И. А. Кудрявцева

КЛАССИФИКАЦИЯ ПАРАДИГМ ПРОГРАММИРОВАНИЯ В КОНТЕКСТЕ ТЕОРЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Автором представлена классификация парадигм программирования в контексте теоретического программирования, основанного на ряде следующих математических дисциплин: алгебра, математическая логика, теория алгоритмов, теория формальных языков и грамматик, теория автоматов. В соответствии с выбранной структурой парадигмы программирования классификация построена на уровне математических оснований и языков программирования, не включая технологии программирования. В результате отмечается, что в контексте теоретического программирования акцент смещается в сторону функциональной парадигмы программирования.

Ключевые слова: парадигма программирования, теоретическое программирование, математические основания парадигм программирования, формальные языки, порождающие модели, вычислительные модели, императивные языки программирования, декларативные языки программирования.

I. Kudryavtseva

CLASSIFICATION OF PROGRAMMING PARADIGMS IN THE CONTEXT OF THEORETICAL PROGRAMMING

The author presents a classification of programming paradigms in the context of theoretical programming, based on a number of the following math subjects: algebra, mathematical logic, theory of algorithms, theory of formal languages and grammars, automata theory. In accordance with the structure of the selected programming paradigms classification is based on the level of mathematical bases and programming languages, not including technologies of programming. As a result, it is noted that in the context of theoretical programming emphasis is shifting towards a functional programming paradigm.

Keywords: programming paradigm, theoretical programming, mathematical foundations of programming paradigms, formal languages, generative models, computational models, imperative programming languages, declarative programming languages.

Объектом изучения теоретического программирования являются реальные компьютерные программы, представляющие собой текст на фиксированном языке программирования, либо цепочку битов памяти компьютера. Однако многообразие языков программирования, отражающих разные подходы к организации вычислительного процесса, приводит к необходимости в построении их классификации.

Концептуально языки программирования принято классифицировать в соответствии с парадигмами программирования, то есть с помощью некоторых концепций, принципов и абстракций. В силу того, что термин «*парадигма программирования*» определяется достаточно абстрактно, то и классификации парадигм программирования строятся в основном на основе признаков, важных автору классификации в соответствии с классами решаемых задач конкретной предметной области.

Вполне очевидно, что теоретическое программирование основано на ряде следующих математических дисциплин: *алгебра*, *математическая логика*, *теория алгоритмов*, *теория формальных языков и грамматик*, *теория автоматов*.

Поскольку неотъемлемой частью теоретического программирования является взаимопроникновение математики и программирования, выберем за основу определения термина «*парадигма программирования*», представленные в работе [3]. Наш выбор основан на приведённой структуре описания парадигмы программирования, основными компонентами которой являются *математические основания*, *языки программирования* и *технологии программирования*.

Выполним классификацию парадигм программирования в контексте теоретического программирования только на уровне математических оснований и языков программирования с опорой:

(а) на классификацию формальных языков [4, с. 208], определяющих содержание обучения математическим основаниям информатики;

(б) на классификацию парадигм программирования [5, с. 85–120], проведённую на основе анализа различных классификаций языков программирования и не утратившую свою фундаментальность со временем.

Ниже мы ограничимся лишь двумя уровнями классификации парадигм программирования (математические основания и языки программирования), не затрагивая технологии программирования. Связано это с тем, что технологии программирования определяют процесс сборки программы посредством инструментальной единицы декомпозиции программы, которая отражена в математических основаниях и выражена в языках программирования.

Кроме этого, будем опираться на философские концепции, определяющие природу математических знаний [11, с. 341,516,518; 2; 6, с. 286]:

(1) *формализм* — направление в математике, пытающееся получить решение проблем основания математики при помощи формально-аксиоматических построений;

(2) *финитизм* — философская концепция, отрицающая объективно реальное содержание категории бесконечного, исходящая из того, что бесконечность не имеет места ни во Вселенной, ни в микромире, ни в нашем мышлении. Основание этого финитизм видит в том, что человек в опыте всегда имеет дело с конечными вещами и их свойствами;

(3) *конструктивизм* с опорой на теорию множеств с уточнением понятия «алгоритм». *Конструктивное направление в математике* — это математическое мировоззрение, связанное с признанием исследования конструктивных процессов и конструктивных объектов основной задачей математики. Основная идея конструктивизма — идея алгоритмического построения логических и математических объектов;

(4) *операционализм* — направление в философии, по своей природе являющееся синтезом *логического позитивизма* и *прагматизма*. Суть операционализма выражена в идее операционального анализа, которую *П. Бриджмен* [13, с. 8] трактовал так: «нам неизвестно значение понятия до тех пор, пока не определены операции, которые используются нами или нашими коллегами при применении этого понятия в некоторой конкретной ситуации»;

(5) *структурализм* — философский подход, в основе которого лежит идея структуры. В центре внимания данного подхода оказалась лингвистика *Ф. де Соссюра* (по [2, с. 94–98]), который рассматривал язык в качестве знаковой системы.

Проведём классификацию парадигм программирования на уровне математических оснований (*схемы 1, 2, 3*).

Математические основания парадигм программирования — это формально-языковые средства, заимствованные из математики, позволяющие описывать конструктив-

ный процесс вычислений, тем самым определяя особенности синтаксиса и семантики языков программирования, входящих в рассматриваемую парадигму.

Конструктивный процесс представляет собой вывод конструктивного объекта из уже построенных объектов по определённым правилам вывода, применение которых осуществляется либо по точному предписанию (*алгоритму*), либо с помощью выбора правила некоторого *исчисления* в соответствии со сложившейся ситуацией.

Схема 1

Структура *схемы 1* обосновывается смыслами понятий, используемых в ней, поэтому напомним некоторые определения.

Язык категориального описания (язык теории категорий) — формальный язык описания объектов с помощью их соответствий (морфизмов).

Языки теоретико-множественного описания — формальные языки, метаязыком которых является язык канторовской (наивной) теории множеств.

Конструктивный процесс, заданный с помощью алгоритма, можно описать посредством математического объекта — *вычислимой функции* (теория рекурсивных функций), а конструктивный процесс, заданный с помощью исчисления, можно описать с помощью *формальных языков комбинаторной* и *математической логики*.

Комбинаторная логика (*в широком смысле*) — это раздел *логики*, посвящённый изучению и анализу таких понятий и методов, как «*переменная*», «*функция*», операция «*подстановка*», классификация предметов по типам или категориям.

Выделим элементы комбинаторной логики, составляющие математические основания *функциональной парадигмы программирования*:

(1) бестиповое λ-исчисление;

(2) системы типизированного лямбда-исчисления с явным приписыванием типов (λ-*куб Барендрегта*);

(3) исчисление комбинаторов (в том числе с типами).

Математическая логика — это раздел математики, посвящённый изучению *математических теорий* и, в частности, основного для математики понятия — понятия «*математическое доказательство*».

Элементы математической теории — аксиома, теоремы, доказательства — могут рассматриваться с точки зрения их смысла и как символьные объекты.

Выделим элементы математической логики, составляющие математические основания *логической парадигмы программирования*: гильбертовское исчисление первого порядка, исчисление дедуктивных эквивалентностей, секвенциальное исчисление (моносукцедентное, многосукцедентное), генценовское исчисление натурального вывода.

Языки конструктивного описания — формальные языки, способствующие индуктивному описанию объекта.

Конструктивный процесс, заданный с помощью алгоритма, можно описать посредством вычислительных моделей, а конструктивный процесс, заданный с помощью исчисления, можно описать с помощью порождающих моделей.

Порождающие модели — известные семейства исчислений.

Вычислительные модели — это формальные конструкции с механизмом, позволяющим уточнить понятие «алгоритм», моделирующие понятия «*вычислительный процесс*», «*вычислимая функция*», «*исполнитель*».

Построение классификации продолжим в соответствии с существующими классификациями представительных вычислительных моделей (по работам [10; 8, с. 294]) и представительных порождающих моделей [8, с. 296].

Представительные вычислительные модели позволяют описывать любой алгоритм. *Представительность порождающей модели* обеспечивает содержание в ней исчисления, эквивалентного любому заданному исчислению.

Математические основания *императивной парадигмы программирования* составляют представительные вычислительные модели: ***м***ашина ***П***оста-***У***спенского (МПУ), ***р***авнодоступная ***а***дресная ***м***ашина (РАМ), ***р***авнодоступная ***а***дресная машина ***с*** хранимой ***п***рограммой (РАСП) и ***м***ашина с ***н***еограниченными ***р***егистрами (МНР).

Математические основания *продукционной парадигмы программирования* составляют представительные вычислительные модели (***м***ашина ***Т***ьюринга (МТ), ***н***ормальные ***а***лгорифмы ***М***аркова (НАМ)) и представительные порождающие модели (полусистемы Туэ).

Непредставительные вычислительные модели — это математические структуры (называемые *автоматами*), которые обладают ограниченными возможностями порождения и распознавания языков в алфавите, что приводит к узости алгоритмических языков, которые могут быть построены на базе данных моделей.

Такие непредставительные модели, как ***к***онечные ***а***втоматы (КА), клеточные автоматы, автоматы Уотсона — Крика (используемые в молекулярных вычислениях), ***м***агазинные ***а***втоматы (МА), помеченные сети Петри и КС-грамматики, составляют парадигму «*программирование от состояний*».

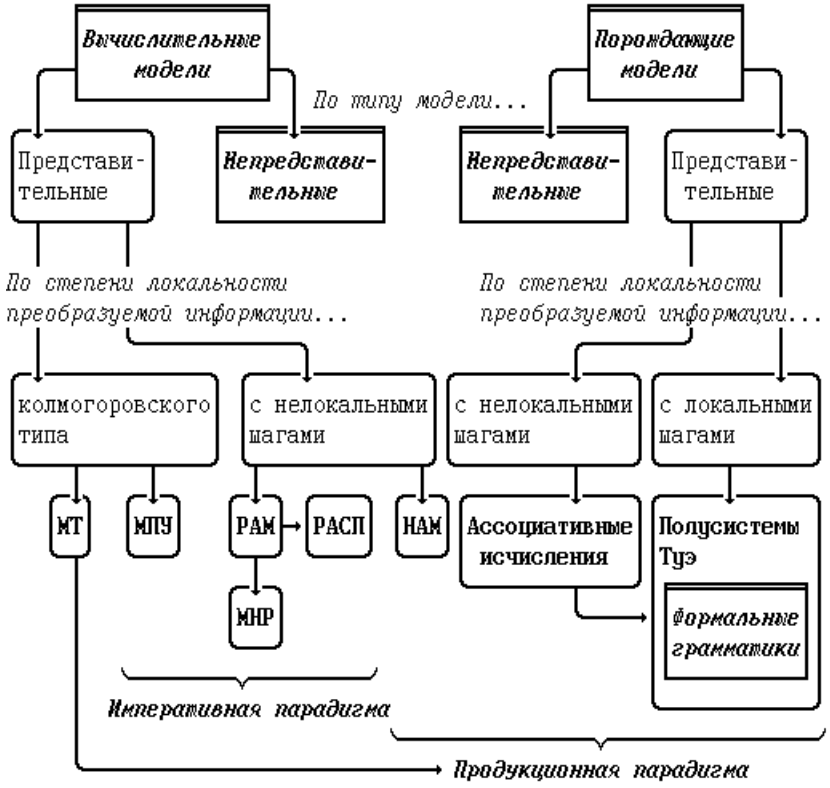


Схема 2

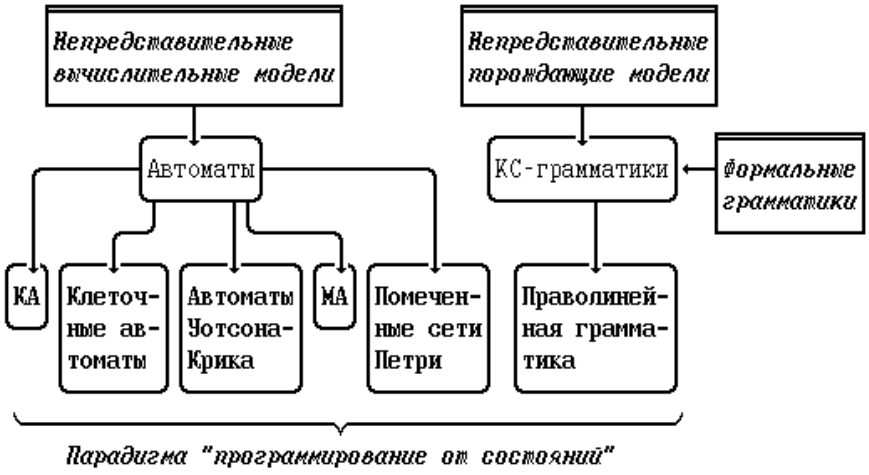


Схема 3

Итак, математические основания парадигм программирования задают специфику операционной семантики языков программирования (*схемы 4*, *5*).

Структура *схемы 4* обосновывается смыслами понятий, используемых в ней, поэтому напомним некоторые определения.

Языки императивные [7] — это класс языков программирования (языков манипулирования данными) в системах управления базами данных. Программа, написанная на императивном языке, явно указывает способ получения желаемого результата, не определяя

при этом его ожидаемых свойств. Процедура получения желаемого результата имеет вид последовательности операций, поэтому для императивных языков характерно указание порядка выполнения операторов. В основе такого программирования лежат извлечение из памяти значения некоторой переменной, совершение над ним действия и сохранение нового значения с помощью оператора присваивания. Императивные языки тесно связаны с фон-неймановской моделью вычислений.

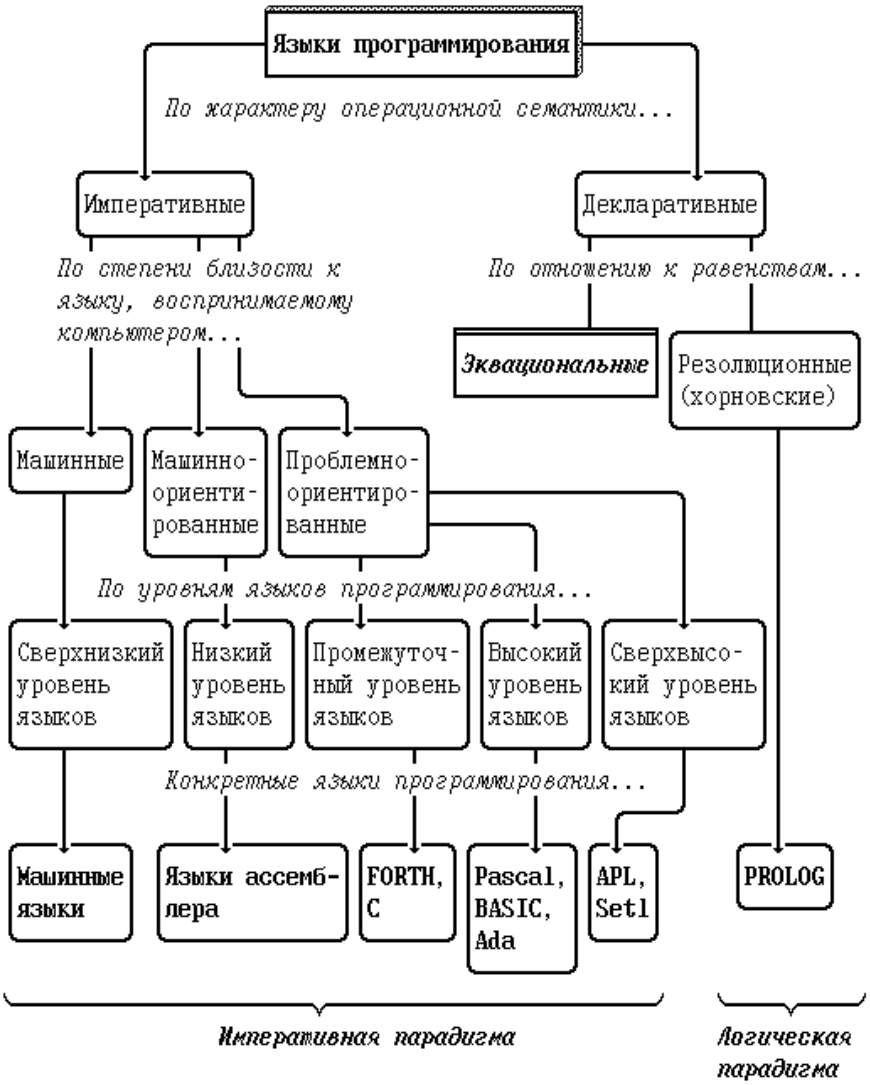


Схема 4

На *схеме 4* отражена классификация императивных языков, построенная *М. В. Швецким* [5, с. 109].

Декларативные языки [7] — это класс языков программирования, программа на которых задаёт связи и отношения между объектами и величинами и не определяет последовательность выполнения действий. В программе в явном виде указывается, какими свойствами должен обладать результат, но не говорится, каким способом он будет получен. В силу этого такие понятия, как упорядочение и поток команд управления, не имеют никакого отношения к этим языкам, и в них отсутствуют операторы присваивания. Декларативные

языки не привязаны к классической фон-неймановской модели вычислений, и в типичном случае алгоритм достижения требуемого результата может иметь высокую степень параллелизма.

Декларативные языки по отношению к равенствам делятся на эквациональные и резолюционные.

Резолюционные (*хорновские*) *языки* (в узком смысле) [7] — класс языков программирования и подкласс декларативных языков, которые основаны на символической логике. Программа может рассматриваться как набор логических формул совместно с теоремой (запросом), которая должна быть доказана. Формулы предназначены для представления различных фактов (данных) и правил вывода. Для хранения и эффективной работы с фактами и правилами привлекается сопоставление с образцом (*унификация*). При этом программист избавляется от необходимости определения точной последовательности шагов для выполнения вычислений.

Этот класс языков составляет *логическую парадигму программирования*, основанную на использовании механизма доказательства теорем, позволяющего выяснить, противоречиво ли некоторое множество логических формул.

Эквациональные языки — класс языков программирования, программа на которых задаётся посредством равенств, то есть формул вида $p = q$, где p и q — термы, построенные из функциональных и предикатных символов, переменных и констант.

По семантике равенств, как отмечено в работе [5, с. 117], различают два класса эквациональных языков программирования (*схема 5*): функциональные и продукционные.

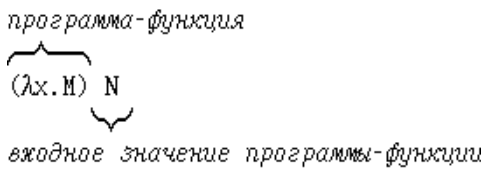
Логические вычисления, определённые в программах на эквациональных языках, подразумевают собой машины для вычислений другой архитектуры, отличной от традиционной.

Как и ранее, структура *схемы 5* обосновывается смыслами понятий, используемых в ней, поэтому напомним некоторые определения.

Языки функционального программирования (по литературе [5, с. 117]) — это подкласс декларативных языков, основанный на идеях лямбда-исчисления и теории рекурсивных функций. Основное понятие — *функция*. В этих языках нет понятия переменной и присваивания, поэтому значение функции зависит только от её аргументов и не зависит от порядка вычислений. Функциональная программа состоит из совокупности определений функций, которые, в свою очередь, представляют собой вызовы других функций и предложений, управляющих последовательностью вызовов. Функции часто либо прямо, либо опосредованно вызывают сами себя. Каждый вызов возвращает некоторое значение в вызвавшую его функцию, вычисление которой после этого продолжается; этот процесс повторяется до тех пор, пока запустившая вычисления функция не выдаст конечный результат пользователю. Повторные вычисления осуществляются посредством рекурсии, являющейся основным средством функционального программирования.

Стратегией вычислений в языках функционального программирования называется порядок выполнения β-редукции.

Будем говорить, что *происходит* β-*редукция вхождения редекса в* λ-*терм* ***P*** (по работе [12, с. 123]), если β-редекс



имеет вхождения в ***P***, и *одно из них* заменяется λ-термом $\left(M\right)_N^x$, который отображает операцию подстановки в терм M вместо свободного вхождения предметной переменной x терма N.

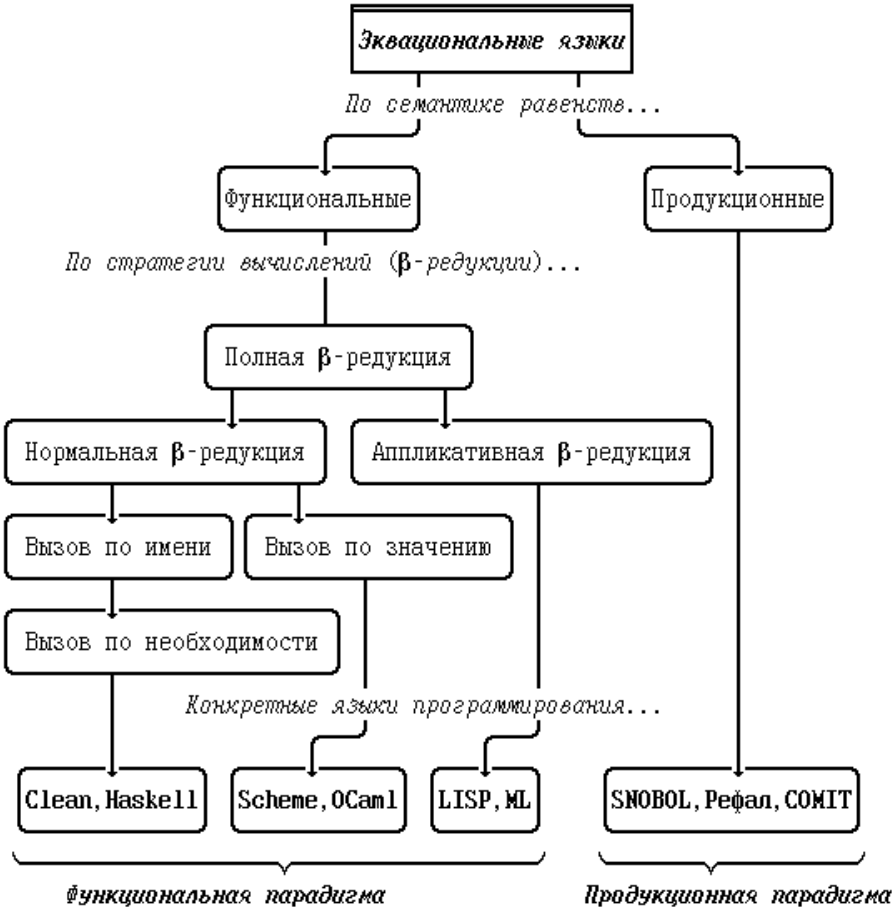


Схема 5

При *полной* β-*редукции* может сработать любой редекс.

При *нормальной* β-*редукции* всегда сначала сокращается *самый левый, самый внешний редекс*.

Стратегия вызова по имени является более строгим вариантом стратегии нормального порядка: она не позволяет проводить редукцию внутри абстракций (безымянных функций-термов, начинающихся с λ).

Стратегия вызова по необходимости является оптимизированной версией стратегии вызова по имени, в которой вместо того, чтобы повторно вычислять значение аргумента при каждом использовании, при первом вычислении все вхождения аргумента заменяются значением, и, таким образом, отпадает необходимость вычислять его заново в следующий раз.

В соответствии со *стратегией вызова по значению* сокращаются только самые внешние редексы, и, кроме того, редекс срабатывает только в том случае, если его правая часть уже сведена к *значению* — замкнутому λ-терму, который уже вычислен и не может быть редуцирован далее.

Стратегия *вызова по значению строга* в том смысле, что аргументы функции всегда вычисляются независимо от того, используются они в теле функции или нет.

С другой стороны, *нестрогие* (или *ленивые*) *стратегии вычисления — вызов по имени* или *по необходимости* — вычисляют только те аргументы, которые действительно используются в вычислениях.

Языки продукционного программирования [7] — класс декларативных языков программирования, основанных на подходе к программированию, при котором программа задается совокупностью правил (продукций) без явного указания последовательности их применения. Правила содержат либо условие и действия, которые должны быть выполнены в случае истинности этого условия, либо условие и совокупность других условий, достаточных для истинности этого условия.

Относительно парадигмы «*программирование от состояний*» отметим, что в неё входят произвольные языки программирования, так как она требует мышления не в терминах семантики конструкций языков программирования, а в средствах описания автоматов, используемых как средства для моделирования вычислительных процессов.

Помимо представленных классов языков программирования, относящихся к той или иной парадигме программирования, существуют языки, поддерживающие несколько парадигм программирования.

Например, *эзотерические языки программирования* (от гр. *esoterikos — внутренний*), заключающие в себе идеи, возможно, о языках программирования будущего. К подобным языкам можно отнести (по работе [9]): Spaghetti, Come Here, Befunge, Whitespace, Malborge, Nil, Unlambda, Iota, Joy.

Анализ теоретических исследований в области теоретического программирования показывает, что обычно используют математический аппарат, содержащий следующие взаимосвязанные разделы (по литературе [1, с. 27–28]):

(1) λ-*исчисление*, в котором изучаются способы построения чистого исчисления функциональной абстракции и аппликации функций; λ-исчисление выполняет важную роль при изучении оснований математики, является основным аппаратом исследования систем типизации и инструментом для разработки и применения языков программирования;

(2) *комбинаторная логика*, в которой показано, что связанные переменные можно исключить без потери выразительных возможностей формальной системы. Комбинаторная логика нашла применение для построения оснований математики, а также в развитии методов и средств реализации языков программирования;

(3) *теория типов языков программирования*;

(4) *теория категорий*, в которой рассматривается язык математики, отличный от языка теории множеств и содержащий всего две сущности (*объекты* и *морфизмы*), причём морфизмы, в свою очередь, можно считать объектами.

Перечисленные выше направления (λ-исчисление, комбинаторная логика, теория типов, теория категорий) трудно рассматривать изолированно: они взаимно переплетены, и современная точка зрения заключается в их *совместном изучении*.

Более того, именно эти четыре раздела являются основой для конструирования языков программирования.

Из представленных компонентов теоретического программирования в классификацию парадигм программирования вошли: язык категориального описания, языки комбинаторной логики и сами парадигмы программирования. Поэтому становится очевидным, что *в контексте теоретического программирования* смещение происходит в направлении от

императивной парадигмы (близкой к архитектуре компьютеров) в сторону функциональной. В частности, это связано с тем, что языки функционального программирования обеспечивают написание более короткого кода по сравнению с императивными языками.

Отметим, что в контексте компьютерных исследований, связанных с направлением «интеллектуальные информационные системы», на первый план выходит понятие «*исчисление*», связанное со смещением в сторону декларативного программирования и подразумевающее пересмотр архитектуры существующих вычислительных машин.

СПИСОК ЛИТЕРАТУРЫ

1. *Вольфенгаген В. Э.* Комбинаторная логика в программировании. Вычисления с объектами в примерах и задачах. М.: АО «Центр ЮрИнфоР», 2003. 336 с.

2. *Канке В. А.* Основные направления и концепции науки. Итоги XX столетия. М.: Логос, 2000. 320 с.

3. *Коротков А. В., Кудрявцева И. А.* К определению понятия «парадигма программирования» // Теоретические и методические проблемы обучения в школе и вузе (математика, информатика): Межвузовский сборник научных трудов. СПб.; Мурманск, 2005. С.107–112.

4. *Лаптев В. В., Рыжова Н. И., Швецкий М. В.* Методическая теория обучения информатике. Аспекты фундаментальной подготовки. СПб.: Изд-во Санкт-Петербургского университета, 2003. 352 с.

5. *Лаптев В. В., Швецкий М. В.* Методическая система фундаментальной подготовки в области информатики: теория и практика многоуровневого педагогического университетского образования. СПб.: Изд-во Санкт-Петербургского университета, 2000. 508 с.

6. Математический энциклопедический словарь. М.: Сов. энциклопедия, 1988; 1995. 847 с.

7. *Першиков В. И., Савинков В. М.* Толковый словарь по информатике. М.: Финансы и статистика, 1995. 543 с.

8. *Рыжова Н. И., Голанова А. В., Швецкий М. В.* Упражнения по теории алгоритмов: Учебное пособие для студентов математического факультета. СПб.: Дмитрий Буланин, 2000. 304 с.

9. *Сондерс М.* Лингва эзотерика // LINUX Format, март 2007, 3 (90). С.42–45.

10. *Успенский В. А., Семенов А. Л.* Теория алгоритмов: основные открытия и приложения. М.: Наука, 1987. 288 с.

11. *Философский* словарь. М.: Политиздат, 1987. 590 с.

12. *Хиндли Дж. Р.* Комбинаторы и лямбда-исчисление. Краткий обзор / Математическая логика в программировании: Сб. статей 1980–1988 гг. М.: Мир, 1991. С.119–140.

13. *Bridgman P. W.* The nature of some of our physical concepts. N. Y., 1952.

REFERENCES

1. *Vol'fengagen V. Je.* Kombinatornaja logika v programmirovanii. Vychislenija s objektami v primerah i zadachah. M.: AO «Tsentr JurInfoR», 2003. 336 s.

2. *Kanke V. A.* Osnovnye napravlenija i kontseptsii nauki. Itogi XX stoletija. M.: Logos, 2000. 320 s.

3. *Korotkov A. V., Kudrjavtseva I. A.* K opredeleniju ponjatija «paradigma programmirovanija» // Teoreticheskie i metodicheskie problemy obuchenija v shkole i vuze (matematika, informatika): Mezhvuzovskij sbornik nauchnyh trudov. SPb.; Murmansk, 2005. S.107–112.

4. *Laptev V. V., Ryzhova N. I., Shvetskij M. V.* Metodicheskaja teorija obuchenija informatike. Aspekty fundamental'noj podgotovki. SPb.: Izd-vo Sankt-Peterburgskogo universiteta, 2003. 352 s.

5. *Laptev V. V., Shvetskij M. V.* Metodicheskaja sistema fundamental'noj podgotovki v oblasti informatiki: teorija i praktika mnogourovnevogo pedagogicheskogo universitetskogo obrazovanija. SPb.: Izd-vo Sankt-Peterburgskogo universiteta, 2000. 508 s.

6. Matematicheskij entsiklopedicheskij slovar'. M.: Sov.jentsiklopedija, 1988; 1995. 847 s.

7. *Pershikov V. I., Savinkov V. M.* Tolkovyj slovar' po informatike. M.: Finansy i statistika, 1995. 543 s.

8. *Ryzhova N. I., Golanova A. V., Shvetskij M. V.* Uprazhnenija po teorii algoritmov: Uchebnoe posobie dlja studentov matematicheskogo fakul'teta. SPb.: Dmitrij Bulanin, 2000. 304 s.

9. *Sonders M.* Lingva ezoterika // LINUX Format, mart 2007, 3 (90). S.42–45.

10. *Uspenskij V. A., Semenov A. L.* Teorija algoritmov: osnovnye otkrytija i prilozhenija. M.: Nauka, 1987. 288 s.

11. Filosofskij slovar'. M.: Politizdat, 1987. 590 s.

12. *Hindli Dzh. R.* Kombinatory i ljambda-ischislenie. Kratkij obzor / Matematicheskaja logika v programmirovanii: Sb. statej 1980–1988 gg. M.: Mir, 1991. S.119–140.

13. *Bridgman P. W.* The nature of some of our physical concepts. N.Y., 1952.

УДК 004.932.2

Ю. В. Шуплецов

ПРЯМОЕ ВЫЧИСЛЕНИЕ МУЛЬТИФРАКТАЛЬНОГО СПЕКТРА ДЛЯ ИЗОБРАЖЕНИЙ БИОМЕДИЦИНСКИХ ПРЕПАРАТОВ

(Работа выполнена при поддержке гранта РФФИ N 13-01-00782)

Одним из методов получения статистических характеристик цифровых изображений является так называемый мультифрактальный формализм. В его основе лежит предположение о том, что многие процессы, особенно биологические, развиваются по степенным законам. Это означает, что, рассматривая изображение как фазовый портрет изучаемого процесса, мы можем характеризовать его с помощью такого неоднородного распределения некоторой меры, которое может быть описано с использованием не одной, а нескольких шкал (масштабов). Мультифрактальный формализм описывает статистические свойства этих мер в терминах сингулярных спектров или в терминах обобщённых размерностей.

Ключевые слова: полутоновые изображения, фрактальный анализ, фультифрактальный спектр.

Yu. Shupletsov

A DIRECT CALCULATION OF THE MULTIFRACTAL SPECTRA OF BIOMEDICAL ITEMS

One of the methods to obtain statistical characteristics of digital images is a so called multifractal formalism. The method is based on the assumption that most of images can be considered as phase portraits of complex dynamical systems and can be described by measure distributions. This distribution can be set using not one but several scales (measures). Multifractal formalism describes the statistical properties of the measure in terms of singular spectra or generalized dimensions. In this paper we investigate the possibility of applying the method of direct calculation of multifractal spectra to the analysis of certain classes of biomedical items' images. This method offers an image partition on a especially constructed subsets and calculation of fractal dimensions for each of them. In some cases the method is more preferable than the calculation of generalized dimensions.

Keywords: gray-scale images, fractal analysis, multifractal spectrum.

В данной работе мы применили прямое вычисление мультифрактального спектра к анализу изображений некоторых классов биомедицинских препаратов. Изображение разбивается на набор множеств уровня, в каждом из которых содержатся точки, обладающие близкими характеристиками. Набор фрактальных размерностей множеств уровня образует