая работа отражает результаты исследований, частично поддержанных грантом Российского Фонда Фундаментальных Исследований, 04-07-90156

1. Введение

С момента своего возникновения комбинаторная логика и ламбда-исчисление были отнесены к “неклассическим” логиками. Дело заключается в том, что комбинаторная логика возникла в 1920-х гг., а ламбда-исчисление (λ-исчисление) — в 1940-х гг. как ветвь метаматематики с достаточно очерченным предназначением — дать основания математике (см. [8]). Это означает, что сконструировав требуемую ‘прикладную’ математическую теорию — предметную теорию, — которая отражает процессы или явления в реальной внешней среде, можно воспользоваться ‘чистой’ метатеорией как оболочкой для выяснения возможностей и свойств предметной теории.

Комбинаторная логика и ламбда-исчисление — это такие формальные системы, в которых центральной разрабатываемой сущностью является представление об объекте. В первой из них — комбинаторной логике, — механизм связывания переменных в явном виде отсутствует, а во второй он имеется. Наличие явного механизма связывания предполагает и наличие связанных переменных, но тогда есть и свободные переменные, а также механизмы замещения формальных параметров — связанных переменных, — на фактические параметры, то есть *подстановка*.

Изначальным назначением комбинаторной логики был именно анализ *процесса подстановки*. В качестве ее сущностей планировалось использовать объекты в виде *комбинаций констант*. Ламбда-исчислению отводилась роль средства уточнения представлений об *алгоритме* и *вычислимости*. Как следствие, комбинаторная логика дает в руки инструмент для анализа процесса подстановки. Через короткий промежуток времени ока-

залось, что обе эти системы можно рассматривать как *языки программирования* (см. [1], [9], а также книгу [4] и библиографию к ней).

В обеих системах исчисляются объекты, они являются исчислениями или языками *высших порядков*, то есть имеются средства описания отображений или операторов, которые определяются на отображениях или операторах, а в качестве результата вырабатывают также отображения или операторы. Самое существенное, что именно *отображение* считается объектом. В этом их принципиальное отличие от всего многообразия других систем, в которых первичной сущностью обычно считают представление о *множестве* и его элементах.

К настоящему времени оба эти языка не только стали основой для всей массы исследований в области computer science, но и широко используются в теории программирования. Развитие вычислительной мощности компьютеров привело к автоматизации значительной части теоретического — логического и математического, — знания, а комбинаторная логика вместе с ламбда-исчислением признаются основой для рассуждений в терминах объектов (см. [6], [7]).

Без овладения их методами нельзя полноценно развить базовую технику вычислений с объектами, поскольку все еще распространенный в объектных языках программирования и проектирования теоретико-множественный стиль заставляет вязнуть в обилии второстепенных деталей, упуская из виду действительно существенные моменты взаимодействия объектов.

2. Зачем нужно исчислять объекты

Работа за компьютерами с оболочкой, способной взять на себя заботы об управлении объектами программного обеспечения, закладывает основу самой современной на сегодня методики программирования. В настоящее время с объектами работают сотни прикладных программ таких, как Windows, AutoCAD, Designer и многих других. С другой стороны инструментальные системы программирования Small Talk, Actor, . . . , C++, C♯, Java и ряд других требуют от программиста систематических рассуждений в терминах объектов и связей между ними, которые в свою очередь могут рассматриваться как объекты. Программирование в терминах объектов требует создания и поддержания собственной математической культуры, дающей весь спектр стимулирующих идей (см. [10], а также раздел *Круг вопросов* в [8], с.9-39). Программист при решении вполне конкретной задачи становится исследователем, от которого требуется создание собственного языка со своими возможностями. Эти возможности не всегда интуитивно очевидны, и могут потребоваться чисто математические оценки их выразительных возможностей. Кроме того, часто требуется не просто написать некоторый программный код, но и выполнить его оптимизацию, не теряя свойства эквивалентности исходному коду. Все это требует для аккуратного и профессионального проведения работы своей собственной “математической оболочки”, в которой поддерживаются все значимые и интересные математические приложения.

3. Основное – адекватный способ мышления

Хорошо известно, что в практике программирования сложились различные подходы, которые развиваются по различным направлениям. Бросающиеся в глаза различия проявляются в разном способе осмысления и написания программ. Большинство программистов занимается *процедурным* программированием. Но кроме него есть и программирование, *основанное на правилах*, *логическое* программирование, *параллельное* программирование, *визуальное* программирование, *программирование* в терминах потоков данных. При желании этот перечень можно продолжить, но он, очевидно, будет неполон, если в него не включить также и *объектно-ориентированное* программирование, которое имеет явно выраженную тенденцию роста.

3.1. Подходы и стили программирования

Подходов и стилей программирования много, и это отражает картину совершенствования и распространения все новых и новых компьютерных архитектур. Возникающие архитектуры ориентированы на новые подходы к программированию, которые еще только зарождаются в исследовательских лабораториях.

Обилие и разнообразие подходов к программированию в computer science отражается в развитии и распространении различных подходов к построению математики. Действительно, математических теорий построено удивительно много, и каждая из них является совершенно своеобразным язы-

ком общения сравнительно ограниченного круга специалистов, которые хорошо понимают друг друга. Однако попытка “непосвященного” понять практическую пользу и значимость нового математического языка наталкивается на препятствия. Прежде всего оказывается необходимым перестроить собственный стиль мышления, чтобы на известные трудности взглянуть под новым углом зрения. Так распространение объектно-ориентированного программирования требует и привлечения других способов рассуждения, которые зачастую радикально отличаются от стереотипов рассуждения в процедурном программировании.

3.2. Рассуждения в терминах объектов

Точно также лишь немногие и сравнительно молодые математические теории ориентированы на рассуждения в терминах *объектов*, а не в терминах *операторов*, как это следует из опыта изучения математического анализа в большинстве университетов, в том числе, и технического или компьютерного профиля. К сожалению, программисту не удается прослушать университетский курс, закладывающий основы математического мышления в терминах объектов. В лучшем случае дело ограничивается сообщением чисто математических результатов, полученных в *комбинаторной логике*, *λ-исчислении* или *теории категорий*, которые не так-то просто преломить на практическое программирование без известной теоретической искушенности.

Можно утверждать, что комбинаторная логика значительно повлияла на современную картину программирования. Начинаясь как наука о *природе*

подстановок в математических теориях, она породила *функциональное* программирование, программирование в терминах *суперкомбинаторов*, а также некоторые другие чрезвычайно плодотворные подходы к программированию. В частности, только по-настоящему проникнув в сам дух комбинаторной логики, можно понять в деталях и практически применить систему программирования с заранее нефиксированной системой инструкций.

3.3. Теория вычислений

Парадигмы программирования 90-х гг. в сильной степени выросли из математического способа рассуждений, принятого в *теории вычислений*. В частности, одной из ее начальных посылок была концепция 'протекания информации' вдоль некоторого 'возможного' русла, что привело к возникновению весьма плодотворной концепции программы, управляемой потоком данных. Другой пример связан с идеей использования некоторой части комбинаторной логики, построив в ней специальные объекты-инструкции. Эти объекты образуют систему команд *категориальной абстрактной машины*, которая может быть с успехом положена в основу вполне практических (но объектно-ориентированных) систем программирования (см. [3]). Более того, правила комбинаторной логики позволяют *оптимизировать компилируемый программный код*, редуцируя его к некоторой нормальной форме. Для специалистов в комбинаторной логике это почти само собой разумеется с самого начала, поскольку в этом состояла одна из целей разработки комбинаторной логики как математической дисциплины.

Современные исследования в области computer science показывают, что комбинаторная логика и ее различные категориальные диалекты становятся необходимым математическим языком программиста, пользуясь которым он обменивается идеями со своими коллегами. Дело как раз в том, что одним из предметов ее исследования являются объекты и построение различных исчислений объектов, которые удовлетворяют кругу вопросов каждой конкретной прикладной задачи. Другими словами, решение всякой задачи требует построения специального точного языка. Как хорошо известно программистам, это язык интерфейса программного обеспечения. В терминах специалиста в computer science это специализированный диалект комбинаторной логики.

3.4. Объекты и системный подход

Если программистом для разработки избирается объектно-ориентированный подход, то скорее всего будет ошибкой подгонять решаемую задачу под какую-нибудь заранее известную математическую модель. Возможно, значительно лучше будет поискать нестандартное решение, которое в точности охватывает специфические особенности, связанные с самой природой прикладной области. В computer science для этого избирается *метатеория*, в рамках которой проводится исследование и которая "настраивается" на специфику прикладной области. Один из способов настройки — это *погружение* прикладной теории ("меньшей" теории) в чистую метатеорию ("большую теорию"). Кроме того, с математической точки зрения в рамках комбинаторной логики удобно строить подтеории — специальные матема-

тические модули, которые в готовом виде задают механизмы вычислений, имеющие самостоятельное значение (см. [2], [10]). Такие рассуждения легко найдут отклик у программиста, вынужденного заниматься большим программным проектом, когда преимущества рассуждений в терминах объектов и их свойств становятся особенно очевидными. Комбинаторная логика позволяет на математически идеализированных объектах предварительно “проиграть” все наиболее сложные и тонкие моменты взаимодействия механизмов большого программного проекта (см. [11]).

4. Аппликативные вычислительные системы

Традиционно в состав аппликативных вычислительных систем, или ABC, включают системы исчислений объектов, основанные на комбинаторной логике и ламбда-исчислении. Единственное, что существенно разрабатывается в этих системах — это представление об *объекте*. В комбинаторной логике единственный метаоператор — *аппликация*, или, по иной терминологии, *приложение* одного объекта к другому. В ламбда-исчислении два метаоператора — *аппликация* и функциональная *абстракция*, позволяющая связывать одну переменную в одном объекте.

Возникающие в этих системах объекты ведут себя как функциональные сущности, имеющие следующие особенности:

- число аргументных мест, или арность объекта заранее не фиксируется, но проявляет себя постепенно, во взаимодействиях с другими объектами;
- при конструировании составного объекта один из исходных объектов — функция, — применяется к другому — аргументу, — причем в других контекстах они могут поменяться ролями, то есть функции и аргументы рассматриваются как объекты на равных правах;
- разрешается самоприменимость функций, то есть объект может применяться сам к себе.

Вычислительные системы с таким наиболее общими и наименее ограничительными свойствами оказываются в центре внимания современного сообщества computer science. Именно они в настоящее время обеспечивают необходимые метатеоретические средства, позволяя исследовать свой-

ства целевых прикладных теорий, дают основу построения семантических средств языков программирования и обеспечивают средства построения моделей данных/метаданных в информационных системах.

Во второй половине 1970-х — начале 1980-х произошел взрыв в развитии аппликативных вычислительных систем, приведший к развитию целого спектра направлений исследований и изобилию научных публикаций. Они написаны специальным языком, их прочтение и понимание, не говоря уже об овладении существом дела, требовало основательной теоретической подготовки. Причин тому было несколько.

Во-первых, в 1980 году теория аппликативных вычислений все еще активно развивалась, а одной из целей авторов публикаций было пробудить интерес и привлечь к исследования в этой области математически одаренных студентов.

Во-вторых, с течением времени изменились роль и место, которые отводились в учебных программах для аппликативных систем. Если раньше для овладения важнейшими идеями требовалось изрядное знакомство с целым набором метаматематических дисциплин, то теперь изучение основных элементов аппликативных вычислений входит в стандартную программу, которая обязательна для студентов младших курсов. Это означает, что в соответствующих курсах предпочтение отдается содержательному методу изложения, который вовлекает только элементарные средства, делая предмет доступным.

В-третьих, за это время курсы по компьютерным наукам превратились из математизированных и концептуальных в довольно рецептурные, дающие

нечто наподобие “быстрого взгляда” на массу готовых решений, методов, технологий и рекомендаций по использованию.

В-четвертых, отношение к компьютерным наукам стало в значительной мере практическим и даже потребительским. В большинстве случаем ожидается, что овладение тем или иным разделом компьютерных наук должно дать немедленную отдачу. У студентов имеется даже вполне заметное нежелание выполнять упражнения или решать задачи творческого и теоретического плана, носящих принципиальный характер. С целью сохранения уровня обучения аппликативным вычислениям получают развитие примеры и упражнения различного характера из области программирования, причем их решения даются элементарными средствами, не предполагая никакой предварительной математической подготовки. Обычно предполагается, что обучающийся знаком с началами программирования, имея некоторое представление о структурах данных.

5. Компьютинг и обучение вычислениям с объектами

За последние двадцать лет одним из наиболее интенсивно разрабатываемым понятием было и остается представление о вычислении (computing), которое оказалось оснащенным комплексом дисциплин, состав и содержание которых непрерывно трансформируется (см. раздел *Вступление* в [8]). Только в самое последнее время специальной объединенной комиссией Ассоциации по Вычислительной Технике (Association for Computing Machinery, ACM) и Компьютерным сообществом Института Инженеров по Электротехнике и Электронике (IEEE Computer Society) был представлен отчет, содержащий рекомендации по преподаванию информатики (и computer science) и типовым учебным планам этой дисциплины. Вместе с тем и процесс научного становления этой дисциплины нельзя считать завершенным.

Хорошо известно, что анализ такого емкого понятия, как 'вычисление' нуждается в подходящем понятийном запасе. Действительно, информатика шире, чем собственно компьютерные науки, а последние несводимы только к программированию. При обучении компьютингу в продвинутых курсах отбираются и подвергаются обсуждению только такие формализации, которые напрямую способствуют изучению вопроса, ка́к же решаются задачи в этих областях. Отличительной особенностью такого подхода является систематическое изложение круга вопросов вычислений с объектами — интенсивно развивающегося направления, которое особенно важно для успешного применения информационных технологий.

Формальные средства вычислений с объектами действительно позволяют охватить все аспекты решения задач, начиная с их аккуратной поста-

новки — чтобы прийти к правильно построенному решению, — вплоть до построения стратегий, которые приводят к решениям шаг за шагом (пошаговые стратегии), и указать аналитические методы, которые дают возможность оценить и сравнить решения.

Этот путь изучения вычислений формирует квалифицированных программистов, поскольку именно программирование является той средой, которая используется для обсуждения всех указанных понятий. В частности, аппликативные вычислительные системы являются базовыми для развития методов и средств вычислений с объектами, а последние нужны тем, кто заинтересован в изучении методов постановки, решения и анализа задач.

6. Заключение

Круг затронутых в работе вопросов нельзя отождествлять с программированием, хотя в известном смысле она предназначена для адептов программирования. Базовый запас метатеоретических средств рассмотрен с позиций систематического использования операций абстракции и аппликации — применения, или приложения, одних объектов к другим, — что позволяет охватить и принципиально важный круг вопросов, относящихся к основам программирования и языкам программирования.

Методы и средства вычислений с объектами, которые обсуждены в настоящей работе, находят отражение в большинстве разделов учебных курсов по информатике и, наряду с другими методами, способствуют развитию компьютерных наук и программирования.

7. Благодарности

Выражаю искреннюю признательность С. Г. Маслову, способствовавшему выходу в свет настоящей работы.

Список литературы

1. Булкин М.А., Габович Ю.Р., Пантелеев А.Г. *Методические рекомендации по программированию и эксплуатации интерпретатора для алгоритмического языка ЛИСП в ОС ЕС.* – Киев : НИИАСС, 1981. – 91 с.
2. Вольфенгаген В.Э., Гольцева Л.В. *Аппликативные вычисления на основе комбинаторов и λ-исчисления.* – (Руководитель проекта “Аппливативные вычислительные системы” к.т.н. Л.Ю. Исмаилова.) – М.: МИФИ, 1992. – 41 с.
   Основы аппликативных вычислительных систем изложены элементарными средствами, что обеспечивает студентов и аспирантов кратким и исчерпывающим руководством, которое может использоваться ‘для первого чтения’. Настоящее руководство в различных вариантах течение ряда лет использовалось для проведения практический занятий и лабораторных работ по соответствующим разделам курса компьютерных наук. Охватываются вопросы использования комбинаторов и λ-исчисления при реализации аппликативных вычислений. Приводится необходимый теоретический минимум, основное внимание уделено выполнению упражнений, иллюстрирующих применения основных вычислительных идей, понятий и определений. Для облегчения овладения предметом руководство снабжено простой обучающей программой, которая может использоваться в качестве вводного лабораторного практикума. При практической работе с обучающей программой следует иметь ввиду, что решение задач предполагает проведение дополнительных преобразований с целью оптимизации выражений, устранения в них переменных, упрощения целевого исполняемого выражения. Аппликативные вычисления излагаются как набор таких методов и средств.
3. Вольфенгаген В.Э. *Категориальная абстрактная машина.*– М.:МИФИ, 1993. – 96 с.; – 2-е изд. – М.: АО “Центр ЮрИнфоР”, 2002. – 96 с. Работа содержит изложение базовых моделей вычислений, применяемых в компьютерных науках. Изложены основы λ-исчисления и комбинаторные исчисления.
   Основное внимание уделено подробному рассмотрению техники вычисления значения конструкций языков программирования, включая компилирование кода, его оптимизацию и исполнение на примере категориальной абстрактной машины. Изложение построено на примерах возрастающей сложности.
4. Вольфенгаген В.Э. *Конструкции языков программирования. Приемы описания.*– М.: АО “Центр ЮрИнфоР”, 2001. – 276 с.

В работе изложены основы, касающихся разработки, реализации и применения конструкций как императивных, так и функциональных языков программирования. Значительное внимание уделяется применению денотационной семантики, позволяющей в полной мере извлечь преимущества объектно-ориентированного подхода, что, в конечном счете, позволяет построить результирующую вычислительную модель чисто функционального типа.

5. Вольфенгаген В.Э. *Логика. Конспект лекций: техника рассуждений.*— М.: АО "Центр ЮрИнфоР", 2001. — 137 с.; — 2-е изд. — М.: АО "Центр ЮрИнфоР", 2004. — 229 с.
Последнее издание значительно переработано и расширено элементами техники семантических рассуждений с применением классов и отношений, что особенно важно для работы с электронными формами информации. Рассмотрены способы переформулирования текста фактического типа на символьный язык, допускающий применение классических логических средств. Показаны приемы и способы записи аргументации и проверки ее значимости. На большом числе примеров проиллюстрирована техника логических рассуждений, выводов и доказательств. Отмечены способы включения в вывод аннотаций (комментариев), пользуясь которыми можно проверить истинность или установить ложность приводимых доводов.
6. Вольфенгаген В. Э. *Комбинаторная логика в программировании. Вычисления с объектами в примерах и задачах.* — М.: МИФИ, 1994. — 204 с.; 2-е изд., М.: АО «Центр ЮрИнфоР», 2003. — 336 с.
Изложен основной круг задач, сводимых к исчислению объектов — "от простого к сложному". Конкретный вариант исчисления выбирается в зависимости от решаемых вычислительных задач. В ходе последовательного решения задач читатель овладевает основными методами и средствами комбинаторной логики и λ-исчисления. Все задачи снабжены подробными и элементарными решениями.
7. Wolfengagen V.E. *Combinatory logic in programming.* — Moscow, "Center JurInfoR" Ltd., 2003. — 336 p.
8. Вольфенгаген В. Э. *Методы и средства вычислений с объектами. Аппликативные вычислительные системы* — М.: JurInfoR Ltd., «Центр ЮрИнфоР», 2004. — xvi+789 с.
Систематически рассмотрены модели, методы и средства, для которых центральной сущностью является представление об объекте. Применен подход, основанный на использовании операций аппликации и абстракции, что позволило выполнить замкнутое изложений техники аппликативных вычислений, оставаясь в рамках элементарных средств. Книга основана на материале, который в различных вариантах использовался для проведения занятий по соответствующим разде-

лам курса компьютерных наук. Приводится необходимый теоретический минимум, соответствующий мировым стандартам, иллюстрируются основные вычислительные идеи, понятия и определения.

9. Wolfengagen V.E. *Functional notation for indexed concepts*. — In: Proceedings of The 9th International Workshop on Functional and Logic Programming WFLP'2000, Benicassim, Spain, September 28-30, 2000
   http://www.dsic.upv.es/~wflp2000/
10. Wolfengagen V.E. *Environment with state transitions for Web Information Systems: case study*. — In: Proceedings of the 6-th International Conference on Information Integration and Web-based Application & Services (iiWAS'2004), Jakarta, Indonesia, September 27-29, 2004. — pp. 99-104.
11. Wolfengagen V.E. *Environment with state transitions for Web Information Systems: case study*. — In: Proceedings of the 6th International Workshop on Computer Science and Information Technologies CSIT'2004, Budapest, Hungary, October 17-19, 2004, pp. 77-86
    Строится модель объектов данных/метаданных. Модель погружена в общую вычислительную среду и имеет возможность учета переходов состояния. Использован интенсиональный подход, основанный на применении конструкции функтор-как-объект, моделирующей объекты данных/метаданных. Модель дает основу динамическим вычислениям, учитывая 'состояния знания'. Обеспечивается чисто функциональная и аппликативная вычислительная среда. Приводится анализ случаев построения концептов на основе конструкции переменного домена (переменной области).