



А.Я. Аноприенко • С.В. Иваница

ПОСТБИНАРНЫЙ КОМПЬЮТИНГ И ИНТЕРВАЛЬНЫЕ ВЫЧИСЛЕНИЯ В КОНТЕКСТЕ КОДО-ЛОГИЧЕСКОЙ ЭВОЛЮЦИИ

УНИТЕХ

Министерство образования и науки,
молодежи и спорта Украины

Донецкий национальный
технический университет

А.Я. Аноприенко, С.В. Иваница

**ПОСТБИНАРНЫЙ КОМПЬЮТИНГ
И ИНТЕРВАЛЬНЫЕ ВЫЧИСЛЕНИЯ
В КОНТЕКСТЕ КОДО-ЛОГИЧЕСКОЙ
ЭВОЛЮЦИИ**

М о н о г р а ф и я

Донецк
УНИТЕХ
2011

УДК 004.2
ББК 32.97
А69

Рецензенты:

Каргин А.А., д.т.н., профессор, декан физического факультета Донецкого национального университета;

Башков Е.А., д.т.н., профессор, заведующий кафедрой прикладной математики и информатики ДонНТУ, проректор по научной работе ДонНТУ;

Баркалов А.А., д.т.н., профессор кафедры компьютерной инженерии ДонНТУ.

Публикуется в соответствии с решением Ученого совета Донецкого национального технического университета № 11 от 23 декабря 2011 года.

Аноприенко А.Я., Иваница С.В.

А69 Постбинарный компьютеринг и интервальные вычисления в контексте кодо-логической эволюции. / А.Я. Аноприенко, С.В. Иваница — Донецк: ДонНТУ, УНИТЕХ, 2011. — 248 с.

ISBN 978–966–8248–20–7

Монография посвящена рассмотрению закономерностей кодо-логической эволюции средств и методов компьютеринга и особенностей перехода к постбинарному компьютерингу, рассматриваемому в качестве следующего этапа в развитии средств и методов вычислений. Детально рассмотрены интервальные вычисления и особенности их постбинарной реализации.

Материалы монографии предназначены для исследователей, специалистов, аспирантов и магистрантов, специализирующихся в области компьютерных наук и технологий.

УДК 004.2
ББК 32.97
А69

ISBN 978–966–8248–20–7

© Аноприенко А.Я., 2011
© Иваница С.В., 2011
© ДонНТУ, 2011

СОДЕРЖАНИЕ

Введение	5
----------------	---

ГЛАВА 1. ОБОБЩЕННЫЙ КОДО-ЛОГИЧЕСКИЙ БАЗИС 9

1.1. Эволюция идеи	9
1.2. Двумерное логическое пространство	25
1.3. Трехмерное логическое пространство	34
1.4. Монологика и монокоды	38
1.5. Диалогика и дикоды	45
1.6. Трилогия и трикоды	49
1.7. Тетралогия и тетракоды	53
1.8. Выводы	59

ГЛАВА 2. ПРОСТРАНСТВО И АЛГЕБРА ТЕТРАЛОГИКИ 61

2.1. Основные тенденции перехода к пространству тетралогии	61
2.2. Реализация нольместных и одноместных логических операций	66
2.3. Реализация базовых двуместных логических операций	72
2.4. Представление элементов тетралогии с помощью аксиоматического аппарата теории множеств	77
2.5. Реализация базовых логических операций над элементами тетралогии, представленными с помощью аксиоматического аппарата теории множеств	82
2.6. Выводы	93

ГЛАВА 3. КОДО-ЛОГИЧЕСКАЯ ЭВОЛЮЦИЯ И ИНТЕРВАЛЬНЫЕ ВЫЧИСЛЕНИЯ.....95

3.1. Развитие понятия числа.....	95
3.2. Эволюция алгоритмического базиса	101
3.3. Эволюция интервальных вычислений.....	107
3.4. Сравнение временных затрат при интервальных вычислениях в математических пакетах Scilab и Mathematica	119
3.5. Верификация интервальных вычислений.....	126
3.6. Постбинарные методы реализации интервальных вычислений в компьютерном моделировании.....	133
3.7. Выводы	138

ГЛАВА 4. ТЕТРАКОДЫ И ПОСТБИНАРНЫЕ ВЫЧИСЛЕНИЯ.....139

4.1. От бинарного к постбинарному компьютеру.....	139
4.2. Особенности постбинарного кодирования на примере интервального представления результатов вычислений по формуле Бэйли-Боруэйна-Плаффа	150
4.3. Отображение тетракодов на множестве интервальных чисел	155
4.4. Пример Румпа в контексте традиционных и интервальных и вычислений	173
4.5. Постбинарные вычисления и преодоление ограничений разрядности	189
4.6. Реализация постбинарных форматов чисел с плавающей запятой	196
4.7. Способы представления вещественных чисел в постбинарных форматах	202
4.8. Выводы	217
Заклучение.....	219
Литература.....	221

Введение

Насыщенность современной техносферы компьютерными технологиями постоянно возрастает при практически экспоненциальном росте объемов вычислений. Это существенно актуализирует вопросы обеспечения эффективности, надежности и гибкости средств и методов компьютеринга. Установлено, в частности, что многие техногенные катастрофы последних десятилетий, причину которых традиционно объяснили преимущественно «человеческими факторами», были в первую очередь обусловлены разного рода недостатками компьютерных систем управления и вычислительными ошибками. Но в большинстве случаев ошибки в вычислениях просто остаются незамеченными, существенно искажая полученные результаты.

В качестве типичного примера такого рода можно привести полином Румпа (впервые опубликован еще в 1988 году), который при определенном сочетании значений переменных дает заведомо неправильный результат при всех стандартных значениях точности вычислений с плавающей запятой практически на всех современных компьютерных системах.

В связи с этим следует признать, что в настоящее время созрели все предпосылки для существенной модификации всей системы компьютерных вычислений с целью повышения ее надежности и адекватности современным требованиям. При этом речь может идти о дальнейшем развитии как логической, так и вычислительной составляющей современного компьютеринга. Особо актуальной является разработка такой модификации вычислений с плавающей запятой, которая позволила бы исключить

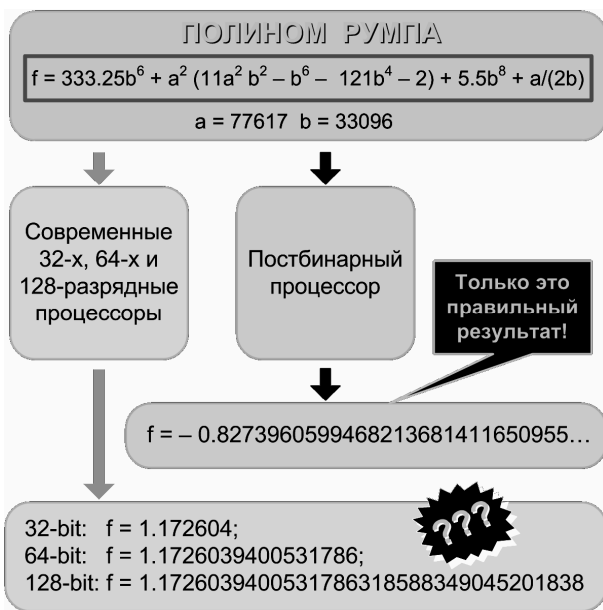
потерю точности (и информации о ней) при представлении входных, промежуточных и результирующих значений в форматах с плавающей запятой.

Наиболее перспективным вариантом модификации представляется расширение существующих стандартных форматов представления чисел с плавающей запятой и алгоритмов работы с ними путем введения ряда изменений, обеспечивающих совместное гибкое использование различных форматов чисел в диапазоне от 32-х двоичных разрядов до неопределенно больших значений. При этом вплоть до разрядности 128 обеспечивается совместимость с существующими форматами, а начиная с разрядности 256 обеспечивается последовательное удвоение разрядности формата по мере необходимости. В качестве основного постбинарного формата рассматривается тетракод. На его базе реализуется представление интервальных типов данных и идентификация реально значимых разрядов численных значений, а в алгоритмы выполнения арифметических операций вносятся такие изменения, которые позволяют автоматически изменять разрядность по мере необходимости. Это обеспечивает «вычисления без округлений», а, следовательно, и без потери точности.

Главной особенностью анализа эволюции средств и методов компьютеринга в данной монографии является предположение о том, что развитие компьютерной логики и арифметики тесно взаимосвязано и взаимообусловлено, в связи с чем рассматривается именно кодо-логическая эволюция как целостное закономерное явление, определившее в свое время переход от добинарного (пребинарного или прабинарного) компьютеринга к современному бинарному. Анализ закономерностей этого перехода позволяет прогнозировать переход в ближайшие десятилетия к постбинарному компьютерингу, неизбежность которого обусловлена

интенсивным развитием современных компьютерных технологий.

В целом предложенные в монографии модификации вычислений с плавающей запятой позволяют реализовать постбинарный процессор, который обеспечивает эффективное распараллеливание вычислительных процессов с учетом реально требуемой точности вычислений и обеспечением необходимой надежности вычислений (см. рисунок ниже).



В первой главе монографии рассматриваются основные особенности и закономерности кодо-логической эволюции, многомерное логическое пространство, различные варианты компьютерной логики и кодирования количественной информации, обосновывается актуальность перехода к постбинарному компьютерингу.

Во **второй главе** детально рассматриваются логическое пространство и алгебра тетралогики. При этом анализируются основные тенденции перехода к пространству тетралогики и рассматривается реализация нольместных, одноместных и двуместных логических операций. Анализируется также представление элементов тетралогики с помощью аксиоматического аппарата теории множеств и реализация базовых логических операций над элементами тетралогики.

В **третьей главе** описывается развитие понятия числа в контексте истории математики и компьютерных вычислений. Рассматривается эволюция алгоритмического базиса интервальных вычислений и предлагаются постбинарные методы реализации интервальных вычислений для компьютерного моделирования и других критичных по производительности и надежности приложений.

В **четвертой главе** анализируются особенности перехода от бинарного к постбинарному компьютерингу и особенности постбинарного кодирования на примере интервального представления результатов вычислений по формуле Бэйли-Боруэйна-Плаффа. Детально анализируется пример Румпа в контексте традиционных и интервальных вычислений и аргументируется использование постбинарных вычислений в качестве достаточно универсального средства обеспечения надежности компьютерных вычислений и преодоления ограничений разрядности. Предлагаются и обосновываются различные варианты реализации постбинарных форматов чисел с плавающей запятой и способы представления вещественных чисел в постбинарных форматах.

В **заключении** приводится обобщенная архитектура постбинарного процессора, реализующего описанные в монографии средства и методы вычислений.

Глава 1

ОБОБЩЕННЫЙ КОДО-ЛОГИЧЕСКИЙ БАЗИС

1.1. Эволюция идеи¹

Концепция обобщенного кодо-логического базиса [1] является развитием ряда идей, впервые сформулированных одним из авторов данной работы еще в 1986 году в результате поиска наиболее эффективных средств и методов кодирования изображений для высокопроизводительных систем машинной графики. Одна из таких идей, в частности, кратко была изложена в работе [2], в которой предлагалось обобщение метода приоритетов для повышения эффективности описания трехмерных объектов и сцен путем использования октантных полидеревьев. В целом можно утверждать, что круг идей, связанных с представлением объектов различной размерности в виде древовидных структур (двоичных, квадрантных, октантных и др.), стал одним из основных источников концепции многомерного (обобщенного) кодо-логического базиса. Материалы исследований в данном направлении частично вошли в кандидатскую диссертацию одного из авторов [3], частично реализованы в виде технических решений, новизна которых подтверждена целым рядом авторских свидетельств на изобретения (см., например, [4, 5]).

Другим источником концепции стал поиск компьютерных **аналогов генетического кодирования**. Благодаря

¹ Раздел подготовлен по материалам публикации [1].

стечению обстоятельств хронология и непосредственные причины формирования идей данного направления зафиксированы с предельной точностью. В порядке объяснения авторской мотивации исследований в области обобщенного кодо-логического базиса уместно привести некоторые детали этих событий. Непосредственно после успешной защиты кандидатской диссертации (октябрь 1987 года), в которой исследовались в числе прочих и вопросы иерархического представления информации применительно к системам компьютерной графики, естественным образом возник вопрос о наиболее эффективных вариантах дальнейшего продолжения исследований в данном направлении. Определяющим моментом можно считать знакомство с работой «Алгоритмы развития» академика Н.Н. Моисеева, где обращалось особое внимание на феномен генетического кода, который, являясь единым для всего живого, строится на базе всего 4-х символов, обеспечивающих не просто сохранение и передачу чрезвычайно объемной наследственной информации [6, с. 15], но и непрерывную эволюцию органической материи в процессе ее усложнения и самоорганизации. При этом утверждалось, что «единый мировой процесс развития — это не просто игра случая, а непрерывное усложнение организации, происходящее в результате взаимодействия объективной необходимости со столь же объективной стохастичностью нашей Вселенной. Реальность такова, что необходимость вовсе не исключает случайность, но определяет потенциальные возможности развития...» [6, с. 19].

Размышления о том, как переход от бинарного кодирования к представлению информации на базе неких 4-х базовых символов мог бы привести к качественно новым свойствам компьютерной обработки информации, привели в итоге к выводу о том, что наиболее естественным реше-

нием является **дополнение традиционных двоичных значений 0 и 1 состояниями их равновероятности и одновременности**. Другими словами, не только НЕ (отрицание), но и логические операции ИЛИ и И должны быть реализованы уже на уровне элементарного кодирования логических состояний. Эти мысли практически сразу же оформились в своеобразную предварительную программу исследований, одним из узловых пунктов которой явилась проверка гипотезы о целесообразности многомерного подхода к представлению не только логического пространства, но и количественных значений на числовой оси. Фактически был поставлен вопрос о разработке нового способа численного кодирования, позволяющего в полной мере использовать преимущества новой логики.

Впоследствии в результате анализа того, что уже было известно о генетических кодах, выявилось настолько много параллелей с идеей нового способа кодирования, что было принято решение отказаться от первоначального его обозначения как «квадрантного» (по аналогии с кодированием двумерных изображений) в пользу определения **«квазигенетическое»**. И уже тогда в качестве основной области применения новых подходов представлялось именно компьютерное моделирование. Сформировалось такое убеждение во многом благодаря также идеям Н.Н. Моисеева, который, в частности, рассматривал модель в первую очередь как своеобразное «упакованное» знание, представляющее собой специфическую форму наиболее эффективного кодирования разнородной информации. При этом, в отличие от обычных способов кодирования, модель за счет специальной систематизации имеющейся информации может нести в себе кроме уже известного и потенциальное знание, выявляемое впоследствии в процессе работы с моделью [6, с. 166].

К сожалению, в силу разных обстоятельств, реально продолжить эти исследования удалось лишь в марте 1992 года, непосредственным поводом для чего явилась подготовка предложений в программу исследований образованной тогда Украинской академии информатизации. Предполагалось также, что данная тема может стать основой диссертационного исследования ассистента кафедры ЭВМ ДонНТУ (тогда еще ДПИ) А.А. Кухтина, принимавшего в то время участие в целом ряде совместных с автором проектов. Основным результатом данного периода стал сделанный в Киеве в ноябре 1992 года доклад «О некоторых возможностях расширения логического базиса информатики» [7], в котором впервые была официально сформулирована и проанализирована концепция двумерного логического пространства в варианте, ставшем впоследствии основой многомерного кодо-логического базиса.

С 1994 года продвижение в развитии идеи многомерного кодо-логического базиса стало более систематичным благодаря началу работы автора над докторской диссертацией в рамках докторантуры под руководством профессора В.А. Святного. Важным этапом при этом стала стажировка в Институте параллельных и распределенных суперЭВМ (Штуттгартский университет, Германия) под руководством профессора А. Ройтера с октября 1994 года по февраль 1995 года. В докладе, сделанном по результатам стажировки на институтском семинаре 31 января 1995 года, были, в частности (в числе прочих вопросов), рассмотрены и перспективы развития «квазигенетической» логики и «квазигенетического» кодирования. При этом было предложено ввести в рассмотрение кроме традиционных логических и арифметических операций специальные «генетические» операции, предполагающие генерацию конкретных «точечных» значений на основе «квазигенетических» ко-

дов, а также различные возможности их модификации или «мутации».

В 1996 году в рассмотрение вводятся понятия «тетра-логика» и «тетракоды» [8], альтернативные определению «квазигенетические», что в дальнейшем позволило обобщить данный подход как на кодо-логический базис, предшествующий бинарному («монологика» и «монокоды»), так и на базисы более высокого порядка, чем четверичные [9]. Особо следует отметить, что в работе [8], кроме двумерного логического пространства, были также рассмотрены и двумерные интерпретации тетракодов. Параллельно в этот период исследовались вопросы возможного использования тетракодов для кодирования изображений [10, 11], а также использование методов стохастической геометрии для анализа и синтеза вычислительных систем и алгоритмов нового поколения [12]. Кроме того, в контексте поиска средств и методов повышения эффективности вычислительного моделирования особое внимание уделялось развитию концепции универсальных моделирующих сред, ориентированных на сетевую (распределенную и/или параллельную) вычислительную инфраструктуру [13–15], что явилось, по сути, первым шагом к концепции расширенного алгоритмического базиса [16].

Важнейшим моментом при этом стало формулирование идеи о **множественности эволюционирующих кодо-логических форм качественного и количественного представления знаний** [9]. При этом утверждалось, что «человеческий интеллект в зависимости от конкретной ситуации и решаемой задачи использует в процессе мышления не одну логическую систему, а некоторое достаточно представительное множество таких систем и связанных с ними количественных представлений. Традиционно используемая двоичная логика и основанные на ней системы

счисления должны рассматриваться при этом в качестве одного из наиболее значимых, но отнюдь не единственных и не достаточных элементов современного интеллектуального инструментария. Другими важными составляющими являются как некоторые более ранние формы мышления и представления количественной информации, так и целый ряд перспективных, которые существуют пока только в зачаточном или не полностью оформившемся виде, но обладают значительным информационным потенциалом» [9, с. 59].

В 1997 году некоторые результаты исследований по многомерному кодо-логическому базису были впервые представлены в англоязычном информационном пространстве. В первую очередь речь идет о докладах на международной конференции по моделированию в Стамбуле [17, 18] и международном конгрессе по научным вычислениям, моделированию и прикладной математике в Берлине [19].

В 1999–2001 гг. удалось существенно продвинуться в исследовании и понимании феномена монокодов и монокодовых вычислительных моделей [20–24]. Это позволило не только понять многие закономерности эволюционного развития методов и средств кодирования количественной информации, но и впервые предложить достаточно обоснованные решения и интерпретации для некоторых достаточно известных трудноразрешимых научных проблем, среди которых в первую очередь можно назвать проблему Фестского диска [21].

В 2002 году возможности практического применения тетралогии была продемонстрирована путем разработки специальной методики модельной и компьютерной поддержки принятия решений в ситуациях когнитивного конфликта [25]. Применение предложенной методики и специальных средств ее реализации для всестороннего анали-

за одной из типичных научных проблем, характеризующихся обилием противоречивой и мало достоверной информации, позволило показать специфику и преимущества новых подходов к решению такого рода задач.

В 2002–2003 годах были также получены существенные результаты в разработке нового поколения распределенных моделирующих сред [26–30], являющихся, по сути, одной из важнейших областей эффективного применения расширенного кодо-логического базиса, который в данном случае должен рассматриваться в неразрывной связи с расширенным алгоритмическим базисом.

В 2000 году в большой обзорной работе «Логика на рубеже тысячелетий» А.С. Карпенко пришел к весьма характерному выводу, суть которого сформулирована следующим образом: «Приходится констатировать, что конец века и конец второго тысячелетия, а именно 1994 г., стал той критической точкой, когда **под невероятным давлением окончательно рухнула конструкция под названием “классическая логика”**» [31].

Свой тезис А.С. Карпенко аргументировал, в частности, следующими фактами: в этом году в Англии и США издается большой сборник работ под названием «Что есть логическая система?» [32]. В том же году практически с таким же названием публикуется философская работа логика с мировым именем Хао Вана [33], которая открывается определениями логики, начиная от Канта и заканчивая Гёделем, и завершается характеристикой логики, данной Л. Витгенштейном в 1921 году: «Логика трактует каждую возможность, и все возможности суть её факты». И в том же году под названием «Что есть истинная элементарная логика?» появляется статья выдающегося логика и философа Яакко Хинтикки [34], в которой развивается новая концепция IF-логики (Independence Friendly — друже-

ственной к независимости). В то же время выходит целый ряд статей, связанных с развитием и применением альтернативных (неклассических) логик. В частности, с 1994 года начала публиковаться целая серия работ Д. Батенса и его учеников, где разработана логика, способная моделировать рассуждения, в ходе которых смысл логических терминов может измениться. Так возникло новое направление исследований, названное «адаптивными логиками» [35].

В итоге А.С. Карпенко детализирует **содержание «революции 90-х» в логике как фактическое исчерпание к этому времени практического потенциала ее простейших форм**: «Необычайный прогресс в хранении и, главное, в обработке информации на основе булевой (классической) логики имеет не только свои естественные физические пределы. Ното-логический универсум не является счётным, а процессы, в нем происходящие, не являются истинностно функциональными. Всё, что можно извлечь из предельного огрубления человеческой логики, впервые представленного работами Шеннона, Шестакова и Накасимы (а это было не что иное, как одна из конкретизаций булевого универсума), как раз извлекает происходящая сейчас компьютерная революция. Но в конечном счете эта конкретизация тупиковая для создания искусственного интеллекта хоть мало-мальски соответствующего человеческому. **Обозначилась явная тенденция к разработке новой логики, которая по своим выразительным средствам намного богаче классической.** Этим объясняется пристальное внимание специалистов к многозначным (бесконечнозначным) и нечеткозначным логикам (которые континуальны) в работах по искусственному интеллекту и других работах» [31].

Но главный смысл наметившихся в 1994 году изменений А.С. Карпенко совершенно справедливо увязал с те-

кущим уровнем развития компьютерных технологий и соответствующим смещением акцентов в развитии логики в направлении получения максимального практического результата: «Тематика абстрактной логики и общетеоретические проблемы обоснования математики... отступают перед новыми тенденциями в развитии логики конца XX века. Логика становится всё более насущной в компьютерных науках, искусственном интеллекте и программировании. Подобное приложение логики порождает большое число новых логических систем, но уже непосредственно нацеленных на их практическое применение» [31].

И действительно, именно к 1994 году в качестве самостоятельного научного направления оформилась новая комплексная дисциплина, получившая в дальнейшем обобщенное название «вычислительный интеллект» (ВИ), и которая, по мнению многих специалистов, в т. ч. основоположника теории нечетких множеств Л. Заде, должна была стать наиболее перспективной альтернативой т. н. «искусственному интеллекту» (ИИ). Одной из основных особенностей ВИ явилась ее ориентация на «мягкие вычисления» («Soft Computing»), концептуально описанные Л. Заде вместе с понятием ВИ именно в 1994 г. [36]. Фактически это означало качественно новый этап и в развитии идей нечеткой логики, получивших к 94-му году как бы «второе дыхание», что выразилось в появлении многочисленных аппаратных и программных реализаций, использующих соответствующие подходы (рис. 1.1).

Концепция «вычислительного интеллекта» в дальнейшем была положена в основу создания нового поколения вычислительной техники, в качестве одного из названий которого часто используется определение Real World Computers (RWC) — «компьютеры реального мира», что

призвано подчеркнуть максимальное приближение новых компьютерных технологий к реально используемым человеком и живой природой средствам и методам кодирования, обработки, преобразования и передачи информации.

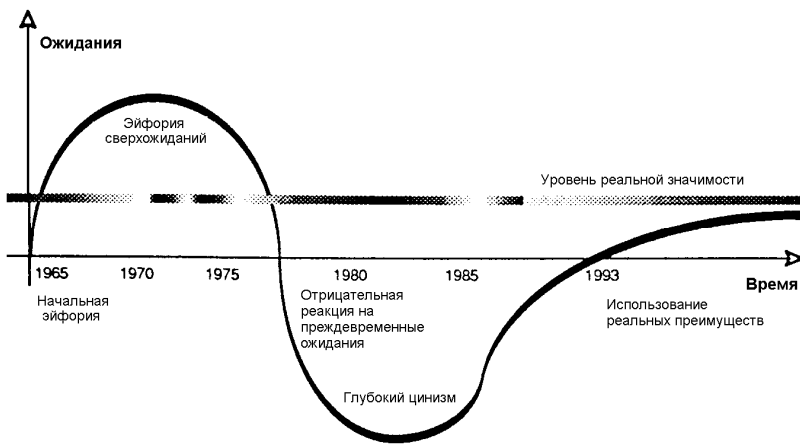


Рисунок 1.1 — Динамика развития и применения идей нечеткой логики [37], достаточно точно отражающая основные тенденции в развитии логических основ компьютеринга во второй половине XX века и нарастание новой волны ожиданий после 1993 года

В работе [9] в связи с этим отмечалось, в частности, что одной из важнейших составляющих «вычислительного интеллекта» является компьютерное моделирование, эффективно использующее весь спектр интеллектуализации вычислительных методов и средств. А в качестве соответствующего направления исследований была предложена концепция расширенного кодо-логического базиса, предназначенная, во-первых, для обобщения и систематизации уже имеющихся в этой области результатов, а во-вторых

— для обеспечения возможности синтеза нового поколения средств и методов моделирования.

Ко всему сказанному следует также добавить, что **1994-й год стал во многом переломным и в развитии самих компьютерных технологий.** Главным при этом явился **глубинный переход от преобладания классической фон-неймановской последовательной парадигмы организации вычислительных процессов к тотальному параллелизму на всех уровнях организации вычислительных структур.** Проявилось это, в частности, в следующем:

Во-первых, в 1994 году фирма Intel заканчивает разработку процессоров нового поколения, получивших обозначение Pentium и реализующих суперскалярную архитектуру, позволяющую выполнять за такт более чем одну инструкцию. В этом же году фирма NexGen на базе производственных площадей IBM начинает выпускать процессор Nx586 с аналогичной архитектурой, который через несколько лет после приобретения NexGen фирмой AMD станет основой для создания основных конкурентов Pentium, а именно процессоров K6, Athlon и последующих [38]. Фактически это означало практически всеобщий переход на суперскалярную параллельную архитектуру начиная с уровня однокристальных процессоров и заканчивая наиболее производительными параллельными системами.

Во-вторых, в июне 1994 года фирмы Intel и HP заключили соглашение о сотрудничестве в области разработки нового поколения 64-разрядных микропроцессоров, на которых приложения для платформы x86 и для Unix-систем работали бы одинаково эффективно. Разрабатываемая архитектура IA-64 предполагала в числе прочего обязательную реализацию параллелизма начиная уже с уровня объ-

единения нескольких инструкций в одной команде (Eclisity Parallel Instruction computing — EPIC). А параллелизм на всех уровнях признавался магистральным направлением дальнейшего развития компьютерных технологий [39].

В-третьих, летом 1994 года в научно-космическом центре NASA Goddard Space Flight Center (GSFC), а точнее в созданном на его основе центре CESDIS (Center of Excellence in Space Data and Information Sciences), был реализован проект Beowulf (www.beowulf.org), название которого превратилось в имя нарицательное для всех последующих кластерных систем такого рода (Beowulf-кластеры). Первоначальный кластер, который и был назван «Beowulf», создавался как вычислительный ресурс проекта Earth Space Sciences Project (ESSP) и состоял из 16-ти узлов на обычных процессорах 486DX4/100MHz с 16 MB памяти и 3-мя сетевыми адаптерами на каждом узле, обеспечивающими связь через 3 обычных Ethernet-кабеля. Основное значение данного проекта заключалось в том, что впервые была наглядно продемонстрирована возможность достижения сверхвысоких показателей производительности на базе кластера, использующего самые обычные компьютерные компоненты массового производства. А это фактически означало начало массового применения параллельных вычислений не только на уровне архитектуры специальных высокопроизводительных систем, но и на сетевом уровне, позволяющем объединять различные компьютерные ресурсы для реализации параллельных вычислений [40].

В-четвертых, с июня 1993 года в университете Мангейма (Mannheim, Германия) совместно с лабораторией Netlib (США) началось ежегодное формирование списка пятиста наиболее производительных вычислительных си-

стем в мире. Уже в 1994 году изменения в этом списке наглядно показали резкое сокращение числа однопроцессорных систем в пользу систем с массовым параллелизмом, а затем и кластерных систем (рис. 1.2).

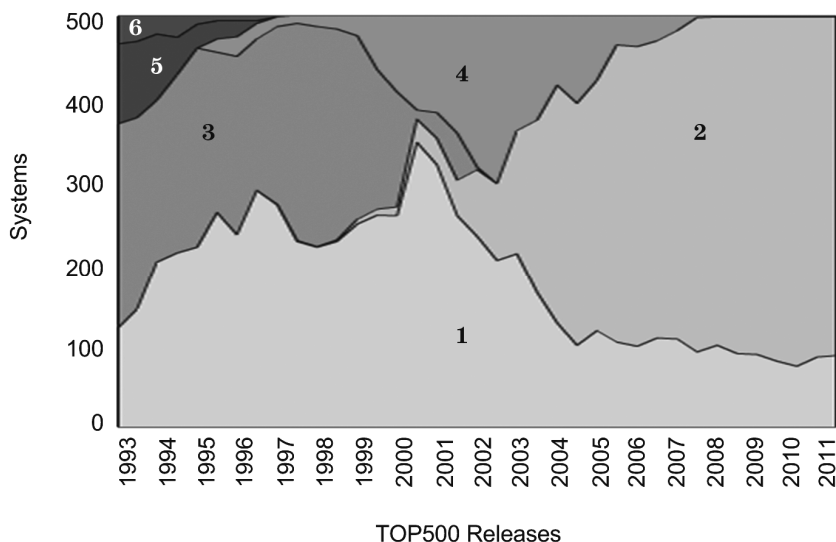


Рисунок 1.2 — Динамика изменения удельного числа систем с различной архитектурой в списке пятиста наиболее производительных вычислительных систем (www.top500.org): **1** – MPP (Massively Parallel Processing) – архитектура вычислительных систем с массовым параллелизмом; **2** – кластерные системы; **3** – SMP (Symmetrical Multiple Processing) – симметричные многопроцессорные системы; **4** – Constellations – суперкомпьютерные платформы на основе открытых компонентов; **5** – однопроцессорные системы; **6** – прочее

В-пятых, к 1994 году Интернет окончательно превратился в глобальную Сеть, стимулом к лавинообразному росту которой стало утверждение к 94-му году стандартов на язык гипертекстовой разметки HTML (версия 2.0) и сопутствующие технологии, на базе чего сформировалась новая информационная инфраструктура, получившая наименование WWW — World Wide Web («Всемирная паутина»).

Знаковым явлением при этом можно считать образование в 1994 году первой компании, ошеломляющий коммерческий успех которой целиком и полностью был основан на существовании Сети. Этой компанией явилась Netscape Communication, основанная разработчиками первого в мире браузера Mosaic и бывшим профессором Стенфордского университета Джимом Кларком, ранее основавшим компанию Silicon Graphics, ставшую одним из лидеров в разработке высокопроизводительных вычислительных систем. Начав с выпуска в 1994 браузера Netscape Navigator, компания в 90-е годы стала одной из наиболее успешных компьютерных корпораций благодаря своему вкладу в развитие клиентских и серверных web-технологий, позволивших в кратчайшие сроки добиться действительно массового использования Сети и превращения ее, по сути, в единую сверхпроизводительную информационно-вычислительную среду [41, с. 106].

В-шестых, в 1994 году закончился многолетний судебный процесс по иску АТТ к университету Беркли, касающийся прав собственности на операционную систему UNIX. Судебное решение о возможности свободного распространения данной системы (наряду с существованием права частной собственности на отдельные ее версии) фактически открыло дорогу повсеместному использованию современных многозадачных операционных систем с мас-

совым параллелизмом процессов, основанном как на эффективном использовании квантования времени, так и на использовании множества процессоров [42, с. 62]. Итогом этого стало и массовое распространение операционных систем семейства Linux (аналогов UNIX, порожденных «Сетью и для Сети»), и переход на использование различных представителей семейства UNIX в абсолютном большинстве высокопроизводительных систем, представленных в списке Top500. Отражением этих процессов стал и ускоренный переход на аналогичные по своей организации операционные системы в фирме Microsoft, что в 1995 году выразилось в выпуске Windows 95, а в последующем — Windows XP, Windows 7 и др., что фактически означало тотальный переход на многозадачность и параллелизм на уровне операционных систем.

В-седьмых, в 1994 году закончилась разработка специального интерфейса передачи сообщений MPI, ставшего первым стандартом передачи сообщений в сетевых и параллельных вычислительных средах [43, с. 111]. Этим, фактически открывалась возможность массовой реализации параллельных приложений в гетерогенных сетевых вычислительных средах, в том числе работающих на базе различных программных платформ.

В-восьмых, с 1994-го фирма SUN начинает адаптировать свою технологию платформенно независимого программирования Oak к среде Интернет, что приводит в 1995 году к появлению технологии Java, не только обеспечившей независимость приложений от аппаратной и программной платформы на базе реализации принципа виртуальной машины, но и создавшей условия для массового перехода от последовательного программирования в среде локального компьютера к параллельному программированию в условиях гетерогенной Сети.

В-девятых, в 1994 году началась третья волна развития клиент-серверных приложений, которая в отличие от первой волны, ориентированной на файловые серверы, и второй, ориентированной на серверы баз данных, впервые в качестве основной среды реализации предполагала распределенные объекты в Сети (рис. 1.3). Первым примером реализации такой инфраструктуры распределенных объектов стала архитектура CORBA [44, с. 53].

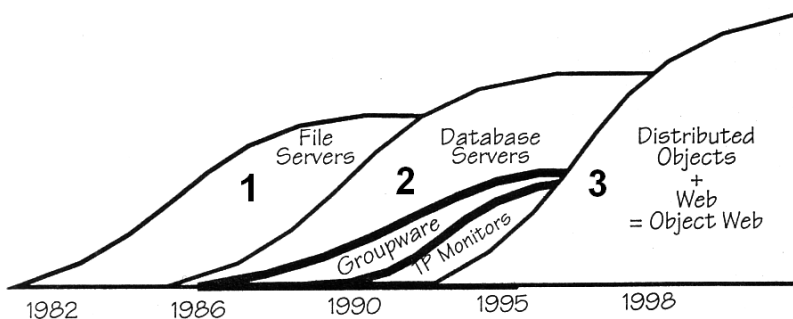


Рисунок 1.3 — Три волны развития технологий «клиент-сервер»: **1** – файловые серверы, **2** – серверы баз данных, **3** – распределенные веб-ориентированные объекты [44, с. 53]

Таким образом, в 90-е годы (кульминацией перемен можно считать 1994 год) произошли качественные изменения как в развитии логических основ, так и в области компьютерных технологий, которые обусловили актуальность соответствующих изменений в кодо-логическом [9] и алгоритмическом [16] базисах современных компьютерных технологий.

Суть данных изменений может быть сведена к утверждению о **переходе от преобладания фиксированной точечной определенности в логических, арифметических**

и алгоритмических основах функционирования компьютерных систем к эволюционирующей множественности и неопределенности.

В 2005 году в работе [1] в порядке развития идей, предложенных в работах [2–30], впервые были рассмотрены и некоторые новые перспективы использования многомерного кодо-логического базиса в вычислительном моделировании и представлении знаний. При этом были проанализированы следующие **пять направлений**, ориентированных на практическое развитие возможностей компьютерного моделирования и представления знаний:

- развитие многомерного логического пространства;
- развитие понятия числа в контексте многомерного логического пространства;
- развитие интерфейсных средств в контексте развития кодо-логического базиса;
- эволюция алгоритмического базиса на основе развития кодо-логических средств;
- переход от господства алгоритмического подхода к преобладанию различных форм комплексного модельного представления.

1.2. Двумерное логическое пространство¹

Необходимым условием при построении логической системы является выполнение главной задачи логики, которая базируется на двух ключевых моментах:

- 1) соблюдение «правильных рассуждений» — отработка механизма перехода к правдивым выходным заключениям исходя из входных предпосылок;

¹ Разделы 1.2–1.7 подготовлены по материалам монографии [45].

2) получение «истинного знания» о предмете размышления для детальной проработки нюансов изучаемых явлений и их соотношениях друг с другом.

Исследование особенностей развития расширенного кодо-логического базиса представляется на сегодня актуальным в связи с тем, что понимание основных закономерностей перехода от монокодового этапа к современному бинарному (дикоковому) этапу позволит более эффективно реализовать переход к следующему (постбинарному, гиперкодовому) этапу в развитии компьютерных технологий. Первые элементы постбинарного компьютеринга начали формироваться уже на протяжении XX века. В частности, в 90-е годы как самостоятельное научное направление оформилась новая комплексная дисциплина, известная в настоящее время под названием «вычислительный интеллект» (см., например, [46–48]).

По мнению основоположника теории нечетких множеств Л. Заде, вычислительный интеллект (ВИ) является наиболее перспективной и значимой альтернативой традиционно понимаемому искусственному интеллекту (ИИ). Одной из важнейших составляющих «вычислительного интеллекта» является компьютерное моделирование, эффективно использующее весь спектр интеллектуализации вычислительных методов и средств.

Предлагаемая **концепция расширенного кодо-логического базиса** актуальна, во-первых, для обобщения и систематизации уже имеющихся в этой области результатов. А, во-вторых, что наиболее существенно, — для обеспечения возможности синтеза новых эффективных методов и средств вычислений. **Основная идея данной концепции базируется на гипотезе о множественности эволюционирующих кодо-логических форм и методов человеческого мышления.**

Другими словами, в основу данного исследования положено представление о том, что человеческий интеллект в зависимости от конкретной ситуации и решаемой задачи использует в процессе мышления не одну логическую систему (относительно простую бинарную логику), а некоторое достаточно представительное множество как более простых, так и более сложных систем (допускающих разного рода неоднозначности) и связанных с ними количественных представлений.

Традиционно используемая в настоящее время двоичная логика и основанные на ней системы счисления должны рассматриваться при этом в качестве одного из наиболее значимых элементов современного интеллектуального инструментария, но отнюдь не единственного и не достаточного. Другими важными составляющими являются как некоторые более ранние формы мышления и представления количественной информации (являющиеся предметом рассмотрения в монографии [45]), так и целый ряд перспективных, которые существуют пока только в зачаточном или не полностью оформившемся виде, но обладают значительным информационным потенциалом.

Традиционные логические системы являются по сути одномерными, так как строятся в пределах оси, соединяющей логические 0 и 1. В простейшем случае классической бинарной логики используются только два противоположных логических значения. В наиболее сложных случаях, при построении непрерывных, в том числе нечетких, логик используется все пространство оси.

Расширенное двумерное логическое пространство (рис. 1.4) может быть порождено базисом, состоящим из ортонормированной системы векторов «Истина» (может обозначаться как T — True или Y — «Yes») и «Ложь» (F —

False или N — «No») с положительной и отрицательной полуосями [7].

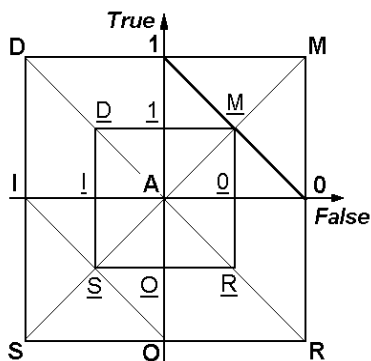


Рисунок 1.4 — Расширенное двумерное логическое пространство

Логические значения при этом могут задаваться либо соответствующими координатами (например, в случае построения непрерывных логик), либо фиксацией характерных точек. В качестве последних должны быть выделены следующие:

1 и **0** — значения «истина» и «ложь» классической логики;

A — абсолютная неопределенность, «непроявленность», неизвестность (обозначение **A** было выбрано исходя из известной критики закона исключения третьего в «Науке логики» Гегеля: «Закон исключения третьего утверждает, что нет ничего такого, что не было бы ни **A**, ни не-**A**. Однако третье есть в самой этой тезе: само **A** есть третье, ибо оно может быть и $+A$ и $-A$ » [49, с. 482], т. е. значение его на момент высказывания утверждения неизвестно, и эта неизвестность и есть фактически тем самым третьим);

M — множественность, многозначность («истина» и «ложь» одновременно);

S — симметричность (инверсная многозначность, отражение **M** относительно точки **A**);

I и **O** — инверсные «истина» и «ложь» (обозначения выбраны по подобию с **1** и **0**, так как предполагается не только симметрия относительно точки **A**, но и относитель-

но оси DR, при этом если 1 и 0 соответствуют положительному выбору некоторых значений из всего возможного множества, то I и O соответствуют отрицательному выбору, т. е. по принципу «все значения, кроме данного»);

D и **R** — мнемонически соответствуют понятиям «дублирование» и «репликация», т.е. формы многозначности, по-разному комбинирующие свойства значений **M** и **S**.

Каждой из перечисленных характерных точек может быть поставлена в соответствие точка, расположенная на половине расстояния между ней и **A**. Значения, соответствующие таким точкам, обозначим аналогичными символами, но с подчеркиванием, что мнемонически может ассоциироваться с дробностью, половинчатостью. Суть данных значений состоит в том, что в них неопределенность принимает вероятностный характер, т. е. равновероятны равноудаленные значения. Например, значение **M** предполагает равновероятность 0 и 1.

Приведенные обозначения существенно отличаются от тех, которые первоначально использовались в работе [8]. Изменение обозначений вызвано в основном двумя причинами: необходимостью улучшения их мнемонических свойств и стремлением к максимальному соответствию используемых обозначений (с учетом возможных их расшифровок) смысловому содержанию.

Смысл предложенных значений может быть проиллюстрирован на одном простом примере. Поведение монеты при бросании принято считать классическим случаем равновероятного события. Однако, если предположить что равновероятность выпадения орла или решки не является обязательной, то можно выделить следующие варианты знания о поведении монеты при бросании: **A** — ничего не известно и возможны любые варианты; **M** — монета ведет себя классически, обеспечивая равновероятность орла и

решки; М — монета при бросании всегда падает на ребро и остается в вертикальном положении, оставляя одновременно открытыми и орла и решку; 1 — при бросании всегда выпадает решка; 0 — всегда орел; $\underline{1}$ — монета доступна для наблюдения после бросания только в половине случаев, при этом каждый раз наблюдается решка; $\underline{0}$ — аналогично предыдущему случаю, но наблюдается орел.

Таким образом, введение новых логических значений позволяет значительно расширить возможности формализованной логической оценки различных нюансов реальных процессов и ситуаций.

В двумерном логическом пространстве могут быть построены различные логические системы, отличающиеся прежде всего количеством используемых логических значений. Возможные логические системы будем обозначать как L_N^K , где K есть количество используемых логических значений или *порядок* логики, а N — порядковый номер логической системы в наборе рассматриваемых логик порядка K . В контексте данного раздела логическую систему будем интерпретировать лишь как множество соответствующих логических значений, т. е. $L_N^K = \{x_1, x_2, \dots, x_K\}$, хотя в общем случае логическая система определяется как множеством логических значений, так и множеством логических функций. С целью терминологического единообразия для наименования логических систем будем использовать слово «логика» в комбинации с греческим корнем, соответствующим значению K . Введем, в частности, в рассмотрение следующие логические системы:

монологика: $L_1^1 = \{1\}$ (в принципе, возможны и другие системы, например, $L_2^1 = \{0\}$, но, с практической точки зрения, достаточно ограничиться L_1^1 , что соответствует рассмотрению и фиксации лишь «положительных» фактов и суждений);

дилогика: $L_1^2 = \{1, 0\}$ — соответствует классической бинарной логике; возможно (но, с практической точки зрения, вряд ли целесообразно) построение и других вариантов дилогики, например

$$L_2^2 = \{1, A\}, L_3^2 = \{A, 0\} \text{ и т.п.};$$

трилогика: $L_1^3 = \{1, 0, A\}$, $L_2^3 = \{1, 0, \underline{M}\}$, $L_3^3 = \{1, 0, M\}$, что покрывает практически все ранее предложенные варианты трилогики;

тетралогика: $L_1^4 = \{1, 0, A, M\}$ и $L_2^4 = \{1, 0, \underline{M}, M\}$, что соответствует ранее предложенным в работе [8] первым вариантам тетралогики; существенный интерес представляют и другие варианты тетралогики, например,

$$L_3^4 = \{1, 0, S, M\}, \text{ а также } L_4^4 = \{1, 0, A, \underline{M}\};$$

$$\textbf{пенталогика: } L_1^5 = \{1, 0, A, \underline{M}, M\},$$

$$L_2^5 = \{1, 0, A, S, M\} \text{ и т.п.};$$

$$\textbf{гексалогика: } L_1^6 = \{1, 0, A, \underline{M}, M, S\} \text{ и др.};$$

$$\textbf{октологика: } L_1^8 = \{1, 0, M, R, O, S, I, D\},$$

$$L_2^8 = \{1, 0, M, S, R, D, A, \underline{M}\} \text{ и др.};$$

$$\textbf{декалогика: } L_1^{10} = \{1, 0, M, R, O, S, I, D, A, \underline{M}\} \text{ и др.};$$

гексадекалогика:

$$L_1^{16} = \{1, 0, M, R, O, S, I, D, \underline{1}, \underline{0}, \underline{M}, \underline{R}, \underline{O}, \underline{S}, \underline{I}, \underline{D}\} \text{ и т.д.}$$

Логики третьего и более высоких порядков, существенно отличающиеся от классической бинарной, целесообразно объединить одним термином, используя для этого обозначение «**гиперлогика**».

Естественно, что перечисленные выше логики отнюдь не исчерпывают всех возможных вариантов, число Q_K которых для каждой из логик K -того порядка определяется количеством K -сочетаний из n различных значений, заданных в логическом пространстве:

$$Q_K = \frac{n!}{K!(n-K)!}. \quad (1.1)$$

Если ограничиться только семью возможными логическими значениями в пределах одного положительного квадранта логического пространства, т. е. принять $n = 7$, то количество всех возможных вариантов диалогии составит $Q_K = 21$. Для трилогии, как и для тетралогии, получим 35 вариантов. Однако, естественно, далеко не все эти варианты равноценны: лишь некоторые из них имеют практическое значение. Поэтому из всего множества вариантов выделены лишь те, которые уже сейчас можно идентифицировать как достаточно продуктивные, в т. ч. — с точки зрения образования на их базе эффективных систем кодирования количественной информации.

Аналогично тому, как бинарная логика является основой двоичной системы счисления, **на базе перечисленных выше логических систем могут быть построены соответствующие системы кодирования количественной информации.** Все вводимые системы кодирования будем рассматривать на машинном уровне, т. е. на уровне двоичной системы счисления, когда кодовый алфавит однозначно совпадает с алфавитом соответствующей логической системы. Системы кодирования при этом могут быть заданы так же, как и соответствующие логические системы. Таким образом, в рассмотрение могут быть введены:

монокоды: $C_1^1 = \{1\}$;

дикокоды: $C_1^2 = \{1, 0\}$ и др.;

трикоды: $C_1^3 = \{1, 0, A\}$, $C_2^3 = \{1, 0, \underline{M}\}$,

$C_3^3 = \{1, 0, M\}$ и др.;

тетракоды: $C_1^4 = \{1, 0, A, M\}$ и $C_2^4 = \{1, 0, \underline{M}, M\}$,

$C_3^4 = \{1, 0, S, M\}$, $C_4^4 = \{1, 0, A, \underline{M}\}$ и др.;

пентакоды: $C_1^5 = \{1, 0, A, \underline{M}, M\}$,

$C_2^5 = \{1, 0, A, S, M\}$ и т. п.;

гексакоды: $C_1^6 = \{1, 0, A, \underline{M}, M, S\}$ и др.;

октокоды: $C_1^8 = \{1, 0, M, R, O, S, I, D\}$,

$C_2^8 = \{1, 0, M, S, R, D, A, \underline{M}\}$ и др.;

декакоды: $C_1^{10} = \{1, 0, M, R, O, S, I, D, A, \underline{M}\}$ и др.;

гексадекакоды:

$C_1^{16} = \{1, 0, M, R, O, S, I, D, \underline{L}, \underline{O}, \underline{M}, \underline{R}, \underline{O}, \underline{S}, \underline{I}, \underline{D}\}$ и т. д.

Аналогично тому, как это было сделано для логических систем, для всех систем кодирования третьего и более высоких порядков может быть введен обобщающий термин «**гиперкоды**».

В совокупности перечисленные логические и кодовые системы образуют **расширенный (обобщенный) кодо-логический базис**.

Следует отметить, что возможности расширения кодо-логического базиса не исчерпываются двумерным логическим пространством, в котором, в частности, не находят своего отражения в явном виде идеи нечеткой логики [36, 48–52]. Этот недостаток может быть устранен **расширением логического пространства до трехмерного путем введения вектора функций принадлежности в качестве третьей составляющей ортонормированного базиса**. При этом двумерному пространству классических функций принадлежности нечеткой логики будет соответствовать плоскость, ортогональная осям «ложь» и «истина» и пересекающая их в точках логических значений 0 и 1.

1.3. Трехмерное логическое пространство

Рассмотренное в работах [7–9] двумерное логическое пространство может быть продуктивно расширено до трехмерного путем введения третьего измерения, соответствующего возможной недостоверности и/или «вариабельности» (т. е. возможной изменчивости) логических значений двумерного пространства (рис. 1.5).

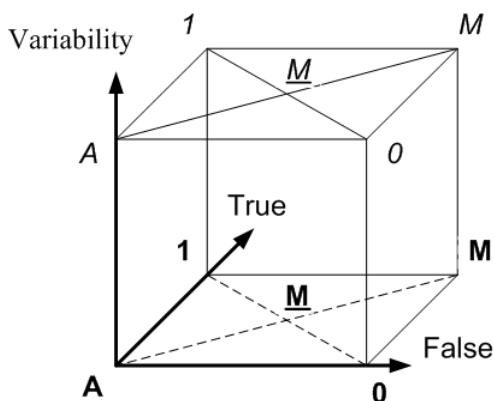


Рисунок 1.5 — Трехмерное логическое пространство с ортогональными осями «истина», «ложь» и «вариабельность»

Традиционные логические системы являются по сути одномерными, так как строятся в пределах оси, соединяющей логические 0 и 1. В простейшем случае классической бинарной логики используются только два противоположных логических значения. В наиболее сложных случаях, при построении непрерывных, в том числе нечетких, логик используется все пространство оси 0–1.

Трехмерное логическое пространство может быть порождено базисом, состоящим из ортонормированной системы векторов, определяющих двумерное логическое

пространство (см. раздел 1.2), а также вектора «Вариабельность» (V — «Variability»), определяющего третье измерение логического пространства. Логические значения при этом могут задаваться либо соответствующими координатами (например, в случае построения непрерывных логик), либо фиксацией характерных точек. В качестве последних прежде всего должны быть выделены следующие: **1** и **0** — значения «истина» и «ложь» классической логики; **A** — абсолютная неопределенность, «непроявленность», неизвестность; **M** — фиксированная множественность, многозначность («истина» и «ложь» одновременно); **M** — фиксированная равновероятность значений «истина» и «ложь», которая может рассматриваться в качестве технически реализуемого аналога состояния **A**.

В дополнение к этим значениям, впервые рассмотренным в работе [7], в трехмерном логическом пространстве дополнительно вводятся соответствующие значения *l*, *0*, *A*, *M* и *M*, модифицируемые при определенных условиях. Интерпретация такой модифицируемости может быть различная: например, адаптивность логики в смысле Д. Батенса [35] или допустимость ценностных изменений в логических значениях в контексте возможной смены или дополнения парадигмы «знания + аргументация» более гибкой парадигмой «знания + оправдания» [56, с.154].

В качестве одного из наиболее перспективных вариантов реализации логических систем в трехмерном базисе можно рассматривать **октологику**, которая в соответствии с введенной в работе [6] системой обозначений может быть описана как следующий кортеж значений: $L_v^8 = \{0, 1, \underline{M}, M, 0, l, \underline{M}, M\}$.

При этом к модифицируемым логическим значениям, кроме простейших логических операций сведения всех значений только к одному из них (таблица 1.1), могут быть

также применимы специальные одноместные модифицирующие операции (таблица 1.2).

Аналогично одноместным операциям могут быть заданы и двуместные операции, которые будут одинаковыми как для модифицируемых, так и для фиксированных значений (таблица 1.3).

Таблица 1.1 — Специальные одноместные модифицирующие операции

Исходные значения	Абсолютная минимизация	Абсолютная максимизация	Абсолютное распараллеливание	Абсолютная стохастизация
0	0	1	M	\underline{M}
1	0	1	M	\underline{M}
M	0	1	M	\underline{M}
\underline{M}	0	1	M	\underline{M}

Таблица 1.2 — Одноместные модифицирующие операции

Исходные значения	Классическая инверсия	Операции с множественностью		
		Инверсия	Максимизация	Минимизация
0	1	0	0	0
1	0	1	1	1
M	M	\underline{M}	M	\underline{M}
\underline{M}	\underline{M}	M	M	\underline{M}

Таблица 1.3 — Двуместные операции октологики

Операнд 1	Операнд 2	И (min)	ИЛИ (max)
0	0	0	0
0	1	0	1
0	М	0	М
0	<u>М</u>	0	<u>М</u>
1	1	1	1
1	М	1	М
1	<u>М</u>	<u>М</u>	1
М	М	М	М
М	<u>М</u>	<u>М</u>	М
<u>М</u>	<u>М</u>	<u>М</u>	<u>М</u>

В качестве простейших частных случаев использования вариабельности могут рассматриваться тетралогика $L_V^4 = \{0, 1, \emptyset, I\}$ и дилогика $L_V^2 = \{\emptyset, I\}$, позволяющие реализовать свойство адаптивности в рамках подходов, характерных для традиционной бинарной логики.

Следует, однако, отметить, что определение «квазигенетическая» в наибольшей степени применимо именно к рассмотренному выше варианту октологики, т. к. именно на ее базе могут быть наиболее полно промоделированы свойства генетического кодирования, в том числе необходимые в специфических ситуациях и допустимые в определенном диапазоне мутации.

1.4. Монологика и монокоды

Весьма существенным представляется введение в рассмотрение понятий монологики и монокодов, что позволяет реализовать новый методический подход к систематизации и вовлечению в круг интересов компьютерных наук чрезвычайно важного массива интеллектуальных достижений человеческой культуры, ранее практически выпавших из рассмотрения в современной информатике. Более того, анализ закономерностей и особенностей перехода от монологики к диалогике и от монокодов к бинарным кодам позволит эффективно использовать этот опыт при переходе к гиперлогике и гиперкодам.

С уверенностью можно констатировать, что **монологика явилась исторически первым логическим построением, освоенным человеческим мышлением.** Этот факт однозначно отражен в особенностях построения так называемого праязыка, наиболее полно реконструированного сегодня на материалах индоевропейской языковой семьи (см., например, [57]). Отмечаются, в частности, следующие реконструированные особенности генетически ранних языковых форм [58]:

Во-первых, господство простых единичных суждений, выражающих и закрепляющих знания о тех предметах, которые в результате практических потребностей рассматривались как предметы отдельные.

Во-вторых, как следствие отсутствие способов синтаксического подчинения и дифференцированной системы союзных отношений, т. е. единственным способом выражения грамматических связей между словами в предложениях праязыка было примыкание, поэтому индоевропейское предложение было способным только к простому соположению слов, выражающих понятия, и, следовательно,

речь могла строиться только в виде упрощенного монолога, состоящего из последовательности простых суждений (даже в классической латыни нередки случаи, когда сложное суждение еще не получает должного языкового выражения).

В-третьих, практически единственным видом умозаключений был вывод от единичного к единичному — явление, ярко выраженное сегодня в мышлении детей дошкольного возраста.

Сложноподчиненные предложения с подчинительными союзами (так, как, потому что, ибо и т. п.) впервые появляются только после распада индоевропейской языковой общности и образования современных языковых семей, приспособленных для тонкой передачи не только высказываний повышенной сложности, но и их зависимости друг от друга. А это явление датируется примерно 5–3 тысячелетиями до н. э.

В качестве примера можно обратиться к «Ригведе» (РВ) — древнейшему из сохранившихся текстов достаточно большого объема [59]. Та часть РВ, которая на сегодня уверенно идентифицирована как наиболее древняя, уходящая корнями устной традиции в эпоху индоевропейской общности, целиком и полностью выдержана в упрощенной монологической форме и состоит из логически слабо связанных последовательностей констатирующих суждений, призывов, заклинаний и риторических вопросов. Из более чем тысячи гимнов РВ не более двух десятков с большей или меньшей достоверностью можно назвать диалогами, причем лишь в их зачаточной неразвитой форме. Даже в наиболее поздних частях РВ для диалогов характерны такие явления, как отсутствие достаточно ясной связи между репликами, впечатление об отсутствии каких-

то звеньев в развитии событий, частая невозможность уверенной идентификации авторов реплик [59, с. 492].

В контексте нашего изложения важно также отметить характерную для РВ неразвитость логического отрицания и вытекающей из него системы логических противопоставлений. Заметно это прежде всего в ярко выраженной и довольно хаотичной многозначности лексики, доходящей до того, что некоторые слова могут объединять в себе прямо противоположные значения, как, например, *ari* — это может означать в зависимости от контекста и «друг» и «враг», а *mau* — это и «сверхъестественная мудрость» и «обман». При этом конкретное значение весьма существенно зависит от контекста и не всегда может быть определено с достаточной степенью уверенности.

Характерным для РВ является также отсутствие каких-либо намеков на возможность логического получения знаний, что обусловлено невозможностью построения на базе монологии сколь-нибудь развитой системы логических операций. Авторы Ригведы обозначаются словом «риши», которое, кроме значения «мудрец», имеет также значение «поэт»: «Поэт в обществе ариев был носителем той мудрости, которая в моменты озарения открывается богами отдельным избранным лицам. Поэт молит богов о том, чтобы ему были дарованы эти мгновения просветления, когда перед ним раскрывается божественная истина, скрытая от обычных людских взоров. Мудрость — это раскрывающаяся на мгновение картина. Способ ее постижения — видение. Видит поэт внутренним взором, интуицией, внезапная вспышка которой озаряет для него божественную картину истины... В смене этих откровений заключалось познание мира, кодируемое словом *dhi* — «мысль, представление, взгляд, понятие, интуиция, познание, разум» [59, с. 458]. Древний индоевропейский корень *dhi* достаточно хорошо

просматривается в современном слове «**вдохновение**», означающем такие внелогические понятия как «воодушевление», «наитие» и «творческий подъем». Еще более явно такие связи прослеживаются в современном украинском языке, где, например, с внелогическим оформлением и синтезом знаний связано слово «**натхнення**».

Из логических операций с монологикой уверенно может быть связана лишь **импликация** (лат. *implicatio* — сплетение), соответствующая в современном обыденном языке связке «если..., то ...». При отождествлении импликации с логическим следованием в форме $x \rightarrow y$ содержание ее можно свести к следующим утверждениям: «если высказывание x истинно, то оно следует из любого высказывания y », и «если x ложно, то из него следует любое y ». В современную формальную логику данные утверждения вписываются не без проблем, в связи с чем возникло понятие «**парадокс материальной импликации**» [60, с. 218]. Одной из причин такой ситуации является, по-видимому, реликтовость данной операции, унаследованной бинарной логикой из монологики, где она еще до оформления ее в языковую конструкцию являлась основой построения простейших суждений «от единичного к единичному».

В алгоритмическом плане монологике соответствует простая последовательность операторных вершин, выполнение которой реализуется в соответствии с правилом «если выполнен текущий оператор, то переходи к выполнению следующего». Большинство современных инструкций по подготовке к эксплуатации технических устройств, например, является именно такими простейшими алгоритмами.

Монокоды несколько упрощенно можно определить как коды без ноля. Другими характерными признаками монокодов являются их непозиционность и представление

значений соответствующим количеством определенных предметов или знаков. Другими словами, **в случае монокодов некоторое количество чего-либо прямо репрезентуется соответствующим количеством счетных знаков или предметов.** Простейшими примерами монокодов являются нарастающие ряды зарубок или других однородных меток, которые не только сегодня служат простейшим средством для последовательного подсчета каких-либо событий, но и, по многочисленным археологическим свидетельствам, являлись на ранних этапах развития цивилизации единственным средством фиксации числовых значений (см., например, [61]).

Многие из первичных форм и приложений монокода сохранились в употреблении и сегодня. Наиболее типичный пример: точечные обозначения на игральные кости, использование которых в обиходе древнейших носителей индоевропейского праязыка подтверждается не только древнеиндийскими тестами, но и целым рядом археологических находок (см., например, [62]). Другим наглядным примером являются счеты, ведущих свою родословную от древнейшего счетного прибора — абака.

Несмотря на кажущуюся примитивность, уже простейшие формы монокода могли использоваться для весьма сложных вычислений и, что особенно важно, построенные довольно развитых средств вычислительного моделирования. Наиболее ярким (и пока фактически уникальным) примером такого рода является хранящаяся в Эрмитаже костяная пластина (рис. 1.6), возраст которой, по разным оценкам, может составлять от 15-ти до 25-ти тысяч лет. Детальная реконструкция и расшифровка точечных узоров на пластине позволяет достаточно уверенно идентифицировать ее как тщательно продуманный вычислительный прибор, позволяющий относительно просто отслеживать и

прогнозировать основные календарные и астрономические события, а также изменения в видимом положении небесных тел [45, 58].

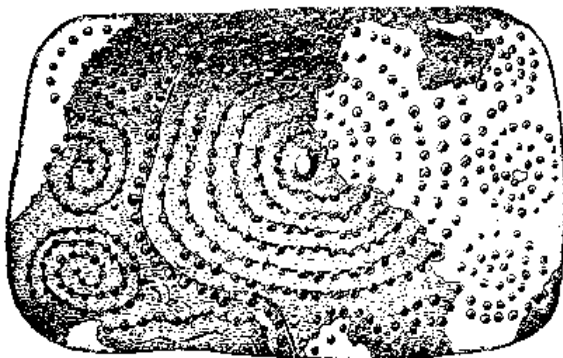


Рисунок 1.6 — Древняя пластина с точечными узорами монокода, интерпретируемыми как довольно сложная вычислительная модель [45]

Одной из основных проблем при работе с монокодами является представление больших чисел. Поэтому развитие монокодов шло в основном по пути введения специальных знаков для определенных количеств, что должно было облегчать представление относительно больших значений. Так, например, уже на ранних стадиях (3200 г. до н. э.) развития цивилизации Древнего Египта существовали отдельные обозначения для чисел до девяти (в виде вертикальных черточек), десятков (короткий изогнутый отрезок веревки), сотен (спирально свернутый отрезок веревки, по форме напоминающий узоры на упомянутой выше пластине), а также — тысяч, десятков, сотен тысяч и миллиона [63]. Известны также достижения пифагорейской школы в области так называемых фигурных чисел. Одной из

наиболее развитых систем монокода являлись греческая и кириллическая «алфавитные цифири», позволявшие представлять цифровые значения символами алфавита со специальными обозначениями (рис. 1.7). Такая форма записи чисел была общеупотребительной в России вплоть до XVIII века. Из сохранившихся сегодня в употреблении развитых форм монокода необходимо отметить в первую очередь римскую систему нумерации.

А 1	Т 10	Р 100	*А 1000	АІ 11
В 2	К 20	Е 200	*В 2000	КІ 12
Г 3	Л 30	Т 300	*Г 3000	ПІ 13
Д 4	М 40	У 400	*Д 4000	МІ 14
Е 5	Н 50	Ф 500	*Е 5000	ЕІ 15
Ѕ 6	Ѕ 60	Х 600	(Ѕ) 60000	ЅІ 16
З 7	О 70	У 700	(З) 70000	КА 21
Н 8	П 80	Ѡ 800	(Н) 800000	НЕ 55
Д 9	Ч 90	Ц 900	(Д) 900000	РН 108

Рисунок 1.7 — Одна из наиболее развитых форм монокода: числовые значения символов алфавита (в данном случае кириллицы)

Заметим, что ранние формы монокода были максимально удобны для последовательного инкрементного (или декрементного) счета. Поздние формы монокода были относительно хорошо приспособлены для компактного представления натуральных чисел вплоть до миллионов. Однако уже простое сравнение чисел, представленных монокодами, а тем более — выполнение с ними основных арифметических операций, являлось довольно сложной задачей. В связи с этим практически повсеместно для вы-

числительных операций использовались специальные средства типа абака, существенно облегчающие манипуляции с монокодами.

Абак оставался в качестве основного вычислительного средства вплоть до широкого распространения десятичной «арабской» системы, включившей нуль в качестве одной из равноправных цифр.

1.5. Диалогика и дикоды

Важнейшей предпосылкой перехода от монологии к бинарной логике (**диалогике**) явилась необходимость в четком оформлении понятия отрицания и соответствующей разработке системы противопоставлений. В монологии отрицание как таковое еще четко не оформлено и может пониматься в основном как «непроявленность», неясность, недоступность для понимания. Так, например, в «Гимне о сотворении мира» в РВ отрицание интенсивно используется для описания непознаваемой ситуации «до сотворения»: «Не было не-сущего, и не было сущего тогда... Не было ни смерти, ни бессмертия тогда... Не было ни признака дня [или] ночи...» [64, с. 44], что в некоторых вариантах перевода на современный язык может звучать вполне абсурдно: «Было не было и Не-было тоже...» [65, с. 125]). Для античной же науки характерен как раз повышенный интерес к четкой проработке проблемы отрицания.

Для современного исследователя чрезмерное увлечение пифагорейской школы учением о противоположностях представляется иногда весьма наивным, оправданным лишь для первых шагов познания. Однако явление «гипериспользования», когда применимость новшества испытывается везде, где это возможно, наблюдается повсеместно и в современной науке в процессе любых действительно

многообещающих нововведений. Поэтому правильнее считать «науку о противоположностях» не первыми шагами научного познания как такового, начавшегося значительно раньше, в «эпоху монологики», а начальным этапом «эпохи дилогики».

Связанной с этим «болезнью роста» можно считать и гипертрофированное использование древнегреческими философами диалоговой формы научных трактатов, несколько раздражающей своей навязчивостью современного читателя. Любопытно отметить, что последний ярко выраженный всплеск повышенного интереса к диалогу в научных текстах имел место в XV веке, когда в Европе окончательно утверждалась в качестве основной системы записи чисел индийско-арабская система с нулем [66, с. 111]. А именно наличие и регулярное использование специального знака для нуля является наиболее характерным признаком бинарных кодов (диконов), построенных на основе дилогики. Основной современной формой дилогики стала бинарная логика, оперирующая значениями «истина» и «ложь». Однако определенный интерес могут представлять и другие её варианты, оперирующие, например, такими парами логических значений, как «истина» и «неизвестность» или «истина» и «многозначность».

Дикоды могут быть определены как «позиционные системы с нулем», основанные на использовании дилогики. Наиболее простой и очевидной формой дикода является классический двоичный (бинарный) код, к которому могут быть сведены и все другие из используемых сегодня в вычислительной технике систем счисления: троичная, восьмеричная, десятичная, шестнадцатиричная и пр.

«Изобретение нуля» по праву считается одним из важнейших шагов на пути к современной математике. Достаточно отметить тот факт, что в математический язык поня-

тие «алгоритм» пришло вместе с нулем [66, с.93]: одним из первых источников, принесших вместе с десятичной позиционной системой понятие нуля в Западную Европу, стал латинский перевод в XII веке книги известного арабского мыслителя IX века аль-Хорезми, которая в переводе называлась «Об индийском числе, сочинение Алгоризми». Вынесенное в заглавие латинизированное имя автора как раз и стало прообразом слова «алгоритм».

Практически одновременно, от названия другой книги аль-Хорезми, сформировалось и понятие «алгебра», что отнюдь не случайно. Ибо с введением нуля, а фактически, в нашей интерпретации, при переходе от монокодов к дикодам, появилась реальная возможность достаточно простой алгоритмизации основных арифметических действий, что послужило стимулом и основой развития алгебраического метода в математике. Процесс перехода от монокодов к дикодам в Европе растянулся на несколько столетий и проходил в острой борьбе, как тогда считали, двух наук: одной — математики на абаке, другой — математики без абак, на бумаге. Эта борьба известна в истории математики как борьба абакистов и алгоритмиков [67, с. 50].

Утверждение дикодов в качестве основной формы представления числовых значений открыло дорогу не только интенсивной алгоритмизации и алгебраизации математики, но и определило переход от абак к механическим арифмометрам, а также — к весьма своеобразным механическим устройствам вычислительного моделирования, работающий по принципу часового механизма (см., например, [68]).

Главный же успех дикодов был обеспечен электронными вычислительными машинами, в которых они оказались наиболее эффективными именно в своей простейшей двоичной форме. Однако с переходом к так называемым

ненеймановским архитектурам, которые в настоящее время представлены в первую очередь массивно параллельными и сетевыми вычислительными структурами, начинает все более остро ощущаться ограниченность бинарных кодов как практически единственных методов кодирования числовых значений в ЭВМ.

Эта ограниченность выражается прежде всего в следующем:

- интенсивное распространение новых методических, вычислительных и алгоритмических подходов, например, т. н. мягких вычислений, генетических алгоритмов и т. п., требует соответствующей поддержки их как на аппаратном уровне, так и на уровне форматов данных и форм кодирования числовой информации; при этом желательно обеспечивать это не специфическими для каждого из подходов средствами, а максимально универсальными, что в рамках ориентации исключительно на дикоды представляется крайне затруднительным;
- в современных массивнопараллельных системах удельный вес межпроцессорного информационного обмена соизмерим, а порой и превосходит удельный вес чисто вычислительных операций (см., например, [69]), что требует максимального повышения компактности кодирования информации для внешнего обмена, в т.ч. даже за счет возможного повышения трудоемкости ее внутрипроцессорной обработки — кардинально же изменить здесь ситуацию на базе дикодов не представляется возможным;
- расширение применения различных форм вычислительного моделирования, в том числе с использованием массивно параллельных структур, требует эффек-

тивного численного описания различных сложных структур реального мира, в т. ч. характеризующихся вариативностью параметров, а также той или иной степенью регулярности структурной организации, что также практически не поддерживается традиционными дикодами.

1.6. Трилогика и трикоды

Простейшим строгим обобщением классической логики явилась трилогика — троичная логика, в которой к двум традиционным значениям «истина» и «ложь» добавляется в какой-либо форме значение неопределенности. При этом особый интерес представляет тот факт, что и в данном случае развитие логических систем было тесно связано с дальнейшей разработкой понятия отрицания.

Впервые формальная трехзначная пропозициональная логика была построена Лукасевичем в 1920 году [70]. В ней «истина» обозначена как 1, «ложь» — 0, «нейтрально» («возможно») — $1/2$. В качестве основных функций рассматриваются отрицание и импликация, а производными от них считаются конъюнкция и дизъюнкция, определяемые соответственно как минимум и максимум значений аргументов. Характерной особенностью логики Лукасевича является нейтральность операции отрицания в отношении значения «возможно».

Обобщенная n -значная система Поста (1921 год, [56]) предполагала уже введение двух видов отрицания: циклического N^1 ($[N^1x] = [x] + 1$ при $[x] < n$ и $[N^1n] = 1$) и симметричного N^2 ($[N^2x] = n - [x] + 1$). При $n = 2$ эти отрицания совпадают, но уже при $n = 3$ они по-разному оперируют с логическими значениями. Существенным при этом является

ся единообразие влияния каждого из видов отрицания на весь набор логических значений.

Следующим существенным шагом в развитии трилогики является трехзначная система советского логика Бочвара (1938 г., [71]), построенная на разделении высказываний на имеющие смысл (т. е. истинные или ложные) и бессмысленные. При этом «истина» обозначается как R , «ложь» — F , «бессмысленность» — S . Для данного набора значений таблично задаются уже три следующих вида отрицаний: $\neg a$ — внешнее отрицание; $\sim a$ — внутреннее отрицание; \bar{a} — внутреннее отрицание внешнего утверждения. Следует отметить, что данное построение, пожалуй, впервые для разработок в области неклассических логик оказалось весьма полезным практически для разрешения ряда парадоксов классической математической логики методом формального доказательства бессмысленности определенных высказываний. В частности, с помощью своей системы Бочвар смог разрешить парадокс Рассела о множестве всех нормальных множеств, доказав несуществование такого множества.

Среди последующих работ в области трилогики может быть выделена трехзначная система Рейхенбаха (1946 год, [72]), которая заслуживает внимания хотя бы потому, что в ней многозначная логика впервые вводится исходя не из исключительно внутренних потребностей математики или логики как таковой, а ввиду потребностей несколько менее абстрактной специальной науки. Рейхенбах построил свою трехзначную систему для описания явлений квантовой механики. Основным положением системы является утверждение о том, что говорить об истинности или ложности высказываний правомерно лишь тогда, когда возможно осуществить их проверку. Если же нельзя ни подтвердить истинность высказывания («верифицировать»), ни опро-

вергнуть его с помощью проверки, то такое высказывание должно оцениваться третьим значением — неопределенно. Характерным примером являются высказывания о ненаблюдаемых объектах в микромире. Рейхенбах ввел 3 вида отрицаний: циклическое, диаметральное и полное, которые различаются в основном их действием в отношении «неопределенности». В современных условиях наибольший и постоянно возрастающий интерес к различным вариантам трилогики проявляют специалисты в области разработки интеллектуальных систем. В работе [73], например, вводится понятие информационного ноля Θ и троичная шкала $\text{Bit} = \{0, 1, \Theta\}$, которая выражает абстрактную семантику номинативной (да, нет, не знаю) и логической (истина, ложь, истинность неизвестна) шкал. Информационный ноль в данном контексте определяет формализованную внутреннюю неопределенность переменной $x = \Theta$ в двоичной шкале. Операции классической логики переносятся при этом в трилогику следующим образом: если вариации неопределенных входных значений изменяют результат операции, то ей присваивается неопределенное значение Θ , в противном случае неопределенности «поглощаются» и троичная функция имеет вполне определенное значение.

Обобщая сложившиеся на сегодня наиболее распространенные подходы к реализации трилогики в компьютерных науках, ее операции можно свести в таблицу 1.4.

В приведенной таблице неопределенность обозначена в своем крайнем выражении как «неизвестность» A , $\neg a$ есть отрицание утверждения « a » (понимаемое в его простейшем виде, характерном для бинарной логики), $a + b$ и $a \cdot b$ есть соответственно логическая сумма (дизъюнкция) и произведение (конъюнкция), $a \rightarrow b$ — импликация (следование), $a \leftrightarrow b$ — эквиваленция, $a \oplus b$ — сумма по модулю

2, $a \downarrow b$ — стрелка Пирса (отрицание дизъюнкции), $a \mid b$ — штрих Шеффера (отрицание конъюнкции). Данная таблица будет выглядеть аналогично при замене А на другие виды неопределенности, например, \underline{M} или \overline{M} .

Таблица 1.4 — Реализация операций трилогики

$a \ b$	$\neg a$	$a + b$	$a \cdot b$	$a \rightarrow b$	$a \leftrightarrow b$	$a \oplus b$	$a \downarrow b$	$a \mid b$
0 0	1	0	0	1	1	0	1	1
0 А	1	А	0	1	А	А	А	1
0 1	1	1	0	1	0	1	0	1
А 0	А	А	0	А	А	А	А	1
А А	А	А	А	А	А	А	А	А
А 1	А	1	А	1	А	А	0	А
1 0	0	1	0	0	0	1	0	1
1 А	0	1	А	А	А	А	0	А
1 1	0	1	1	1	1	0	0	0

Трикоды (троичные коды, в которых каждый разряд представлен **тритами**) определяют способ представления данных в виде комбинации трех знаков. Аналогично дикодам трикоды также могут быть определены как «позиционные системы с нулем», основанные на использовании трилогики. По сути, троичные коды не являются троичной системой счисления, но используются в ней как основа, причем двоичный код может использоваться для кодирования троичных чисел (как, впрочем, и в системах счисления с любым основанием).

Идея представления троичной логики с помощью двоичного кода была успешно воплощена в малой ЭВМ «Сетунь», разработанной в 1959 году в вычислительном центре Московского государственного университета под руководством Н.П. Брусенцова. Для не имеющей аналогов в истории вычислительной техники машины «Сетунь» была разработана троичная ферритодиодная ячейка [74], работающая в двухбитном троичном коде, т. е. один трит записывался в два двоичных разряда 00, 01 и 10 (состояние 11 не использовалось).

1.7. Тетралогики и тетракоды

Если не считать n -значной системы Поста, которая, с логической точки зрения, при $n > 3$ представляется достаточно тривиальной и не вносящей ничего принципиально нового по сравнению с трилогикой, то можно считать, что первый шаг в области тетралогики был сделан также Лукасевичем (1957 г., [75]). Суть предложенного подхода заключалась в том, что неопределенность разделялась на два логических значения: «вероятность» (как приближение к «истине») и «невероятность» (как приближение к «лжи»). Причем данный шаг интерпретировался как явное выражение тех идей, которые в зародышевом виде содержались уже в аристотелевой логике. Следует признать, однако, что, как и система Поста, данная логика существенно не продвинула развитие логических идей по сравнению с трилогикой, так как все построения по-прежнему оставались в одномерном пространстве — в пределах оси «0–1».

В то же время академик Б.В. Раушенбах, известный специалист в области механики (более 20 лет заведовал кафедрой механики Московского физико-технического института), к концу 90-х годов XX века убедительно показал,

что математика к настоящему времени по существу уже оперирует математическими объектами, обладающими всеми логическими свойствами троичности, и в качестве примера одного из таких объектов привел вектор с тремя ортогональными составляющими [76].

Другими попытками преодоления одномерности логики можно считать древнекитайскую концепцию «гармонии противоположностей» (через взаимопроникновение и взаимодополнение противоположных начал Инь и Ян) и диалектическую концепцию «единства и борьбы противоположностей».

В вычислительной технике возможность и необходимость выхода за пределы одномерного логического пространства впервые была достаточно четко декларирована в 1976 году американским математиком Н. Белнапом в работах «Как нужно рассуждать компьютеру» и «Об одной полезной четырехзначной логике» [77], в которых была предложена четырехзначная логика со следующими значениями истинности: Т — «только Истина» (True); F — «только Ложь» (False); N — «ни Истины, ни Лжи» (None); В — «и Истина и Ложь» (Both). Необходимость четырехзначной логики обосновывалась тем, что входные данные могут поступать в компьютер из различных независимых источников, что может привести к достаточно типичной ситуации: появлению противоречивой информации. Предложенная логика рассматривалась как средство практического преодоления такой ситуации.

В 1996 году независимо и практически одновременно вводится специальное понятие «тетралогика» для обозначения четырехзначной логики в работах [8] и [73]. В частности, в работе [73] введение данного понятия аргументировалось следующим образом: «Простейший учет внешней неопределенности состоит в переходе к тетралогике с фа-

тальным (квадратным) нулем, который метит абсурдные ситуации внешней неопределенности фактических и априорных знаний в шкале (0, 1, Θ , Π), и наличие на значимом входе любой функции квадратного нуля порождает на выходе функции знак Π . При отсутствии в процессе фатальных ошибок и внутренних неопределенностей трилогика и тетралогика воспроизводят классическую логику».

В работе [8] и в данной главе **тетралогика трактуется существенно шире.**

Во-первых, предполагается возможность построения различных вариантов тетралогики, включающих в качестве логических значений, кроме классических 1 («истина») и 0 («ложь»), также различные парные комбинации следующих значений: A («неопределенность», «непроявленность» — аналогично N в логике Белнапа), M («множественность» — аналогична B в логике Белнапа), I («возможность», «равновероятность» — аналогична значению «возможно» в трилогике Лукасевича), S («симметричность»), R («возможность симметричности») и другие, представленные на рис. 1.8.

Во-вторых, четко декларируется *три вида* логических значений, отличных от классических: *первый* вид

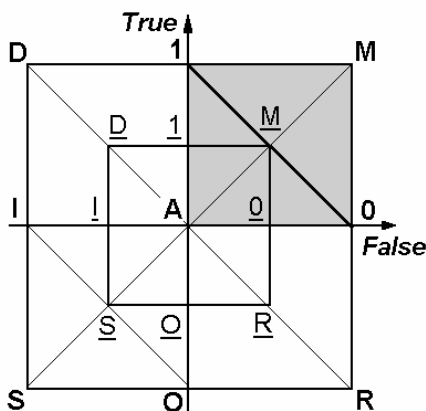


Рисунок 1.8 — Расширенное двумерное логическое пространство: серым цветом выделен основной квадрант, включающий все традиционные логические значения

— это значение полной неопределенности A ; *второй* — неопределенность однозначных значений (\underline{M} , \underline{S} и др., расположенные в логическом пространстве на границах квадрата \underline{MRSD}); и *третий* — выражающий многозначность или множественность значений (M , S и другие значения на границах квадрата $MRSD$), что, соответственно, в зависимости от конкретных ситуаций и задач, позволяет с максимальной эффективностью реализовывать и использовать одни или другие варианты тетралогии.

В-третьих, что наиболее существенно, тетралогика в различных ее проявлениях рассматривается прежде всего как основа для построения эффективных систем кодирования количественной информации, обладающих по сравнению с традиционными рядом качественных преимуществ.

«Проблема отрицания» в тетралогике и других гиперлогиках требует существенно более глубокой проработки, чем в диалогике. В первом приближении различные альтернативы операции логического отрицания могут задаваться как симметричные преобразования относительно тех или иных осей двумерного логического пространства. Например, отрицание в диалогике и трехзначной логике Лукасевича есть симметричное отражение относительно оси AM .

Подобно тому, как переход от монокодов к дикодам открыл принципиально новые возможности для развития математики и новых вычислительных средств, переход к **тетракодам** также может открыть новые горизонты как в теоретическом плане, так и с точки зрения практической эффективности.

Основные особенности тетракодов можно проиллюстрировать, используя отображения трехразрядных кодовых комбинаций на числовую ось, представляющую целочисленные значения от 0 до 7 (эта операция по сути своей аналогична преобразованию тетракодов в простейший мо-

нокод, который для небольших численных значений является существенно более наглядным, чем прочие коды). При этом введем следующие символьные обозначения, позволяющие отображать не только конкретное единичное значение на числовой оси, но и различные комбинации, образуемые при кодировании количественной информации тетракодом: «□» — пустые значения; «■» — непустые определенные значения; «◇» — непустые неопределенные значения (задаваемые символом А); «◆» — непустые неопределенные значения (задаваемые символом М).

Простейшие комбинации типа 000 или 001 отображаются на числовой оси единичными значениями соответственно как [■□□□□□□□] или [□■□□□□□□] и т. п. В случае использования символа множественности М значения на числовой оси генерируются путем последовательной подстановки вместо М значений 0 и 1. При этом осуществляется перебор всех возможных комбинаций таким образом, что количество одновременно кодируемых значений будет 2^m , где m — количество разрядов, содержащих М.

Например:

$$\begin{aligned} 00M &= [\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare]; \\ 0M0 &= [\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare]; \\ 1MM &= [\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare]. \end{aligned}$$

Аналогично интерпретируется символ S, но с одним существенным различием: при подстановке значения 1 все разряды, расположенные правее данного, инвертируются, что соответствует симметричному отображению значений на числовую ось.

Например:

$$\begin{aligned} 0S1 &= [\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare]; \\ S01 &= [\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare]; \\ SM0 &= [\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare]. \end{aligned}$$

При использовании символа «возможно» M генерация конкретного значения осуществляется путем подстановки 0 или 1, выбираемых случайно для каждого из разрядов, содержащих M (как правило, при каждом использовании кода).

$$\begin{aligned}\text{Например:} \quad 1\underline{M}1 &= [\square\square\square\square\diamond\square\diamond]; \\ \underline{M}0\underline{M} &= [\diamond\diamond\square\square\diamond\diamond\square\square].\end{aligned}$$

Символ полной неопределенности A при каждом использовании содержащих его кодов может интерпретироваться по-разному и в зависимости от конкретной ситуации может заменяться на какой-либо другой символ (0, 1, M, S и др.).

$$\begin{aligned}\text{Например:} \quad 1AA &= [\square\square\square\square\diamond\diamond\diamond\diamond]; \\ 01A \rightarrow 01\underline{M} &= [\square\square\diamond\diamond\square\square\square\square]; \\ 00A \rightarrow 00M &= [\blacksquare\blacksquare\square\square\square\square\square\square].\end{aligned}$$

Интерпретация символов I и O аналогична 1 и 0, но «с точностью до наоборот» и соответствует формуле «все значения кроме».

$$\begin{aligned}\text{Например:} \quad OOO &= [\square\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare]; \\ 1II &= [\square\square\square\square\blacksquare\blacksquare\blacksquare\square]; \\ O10 &= [\square\square\blacksquare\square\blacksquare\blacksquare\blacksquare\blacksquare].\end{aligned}$$

Символы D и R могут быть интерпретированы как «множественность с инверсией» и «симметрия с инверсией» соответственно.

$$\begin{aligned}\text{Например:} \quad D01 &= [\square\blacksquare\square\square\blacksquare\square\blacksquare\blacksquare]; \\ R01 &= [\square\blacksquare\square\square\blacksquare\blacksquare\square\blacksquare].\end{aligned}$$

Для кодирования детерминированных структур, процессов и сигналов с наибольшей эффективностью могут

использоваться тетракоды $C_3^4 = \{1, 0, S, M\}$ или октокоды $C_1^8 = \{1, 0, M, R, O, S, I, D\}$.

В случае же, когда объект кодирования характеризуется стохастичностью, наиболее целесообразно использование гиперкодов типа

$$C_1^4 = \{1, 0, A, M\}, C_2^4 = \{1, 0, \underline{M}, M\}, C_4^4 = \{1, 0, A, \underline{M}\},$$

$$C_2^8 = \{1, 0, M, S, R, D, A, \underline{M}\} \text{ или}$$

$$C_1^{16} = \{1, 0, M, R, O, S, I, D, \underline{1}, \underline{0}, \underline{M}, \underline{R}, \underline{O}, \underline{S}, \underline{I}, \underline{D}\}.$$

Таким образом, представление чисел в виде гиперкодов дает возможность гибкого задания не только фиксированных «точечных» значений на числовой оси (как в традиционном кодировании), но и множество значений одновременно. По сравнению с обычным бинарным кодом количество бит, требуемых для кодирования чисел возрастает при этом в $\log_2 K$ раз, где K — количество возможных значений в каждом разряде числа при его машинном представлении (см. раздел 1.2).

Для представления тетракодов, например, требуется удвоенная длина кодовых слов. Однако повышение степени информативности получаемых за счет этого кодов вполне оправдывает увеличение затрат на кодирование. Экспериментально, в частности, доказана эффективность использования тетракодов для представления различных произвольных структур на примере кодирования изображений [10, 11].

1.8. Выводы

Таким образом, рассмотренные в данной главе основные вопросы, связанные с развитием, текущим состоянием и ближайшими перспективами концепции обобщенного кодо-логического базиса, позволяют предположить, что

актуальность исследований и разработок в данном направлении будет в дальнейшем возрастать, что в итоге может привести к определенному качественному скачку в развитии компьютерного моделирования, а также — технологий кодирования и представления знаний.

В то же время к числу важнейших составляющих расширенного или обобщенного кодо-логического базиса следует относить также монологику и монокоды, на базе которых зародилось и длительное время развивалось вычислительное моделирование в виде так называемых монокодовых вычислительных моделей.

Предложенная в данной главе концепция позволяет реализовать существенно новый подход к повышению эффективности методов и средств «вычислительного интеллекта», суть которого состоит в интеграции преимуществ различных кодо-логических систем и получении за счет этого значительного синергетического эффекта. Первостепенное значение возможности расширенного кодо-логического базиса имеют для высокопроизводительного компьютерного моделирования сложных систем, где требуются развитые возможности достаточно компактного и формализованного представления информации о явлениях и объектах реального мира, с трудом поддающихся формализации.

Глава 2

ПРОСТРАНСТВО И АЛГЕБРА ТЕТРАЛОГИКИ

2.1. Основные тенденции перехода к пространству тетралогии¹

Так как логика служит одним из инструментов практически любой науки, то с появлением и широким распространением вычислительной техники логика оказала значительное влияние на развитие искусственного (а позже — и вычислительного) интеллекта.

Классическая булева логика [2] основана всего на двух логических состояниях: истина (логическая единица) и ложь (логический ноль). Такая логика из-за простоты реализации получила широкое распространение в вычислительной технике и более точное название: двоичная (двузначная) логика. Считается, что двоичная логика в достаточной степени способна выполнять основную функцию классической логики: исследование того, как из одних утверждений можно выводить другие. Но реальное мышление не сводится к манипуляциям только двумя логическими значениями, поскольку часто приходится иметь дело с противоречивой информацией, например, поступившей от нескольких независимых источников. Используемый двузначную логику компьютер не является достаточно совершенным устройством, которое, столкнувшись с противоречием, было бы способно сделать нечто большее, чем просто зафиксировать его существование. В частно-

¹ Разделы 2.1–2.3 подготовлены по материалам публикации [1].

сти, Н. Белнап в 1976 году высказал предположение, что совершенное устройство должно обладать некоторой стратегией, с тем чтобы, обнаружив противоречивость определенных представлений, иметь возможность отказаться от них [3]. Несмотря на постоянное увеличение вычислительной мощности современных компьютерных систем, существующие логические основы их работы не позволяют в должной степени приблизить искусственный интеллект к человеческому, что делает практически недостижимым одно из требований к искусственному интеллекту: **выполнение функций (в частности, творческих), которые традиционно считаются прерогативой человека** [4]. Поэтому современное состояние «логического аппарата» компьютерных технологий представляет собой рубеж, преодоление которого положит начало созданию компьютерной техники нового поколения, в качестве обобщающего названия для которой целесообразным представляется использование определения «постбинарный компьютеринг» [5–7].

Двоичная логика и основанные на ней системы счисления представляют собой логическую систему, которая по своей сути является одномерной, поскольку строится в пределах оси, соединяющей логические «0» и «1». Такая одномерная логическая система не единственна и не достаточна, однако она является одним из наиболее значимых элементов современного интеллектуального инструментария. В целом можно констатировать, что XX век ознаменовался прогрессирующим нарастанием протеста против двузначности: это и отвержение закона исключенного третьего, и различные попытки преодоления «парадокса материальной импликации», и введение трехзначной логики, и общее усиление активности в области многозначных логик, и, наконец, появление нечеткой логики Заде, справед-

ливо квалифицируемой «как вызов, брошенный европейской культуре с ее дихотомическим видением мира в жестко разграничиваемой системе понятий» [8].

Другими важными составляющими являются как некоторые более ранние формы мышления и представления количественной информации, так и множество перспективных, существующих пока в не полностью оформившемся виде, но все же обладающих значительным информационным потенциалом. К одной из таких перспективных составляющих можно отнести двумерное логическое пространство, которое может быть порождено базисом, состоящим из ортонормированной системы векторов «истина» и «ложь» с положительными и отрицательными осями (рис. 1.4). Таким образом, введение новых логических значений позволяет значительно расширить возможности формализованной логической оценки разнообразных реальных процессов и ситуаций, что является существенным шагом к «очеловечиванию» машинной логики.

В зависимости от количества используемых логических значений в двумерном логическом пространстве возможно построение ряда различных логических систем. В частности, к таким системам можно отнести монологику (1)¹, дилогику (2), трилогику (3), тетралогику (4), пенталогику (5), и т. п. На базе полученных логических систем могут быть построены и введены в рассмотрение системы кодирования количественной информации, которые задаются так же, как и соответствующие им логические системы. Такими системами кодирования могут быть монокоды, дикоды, трикоды, тетракоды, пентакоды и т. п. (см. разделы 1.4–1.7).

¹ В скобках указано количество используемых логических значений в каждой приведенной логической системе

В разделе 1.2 было предложено логики третьего и более высоких порядков объединить одним термином «гиперлогика», а все соответствующие этим логикам системы кодирования — термином «гиперкоды», что в совокупности является кодо-логическим базисом постбинарного компьютеринга. В совокупности с традиционными бинарными логикой и кодами, а также с монологикой и монокодами гиперлогика и гиперкоды образуют более общее понятие: «расширенный (обобщенный) кодо-логический базис».

В данной главе рассматриваются основные особенности реализации постбинарных логических операций, использующих четыре логических значения (состояния). При этом в качестве наиболее перспективного варианта рассматривается тетралогика (рис. 2.1), включающая в себя наряду с классическими «истиной» (1) и «ложью» (0) значения «неопределенности» (A) и «множественности» (M).

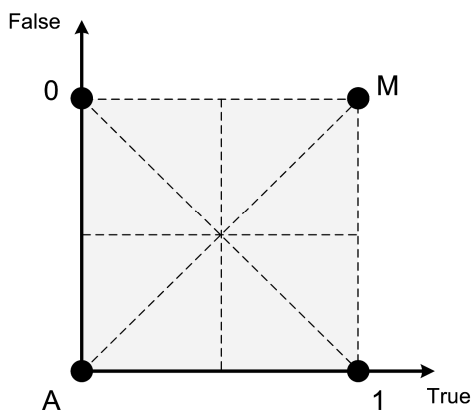


Рисунок 2.1 — Двумерное логическое пространство с указанными логическими значениями рассматриваемого варианта тетралогии

Полученные результаты могут служить основой для дальнейшего развития так называемой «алгебры тетралогии», которую можно рассматривать как важный этап развития постбинарной логики будущего. Данный вариант тетралогии представим в виде множества четырех состояний $L_4 = \{0, 1, A, M\}$. Состояния тетралогии

могут кодироваться тетракодом, представленным в виде значений $S_4 = \{0, 1, A, M\}$ и используемым по аналогии с бинарным кодом для поразрядного представления количественных значений. При этом в отличие от двоичной логики, где n -разрядное пространство определено 2^n двоичными значениями, в тетралогике количество увеличивается до $2^{2n} = 4^n$ значений тетракодов.

По аналогии с понятием «бит» (*англ.* binary digit — один разряд в двоичной системе счисления) для поразрядного представления тетракода целесообразно ввести понятие **тетрит** (*англ.* **tetrit**), являющееся соответствующей трансформацией корневой основы «тетра» (*англ.* tetra), связанной со значением «четыре».

Следует отметить, что параллельно понятие «tetrit» было предложено использовать как средство для упрощения и разъяснения семантики обработки исключительных ситуаций при интервальных вычислениях [9], что в дальнейшем можно рассматривать как альтернативный (более узкий) вариант использования данного термина.

Тетралогика в простейшем варианте является конечнозначной и, согласно теории функциональных систем [10], может быть описана классом k -значных функций при $k = 4$. Таким образом, имеет место запись так называемой тетрафункции $\text{Tr}^n \rightarrow \text{Tr}$, где $\text{Tr} = \{0, 1, A, M\}$ — выбранное множество тетралогии, $n \in \mathbb{Z}$, $n \geq 0$ — арность или местность функции. В данном представлении тетрафункции, элементы Tr^n можно назвать векторами тетралогии (тетравекторами). В случае $n = 0$ тетрафункция превращается в константу — одно из представленных состояний тетритов. Максимальное количество возможных n -местных логических операций тетралогии определимо как число функций N_k^n в k -значной системе:

$$N_k^n = (k)^{k^n \cdot m}, \quad (2.1)$$

где k — число знаков (основание системы счисления), n — число аргументов (входов), а m — число выходов [11]. Таким образом, в тетралогике определено $N_4^n = (4)^{4^n \cdot m}$ простейших n -арных тетрафункций с m -арным результатом (выходом).

В разделах 2.2–2.3 определены возможности выполнения основных операций тетралогии, в частности реализации ряда унарных и бинарных поразрядных логических операций с тетритами. При этом проанализированы свойства логических операций тетралогии и выполнение некоторых законов традиционной алгебры логики.

2.2. Реализация нольместных и одноместных логических операций

Согласно (2.1) в тетралогике могут быть определены $N_4^0 = (4)^{4^0 \cdot 1} = 4^1 = 4$ простейших нольместных (нульарных, $n = 0$) логических тетрафункций, которые имеют следующие обозначения: 0 — логический тождественный ноль; 1 — логическая тождественная единица; А — неопределенность логического состояния, принимающая значение тождественных ноля или единицы; М — множественность логического состояния, принимающая значение тождественных ноля и единицы.

Одноместные (унарные) функции тетралогии определены в количестве $N_4^1 = (4)^{4^1 \cdot 1} = 4^4 = 256$ (в отличие от 4-х и 27-ми унарных функций двоичной и троичной логики). Количество возможных унарных функций тетралогии также равно числу размещений с повторениями \bar{A}_n^k при

$k = n = 4$: $\overline{A}_n^k = n^k = 4^4 = 256$. В табл. 2.1 приведены названия и обозначения 16-ти базовых унарных тетрафункций. Столбец «№» обозначает номер тетракода в упорядоченном множестве из 256 значений: 0000, 0001, 000A, 000M, ..., MMM0, MMM1, МММА, ММММ. При этом каждый разряд такого значения является результатом унарной операции f над входным разрядом x .

На рис. 2.2 и 2.3 приведена графическая интерпретация некоторых базовых унарных тетрафункций для тетрита X , определенного на двумерном логическом пространстве как множество $X \in C_4$.

Для унарных логических операций инверсной группы справедлив положенный в основу классической логики **закон двойного отрицания** [12], который в контексте тетралогии представляет явление, расширяющее не только концепцию двойного отрицания как такового, но и принадлежность к его определенному виду. Соответствующая речевая конструкция данного закона может быть, например, расширена до следующего выражения: «если известно, как именно неверно, что неверно X , то аналогичным образом известно, что верно X ».

В классической логике двойное отрицание высказывания X является следствием суждения X , поэтому в рамках тетралогии имеет место тавтология:

$$X \rightarrow \text{SWAP_}\Psi(\text{SWAP_}\Psi(X))$$

при $\Psi = \{A/M, 0/1, FL, TR\}$.

Обратное утверждение $\text{SWAP_}\Psi(\text{SWAP_}\Psi(X)) \rightarrow X$ при $\Psi = \{A/M, 0/1, FL, TR\}$ также не противоречит закону двойного отрицания в классической логике. Здесь Ψ определяет один из видов отрицания (инверсии) и выражает

первую часть адаптированной под тетралогiku речевой конструкции: «если известно, как именно...».

Таблица 2.1 — Базовые унарные функции тетралогики

№	x				Название функции	Обозначение
	A	1	M	0		
	f(x)					
1	0	0	0	0	тождественный ноль	O(x)=0
29	0	1	M	0	минимизация неопределенности	MIN_A(x)
40	0	A	1	M	циклический сдвиг (поворот, вращение) назад на 1 (1/4 оборота) или вперед на 3 (3/4 оборота)	SHIFT_1B(x), SHIFT_3F(x)
55	0	M	1	A	вероятностная инверсия по оси «FALSE»	SWAP_FL(x)
86	1	1	1	1	тождественная единица	I(x)=1
93	1	1	M	0	максимизация неопределенности	MAX_A(x)
100	1	A	0	M	вероятностная инверсия по оси «TRUE»	SWAP_TR(x)
115	1	M	0	A	циклический сдвиг (поворот, вращение) вперед на 1 (1/4 оборота) или назад на 3 (3/4 оборота)	SHIFT_1F(x), SHIFT_3B(x)
142	A	0	M	1	инверсия около неопределенности и множественности (традиционная инверсия)	\bar{x} , #x, SWAP_A/M(x)
145	A	1	0	0	минимизация множественности	MIN_M(x)
149	A	1	1	0	максимизация множественности	MAX_M(x)
157	A	1	M	0	тождественная функция, повторитель	x , SHIFT_0B(x), SHIFT_0F(x),
171	A	A	A	A	абсолютная неопределенность	A(x)=A
202	M	0	A	1	циклический сдвиг (поворот, вращение) вперед на 2 (1/2 оборота) или назад на 2 (1/2 оборота)	SHIFT_2F(x), SHIFT_2B(x)
217	M	1	A	0	инверсия около нуля и единицы	SWAP_0/1(x)
256	M	M	M	M	абсолютная множественность	M(x)=M

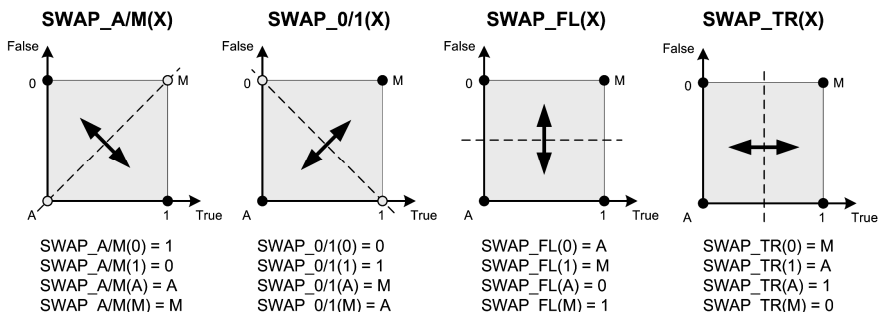


Рисунок 2.2 — Графическая интерпретация унарных логических операций инверсной группы (swap group)

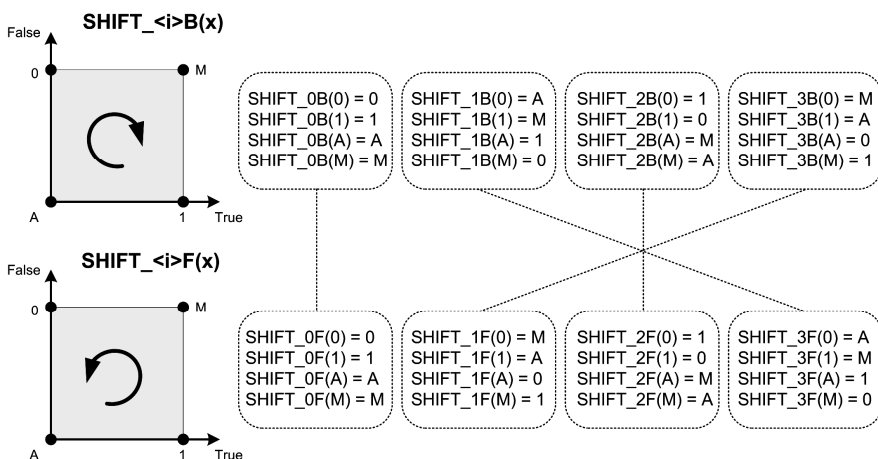


Рисунок 2.3 — Графическая интерпретация унарных логических операций сдвиговой группы (shift group)

В обобщенной записи закона двойного отрицания инверсных операций тетралогии можно воспользоваться свойствами математической равнозначности:

$$X \leftrightarrow \text{SWAP_}\Psi(\text{SWAP_}\Psi(X)).$$

Для унарных логических операций сдвиговой группы определены направления сдвига. Так, под словом «назад» (сдвиговая операция $\text{SHIFT_}\langle i \rangle B$, B от англ. *Backward* — назад) следует понимать направление сдвига, совпадающее с направлением оси «False» двумерного логического пространства (рис. 2.3). Аналогичным образом под словом «вперед» (сдвиговая операция $\text{SHIFT_}\langle i \rangle F$, F от англ. *Forward* — вперед) — направление сдвига, совпадающее с направлением оси «True». Целое положительное число $\langle i \rangle$ ($i=0 \cup i \in N$) определяет количество сдвигов (или «поворотов» условной плоскости в концепции двумерного логического пространства).

На рис. 2.3 пунктирными линиями показаны соответствия при выполнении сдвиговых унарных операций тетралогии. При $i=0$ сдвиговые унарные операции $\text{SHIFT_}\langle i \rangle \vartheta$ при $\vartheta = \{B, F\}$ идентичны и соответствуют тождественной функции, т. е. являются повторителями:

$$X \rightarrow \text{SHIFT_}0B(X) \rightarrow X, \quad X \rightarrow \text{SHIFT_}0F(X) \rightarrow X.$$

Очевидно, что $X \leftrightarrow \text{SHIFT_}0B(X)$, $X \leftrightarrow \text{SHIFT_}0F(X)$, откуда следует соответствие

$$\text{SHIFT_}0B(X) \leftrightarrow \text{SHIFT_}0F(X).$$

Аналогичным образом прослеживаются следующие идентичности (равнозначности):

$$\text{SHIFT_}3B(X) \leftrightarrow \text{SHIFT_}1F(X);$$

$$\text{SHIFT_}1B(X) \leftrightarrow \text{SHIFT_}3F(X);$$

$$\text{SHIFT_}2B(X) \leftrightarrow \text{SHIFT_}2F(X).$$

Множество сдвиговых унарных операций $\text{SHIFT}_{<i>\vartheta$ ($\vartheta = \{B, F\}$) определено также для $i > 3$ при условии, что для любого $\omega \in N$ количества сдвигов (поворотов) справедливо равенство $i = (\omega)_{\text{mod}4}$. В частности, циклическое вращение вперед или назад на 4 ($i = 4$) условная плоскость логического пространства совершает $4/4 = 1$ полный оборот, повторяя его предыдущее состояние. В таком случае $i = (4)_{\text{mod}4} = 0$, и функция циклического сдвига принимает вид $\text{SHIFT}_{0\vartheta}$ при $\vartheta = \{B, F\}$. Такая организация логических сдвиговых операций подчиняется **закону n -ного сдвига в n -ичной логике**, согласно которому n сдвигов вперед/назад равносильны повторению (утверждению). Ниже приведены примеры унарных тетрафункций сдвиговой группы, возвращающих значение тетрита X после поворота логического пространства на $i/4$ оборота при значениях i , приведенных к диапазону значений $0 \div 3$:

$$\text{SHIFT}_{16B}(X) = \text{SHIFT}_{0B}(X);$$

$$\text{SHIFT}_{18F}(X) = \text{SHIFT}_{2F}(X);$$

$$\text{SHIFT}_{25B}(X) = \text{SHIFT}_{1B}(X);$$

$$\text{SHIFT}_{31F}(X) = \text{SHIFT}_{3F}(X).$$

Унарные функции $\text{MIN}_{\Theta}(X)$ и $\text{MAX}_{\Theta}(X)$ при $\Theta = \{A, M\}$ — тетрафункции, приводящие неоднозначное значение тетрита, выраженное в состоянии неопределенности A или множественности M , к определенным (однозначным) значениям логических нуля и единицы.

Здесь в качестве минимального возвращаемого логического состояния выбрано значение нуля, а в качестве максимального возвращаемого логического состояния — значение единицы.

Таким образом,

$$\text{MIN}_A(X) = \begin{cases} 0, & \text{if } X = A, \\ X, & \text{if } X \neq A; \end{cases}$$

$$\text{MAX}_A(X) = \begin{cases} 1, & \text{if } X = A, \\ X, & \text{if } X \neq A; \end{cases}$$

$$\text{MIN}_M(X) = \begin{cases} 0, & \text{if } X = M, \\ X, & \text{if } X \neq M; \end{cases}$$

$$\text{MAX}_M(X) = \begin{cases} 1, & \text{if } X = M, \\ X, & \text{if } X \neq M. \end{cases}$$

Рассмотренные нульарные и унарные функции тетралогии представляют собой теоретическую основу, на базе которой возможна аппаратная реализация 4-х безвыходных и 16-ти (из 256-ти возможных) базовых одновходовых логических элементов тетралогии с унарным выходом.

2.3. Реализация базовых двуместных логических операций

Двуместные функции тетралогии позволяют создать множество полных систем функций представленной логики. Из (2.1) общее число двуместных (бинарных) логических операций тетралогии определено в количестве

$$N_4^2 = (4)^{4^2 \cdot 1} = 4^{16} = 4\,294\,967\,296$$

(к примеру, в двоичной логике определено всего 16, а в троичной логике — 19 683 двуместных операций). В частности, для достаточного набора тетрафункций возможна адаптация критерия Поста и для алгебры тетралогии, в котором появится возможность сформулировать необходимое и достаточное условие для того, чтобы некоторый

набор тетрафункций обладал достаточной выразительностью для представления любой тетрафункции из полного набора. Однако следует учесть, что на данный момент в тетралогике, как и в любой k -значной логике не существует полного описания замкнутых классов при $k > 2$. Полное описание системы замкнутых классов для двузначной логики ($k = 2$) было предложено Эмилем Постом в 1940 году [13, с. 9]. Отсутствие замкнутых классов является одним из барьеров, сдерживающих развитие и распространение k -значной (гипер-) логики.

В качестве базовых двуместных тетрафункций можно выделить конъюнкцию и дизъюнкцию как операции, которые определены для непустого множества $\{0, 1, A, M\}$, элементами которого являются высказывания.

Следует отметить, что высказывания в тетралогике могут быть истинными, ложными, содержащими или истину или ложь, содержащими и истину и ложь одновременно. Таким образом, под алгеброй тетралогии следует понимать ту ее часть (раздел), которая определяет логические операции над высказываниями.

Логическое умножение (конъюнкция) — логическая операция, по своему применению максимально приближенная к союзу «и». Словосочетание «логическое умножение» может быть равнозначно заменено выражениями «И» или «логическое И». Для обозначения операции логического сложения в данной работе используется знак « \wedge », т. е. $\wedge \equiv$ «И». В традиционной логике высказывание $X \wedge Y$ истинно тогда и только тогда, когда оба высказывания X , Y истинны. В тетралогике в качестве одного из способов определения операции конъюнкции может использоваться схема $X \wedge Y = \min(X, Y)$ где $X, Y \in \{0, 1, A, M\}$, при которой сохраняется совместимость с булевой алгеброй для значений высказываний «0» и «1».

В контексте выражения $X \wedge Y = \min(X, Y)$ можно определить бинарную тетрафункцию $\text{BMIN}(X, Y)$ ¹, которую в силу наличия неявных, не поддающихся однозначному сравнению, значений тетритов (неопределенность А, многозначность М) целесообразно разбить на две функции минимума $\text{BMIN_A}(X, Y)$ и $\text{BMIN_M}(X, Y)$. Так, для функции BMIN_A (BMIN_M), значение одного тетрита-операнда, равное А (М), принимается **меньшим** (уступающим) по отношению к значению другого тетрита-операнда, равного М (А). В остальных случаях значение логического нуля принято как минимальное значение, а значение логической единицы — как максимальное. Аналогичным образом определима бинарная тетрафункция максимума $\text{BMAX}(X, Y)$. Следовательно, если $\text{BMIN_A}(A, M) = A$ ($\text{BMIN_M}(A, M) = M$), то $\text{BMAX_A}(A, M) = M$ ($\text{BMAX_M}(A, M) = A$).

Реализация объявленных тетрафункций $\text{BMIN}(X, Y)$ и $\text{BMAX}(X, Y)$ приведена в табл. 2.2–2.5.

Таким образом, следует различать операцию конъюнкции тетритов как операцию, определенную тождествами $X \wedge Y = \text{BMIN_A}(X, Y)$; $X \wedge Y = \text{BMIN_M}(X, Y)$.

В первом случае для символа конъюнкции « \wedge » приняты обозначения « $\wedge^{<A>}$ » или « $\underline{\wedge}$ », во втором — « $\wedge^{<M>}$ » или « $\overline{\wedge}$ ». Реализация операции конъюнкции для тетритов-операндов тетрафункции полностью соответствует приведенной в табл. 2.2–2.3 тетрафункции BMIN , причем

$$\wedge^{<A>} \equiv \text{BMIN_A}, \quad \wedge^{<M>} \equiv \text{BMIN_M}.$$

¹ Буква «В» в названии функции MIN (а далее и в названии функции MAX) указывает на то, что она бинарная (от англ. binary), во избежание путаницы с определенной ранее унарной функцией минимума (максимума).

Таблица 2.2 —
Функция BMIN_A

BMIN_A	0	1	A	M
0	0	0	0	0
1	0	1	A	M
A	0	A	A	A
M	0	M	A	M

Таблица 2.3 —
Функция BMIN_M

BMIN_M	0	1	A	M
0	0	0	0	0
1	0	1	A	M
A	0	A	A	M
M	0	M	M	M

Таблица 2.4 —
Функция BMAX_A

BMAX_A	0	1	A	M
0	0	1	A	M
1	1	1	1	1
A	A	1	A	M
M	M	1	M	M

Таблица 2.5 —
Функция BMAX_M

BMAX_M	0	1	A	M
0	0	1	A	M
1	1	1	1	1
A	A	1	A	A
M	M	1	A	M

Логическое сложение (дизъюнкция) — логическая операция, по своему применению максимально приближенная к союзу «или» в смысле «или то, или это, или оба сразу». Словосочетание «логическое сложение» может быть равнозначно заменено выражениями «ИЛИ», «логическое ИЛИ», «включающее ИЛИ». Для обозначения операции логического сложения в данной работе используется знак « \vee », т. е. $\vee \equiv$ «ИЛИ».

В традиционной логике высказывание $X \vee Y$ истинно тогда и только тогда, когда хотя бы одно из высказываний X , Y истинно. В тетралогике в качестве одного из способов

определения операции дизъюнкции может использоваться схема $X \vee Y = \max(X, Y)$ где $X, Y \in \{0, 1, A, M\}$, при которой также сохранена совместимость с булевой алгеброй для значений высказываний «0» и «1».

Таким образом, следует различать операцию дизъюнкции тетритов как операцию, определенную тождествами

$$X \vee Y = \text{BMAX_A}(X, Y); \quad X \vee Y = \text{BMAX_M}(X, Y).$$

В первом случае для символа дизъюнкции « \vee » приняты обозначения « $\vee^{<A>}$ » или « $\underline{\vee}$ », во втором — « $\vee^{<M>}$ » или « $\bar{\vee}$ ». Реализация операции дизъюнкции для тетритов-операндов тетрафункции полностью соответствует приведенной в табл. 2.4–2.5 тетрафункции BMAX, причем

$$\vee^{<A>} \equiv \text{BMAX_A}, \quad \vee^{<M>} \equiv \text{BMAX_M}.$$

Следует отметить, что полученные логические операции конъюнкции и дизъюнкции тетралогии не нарушают ни один из основных законов, присущих аналогичным операциям традиционной логики. Поскольку выполнение законов логики позволяет проверять правильность рассуждений и доказательств, а нарушения этих законов приводят к логическим ошибкам и вытекающим из них противоречиям, то можно утверждать о корректности сформулированных логических операций тетралогии.

Рассмотрим выполнение законов коммутативности, ассоциативности и дистрибутивности для логических операций конъюнкции и дизъюнкции тетралогии:

1) **Законы коммутативности** (переместительности):

$$X \wedge Y \equiv Y \wedge X, \quad X \vee Y \equiv Y \vee X.$$

Например,

$$A \underline{\wedge} M \equiv M \underline{\wedge} A \Leftrightarrow A \equiv A; \quad A \bar{\wedge} M \equiv M \bar{\wedge} A \Leftrightarrow M \equiv M;$$

$$A \underline{\vee} M \equiv M \underline{\vee} A \Leftrightarrow M \equiv M; \quad A \bar{\vee} M \equiv M \bar{\vee} A \Leftrightarrow A \equiv A.$$

2) **Законы ассоциативности** (сочетательности):

$$(X \wedge Y) \wedge Z \equiv X \wedge (Y \wedge Z), \quad (X \vee Y) \vee Z \equiv X \vee (Y \vee Z).$$

Например, $(A \bar{\wedge} M) \bar{\wedge} M \equiv A \bar{\wedge} (M \bar{\wedge} M) \Leftrightarrow M \equiv M;$

$$(M \bar{\vee} A) \bar{\vee} A \equiv M \bar{\vee} (A \bar{\vee} A) \Leftrightarrow A \equiv A.$$

3) **Законы дистрибутивности** (распределительности):

$$X \wedge (Y \vee Z) \equiv (X \wedge Y) \vee (X \wedge Z),$$

$$X \vee (Y \wedge Z) \equiv (X \vee Y) \wedge (X \vee Z).$$

Например,

$$A \bar{\wedge} (M \bar{\vee} A) \equiv (A \bar{\wedge} M) \bar{\vee} (A \bar{\wedge} A) \Leftrightarrow A \equiv A;$$

$$A \underline{\wedge} (M \underline{\vee} A) \equiv (A \underline{\wedge} M) \underline{\vee} (A \underline{\wedge} A) \Leftrightarrow A \equiv A.$$

2.4. Представление элементов тетралогик с помощью аксиоматического аппарата теории множеств¹

Современная теория множеств строится на системе аксиом **Цермело-Френкеля** (ZF — Zermelo-Fraenkel), из которых выводятся все теоремы и утверждения теории множеств. К системе аксиом ZF часто добавляют аксиому выбора и называют системой Цермело-Френкеля с аксиомой выбора (ZFC — Zermelo-Fraenkel set theory with the axiom of Choice) [15, с. 157–166].

В контексте тетралогик система ZFC также представляет интерес, поскольку всякое логическое пространство может быть «переведено» на язык теории множеств таким образом, что полученные теоремы станут теоремами о множествах, доказуемыми из аксиом ZFC. Иными словами, поскольку любой объект можно считать множеством, то, соответственно, появляется возможность периформу-

¹ Разделы 2.4, 2.5 подготовлены по материалам публикации [14].

лизовать какое-либо утверждение, не нарушив его первоначальную истинность.

Таким образом, пространство тетралогии (тетралогическое пространство) можно представить совокупностью четырех непустых множеств: Q, J — содержащих по одному фиксированному логическому элементу («истина», «ложь»); A, M — содержащих, в конечном представлении, по два элемента, представляющих собой результат логической операции над элементами множеств Q, J («истина» или «ложь», «истина» и «ложь»). Математическая запись определения множеств Q, J, A и M :

$$\forall q \exists Q (q \subseteq Q \rightarrow q = 0); \quad (2.2)$$

$$\forall j \exists J (j \subseteq J \rightarrow j = 1); \quad (2.3)$$

$$\forall a \exists A (a \subset A \rightarrow a \in Q \vee a \in J); \quad (2.4)$$

$$\forall m \exists M (m \subset M \rightarrow m \in Q \wedge m \in J). \quad (2.5)$$

Из записей (2.2) и (2.3) очевидно, что множества Q и J представляют собой множества фиксированных значений $\{0\}$ и $\{1\}$ соответственно ($q \in Q \rightarrow q = \text{const } 0; j \in J \rightarrow j = \text{const } 1$). Запись (2.4) показывает, что множество A определимо как сумма (объединение) множеств Q и J ($a \in (Q \cup J)$) и словесно может быть описано как «или 0 или 1». Запись (2.5) показывает, что множество M определимо как произведение (пересечение) множеств Q и J ($m \in (Q \cap J)$) и словесно может быть описано как «и 0 и 1».

Однако для полного представления гиперлогического пространства необходимо ввести еще одно множество — универсум U , представляющее собой пространство, содержащее все четыре ранее объявленных множества. Таким образом,

$$\forall q \forall j \forall a \forall m \exists U = \{ u : u \in q \vee u \in j \vee u \in a \vee u \in m \}. \quad (2.6)$$

Из записи (2.6) следует, что множество U представляет собой один из элементов тетракода, имеющего возможность каждый элемент множества $\{Q, J, A, M\}$ при определенных обстоятельствах представлять в виде логического «0» или «1».

На основании принятых утверждений (2.2–2.5) можно определить понятия каждого элемента тетралогики:

$Q = \{0\}$ — множество «ложь», представляющее значение «ложь» (логический 0) классической логики;

$J = \{1\}$ — множество «истина», представляющее значение «истина» (логическая 1) классической логики;

A — множество абсолютной неопределенности, «непроявленности» (на данный момент, т. е. на момент фиксации логического состояния (высказывания), неизвестно, или «истина» или «ложь»), которая может быть выражена объединением множеств «истина» и «ложь»;

M — множественность, многозначность (и «истина» и «ложь» одновременно, т. е. невозможна однозначная фиксация высказывания), которая может быть выражена пересечением множеств «истина» и «ложь».

В графическом представлении тетралогического пространства (рис. 2.4) можно выделить следующее: если U — прямоугольник $PKLN$ и Q — прямоугольник $PKCE$, то J — прямоугольник $ECLN$, A — прямоугольник $BKLD$ и M — прямоугольник $PBDN$.

Иногда удобно опускать буквенное обозначение

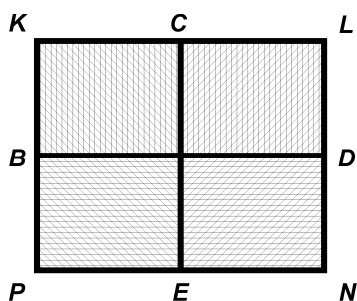


Рисунок 2.4 — Графическое представление пространства тетралогики

множеств Q и J , заменяя их, в силу определений (2.2) и (2.3), числовыми эквивалентами 0 и 1 соответственно. В этом случае множество U можно представить в виде множества элементов $\{0, 1, A, M\}$.

На заключительном этапе представления элементов тетралогии с учетом их определений (2.2–2.6) и графического представления (рис. 2.4) можно ввести следующие закономерности:

- $0 = Q$ (в частности, $q \in \{0\}$);
- $1 = J$ (в частности, $j \in \{1\}$);
- $(A \cap M) \subset (A \cup M) \subset U$ — для множеств A и M на универсуме U определены операции объединения и пересечения;
- $a \in (Q \cup J), \quad q \subset A, \quad j \subset A, \quad (Q \cup J) \neq \emptyset \Rightarrow a \notin \emptyset \vee a \in \{0, 1\}$ — формирование множества неоднозначности A ;
- $m \in (Q \cap J), \quad q \subset M, \quad j \subset M, \quad (Q \cap J) \neq \emptyset \Rightarrow m \notin \emptyset \vee m \in \{0\} \wedge m \in \{1\}$ — выделение противоречия при формировании множественности M : **множество M не может быть пустым, однако для множеств Q и J не определена операция пересечения.**

Возникшее противоречие в данном представлении является достаточно актуальным, так как позволяет объяснить ряд существующих спорных моментов: **во-первых**, данное противоречие проявляется и в определении самой «множественности», поскольку при одновременном поступлении противоречивых высказываний попытка зафиксировать сразу два значения «истина» и «ложь» приведет к логическому парадоксу, при котором компьютер (являющийся по своей сути классическим двузначным логиче-

ским устройством) должен полностью отказаться от принятия и дальнейшей обработки противоречивой информации; **во-вторых**, из двух значений (множеств) тетралогии A и M данное противоречие **позволяет в ряде случаев выделить множественность M как доминирующее над неопределенностью A состояние** и в качестве альтернативного выхода предложить поочередную подстановку значений «1» и «0» (в контексте традиционного бинарного компьютеринга) в содержащее эту множественность тетракодовое представление числа.

Такая поочередная подстановка фиксированных «0» (элемент множества Q) и «1» (элемент множества J) делает возможным приведение множества M к виду «плавающего» (неопределенного) подмножества множества U , попеременно принимая значения подмножеств $M1$ и $M2$. Графическое представление подмножеств $M1$ и $M2$ на множестве U (причем $Q \subset U, J \subset U$) приведено на рис. 2.5.

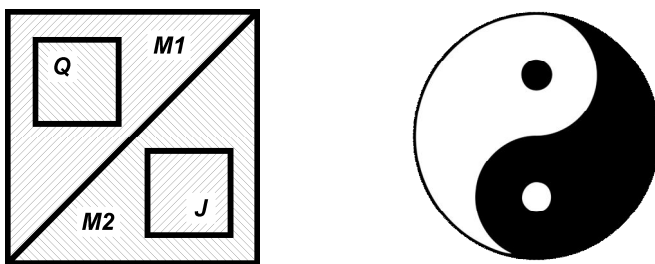


Рисунок 2.5 — Аналогия представления подмножеств $M1$ и $M2$ (слева) с изображением Великого предела китайской модели мира (справа)

Нетрудно заметить что данное графическое представление $M1$ и $M2$ по своей структуре напоминает изображе-

ние так называемого «Великого предела», ставшего эмблемой китайской классической философии, в которой отображено предельное состояние бытия со сложным взаимодействием сил инь (представленной, например, логическим «0») и ян (представленной, например, логической «1») [16].

Подобное «родство» элементов гиперлогики с философской сущностью бытия существенно расширяет применение как соответствующей логики, так и расширенного кодо-логического базиса в целом. Наряду с важностью его применения с целью обеспечения более реалистичного высокопроизводительного компьютерного моделирования сложных систем становится возможным также создание вычислительных моделей, способных представить в вычисляемом виде большинство противоречий окружающего мира.

2.5. Реализация базовых логических операций над элементами тетралогики, представленными с помощью аксиоматического аппарата теории множеств

Реализация базовых логических операций отрицания «НЕ» («NOT»), логического сложения «ИЛИ» («OR»), логического умножения «И» («AND») на тетралогическом множестве-универсуме U производится также с помощью аксиоматики теории множеств.

Отрицание в логике — унарная (одноместная) операция над высказываниями, результатом которой является высказывание (в известном смысле), «противоположное» исходному. Для обозначения отрицания множества примем следующие обозначения: если φ — элемент множества Φ , являющегося подмножеством универсума U , то

отрицание φ — элемент $\bar{\varphi}$ множества $\bar{\Phi}$, при этом справедлива следующая запись:

$$\varphi \in \Phi, \Phi \subset U \Rightarrow \bar{\varphi} \in \bar{\Phi}, \bar{\Phi} \subset U \Rightarrow \bar{\varphi} \in \bar{\Phi} \neq \varphi \in \Phi, \varphi, \bar{\varphi} \in U.$$

Однозначно представимые «ложь» и «истина» по своему определению являются элементами классической двоичной логики, т. е. для множеств $Q = \{0\}$ и $J = \{1\}$ справедливы следующие равенства:

$$\bar{Q} = J, \bar{J} = Q \Leftrightarrow \bar{0} = 1, \bar{1} = 0.$$

Действительно, для множеств $Q \subset U, J \subset U$, для которых $Q \cup J = U$, возможны следующие соотношения:

$$\forall q \in Q, U \exists \bar{Q} \left(\bar{Q} = \{ \bar{q} : \bar{q} \notin Q \vee \bar{q} \in U \rightarrow \bar{q} \in J \} \right);$$

$$\forall j \in J, U \exists \bar{J} \left(\bar{J} = \{ \bar{j} : \bar{j} \notin J \vee \bar{j} \in U \rightarrow \bar{j} \in Q \} \right).$$

При этом $\bar{Q} \cup \bar{J} = J \cup Q = Q \cup J = U$ в силу коммутативности операции объединения множеств.

Для множества A , каждый элемент которого $a \in (Q \cup J) \Leftrightarrow a \in U$, операция отрицания может быть определена следующим высказыванием:

$$\forall a \in A, U$$

$$\exists \bar{A} \left(\bar{A} = \{ \bar{a} : \bar{a} \notin A \vee \bar{a} \in U \rightarrow \bar{a} \notin (Q \cup J) \vee \bar{a} \in (\bar{Q} \cup \bar{J}) \rightarrow \bar{a} \in A \} \right).$$

Это означает, что если $a \in (Q \cup J)$, то $\bar{a} \in (\bar{Q} \cup \bar{J})$, откуда в силу равенства $\bar{Q} \cup \bar{J} = Q \cup J$ получаем, что a и \bar{a} принадлежат одному множеству, т. е. $\bar{\bar{A}} = A$ — верное равенство.

Для определения операции отрицания множества M необходимо наличие условия $J \cap Q \neq \emptyset$ (в обход противо-

речию, возникшему при определении множественности M). С учетом наличия условия $J \cap Q \neq \emptyset$ справедливо равенство $(J \cap Q = \bar{Q} \cap \bar{J}) \subset U$. Тогда каждый элемент множества M такой, что $m \in (Q \cap J) \Leftrightarrow m \in U$, и операция отрицания для M может быть определена следующим высказыванием:

$$\forall m \in M, U$$

$$\exists \bar{M} \left(\bar{M} = \left\{ \bar{m} : \bar{m} \notin M \vee \bar{m} \in U \rightarrow \bar{m} \notin (Q \cap J) \vee \bar{m} \in (\bar{Q} \cap \bar{J}) \rightarrow \bar{m} \in M \right\} \right).$$

Это означает, что если $m \in (Q \cap J)$, то $\bar{m} \in (\bar{Q} \cap \bar{J})$, откуда в силу равенства $\bar{Q} \cap \bar{J} = Q \cap J$ получаем, что m и \bar{m} принадлежат одному множеству. Тогда $\bar{M} = M$ — верное равенство.

При наличии дополнительных условий, подтверждающих доминирование множественности над неопределенностью (в этом случае множественность обозначается с индексом «d» — M_d), в результате операции отрицания M_d происходит также отрицание доминирования, т. е. $\overline{M_d} = M$ и, следовательно, $\bar{M} = M_d$.

Полученные отрицания значений множества $u \in U$ представлены в таблице 2.6. В таблице 2.6 и далее (таблицы 2.7 и 2.8) индекс «tt» означает, что значение может быть получено в соответствии с обычной таблицей истинности (truth table) двоичной логики, в то время как индекс «st» указывает на то, что для получения данного значения пришлось применять аксиомы теории множеств (set theory).

Для любого элемента гиперкодового множества возможна операция двойного отрицания, приводящая к исходному значению элемента:

$$\bar{\bar{0}} = 0, \bar{\bar{1}} = 1, \bar{\bar{A}} = A, \bar{\bar{M}} = M.$$

Таблица 2.6 — Операция отрицания значений множества $u \in U$

u	0	1	A	M	M_d
\bar{u}	1^{tt}	0^{tt}	A^{st}	M_d^{st}	M^{st}

Для обозначения **логического сложения (дизъюнкции)** элементов тетралогии (см. раздел 2.3) примем следующие обозначения: если φ_1 — элемент множества Φ_1 , а φ_2 — элемент множества Φ_2 , причем каждое из множеств Φ_1 и Φ_2 является подмножеством универсума U , то справедлива следующая запись:

$$\varphi_1 \in \Phi_1, \varphi_2 \in \Phi_2, \Phi_1, \Phi_2 \subset U \Rightarrow (\varphi_1 \vee \varphi_2) \in (\Phi_1 \cup \Phi_2) \subset U.$$

При этом для множеств $Q = \{0\}$ и $J = \{1\}$ справедливы следующие равенства:

$$Q \cup Q = Q = \{0\} \text{ и } J \cup J = J = \{1\},$$

с учетом того, что операция объединения $\Phi_1 \cup \Phi_2$ определена как множество $\Phi = \{\varphi : \varphi \in \Phi_1 \vee \varphi \in \Phi_2\}$.

В двоичной логике высказывание $\varphi_1 \vee \varphi_2$ истинно тогда и только тогда, когда хотя бы одно высказывание φ_1 или φ_2 истинно. Так, для множеств Q и J операция дизъюнкции определена следующим образом:

$$Q \vee Q = Q = \{0\}, \quad Q \vee J = J = \{1\}, \quad J \vee J = J = \{1\}.$$

В записи (2.4) множество A определено как множество, состоящее из элементов множеств Q или J , т.е. $A = \{0; 1\}$. Поэтому для множеств Q , J и A операция дизъюнкции может быть определена следующим образом:

$$A \vee Q = A = \{0, 1\}, \quad A \vee J = J = \{1\}, \quad A \vee A = A = \{0, 1\}.$$

Последнее равенство — операция дизъюнкции двух множеств A — в силу их представления как неопределенности истины или лжи отображена в самом множестве A (эквивалентно высказыванию «неопределенность или неопределенность есть неопределенность»).

В записи (2.5) множество M определено как множество, состоящее из элементов множеств Q и J . Поэтому для Q , J и M операция дизъюнкции может быть определена следующим образом:

$$M \vee Q = M, \quad M \vee J = J = \{1\}, \quad M \vee M = M.$$

Последнее равенство — операция дизъюнкции двух множеств M , в силу их представления как многозначности, т. е. одновременности появления истины и лжи, также отображена в самом множестве M (эквивалентно высказыванию «многозначность или многозначность есть многозначность»).

Операция дизъюнкции двух множеств A и M представляет собой нестандартный случай, решение которого основано на противоречии представления множественности. При наличии дополнительных условий, подтверждающих доминирование множественности над неопределенностью (в этом случае множественность обозначается с индексом «d»), возможно отображение результата операции $A \vee M_d$ на множестве M , т. е. $A \vee M_d = M$. В противном случае результатом операции $A \vee M$ являются значения множества A , т. е. $A \vee M = A$.

В тетралогике, как в частном случае многозначной логики, операция дизъюнкции может определяться различными способами. Вполне применима схема $\varphi_1 \vee \varphi_2 = \max(\varphi_1, \varphi_2)$, где $\varphi_1, \varphi_2 \in \{0, 1, A, M\}$. В иных способах, как правило, стараются сохранить совместимость с булевой алгеброй для значений операндов 0 и 1 [17, с. 19–24]. Однако остается открытым вопрос о распределении значимости (доминирования) между состояниями A и M .

Полученные значения операции дизъюнкции элементов множества $u \in U$ представлены в таблице 2.7.

Таблица 2.7 — Операции дизъюнкции значений множества $u \in U$

\vee	0	1	A	M	M_d
0	0^{tt}	1^{tt}	A^{st}	M^{st}	M_d^{st}
1	1^{tt}	1^{tt}	1^{st}	1^{st}	1^{st}
A	A^{st}	1^{st}	A^{st}	A^{st}	M_d^{st}
M	M^{st}	1^{st}	A^{st}	M^{st}	M_d^{st}
M_d	M_d^{st}	1^{st}	M_d^{st}	M_d^{st}	M_d^{st}

В таблице 2.7 индекс «d» означает, что такое значение может быть получено в случае установленного доминирования множественности над неопределенностью, т. е. как результат операции $A \vee M_d$ при $\max(A, M) = M$. В этом случае справедливы следующие равенства: $A \vee M_d = M_d$, $A \vee M = A$, а также $M \vee M_d = M_d$, (но $M \vee M = M$ и $M_d \vee M_d = M_d$).

Дизъюнкция в тетралогике может быть бинарной (два операнда), тернарной (три операнда) или n -арной (n опе-

рандов). Операция дизъюнкции тетралогии сохранила правила дизъюнкции классической алгебры логики: результат равен 0, если все операнды равны 0; результат равен 1, если хотя бы один из операндов равен 1.

Для обозначения **логического умножения (конъюнкции)** элементов тетралогии (см. раздел 2.3) примем следующие обозначения: если φ_1 — элемент множества Φ_1 , а φ_2 — элемент множества Φ_2 , причем каждое из множеств Φ_1 и Φ_2 является подмножеством универсума U , то справедлива следующая запись:

$$\varphi_1 \in \Phi_1, \varphi_2 \in \Phi_2, \Phi_1, \Phi_2 \subset U \Rightarrow (\varphi_1 \wedge \varphi_2) \in (\Phi_1 \cap \Phi_2) \subset U.$$

При этом если для множеств $Q = \{0\}$ и $J = \{1\}$ определена операция пересечения, то справедливы следующие равенства: $Q \cap Q = Q = \{0\}$ и $J \cap J = J = \{1\}$, с учетом того, что операция объединения $\Phi_1 \cap \Phi_2$ определена как множество $\Phi = \{\varphi: \varphi \in \Phi_1 \wedge \varphi \in \Phi_2\}$.

В двоичной логике высказывание $\varphi_1 \wedge \varphi_2$ истинно тогда и только тогда, когда оба высказывания φ_1 или φ_2 истинны. Так, для множеств Q и J операция дизъюнкции определена следующим образом:

$$Q \vee Q = Q = \{0\}, \quad Q \vee J = Q = \{0\}, \quad J \vee J = J = \{1\}.$$

Согласно записи (2.4) множество A определено как множество, состоящее из элементов множеств Q или J , объединенных операцией дизъюнкции. Поэтому для множеств Q , J и A операция конъюнкции может быть определена следующим образом:

$$A \wedge Q = Q = \{0\}, \quad A \wedge J = A = \{0, 1\}, \quad A \wedge A = A = \{0, 1\}.$$

Последнее равенство — операция конъюнкции двух множеств A отображена в самом множестве A (эквивалент-

но высказыванию «неопределенность и неопределенность есть неопределенность»).

В записи (2.5) множество M определено как множество, состоящее из элементов множеств Q и J , связанных между собой операцией конъюнкции. Поэтому для множеств Q , J и M операция конъюнкции может быть определена следующим образом:

$$M \wedge Q = Q = \{0\}, \quad M \wedge J = M, \quad M \wedge M = M.$$

Последнее равенство — операция конъюнкции двух множеств M также отображена в самом множестве M (эквивалентно высказыванию «многозначность и многозначность есть многозначность»).

В тетралогике, в качестве одного из способов определения операции конъюнкции, может использоваться схема $\varphi_1 \wedge \varphi_2 = \min(\varphi_1, \varphi_2)$, где $\varphi_1, \varphi_2 \in \{0, 1, A, M\}$, при которой сохраняется совместимость с булевой алгеброй для значений операндов 0 и 1. Поэтому операция конъюнкции двух множеств A и M рассматривается следующим образом: $A \wedge M_d = A$ или $A \wedge M = M$. Полученные значения операции конъюнкции элементов множества $u \in U$ представлены в таблице 2.8.

В таблице 2.8 индекс «d» показывает, что такое значение может быть получено в случае установленного доминирования множественности над неопределенностью, т. е. как результат операции $A \wedge M_d$ при $\min(A, M) = A$. В этом случае справедливы следующие равенства: $A \wedge M_d = A$, $A \wedge M = M$, а также $M \wedge M_d = M$ (но $M \wedge M = M$ и $M_d \wedge M_d = M_d$).

Конъюнкция в тетралогике также может быть бинарной, тернарной или n -арной. Операция конъюнкции тетра-

логики также сохранила правила конъюнкции классической алгебры логики: результат равен 1, если все операнды равны 1; результат равен 0, если хотя бы один из операндов равен 0.

Таблица 2.8 — Операции конъюнкции значений множества $u \in U$

\wedge	0	1	A	M	M_d
0	0^{tt}	0^{tt}	0^{st}	0^{st}	0^{st}
1	0^{tt}	1^{tt}	A^{st}	M^{st}	M_d^{st}
A	0^{st}	A^{st}	A^{st}	M^{st}	A^{st}
M	0^{st}	M^{st}	M^{st}	M^{st}	M^{st}
M_d	0^{st}	M_d^{st}	A^{st}	M^{st}	M_d^{st}

Полученные операции отрицания, дизъюнкции и конъюнкции тетралогии сохранили в себе все важнейшие равносильности алгебры классической логики.

Таким образом, для значений $x, y, z \in \{0, 1, A, M\}$ верны следующие равенства:

$$\overline{\overline{x}} = x; \quad (2.7)$$

$$x \vee y = y \vee x; \quad (2.8)$$

$$x \wedge y = y \wedge x; \quad (2.9)$$

$$(x \vee y) \vee z = x \vee (y \vee z); \quad (2.10)$$

$$(x \wedge y) \wedge z = x \wedge (y \wedge z); \quad (2.11)$$

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z); \quad (2.12)$$

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z); \quad (2.13)$$

$$\overline{x \vee y} = \overline{x} \wedge \overline{y}; \quad (2.14)$$

$$\overline{x \wedge y} = \overline{x} \vee \overline{y}; \quad (2.15)$$

$$x \vee x = x; \quad (2.16)$$

$$x \wedge x = x; \quad (2.17)$$

$$1 \wedge x = x \quad (1 \vee x = 1); \quad (2.18)$$

$$0 \vee x = x \quad (0 \wedge x = 0). \quad (2.19)$$

Равенства (2.9) и (2.11), (2.8) и (2.10) означают коммутативность и ассоциативность дизъюнкции и конъюнкции соответственно. Ассоциативность дизъюнкции и конъюнкции позволяет опускать скобки в дизъюнкциях и конъюнкциях нескольких переменных, коммутативность — располагать члены таких дизъюнкций и конъюнкций в любом порядке. Равенства (2.12) и (2.13) — это дистрибутивные (распределительные) законы дизъюнкции и конъюнкции, которые позволяют преобразовывать выражения так, чтобы операции в них выполнялись в обратном порядке (например, если в исходном выражении вначале выполнялась дизъюнкция, а потом конъюнкция, то можно получить равносильную формулу, в которой вначале выполняется конъюнкция, а потом дизъюнкция). Равенства (2.14) и (2.15) (так называемые законы де Моргана) позволяют выразить конъюнкцию через дизъюнкцию и отрицание, а дизъюнкцию — через конъюнкцию и отрицание. Эти же соотношения используются для перенесения отрицаний, применяемых к сложным высказываниям, на составляющие их простые [18, с. 35–36].

В качестве обобщения вышеизложенного «поведения» базовых логических операций тетралогике можно также предложить ряд операций, которые сведены в таблице 2.9.

Таблица 2.9 — Базовые логические операции тетралогики

x	0	0	0	0	0	1	1	1	1	A	A	M_d M	M_d M	M_d M	M_d M
y	0	0	1	A	M_d M	0	1	A	M_d M	0	1	A	M_d M	M_d M	M_d M
$x \oplus y$	0	1	A	M	M_d M_d	1	0	A	M	A	A	M M_d	M M_d	A	M
$x \downarrow y$	1	0	A	M	M_d M_d	0	0	0	0	A	0	A	M M_d	M M_d	M M_d
$x \leftrightarrow y$	1	0	A	M_d M	0	1	A	M_d M	M_d M	A	A	A	M_d M	M_d M	M_d M
$x \rightarrow y$	1	1	1	0	0	1	A	M_d M	M_d M	A	1	A	M_d M	1	M_d
$x \leftarrow y$	1	0	A	M	M_d M_d	1	1	A	1	A	A	M M_d	M_d M	M_d A	M_d
$x y$	1	1	1	1	1	0	A	M	M_d M_d	1	A	A	M M_d	M M_d	M M_d

В таблице 2.9 обозначены следующие функции: $x \oplus y$ — сумма по модулю 2 (исключающее «ИЛИ», «XOR»); $x \downarrow y$ — отрицание дизъюнкции (стрелка Пирса — функция «ИЛИ-НЕ»); $x \leftrightarrow y$ — эквиваленция; $x \rightarrow y$, $x \leftarrow y$ — импликация (следование); $x | y$ — отрицание конъюнкции (штрих Шеффера — функция «И-НЕ»).

Наличие данных логических операций позволяет получить достаточно полное представление о тетралогическом пространстве и обеспечить практическую реализацию тетралогии в компьютерных системах.

2.6. Выводы

Тетралогика представляет собой существенно расширенную по сравнению с традиционной бинарной логикой логическую систему. Это связано прежде всего с тем, что в тетралогике имеются функции, не выражимые в двузначной логике.

В данной главе рассмотрены основные операции тетралогии как одного из важнейших вариантов постбинарной логики. Показано одно из направлений в формировании математического аппарата постбинарной логики. Выявлено противоречие неявных состояний неопределенности и множественности. Предложен подход, в котором данное противоречие преодолевается путем приведения множественности к доминирующему над неопределенностью состоянию.

В данной главе проанализирована реализация 16-ти базовых унарных поразрядных операций тетралогии, объединенных в две группы — инверсную и сдвиговую, и 4 двуместные поразрядные операции тетралогии. Показано, что при этом соблюдаются следующие законы:

– двойного отрицания для реализованных логических унарных операций инверсной группы;

– n -го сдвига в n -значных логических системах для реализованных унарных операций сдвиговой группы;

– коммутативности, ассоциативности и дистрибутивности для реализованных логических бинарных операций.

В дальнейшем на базе проведенного анализа планируется развитие исследований в следующих направлениях [5]:

1. Дальнейшая разработка логических основ тетралогии и других логических систем, реализуемых на базе двумерного (раздел 1.2) и трехмерного (раздел 1.3) логических пространств.

2. Разработка теоретических основ и практическая реализация вычислений на базе тетракодов. В первую очередь это необходимо реализовать в контексте так называемых «нормированных интервальных вычислений» (см. главу 3), ориентированных на специальную форму представления тетракодов, предполагающих использование значений множественности и/или неопределенности в младших разрядах численного представления значений с ограниченной точностью [19, 20].

Глава 3

КОДО-ЛОГИЧЕСКАЯ ЭВОЛЮЦИЯ И ИНТЕРВАЛЬНЫЕ ВЫЧИСЛЕНИЯ

3.1. Развитие понятия числа¹

Аналогично тому, как бинарная логика является основой двоичной системы счисления, на базе рассмотренных логических систем (см. главу 1) могут быть построены соответствующие системы кодирования количественной информации. При этом в рассмотрение могут быть введены переменные (модифицируемые) **тетракоды** ($C_V^4 = \{1, 0, I, 0\}$) и **октокоды**: $C_V^8 = \{0, 1, \underline{M}, M, 0, I, \underline{M}, M\}$, функциональные возможности которых по отражению количественных соотношений и закономерностей реального мира существенно превышают возможности классической двоичной (бинарной) системы.

Двоичная логика и фиксированные системы счисления, сводимые к двоичной, — это характерные черты так называемой Неймановской архитектуры, получившей всеобщее распространение после опубликования в 1946 году отчета с участием Дж. Фон Неймана о проекте первой вычислительной машины с хранимой в памяти программой [2, с. 234]. Однако в условиях сетевой параллельной и распределенной инфраструктуры стали очевидны недостатки и ограничения классического кодо-логического базиса, частично компенсируемые алгоритмически-программным способом. Но, как известно, программная реализация функциональных возможностей хотя и обеспе-

¹ Разделы 3.1 и 3.2 подготовлены по материалам публикации [1].

чивает беспрецедентную универсальность современной компьютерной техники, тем не менее ведет к почти неизбежному снижению производительности на 1–2 порядка по сравнению с аппаратной поддержкой соответствующих функций. А последнее может быть оправдано и целесообразно лишь при переходе к качественно новому кодо-логическому базису.

При этом неизбежным является развитие и самого понятия числа. Несколько упрощая **общую схему развития** в контексте обобщенного кодо-логического базиса, можно утверждать следующее:

- Монокодам соответствуют различные модификации **непозиционных систем** счисления, в которых различное расположение счетных знаков могло нести не весовую нагрузку, а разного рода семантическую и интерфейсную, что детально рассмотрено в работе [3].
- Дикодам, т. е. современному этапу развития, соответствуют преимущественно **позиционные системы** счисления, позволяющие достаточно компактно представлять практически любые «точечные» количественные значения и осуществлять алгоритмические манипуляции с ними.
- Гиперкодам, в т. ч. тетракодам, соответствуют сверхпозиционные или **гиперпозиционные системы** счисления, в которых «точечное» представление количественных значений может рассматриваться лишь как частный случай. В общем случае в гиперпозиционной системе число представляет собой некоторое множество количественных значений, мощность которого и структура определяются конкретными значениями в соответствующих разрядах, т. е. на соответствующих позициях тетракодового представления числа.

В традиционной математике эволюция понятия числа также может быть привязана к различным **этапам развития обобщенного кодо-логического базиса**:

Натуральные числа явились прямым следствием перехода от непозиционных монокодов вначале к частично позиционным системам (например, римская нумерация), а затем и к полностью позиционным системам счисления. Характерной переходной формой от монокодов к дикодам в виде натуральных чисел являются так называемые фигурные числа Пифагора [4, с. 63–64]. Уходящие в седую древность истоки такой формы представления чисел хорошо характеризует высказывание известного математика XIX века Леопольда Кронекера: «Бог создал натуральные числа, все прочее — творение человека» [5, с. 24].

Рациональные числа явились одним из наиболее существенных шагов на пути «характерного для математики принципа обобщения. Переход путем обобщения от натуральных чисел к рациональным удовлетворяет одновременно и теоретические потребности в снятии ограничений, которые наложены на вычитание и деление, и вместе с тем — практические потребности в числах, пригодных для фиксации результатов измерений» [6, с. 81]. Другими словами, естественный переход к наглядному представлению натуральных чисел, выраженных в виде позиционных кодов, на числовой оси, которую в ее положительной части можно считать специфической переходной формой монокодового представления, неизбежно привел к экстраполяции оси в отрицательную область и фиксации в качестве одного из равноправных чисел нулевого значения.

Действительные числа явились прямым и естественным следствием перехода от дискретного представления числовой оси, что можно считать рудиментом монокода, к непрерывной оси, что позволило решить проблему ирра-

циональных чисел и означало окончательный переход к дикодам, позволяющий максимально использовать все преимущества новой формы представления чисел.

Комплексные числа явились, по сути, первым следствием необходимости расширения понятия числа за пределы точечного представления на континууме действительных чисел, вызванным, однако, исключительно внутренними потребностями алгебры дикоков. В 1843 году У.Р. Гамильтоном вводятся кватернионы как частный случай гиперкомплексных чисел. Кватернионы как составные числа оказались достаточно удобными для описания изометрий трёхмерного и четырёхмерного Евклидовых пространств и поэтому получили широкое распространение в механике.

Алгебраические и трансцендентные числа явились наиболее значимым в XIX веке обобщением понятий соответственно рациональных и действительных чисел [4, с. 131], впервые позволившим выйти за пределы точечного представления и ввести элементы множественности числового представления, характерные для гиперкодов. Правда, это нововведение реализовывалось лишь алгебраическим путем, не затрагивающим, фактически, кодо-логический базис. Дальнейшим развитием идей данного направления явилось формирование теории числовых полей [4, с. 145–163], которую в совокупности с интенсивно развивавшейся на протяжении всего XX века теорией множеств можно считать основой перехода к гиперкодам в XXI веке.

Гиперкоды, включая такие частные случаи как тетракоды, обеспечивающие множественность и вероятностность числовых значений, а также октокоды, позволяющие дополнительно ввести свойство случайной или контекстно-зависимой модифицируемости тетракодовых значений, можно считать новым этапом в развитии понятия числа.

Однако в данном случае переход к новому этапу обусловлен не алгебраическими потребностями, а сугубо компьютерными, в том числе связанными с потребностями компьютерного представления параметров разного рода сложных систем и процессов.

В этой связи следует отметить, что для теоретического анализа обобщенного кодо-логического базиса в целом наиболее продуктивным представляется именно **теоретико-множественный подход**. При этом монокоды могут рассматриваться в качестве простейшей алгебраической системы (модели), представленной полугруппой, обладающей только свойством ассоциативности. Такая полугруппа традиционно представляется моноидом [6, с. 142], т. е. полугруппой с единицей [7, с. 58]. Все более сложные варианты бинарных кодов (дикодов) и гиперкодов могут рассматриваться как различные варианты полугрупп, групп, колец и полей.

Усложнение структуры числа происходило параллельно по двум направлениям: кроме уже рассмотренного «алгебраического» усложнения, итог которого комплексные и гиперкомплексные числа, важным фактором стало «вычислительное» усложнение, предполагавшее масштабирование значений с целью обеспечения максимальной компактности для представления максимально широкого диапазона значений. Истоком такого усложнения можно считать взвешенные монокоды, примером чего является, в частности, древнеегипетская система счисления [3]. Но наиболее эффективно масштабирование реализовывалось на базе дикодов путем представления значений в виде мантиссы и порядка, что в вычислительной технике нашло свое выражение в специальном формате с т. н. плавающей запятой.

Неявное использование плавающей запятой появилось практически одновременно с использованием самых ранних прототипов позиционных систем счисления. Началом систематического использования плавающей запятой при вычислениях принято считать изобретение логарифмов в 1600 году и логарифмической линейки в 1630 году [2, с. 260]. Конрад Цузе уже в 1936 году в своем первом варианте вычислительной машины использовал двоичное представление с плавающей точкой, которое он называл полулогарифмической нотацией [2, с. 260]. Наиболее революционным изменением в аппаратной реализации арифметики с плавающей запятой считается принятие в конце 80-х годов соответствующего стандарта ANSI/IEEE [2, с. 262], что стимулировало практически повсеместный переход на аппаратную реализацию вычислений с плавающей запятой в современных процессорах.

Однако проблемы точности представления чисел в форматах с плавающей запятой породили еще в 60-е годы многочисленные исследования в области интервальной арифметики [2, с. 279], актуальность которой во многих приложениях, например, в моделировании сложных динамических систем, только повысилась за последние годы (см., например, [8]).

Но на базе традиционной двоичной вычислительной техники возможна лишь алгоритмическая реализация таких подходов, что резко снижает производительность соответствующих вычислений и, следовательно, ограничивает их применение. Гиперкоды позволяют перенести интервальные вычисления на уровень реализации элементарных операций и в связи с этим представляются наиболее перспективным направлением развития **интервальных подходов** в современных компьютерных вычислениях, позво-

ляющим на определенном этапе перейти на аппаратную реализацию соответствующих процессоров.

3.2. Эволюция алгоритмического базиса

Важно также отметить, что развитие кодо-логического базиса непосредственно влияет и на эволюцию интерфейсных средств, что в первую очередь наиболее наглядно проявляется в системах вычислительного моделирования. На рис. 3.1 схематически представлено развитие интерфейсной компоненты в системах вычислительного моделирования в процессе эволюции кодо-логического базиса.

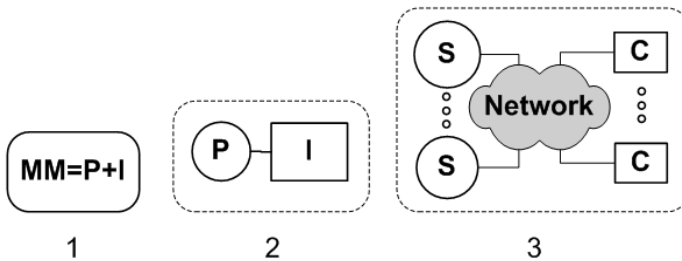


Рисунок 3.1 — Схема развития интегрированных моделирующих сред: **1** — монокодовый этап, в котором монокодовая модель (ММ) представляется как неразрывное единство процессорной части (P) и интерфейсной (I); **2** — дикодовый этап, наиболее характерный для современной вычислительной техники, привел к существенному разделению собственно вычислений (P) и интерфейса (I); **3** — гиперкодовый этап характерен для формирующейся в настоящее время сетевой среды, в которой не просто разделенными, а распределенными структурно и пространственно оказываются вычислительные серверные компоненты (S) и интерфейсные клиентские (C)

При этом следует отметить, что в монокодовых моделях интерфейсные функции выполняют непосредственно размещение, форма и варианты группировки монокодовых счетных элементов. Одним из наиболее наглядных примеров этого являются Мальтинская пластина [3] и Фестский диск [9]. Именно наглядность ранних монокодовых моделей и интегрированность в них вычислительных (в виде некоторых конечных счетных множеств элементов) и интерфейсных компонент можно считать наиболее ранними и естественными проявлениями того, что впоследствии было обозначено как интегрированные моделирующие среды [10, 11], обеспечивающие в числе прочего базовые функции когнитивного вычислительного моделирования [9].

Абстрагирование от количественных значений счетных элементов в дикодовом представлении чисел привело к существенной утрате наглядности и вообще к вырождению интерфейсных функций численного представления. В связи с этим интерфейсные компоненты стали приобретать самостоятельное значение в качестве средств повышения наглядности как в случае управления вычислительным процессом, так и при визуализации результатов и текущих значений. Характерным примером является организация современных операционных систем, в большинстве которых имеет место четкое разделение между основной частью системы и интерфейсом пользователя, который во многих случаях является опциональной частью, как, например, это имеет место в операционных системах семейства Линукс. Графическое представление численных дикодовых значений, хранимых в базе данных или полученных в ходе вычислений или моделирования, в большинстве случаев (например, при использовании столбиковых диаграмм) представляет собой по сути перевод пред-

ставления значений из дикодовой формы в монокодovou как наиболее наглядную для человеческого восприятия.

Переход к сетевой парадигме вычислений еще дальше отделяет интерфейсную часть от вычислительной. При этом последняя сосредоточивается преимущественно в серверах — специальных компьютерах, все чаще лишенных каких-либо интерфейсных средств кроме сетевого подключения. Еще одним характерным моментом сетевой парадигмы является переход от взаимно-однозначного соответствия между вычислительной частью и интерфейсной к реализации разного рода множественных соотношений, когда на базе сетевой среды может устанавливаться взаимосвязь между произвольным числом серверных и клиентских компонент (схема 3 на рис. 3.1). Указанная множественность в контексте перехода к гиперкодовому базису представляется вполне естественной и неизбежной.

В работе [12] впервые была рассмотрена эволюция алгоритмического базиса в контексте развития обобщенного кодо-логического базиса. При этом акцент был сделан на том, что объективно необходимость такой эволюции обусловлена сложностью процессов реального мира. В современных условиях можно констатировать переход от классических последовательных алгоритмов, допускающих лишь простые формы распараллеливания, к своего рода **гипералгоритмам**, основанным на гиперлогике и гиперкодах. Характерными свойствами данной категории алгоритмов являются множественная и сложная структура различных форм распараллеливания процессов, а также множественная стохастичность структуры и динамики вычислительных процессов.

Множественность в современной компьютерной инфраструктуре реализуется в основном по следующим 3-м направлениям:

1. **Распараллеливание с целью повышения производительности.** Отсчет данного направления можно вести с конца 60-х или начала 70-х годов прошлого века, а точнее с системы Иллиак-4, введенной в действие в 1972 году. В настоящее время данный вид относительно простой формы множественности процессов представлен на всех уровнях организации современной вычислительной техники: от фактической многопроцессорности современных микропроцессоров до сверхпроизводительных систем с массовым параллелизмом.

2. **Распараллеливание с целью расширения диапазона масштабируемости решаемых задач.** В качестве примера можно привести сверхбольшие базы данных, наиболее характерным примером которых является поисковая система Google, обеспечивающая на сегодня с помощью параллельной работы сотен тысяч компьютеров индексирование и перманентное ранжирование в режиме реального времени порядка 10-ти млрд. гипертекстовых документов.

3. **Распараллеливание, связанное с естественной распределенностью решаемых задач,** что характерно для большинства клиент-серверных систем. В рамках системы Google простейшая форма распределенности такого рода обусловлена, в частности, необходимостью поддержки и ранжирования в поисковых системах, локализованных по языкам и странам.

При переходе к гипералгоритмам актуальны все три перечисленных направления, но возникает необходимость в специфическом распараллеливании, обусловленная следующими 2-мя причинами:

– **логическая множественность**, ведущая преимущественно к MPMD-ветвлению (множество программ и мно-

жество данных) алгоритмов при проверке логических условий;

– **множественность числового представления**, которая ведет в основном к SPMD-распараллеливанию (одна программа и множество данных).

Множественность представления количественных значений (например, один тетракод может задавать некоторый интервал значений) естественным образом реализуется многозадачностью и многопроцессностью современного программного обеспечения, многопроцессорностью структуры современных вычислительных систем, но самое главное — глобальной массовой многопроцессностью и многопроцессорностью современной сетевой инфраструктуры.

Переход к гипералгоритмам означает фактически отказ от классических форм представления алгоритмов, таких как традиционные блок-схемы, и переход к множественности их описаний, что наиболее полно и адекватно на сегодня представлено в технологии Универсального Языка Моделирования (UML) [13].

Но в более общем виде переход к гипералгоритмам и соответствующей множественности представления может интерпретироваться как переход от преимущественно однозначного алгоритмического описания вычислительных процессов (а, следовательно, и связанных с ними сложных систем и процессов) к преимущественно многозначной форме их представления, все более характерной для современных сложных компьютерных моделей реального мира.

В наиболее радикальной постановке данная концепция (но без привязки к обобщенному кодо-логическому базису и гипералгоритмам) сформулирована А.С. Нариньяни в

ряде его работ [14, 15]. При этом он отмечает два важных различия в подходах, определяющих преимущества модельного подхода:

- **Во-первых**, традиционный алгоритм и неопределенность — это практически несовместимые понятия, а модель вполне может быть недоопределенной.

- **Во-вторых**, традиционный (не интервальный) алгоритм позволяет получать только отдельные точечные решения, а модель в общем случае определяет пространство решений.

Как видим и в этом случае определяющими являются те свойства моделей, которые являются специфическими и характерными для гиперлогики и гиперкодов.

Следует отметить, что и в технологиях программирования наметилась тенденция к созданию приложений на базе разработки их моделей (Model Driven Architecture архитектура — MDA). Концептуальной основой появления MDA стали спецификации объектно-ориентированного программирования, типичным примером которых является CORBA. А перевести данный круг идей в практическую плоскость также предполагается на базе технологий объектно-ориентированного программирования и UML [16]. В целом такого рода процессы вполне могут рассматриваться как практическая реализация тех концепций, о которых пишет А.С. Нариньяни и которые свидетельствуют о назревании своего рода кодо-логической революции в развитии компьютерных технологий, моделирования и представления знаний.

3.3. Эволюция интервальных вычислений¹

Наряду со многими достижениями в области науки и техники в середине XX века началось внедрение более широкой парадигмы понимания природы неопределенности, ранее сформированной в различных областях прикладной математики. Она базировалась на трактовке неопределенности в более широком контексте, расширяющем понятие случайности: неединственность возможных исходов, семантическая вариабельность, многокритериальность для задач оптимизации. Новые подходы к описанию неопределенности вызвали появление концепций многозначных логик и недоопределенных моделей, теорию нечетких множеств и чисел [17, 18], а также **интервальный анализ**, предметом которого является решение задач с интервальными неопределенностями и неоднозначностями в данных, возникающими как при постановке задачи, так и на промежуточных стадиях вычислений [19, 20]. В настоящее время к интервальному представлению факторов неопределенности обращено пристальное внимание инженеров и конструкторов, как к наименее ограничительному и наиболее адекватному описанию начальных условий при практической постановке инженерных задач. Такая неопределенность называется интервальной, поскольку указывает только границы возможных значений некоторой величины (либо пределы ее изменения), знания о которой являются неполными (или частичными). Интервальная неопределенность величины, выраженная своими крайними значениями, может рассматриваться как интервальный параметр, имеющий следующие особенности:

¹ Раздел 3.3 подготовлен по материалам публикации [20].

1) любая величина, имеющая интервальную неопределенность, может быть представлена только своими крайними значениями — границами возможных значений (либо пределами изменения) этой величины;

2) ширина интервала, которым выражена такая величина, является естественной мерой ее неопределенности (неоднозначности).

3) результатом арифметических операций над величинами, имеющими интервальную неопределенность, также является интервальная неопределенность.

В отличие от интервальной неопределенности некоторой величины интервальный параметр (называемый также интервалом) представляет числовые промежутки в качестве основного объекта данных и не содержит никакой дополнительной информации о самой величине. Иными словами, в контексте интервального анализа отрезок \mathbf{x} трактуется как множество возможных значений неизвестной истинной величины, т. е. как ограниченный **интервал** ее неопределенности:

$$\mathbf{x} = [x | x_1 \leq x \leq x_2], \quad (3.1)$$

задаваемый нижней (левой) и верхней (правой) границами x_1 и x_2 ($\mathbf{x} = [x_1, x_2]$), причем $x_1, x_2 \in \mathbf{R}$. В различных источниках границы интервала могут иметь другие обозначения. Например, справедливы тождественности $x_1 \equiv x_- \equiv \underline{x}$ для нижней границы, и $x_2 \equiv x_+ \equiv \bar{x}$ — для верхней границы интервала.

Предполагается, что точное значение переменной достоверно находится внутри интервала \mathbf{x} , который принадлежит множеству $\mathbf{I}(\mathbf{R})$ всех интервалов действительных чисел. При этом для $\mathbf{x} \in \mathbf{I}(\mathbf{R})$ все значения $x \in \mathbf{R}$ считаются равновероятными, следовательно, интервал \mathbf{x} не опреде-

лен никакой вероятностной мерой. Использование таких интервалов в качестве числовых аргументов составляют главную идею интервальных вычислений.

«Интервальная идея» начала интенсивно развиваться в конце XX века в тесной связи с развитием и распространением практических инженерных вычислений. Увеличение темпов развития интервальных вычислений прежде всего, связано с широким распространением ЭВМ, с чем также связано оформление интервального анализа в самостоятельную научную дисциплину [21]. Основы интервального анализа как научной дисциплины были изначально заложены теорией измерений в метрологии, где предполагается, что значение неизвестной величины x характеризуется полученной неточным измерительным прибором недостоверной величины x_0 и известной абсолютной Δ (или относительной δ) ошибкой измерения. Тогда, аналогично (3.1), границы интервала неопределенности x измеряемой величины x_0 выражаются условием

$$x = [x_1, x_2] = [x_0 - \Delta, x_0 + \Delta] = [x_0 \cdot (1 - \delta), x_0(1 + \delta)]. \quad (3.2)$$

Необходимость решения прикладных задач с интервальными параметрами в других областях науки требовала дополнительных подходов и методов. По этой причине в 70–80-х годах прошлого века наблюдается появление как многочисленных работ по интервальным арифметике и анализу, так и большого интереса специалистов к интервальному представлению значений величин. Развитие методов вычислений с интервальными значениями проходило по двум направлениям.

Первое направление — **интервальные вычисления** (англоязычные аналоги этого термина: *reliable computing*, *validated computing*, *interval computing*), теоретической базой которых выступала **интервальная арифметика**. Ин-

тервальные вычисления получили развитие на Западе (прежде всего в Германии) в качестве средства автоматического учета ошибок округления при проведении численного решения задач на ЭВМ. В рамках интервальных вычислений решаются задачи, направленные, во-первых, на анализ интервальной сходимости, устойчивости и точности существующих алгоритмов и, во-вторых, на разработку новых алгоритмов решения задач, обеспечивающих минимальную ошибку результирующего интервала [22]. На сегодняшний день основным международным информационным порталом по интервальным вычислениям является портал Interval Computations [23], поддерживаемый В.Я. Крейновичем в Университете Техаса в Эль-Пасо, США.

Второе направление — **интервальный анализ** или **интервальная математика**, развиваемое учеными бывшего СССР как теоретическая основа для решения практических задач с неопределенностью в исходных данных и параметрах моделей [24–26]. В отличие от западного направления, где применение интервальных вычислений было прежде всего направлено на алгоритмизацию и автоматизацию вычислений, главная цель применения интервального анализа состояла в нахождении области возможных значений результата с учетом структуры данных и функций, заданных в символьном виде. Информационным порталом, посвященным интервальному анализу, является портал «Интервальный анализ и его приложения» [20], куратором которого выступает С.П. Шарый в институте вычислительных технологий Сибирского отделения Российской академии наук, г. Новосибирск.

Общность двух направлений выражена одинаковыми взглядами на представление описания неопределенности, которая «упакована» в интервальной форме. Однако от-

личительной чертой данных направлений является использование различных базовых гипотез и подходов к определению результирующего интервала неопределенности. Нечеткие отличительные границы и близость «интервальной» терминологии обоих направлений привели к появлению в последнее время ряда исследований, в которых методология интервальных вычислений «по умолчанию» применяется к решению задач интервального анализа. В последующем изложении термин «интервальный анализ» будет использоваться как собирательный, включающий в себя представление «интервальных исчислений» обоих направлений.

Интервальный анализ как научное направление сформировался в основном как метод автоматического контроля ошибок округления на ЭВМ, обусловленный тем, что во многих вычислительных задачах возникла потребность не только вычисления приближенных решений, но и гарантированных оценок их близости к точным решениям. Таким образом, ценность интервальных решений заключается в том, что они в целом позволяют получать наиболее достоверные решения исходных задач, учитывающие возможные диапазоны изменения исходных и вычисляемых значений [27]. Вместе с тем для сложных задач применение интервального анализа часто дает неудовлетворительные результаты из-за чрезмерной длины получаемых интервалов. Чаще всего это происходит из-за того, что «пессимистические» оценки точности оказываются на порядок хуже, чем реально достигаемая точность результатов [28, 29]. Кроме этого, возникает естественное противоречие между относительно большим диапазоном интервальных значений, отражающим низкую точность соответствующих значений, и предельно точным заданием границ интервалов.

В то же время все чаще возникают ситуации, когда возможности классических бинарных ЭВМ (ориентированных на использование бинарной логики и арифметики только с двумя возможными логическими состояниями «0» и «1») не отвечают возрастающей сложности и масштабам современных вычислительных задач и компьютерного моделирования. Одно из возможных решений этой проблемы лежит в переходе **к постбинарным вычислениям в контексте кодо-логической эволюции**. При этом появляется возможность максимально использовать тот научно-технический задел, который накоплен в рамках интервальных вычислений, но при этом устранить или минимизировать некоторые проблемы интервального анализа, связанные, например, с противоречием между чрезмерно точным представлением границ интервальных значений и фактической неточностью тех данных, которые они представляют.

Отсчет относительно широкого распространения интервальных методов в компьютерных технологиях можно вести с первого международного симпозиума по интервальному анализу, прошедшего в Великобритании в январе 1968 года [30]. Первая монография, полностью посвященная интервальному анализу, была опубликована Р.Э. Муром в 1966 г. [31]. В этой монографии практически впервые были последовательно изложены основы нового направления в вычислительной математике, а также высказана точка зрения, что первым «интервальщиком» следует считать Архимеда, широко использовавшего в своих расчетах двусторонние приближения, в частности, для определения границ числа π — отношения длины окружности к ее диаметру. Предложенные Муром новые интересные постановки задач и поучительные применения интервальной техники оказали решающее влияние на становление и развитие нового научного направления во всем мире. На рус-

ском языке первая достаточно известная работа по интервальным вычислениям была опубликована Ю.И. Шокиным в 1981 году [32].

Началом истории интервальных вычислений традиционно считают 1931 год, когда Розалиндой Янг (Великобритания) была предложена арифметика для вычислений с множествами чисел [33, с. 260–290]. Но фактически началом современной истории интервальных вычислений является 1951 год, когда П. Двайер ввел в научный оборот специальный случай замкнутых интервалов, обусловленный необходимостью учета погрешностей в численном анализе [34]. В 1956–1958 годах в Польше были опубликованы работы Мечислава Вармуса [35, с. 253–259] и Теруо Сунаги в Японии [36], предлагавшие классическую интервальную арифметику и намечавшие ее приложения. При этом в работе Теруо Сунаги [36, с. 547–564] впервые были использованы и современные термины «интервал», «интервальный». Кроме того, Т. Сунагой были заложены основы интервального алгебраического формализма и даны весьма нетривиальные примеры применений новой техники, к примеру, в численном решении алгебраических уравнений и задачи Коши для обыкновенных дифференциальных уравнений (см. также работу [37, с. 167–188]). В 1959 году начал публиковать свои работы в области интервальных вычислений также Раймон Э. Мур, на базе чего к 1966 году была издана упомянутая выше монография [31].

В России и Советском Союзе «интервальную» историю можно отсчитывать с 20-х годов прошлого века, и связана она с именем видного русского советского математика Владимира Модестовича Брадиса, который широко известен своими математическими таблицами. В.М. Брадис предложил так называемый метод границ — способ организации вычислений, приводящий к достоверным двусто-

ронным границам точного значения вычисляемого результата, фактически аналогичный интервальной арифметике. Работая в Тверском педагогическом институте, он опубликовал целый ряд работ на эту тему [38–40].

В 1962 году в одном из первых выпусков «Сибирского математического журнала» появилась статья Л.В. Канторовича [41, с. 701–709], обозначившего эту тематику как одну из приоритетных для активно набирающей обороты вычислительной науки. В 1982 году было издано учебное пособие Т.Н. Назаренко, Л.В. Марченко по интервальным методам [42], а в 1986 г. — монография С.Л. Калмыкова, Ю.И. Шокина, З.Х. Юлдашева [43]. Обширная и подробная библиография по интервальным анализу и вычислениям имеется, в частности, в работах [44–46]. К настоящему времени разработаны различные приемы интервальных вычислений [22, 45–48] и множество пакетов прикладных программ и алгоритмических макроязыков, реализующих элементы интервального анализа на машинном уровне для нескольких типов ЭВМ [47–49]. В настоящее время готовится также соответствующий стандарт для интервальных вычислений [50].

Авторами по результатам исследований в области интервальных вычислений в 2010–2011 годах были опубликованы работы [21, 51–56], в которых рассматриваются особенности реализации интервальных вычислений в контексте постбинарного компьютеринга. В частности, в работе [51] был предложен способ кодирования границ интервала одним «тетракодовым» значением, причем разряды множественности (M) являются определяющими для позиционирования значений границ интервала на числовой оси, а разряды неопределенности (A) — уточняющими позицию этих чисел. Варьируя количеством разрядов M и A , можно добиться представления границ интервала с необ-

ходимой точностью. В докладе [55] рассмотрены способы решения интервальных полиномиальных функций с контролем точности результирующего интервала. Предложены методы замены и дифференцирования, при использовании которых отбрасываются все «побочные» значения, вызванные чрезмерным расширением интервального результата при вычислениях полиномов с интервальными коэффициентами. В докладе [56] предложен специальный формат вещественных чисел от одинарной до четвертичной точности, способный хранить в одном поле числа обе границы интервала (см. главу 4). Данный формат является одним из способов хранения данных, которые в дальнейшем планируется использовать при моделировании кардинально новых вычислительных алгоритмов и архитектур на базе существующих компьютерных систем.

Одной из причин нарастающего использования интервальных методов является то, что современные классические ЭВМ не учитывают, как правило, факт довольно низкой точности большинства исходных данных. Даже невинно выглядящее дробное число $1/10$ может породить вычислительную проблему: в большинстве случаев компьютер не может выполнять точные вычисления с этим числом [57]. В той мере, в какой точные вычислительные результаты используются для принятия критических решений, неучтенные ошибки вычислений означают повышенный риск.

Например, широко известен такой классический пример, как катастрофа с американской зенитной ракетой Patriot 25 февраля 1991 года в Дхаране (Саудовская Аравия). Он показывает, что может произойти, если существующие вычисления с плавающей точкой будут и далее некритично применяться к новым задачам. В тот день батарея ракет Patriot не смогла перехватить иракскую ракету

по официально названной следующей причине: неадекватное вычисление в формате с плавающей точкой. Дело в том, что система управления ракеты Patriot имеет внутренние системные часы, отсчитывающие время в десятых долях секунды, т. е. для перевода времени в формат с целыми секундами компьютер просто умножает данные на $1/10$. Однако, как уже упоминалось выше, на современных классических ЭВМ дробь $1/10$ не имеет точного внутреннего представления и должна быть приближена подходящей двоичной дробью. В качестве такого приближения американские разработчики взяли 24-битное двоичное число 0.00011001100110011001100, которое меньше, чем $1/10$, примерно на одну миллионную. Эта, на первый взгляд ничтожно малая, погрешность постепенно накапливалась, и после четырех дней непрерывной работы расхождение системного времени с точным достигло $1/3$ секунды, что, в конечном счете, привело к ошибке наведения в 700 метров. В результате ракета Patriot, выпущенная на перехват иракской ракеты, попала в помещение с американскими военными служащими, убив 28 человек [58].

Вышеприведенный и подобные ему (не столь катастрофичные) случаи [58] наглядно демонстрируют, что являющиеся основой современных цифровых ЭВМ числа в формате с плавающей точкой оказываются не вполне адекватными как реальному физическому миру, так и его математическим моделям, в частности, математическому понятию вещественного (действительного) числа.

Основные недостатки современного представления чисел с плавающей точкой заключаются в следующем:

– большинство чисел вещественной оси не могут быть представлены точно числами с плавающей точкой, имеющими конечную длину мантиссы, и, соответственно, свойства арифметических операций над числами с плавающей

точкой отличаются (из-за неизбежных округлений) от свойств идеальных математических операций над вещественными числами;

– число в формате с плавающей точкой не несет никакой информации о точности той величины, которую оно представляет.

Получается, что существующая модель вычислений с плавающей точкой не предназначена ни для адекватного представления исходных значений, ни для отслеживания вычислительных ошибок. В связи с этим постепенно усиливается тенденция к переходу от точечных значений к интервальным, что влечет за собой стремительное развитие интервальной арифметики.

Интервальный тип данных реализуется на современных ЭВМ, как правило, с помощью представления интервала в виде пары чисел — одного для левого конца интервала, а другого для правого. При этом существующее аппаратное обеспечение, в частности, арифметика чисел с плавающей точкой, используется без каких-либо изменений, так как корректность получающейся интервальной арифметики может быть обеспечена так называемыми направленными округлениями. Например, там, где в задачах внешнего интервального оценивания в процессе вычислений требуется округление результата, нижняя граница интервала должна округляться вниз, а верхняя граница интервала — вверх, что не противоречит равенству (3.2). Таким образом, даже неизбежные ошибки округления при вычислениях с плавающей точкой будут строго и систематически учитываются в процессе выполнения интервальной программы. В качестве примера, на рис. 3.2 показано, как иррациональные числа $\sqrt{2}$ и π в различных числовых шкалах представлены в виде различных числовых промежутков (интервалов), причем наглядно проиллюстрирова-

но изменение «ширины» этих интервалов. Интервалы чисел, представленных в вещественном формате (шкала floats) являются достаточными, так как в границы интервалов включены и ошибки округления исходных иррациональных чисел: $\sqrt{2} \in [1.25, 1.5]$; $\pi \in [3.0, 3.5]$.

Данный пример отражает основные положения интервального подхода к вычислениям на ЭВМ: исходные данные и промежуточные результаты представляются граничными значениями, над которыми и производятся все операции. При этом сами операции (прежде всего арифметические) определяются таким образом, что результат соответствующей точной операции обязательно лежит внутри вычисляемых границ.

Иллюстрацией интервального подхода могут служить следующие правила выполнения операций вещественной интервальной арифметики:

$$[x_1, x_2] + [x_3, x_4] = [x_1 + x_3, x_2 + x_4]; \quad (3.3)$$

$$[x_1, x_2] - [x_3, x_4] = [x_1 - x_4, x_2 - x_3]; \quad (3.4)$$

$$[x_1, x_2] \times [x_3, x_4] = [\min(x_1x_3, x_1x_4, x_2x_3, x_2x_4), \max(x_1x_3, x_1x_4, x_2x_3, x_2x_4)]; \quad (3.5)$$

$$[x_1, x_2] / [x_3, x_4] = \left[\min\left(\frac{x_1}{x_3}, \frac{x_1}{x_4}, \frac{x_2}{x_3}, \frac{x_2}{x_4}\right), \max\left(\frac{x_1}{x_3}, \frac{x_1}{x_4}, \frac{x_2}{x_3}, \frac{x_2}{x_4}\right) \right], \quad (3.6)$$

где $0 \notin [x_3, x_4]$.

Возникающая при вычислении границ погрешность учитывается с помощью направленных округлений: мень-

шая из вычисленных границ получается округлением до ближайшего машинного числа с недостатком, а большая — с избытком.

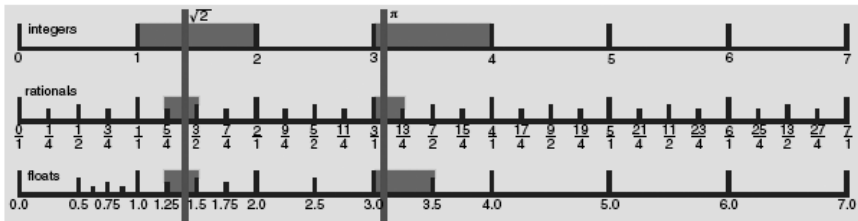


Рисунок 3.2 — Представление иррациональных чисел в виде интервалов (integers – целые числа, rationals – рациональные числа, floats – вещественные числа) [57, с. 485]

Таким образом, интервальный подход позволяет единообразным способом учесть все виды погрешностей вычислительного процесса: приближенно известные исходные данные заключаются в гарантированно содержащие точное значение границы. Погрешности округлений лишь несколько расширяют границы промежуточных результатов, а сам вычислительный метод строится так, чтобы его погрешность также включалась в вычисленные границы конечного результата [59].

3.4. Сравнение временных затрат при интервальных вычислениях в математических пакетах Scilab и Mathematica

Классическая интервальная арифметика образована интервалами

$$x = \{ x_1, x_2 \mid x_1 \geq x_2 \} \subset \mathbf{R}, \quad y = \{ y_1, y_2 \mid y_1 \geq y_2 \} \subset \mathbf{R}$$

такими что, что $x \bullet y = \{ x \bullet y \mid x \in \mathbf{x}, y \in \mathbf{y} \}$ для $\bullet \in \{ +, -, *, / \}$. Это также проиллюстрировано выражениями (3.3–3.6). При этом интервальная функция $f(\mathbf{x})$ для всех $\mathbf{x} \in I(\mathbf{R})$ называется интервальным расширением вещественной функции $f(x)$, если $f(\mathbf{x})$ монотонна по включению и $f(\mathbf{x}) = f(x)$ для всех $x \in \mathbf{R}$.

В качестве основной цели проведения исследования можно выделить **анализ точности и времени выполнения интервальных операций** в поддерживающих интервальную арифметику математических пакетах Wolfram Mathematica 5 [60] и Digiteo SciLab 5.2.2 [61] (для математического пакета SciLab необходимо наличие интервальной надстройки Int4Sci).

В процессе проведения интервальных вычислений в математических пакетах были выполнены следующие задачи:

- получение значений тестовых функций для вычислений: **гиперболический эллипсоид** (Rotated Hyper-Ellipsoid) и **функция Растригина** (Rastrigin's function) с использованием пакетов Mathematica и SciLab;
- выполнение интервальных вычислений данных функций и сравнительный анализ точности полученных результатов в данных пакетах;
- получение и сравнительный анализ временной характеристики T_n (T_n — время вычисления функции в n -количествах циклов вычислений) при вычислениях точечным (обычным) и интервальным способами.

Графики функций Rotated Hyper-Ellipsoid (3.7) и Rastrigin's function (3.8) представлены на рис. 3.3.

$$f = \sum_{i=1}^n \sum_{j=1}^i x_j^2, \quad i \in \{1, \dots, n\}, j \in \{1, \dots, i\}. \quad (3.7)$$

$$f = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i)) + 10n, \quad i \in \{1, \dots, n\}. \quad (3.8)$$

Выполнение интервальных вычислений и получение необходимых результатов работы математических пакетов проводилось на ноутбуке DELL Inspiron N5010 со следующими параметрами конфигурации: CPU Intel CoreTM i3-330M (2.13 GHz); RAM DDRII 3072MB; VGA ATI Radeon HD5470 (GDDR III 1GB); HDD SATA 320GB.

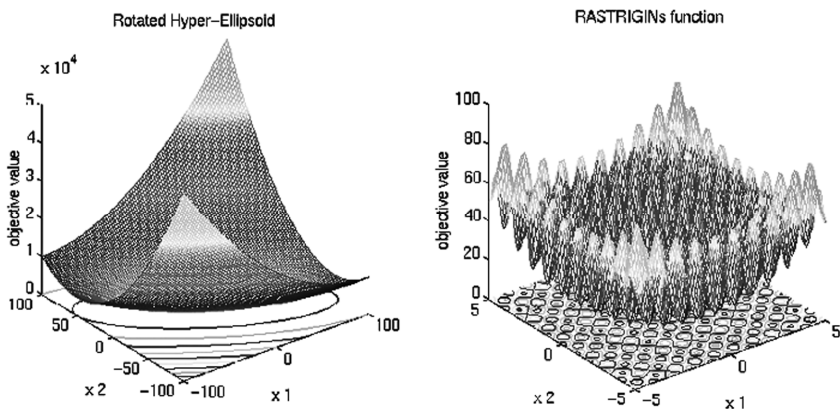


Рисунок 3.3 — Графики тестовых функций Rotated Hyper-Ellipsoid и Rastrigin's function [62]

При выполнении вычислений **интервальной функции гипер-эллипсоида** $f_1(x)$ было выбрано начальное интервальное значение $x = [1.4999, 1.5001]$, представляющее собой интервальное окружение числа 1,5. Для проведения анализа точности полученных результатов при интерваль-

ных вычислениях предварительно было проведено точечное вычисление данной функции при значении аргумента $x = 1.5$:

$$n = 5 \rightarrow f_1(x) = 33.75;$$

$$n = 25 \rightarrow f_1(x) = 731.25;$$

$$n = 100 \rightarrow f_1(x) = 11362.50;$$

$$n = 500 \rightarrow f_1(x) = 281812.50.$$

Результаты и время вычисления интервальной функции $F_1(X)$ в соответствующих математических пакетах приведены в табл. 3.1 (в скобках указаны значения Tn , полученные при выполнении точечных вычислений функции $f_1(1.5)$).

При анализе полученных результатов вычисления функции $f_1(x)$ следует отметить, что результирующие интервалы, полученные при использовании SciLab и Mathematica, во-первых, имеют небольшую ширину и, во-вторых, включают в себя полученные обычным способом значения функции $f_1(x)$. Однако время, затраченное на вычисления в SciLab, в значительной степени (на порядки) больше аналогичного времени вычисления в Mathematica, а, например, при $n = 500$ значение функции $f_1(x)$ в системе Mathematica было вычислено примерно за 1 сек., а в системе SciLab все еще не было получено по истечении 20-ти минут работы вычислительного ядра. Затраты времени при интервальных и обычных вычислениях функции Rotated Hyper-Ellipsoid представлены на рис. 3.4.

При выполнении вычислений **интервальной функции Растригина** $f_2(x)$ было также выбрано начальное интервальное значение $x = [1.4999, 1.5001]$. Как и в предыдущем случае, для проведения анализа точности полученных результатов при интервальных вычислениях предварительно

было проведено точечное вычисление данной функции при значении аргумента $x = 1.5$:

$$\begin{aligned} n = 5 &\rightarrow f_2(x) = 111.25; \\ n = 25 &\rightarrow f_2(x) = 556.25; \\ n = 100 &\rightarrow f_2(x) = 2225.00; \\ n = 500 &\rightarrow f_2(x) = 11125.00. \end{aligned}$$

Таблица 3.1 — Вычисление интервальной функции Rotated Hyper-Ellipsoid

<i>n</i>	<i>Tn, c</i>		<i>Результат f_i(x)</i>	
	<i>SciLab</i>	<i>Mathematica</i>	<i>SciLab</i>	<i>Mathematica</i>
5	0.24960 (0.0312002)	$0 \cdot 10^{-8}$ (0.01560)	33.7455001499999696, 33.7545001500000055	{ 33.745500149999955`, 33.754500150000004` }
25	4.1964269 (0.0156001)	0.0156000 ($0 \cdot 10^{-8}$)	731.152503249993288, 731.347503250010959	{ 731.1525032499974`, 731.3475032500027` }
100	64.132011 (0.0624004)	0.0624001 (0.0312001)	11360.9850504974384, 11364.0150505016991	{ 11360.985050499863`, 11364.015050500127` }
500	> 20 мин (0.3900025)	0.7332013 (0.1872003)	—	{ 281774.9262524849`, 281850.07625251607` }

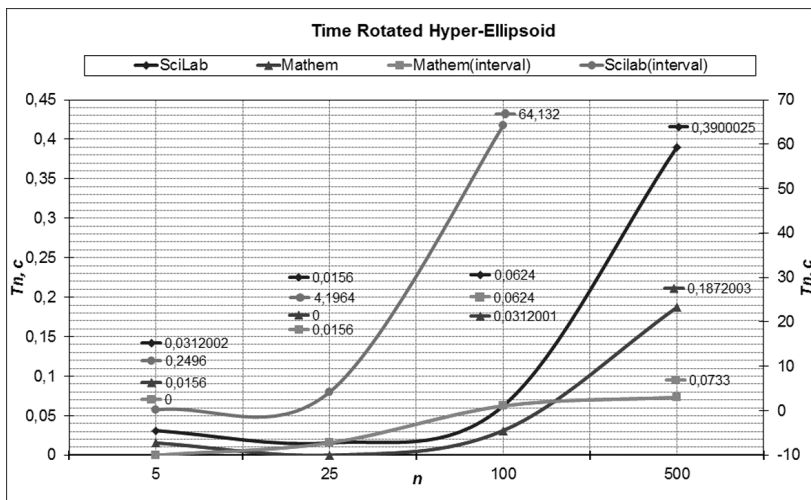


Рисунок 3.4 — Временные параметры вычисления функции Rotated Hyper-Ellipsoid

Результаты и время вычисления интервальной функции $F_2(X)$ в соответствующих математических пакетах приведены в табл. 3.2 (в скобках указаны значения T_n , полученные при выполнении точечных вычислений функции $f_2(1.5)$). Полученные результаты вычисления функции $f_2(x)$ аналогичны результатам вычисления функции $f_2(x)$: оба математических пакета получили корректные результирующие интервалы; пакет Mathematica вновь значительно опередил пакет SciLab по времени выполнения интервальных вычислений.

Для наглядной оценки полученных результатов по быстродействию интервальных вычислений на рис. 3.5 приведены временные параметры вычислений функции Растригина обычным и интервальным способами.

Опираясь на результаты проведения вычислений интервальных функций $f_1(x)$ и $f_2(x)$, можно утверждать следующее: оба математических пакета могут быть использованы в качестве инструментов для выполнения интервальных вычислений, но при этом пакет Mathematica, обладает намного более высоким быстродействием. Пакет SciLab, к сожалению, не может достаточно эффективно использоваться при проведении большого объема интервальных вычислений на персональном компьютере.

Таблица 3.2 — Вычисление интервальной функции Растригина (Rastrigin's function)

n	T_n, c		Результат $f_2(x)$	
	SciLab	Mathematica	SciLab	Mathematica
5	0.1716011 (0.0312002)	$0 \cdot 10^{-8}$ ($0 \cdot 10^{-8}$)	111.248490180395834, 111.251500050000018	{ 111.24849018039583`, 111.25150005000005` }
25	0.624 (0.0468003)	$0 \cdot 10^{-8}$ ($0 \cdot 10^{-8}$)	556.242450901977577, 556.257500250001158	{ 556.2424509019785`, 556.2575002500006` }
100	2.3244149 (0.0656001)	0.0156000 (0.01560)	2224.96980360790212, 2225.03000100000463	{ 2224.969803607908`, 2225.0300010000133` }
500	11.575274 (0.312105)	0.0624002 ($0 \cdot 10^{-8}$)	11124.8490180389672, 11125.1500050002505	{ 11124.849018039376`, 11125.1500050002` }

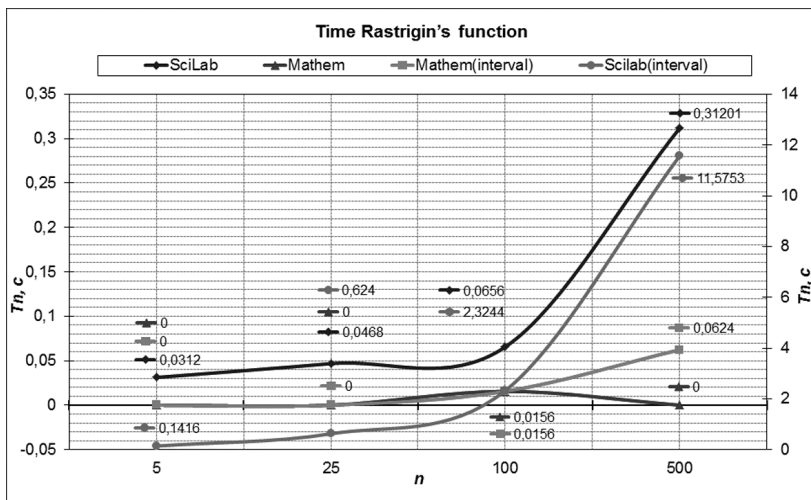


Рисунок 3.5 — Временные параметры вычисления функции Растригина (Rastrigin's function)

3.5. Верификация интервальных значений

Любое действительное число однозначно представимо на числовой оси и приближенно — в памяти электронной вычислительной машины (ЭВМ). Возникшее приближение обусловлено машинными форматами представления вещественных чисел (например, форматами чисел с плавающей запятой), которые конечным набором двоичных значений отображают бесконечное множество действительных чисел [63]. Таким образом, поскольку в ЭВМ с конечной разрядной сеткой невозможно точное представление вещественных чисел, то результат каждого достаточно сложного расчета содержит некоторую ошибку, обусловленную погрешностями округления входных данных и промежуточных результатов. Это означает, что если в вычислении учувствуют приближенно представленные (округленные)

значения, то и результат окажется приближенным. Кроме того, на погрешность, полученную при вычислении, накладывается и погрешность, возникшая при представлении полученного значения в формате с плавающей запятой (ошибка округления). Поэтому в результате накопления погрешности на каждом шаге вычисления степень достоверности результата снижается, приводя порой к совершенно неверному решению вычислительной задачи.

Указанные обстоятельства привели к тому, что в конце XX столетия в компьютерные технологии начинает внедряться более широкая парадигма понимания природы числа — любое число может быть представлено не только в виде точки на числовой оси, а и в виде некоего отрезка-интервала. Однако для сложных задач использование интервального представления числа часто дает неудовлетворительные результаты из-за чрезмерной длины получаемых интервалов, поскольку в ряде случаев оценки точности оказываются на порядок хуже, чем реально достигаемая точность результатов [29, 64]. Для иллюстрации подобной ситуации можно привести следующий пример. Рассмотрим следующую функцию $f(x)$, которая для любого значения x в пределах множества действительных чисел \mathbf{R} принимает нулевое значение:

$$f(x) = x - x = 0, \quad (3.9)$$

Однако число можно представить и в виде множества числовых значений, которое задается его крайними границами. В таком случае данное число — интервальное (определенное на множестве $\mathbf{I}(\mathbf{R})$), а его крайние границы — левая (меньшая) и правая (большая) границы интервала. Все арифметические операции над такими числами образуют класс интервальных арифметических операций (3.3–3.6), которые оперируют значениями границ интервалов-операндов. При переходе к интервальным вычислениям

функция (3.9) примет вид интервальной функции $f(x)$, где аргумент $x = [x_1, x_2]$ — интервальное число с границами x_1 (левая) и x_2 (правая), причем $x_1 \leq x_2$. Тогда

$$f(x) = x - x = [x_1, x_2] - [x_1, x_2] = [-|x_2 - x_1|, |x_1 - x_2|]. \quad (3.10)$$

Таким образом, результирующий интервал $f(x)$ имеет ненулевую длину при $x_1 \neq x_2$, и, следовательно, $x - x \neq 0$, что противоречит (3.9). Аналогично, если $f(x) = g(x) \diamond h(x)$, где \diamond — некоторая арифметическая операция, реальный выходной интервал будет заведомо уже, чем при $[g_1, g_2] \diamond [h_1, h_2]$, т. к. g и h — функции от одной и той же интервальной переменной.

Рассмотрим некоторую интервальную функцию $f(x)$, определенную на всех точках интервала $x = [x_-, x_+]$, т. е. непрерывную и дифференцируемую на рассматриваемом участке x :

$$f(x) = [f_-, f_+] = [f_{\min}(x), f_{\max}(x)], \quad (3.11)$$

где f_- , f_+ — минимум и максимум функции f на области $[x_-, x_+]$. Если все коэффициенты — обычные действительные числа, то задачу нахождения точного (верифицированного) интервала $f(x) = [f_-, f_+]$ можно решить, используя значения производной функции $f(x)$ на участке $[x_-, x_+]$.

Очевидно, что для нахождения экстремумов функции следует найти корни производной данной функции на области определения $x = [x_-, x_+]$ и сравнить значения функции в точках x_- , x_+ и нулях производной. Среди перечисленных значений выбираются максимум и минимум — это и есть границы области значений функции для данного значения x .

Для полиномов такой подход очевиден и легко реализуем. Но для остальных функций подобный подход имеет очевидные слабые места, даже несмотря на то, что математически он полностью верен. Во-первых, допущения, ко-

которые мы приняли для $f(x)$ (определенность, непрерывность, дифференцируемость) в общем случае не обязаны выполняться. Во-вторых, в случае, если все допущения выполнены и возможна выборка производной, для некоторых функций процесс взятия производной будет более трудоемким, чем само вычисление результирующего интервала. В-третьих, если производная определима, нужно еще найти ее действительные корни (иначе применение данного подхода не имеет смысла), а известны сложные уравнения, для которых получить численные решения невозможно, т. к. из-за специфических особенностей функции можно получить совершенно неверные результаты. Если организовать просчет решения производной с интервальными аргументами, то может возникнуть проблема верификации еще одного интервала, т. е. нужно повторно проводить дифференцирование (если это возможно) и т. д. В итоге, процесс верификации можно производить бесконечно, но так и не достичь приемлемого результата. Для подобных «тяжелых» функций $f(x)$ выгодна их аппроксимация на входном интервале x полиномом с необходимой точностью. Это сразу снимает весь комплекс проблем, но достоверность решения будет соответствовать достоверности аппроксимации функции.

Введение интервальных коэффициентов a_i в полиномиальную функцию $f(a_i, x)$ добавляет новые проблемы к получению верифицированных значений. Очевидно, что в таком случае любой действительный корень производной может быть представлен только в виде интервала. Поэтому функция $f(x_i)$, где x_i — корень производной, нуждается в верификации. Сделать это невозможно, так как $x_i = g(a_n, \dots, a_1)$, где $a_i, i = n, n-1, \dots, 0, n \in N$ — коэффициенты $f(x)$, но x_i — точечное значение. Таким образом,

$$f(x_i) = h(a_n, \dots, a_0, g) = h(a_n, \dots, a_0), \quad (3.12)$$

однако g представлено не функцией, что лишает смысла весь процесс верификации.

Для верификации интервальных полиномиальных функций можно использовать замену, которую надо организовать следующим образом. Необходимо выразить интервальную переменную x через единичный интервал $I = [0, 1]$:

$$x = [x_1, x_2] = [0, 1] \cdot (x_2 - x_1) + x_1 = I \cdot d + x_1,$$

где $d = x_2 - x_1$ — расстояние между границами интервала x . Таким образом, полином

$$f(a_i, x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \dots + a_1 \cdot x + a_0$$

принимает вид

$$f(a_i, x, I) = a_n \cdot x_1^n + a_{n-1} \cdot x_1^{n-1} + \dots + a_1 \cdot x_1 + a_0 + g(d, x_1, I). \quad (3.13)$$

Единичный интервал I удобен тем, что при возведении в степень он не расширяется и не сужается. Однако при возведении в степень единичного интервала справедливо выражение $I^n \neq I$, хотя границы этих интервалов и совпадают. Очевидно, что для $x \in (0, 1)$ справедливо неравенство $x^n < x$. Таким образом, для преобразованного выражения так же, как и в первом методе, необходимо применять дифференцирование. Однако выражение (3.13) лучше в плане удобства представления входного интервала и четкого разделения на действительную и интервальную части.

Для полиномов с действительными коэффициентами данный метод позволяет получить те же результаты, что и метод дифференцирования, но обладает большими вычислительными затратами за счет наличия дополнительных преобразований. Однако для полиномиальных функций с интервальными коэффициентами эти преобразования оправдывают себя, т. к. они переносят независимые интер-

вальные значения в свободный член полинома. При этом длина интервала не увеличивается в такой же мере, как в предыдущем методе, что позволяет получить более оптимизированный выходной интервал. Данный метод применим и для функций нескольких переменных $f(x_i)$, $i = 1, \dots, m$, $m \in \mathbb{N}$. В этом случае для каждой переменной вводятся свои собственные единичные интервалы, Ix_1, Ix_2, \dots, Ix_m , так как они независимы друг от друга.

В поддерживающей интервальные вычисления системе компьютерной алгебры Wolfram Mathematica 7.0 [60] была проведена проверка эффективности верификации интервала на примере полиномиальных интервальных функций второй и пятой степеней:

$$f_1(x) = x - x^2; \quad (3.14)$$

$$f_2(x) = x^5 - 5x^4 + 4x^3 + x^2 - x. \quad (3.15)$$

В интервальной математике результирующий интервал функции $f(x)$ вычисляется следующим образом:

$$f(x) = [\min(f(x)), \max(f(x))] \text{ при } x \in x. \quad (3.16)$$

Однако при таких вычислениях наблюдается неконтролируемое увеличение длины интервала. Полученные результаты при вычислении функции $f_1(x)$ при различных входных значениях приведены в табл. 3.3. Получение результирующего интервала $f_2(x)$ при входном интервале $x = [-0.1, 0.1]$:

– средствами интервальной математики:

$$[-0.10451, 0.11401];$$

– с помощью метода дифференцирования:

$$[-0.08649, 0.10549].$$

Таблица 3.3 — Результаты вычисления функции $f_1(x)$ с указанием ширины d результирующих интервалов

i	Входной интервал x_i	Получение результирующего интервала $f_1(x)$			
		средствами интервальной математики		с помощью метода дифференцирования	
1	[0.7, 1.4]	[-1.26, 0.91]	$d_1 = 2.17$	[-0.56, 0.21]	$d_1 = 0.77$
2	[0.3, 0.6]	[-0.06, 0.51]	$d_2 = 0.62$	[0.21, 0.25]	$d_2 = 0.04$
3	[-0.4, 0.2]	[-0.56, 0.2]	$d_3 = 0.76$	[-0.56, 0.16]	$d_3 = 0.72$

Таким образом, при решении полиномов (3.14) и (3.15) были получены интервалы наименьшей длины, что, безусловно, доказывает эффективность предложенного метода. В рамках внедрения данного метода в интервальную математику разработана программная модель *math* [65], способная выполнять вычисление полиномиальных интервальных функций. В качестве вычислительного ядра модели выбрано ядро математического пакета Wolfram Mathematica, которое поддерживает интервальные вычисления. Применение данного программного продукта как самостоятельного вычислительного средства обусловлено, во-первых, возможностью оптимизации промежуточных результатов на различных этапах сложных интервальных вычислений, и, во-вторых, способностью провести полный цикл вычислений с гарантированием точности и максимальным «сужением» границ результирующих интервалов.

3.6. Постбинарные методы реализации интервальных вычислений в компьютерном моделировании

Традиционно аппаратное обеспечение компьютеров поддерживает две числовые системы: целые числа и числа с плавающей точкой. Целочисленная арифметика оперирует конечным подмножеством множества целых чисел и позволяет безошибочно осуществлять адресные вычисления, компиляцию и другие формы трансляции, а также реализовать различные алгоритмы типа поиска и сортировки [66].

Произвольное вещественное число представляется бесконечной систематической (например, десятичной или двоичной) дробью. На практике в научных и инженерных вычислениях вещественные числа приходится представлять в компьютере конечными дробями, чаще всего числами с плавающей точкой. Арифметика чисел с плавающей точкой поддерживается аппаратным обеспечением компьютеров, поэтому выполняется очень быстро, однако каждая операция с вещественным числом может вносить погрешность.

Современные ЭВМ практически полностью базируются на бинарной логике и арифметике, обеспечивающих до недавнего времени практически все потребности компьютерных вычислений [67]. Однако в 90-е годы прошлого века произошли качественные изменения как в развитии логических основ, так и в области компьютерных технологий, которые обусловили актуальность соответствующих изменений как в кодо-логическом [68], так и в алгоритмическом [12] базисе современных компьютерных технологий. Суть данных изменений может быть сведена к переходу от преобладания фиксированной точечной определенности к эволюционирующей множественности

и неопределенности, а также к рассмотрению новых перспектив использования многомерного кодо-логического базиса в вычислительном моделировании и представлении знаний.

При этом возможны различные варианты постбинарного представления численных значений. Например, одной из наиболее универсальных форм компьютерного представления тетракодов является формат с плавающей запятой, в котором представленные соответствующим образом мантисса и порядок должны быть дополнены значениями смещений, представленных обычными бинарными кодами, обеспечивающими совместимость с традиционными двоичными операциями машинной арифметики. Другими словами, если обычные натуральные числа представляются в компьютерных системах в формате $m \cdot 2^e$, где m — мантисса, e — порядок, кодируемые традиционно в виде бинарных кодов, то для универсального представления тетракодов целесообразен формат $(d + m') \cdot 2^{(k+e')}$, где m' — нормализованная мантисса и e' — порядок, представленные соответственно в виде целочисленных тетракодов, а d и k — смещения, представленные в виде классического бинарного кода. При этом смещения, с одной стороны, позволяют достаточно просто решить проблему совместимости гиперкодового представления с традиционной машинной арифметикой, а, с другой стороны, открывают возможности для дальнейшего усложнения структуры и многофункциональности численного представления путем введения элементов гиперкодового кодирования в представление смещений.

Рассмотрим представление десятичной дроби в виде постбинарного интервального числа. В качестве исходного выберем трансцендентное число e , являющееся основанием натурального логарифма. Числовое значение этой ма-

тематической константы возьмем с точностью до 18-го знака после запятой:

$$e_0 = +2,718281828459045235.$$

Естественно, такое значение невозможно точно представить в традиционных, используемых современными процессорами, форматах чисел с плавающей запятой. Так, для формата single (32 бит IEEE 754) заданное число e представляется следующими двоичными полями (S — знак числа, E — смещенная экспонента, M — остаток от мантиссы):

$$S = 0; E = 10000000_2; M = 01011011111100001010100_2.$$

Точное десятичное значение, полученное из данных полей, равно $e_{32b} = +2,71828174591064453125$ (число отображается в окне Windows с еще большим округлением: $F_{32b} = 2,718282$) и имеет значимую погрешность представления для заданной точности:

$$\Delta_1 = e_0 - e_{32b} = +8,254840070375 \cdot 10^{-8}.$$

Заданное число e также невозможно точно представить и в формате double (64 бит IEEE 754):

$$S = 0; E = 10000000000_2;$$

$$M = 0101101111110000101010001011000101000101011101101001_2.$$

$$e_{64b} =$$

$$= +2,718281828459045090795598298427648842334747314453125;$$

$$F_{64b} = 2,71828182845905;$$

$$\Delta_2 = e_0 - e_{64b} =$$

$$= +1,44204401701572351157665252685546875 \cdot 10^{-16}.$$

Рассмотрим интервалы e_1 и e_2 , границы которых соответствуют записи (3.1), а их численное значение получено из (3.2):

$$\begin{aligned}
e_1 &= [e_0 - \Delta_1, e_0 + \Delta_1] = \\
&= [+2,718281755910644531, \\
&\quad +2,718281911007445939].
\end{aligned} \tag{3.17}$$

$$\begin{aligned}
e_2 &= [e_0 - \Delta_2; e_0 + \Delta_2] = \\
&= [+2,718281828459045091, \\
&\quad +2,718281828459045379].
\end{aligned} \tag{3.18}$$

В (3.17) и (3.18) соблюдены соотношения $e_0 \in e_1$ и $e_0 \in e_2$. Представим e_1 и e_2 в форматах single и double с последующим извлечением точных десятичных значений (выделены и подчеркнуты первые отличающиеся разряды дробной части числа):

$$\begin{aligned}
e_{32b} &= [+2,718281\underline{7}4591064453125, \\
&\quad +2,718281\underline{9}843292236328125]; \\
e_{64b} &= \\
&= [+2,718281828459045\underline{0}90795598298427648842334747314453125, \\
&\quad +2,718281828459045\underline{5}3488480814849026501178741455078125].
\end{aligned}$$

Полученные интервалы также содержат заданное значение e_0 : $e_0 \in e_{32b}$ и $e_0 \in e_{64b}$. Для границ интервалов (3.17) и (3.18) также возможен способ кодирования с помощью одного «тетракодового» слова, как это было предложено в работе [26]. Так, интервалы (3.17) и (3.18) можно в итоге представить соответствующими наборами тетракодов:

$$\begin{aligned}
e_{T1} &= \underbrace{0}_S \underbrace{1000000}_E \underbrace{0101101111110000101010M}_M; \\
e_{T2} &= \underbrace{0}_S \underbrace{10000000000}_E \\
&\quad \underbrace{01011011111100001010100010110001010001010111011010MA}_M.
\end{aligned}$$

Такого набора достаточно для получения соответствующей пары двоичных полей при всех $M = 0$ для левой границы и $M = 1$ для правой границы (для обоих границ разряд A принимает случайное значение из набора $\{0; 1\}$) интервалов, точные десятичные значения которых включают в себя все значения интервалов e_{32b} и e_{64b} : $e_{T1} \supseteq e_{32b}$ и $e_{T2} \supseteq e_{64b}$ (строгое равенство достигается при отсутствии значений неопределенности A в тетракоде).

Однако точные десятичные значения могут быть получены из форматов `single` и `double` только с помощью «ручного» просчета. Для машинных вычислений нужно ориентироваться на относительную точность форматов чисел с плавающей запятой, которая составляет 7–8 десятичных цифр для формата `single` и 15–16 десятичных цифр для формата `double`. В данном случае значение (или несколько значений) множественности переходит в старшие разряды, а младшие разряды заполняются значениями неопределенности. Например, границы интервалов (3.17) и (3.18) можно выразить следующим образом

$$\begin{aligned}
 e_{T1} &= \underbrace{0}_{S} \underbrace{1000000}_{E} \underbrace{0101101111MMAAAAAAAAAAAA}_{M}; \\
 e_{T2} &= \underbrace{0}_{S} \underbrace{100000000000}_{E} \\
 &\quad \underbrace{01011011111100001010MMMMAAAAA...A}_{M}.
 \end{aligned}$$

Таким образом, с помощью тетракода можно закодировать границы интервала произвольной точности в одном поле данных. Однако следует учесть, что данный способ эффективен при незначительной ширине интервала $\text{wid}(x) = x_2 - x_1$.

Указанная для e_{T1} и e_{T2} концепция перехода к группам разрядов множественности M и неопределенности A подробно рассмотрена в следующей главе (раздел 4.3).

3.7. Выводы

В 90-е годы прошлого века произошли качественные изменения как в развитии логических основ, так и в области компьютерных технологий, которые обусловили актуальность соответствующих изменений как в кодо-логическом, так и в алгоритмическом базисе современных компьютерных технологий. Суть данных изменений может быть сведена к переходу от преобладания фиксированной точечной определенности к эволюционирующей множественности и неопределенности в представлении чисел, что является закономерным этапом в эволюции понятия числа как постепенно усложняющейся формы представления количественной информации.

Вполне закономерным в этом контексте явилось также появление и развитие интервальных вычислений, которые в настоящее время реализуются преимущественно программным путем, требуют существенных дополнительных вычислительных затрат и в связи с этим получили пока довольно ограниченное применение. В рамках постбинарного компьютинга появляется возможность реализовать основные идеи интервальных вычислений на уровне форматов компьютерного представления чисел и аппаратно реализуемых операций с ними, что позволит существенно повысить производительность интервальных вычислений и расширить область их применения.

Глава 4

ТЕТРАКОДЫ И ПОСТБИНАРНЫЕ ВЫЧИСЛЕНИЯ

4.1. От бинарного к постбинарному компьютерингу

Первая волна интенсивного развития цифровых технологий, отсчет которой можно вести примерно с 1945 года, прошла почти исключительно на базе бинарной логики и арифметики. Однако уже в 90-е годы, когда начался массовый переход к технологиям параллельных вычислений, ограниченность бинарного кодо-логического базиса стала проявляться особенно остро, в связи с чем актуализировались поиски дальнейших, постбинарных, путей развития компьютерных технологий.

Само понятие «компьютеринг» начало использоваться еще в эпоху механических вычислительных устройств, означая первоначально просто процесс реализации каких-либо вычислений. Однако к настоящему времени смысл этого термина существенно расширился, что нашло отражение в следующем современном определении: «Под компьютерингом обычно понимается деятельность, направленная на использование и разработку компьютерных технологий, компьютерной техники и программного обеспечения. Это связанная с компьютерами часть информационных технологий. С этим понятием тесно связаны компьютерные науки, направленные на изучение теоретических и практических основ вычислений и информатики, а также на их реализацию в компьютерных системах» [1].

Такое расширенное понимание компьютеринга сложилось уже в бинарную эпоху, в связи с чем под этим термином традиционно предполагается **бинарный компьютеринг**, в основе которого лежит использование двоичной логики и двоичной системы счисления. Но современное определение компьютеринга вполне применимо и для предыдущего добинарного периода, для чего следует признать целесообразным использование термина **прабинарный (пребинарный) компьютеринг**. Естественно, мы не можем говорить в этом случае, например, о каком-либо программном обеспечении в современном его понимании. Но алгоритмическая составляющая прабинарного компьютеринга заслуживает самого серьезного изучения наряду с логической и вычислительной составляющими, представленными соответственно монологикой и монокодами.

Целесообразным и оправданным в настоящее время следует также признать использование термина **постбинарный компьютеринг**, включающего в себя все, что выходит за рамки двоичной логики и систем счисления, однозначно сводимых к двоичной (точечной и однозначной) системе представления количественной информации.

В целом можно констатировать, что возможности продуктивного системного подхода к исследованию вопросов постбинарного компьютеринга могут быть обеспечены различными способами, следствием чего является естественное многообразие соответствующих исследований. Целесообразно, в частности, назвать такие перспективные направления, как Grid-технологии [2], интервальные вычисления [3], гипервычисления, квантовый и молекулярный компьютеринг [4].

В рамках кодо-логического подхода, развиваемого авторами данной монографии, **важнейшими предпосылка-**

ми для системного исследования проблематики пост-бинарного компьютеринга явились следующие моменты:

Во-первых, введение в научный оборот понятия «кодо-логический базис» (с доопределением «расширенный» или «обобщенный»), предполагающего рассмотрение эволюции логических и арифметических основ компьютерных технологий в неразрывном единстве и в достаточно широкой исторической перспективе и ретроспективе [5–7].

Во-вторых, выявление и анализ как целостного явления добинарного (пребинарного, прабинарного) кодо-логического базиса, основными составляющими которого являются монологика и различные эволюционирующие формы монокодов (см. раздел 1.4, а также [6]). Проведенные в связи с этим исследования позволили в итоге инициировать такое новое междисциплинарное научное направление как археомоделирование, основной задачей которого является выявление и изучение различных средств и методов добинарного вычислительного моделирования [15]. В целом появились основания говорить о целой эпохе прабинарного компьютеринга, предшествовавшей бинарному этапу в развитии вычислительных средств. Исследование закономерностей добинарной эволюции и перехода к бинарному компьютерингу позволило более системно и продуктивно подойти к анализу вопросов перехода к постбинарным вычислениям.

В-третьих, введение в научный оборот таких взаимосвязанных понятий как «тетралогика» и «тетракоды» (см. раздел 1.2), что позволило впервые одновременно выйти за пределы как одномерного пространства двоичной логики, так и точечного (бинарного) представления количественной информации, что явилось решающим шагом к последующим исследованиям постбинарного компьютеринга в контексте кодо-логической эволюции.

Логические, алгоритмические и прочие предпосылки перехода к постбинарному компьютерингу начали складываться уже в первой половине XX века в результате интенсивного развития средств и методов бинарного компьютеринга во второй половине прошлого века, но только на рубеже тысячелетий (в первую очередь благодаря тотальному переходу к параллельным вычислениям и формированию тесно взаимосвязанной глобальной компьютерной инфраструктуры) начала проявляться настоятельная необходимость преодоления многочисленных ограничений традиционного бинарного кодо-логического базиса, ориентированного преимущественно на последовательную (так называемую фон неймановскую) вычислительную архитектуру.

В связи с этим назрела необходимость комплексного исследования всей совокупности вопросов, связанных с переходом к постбинарному компьютерингу, основанному на использовании различных форм гиперлогики и гиперкодов, которые могут рассматриваться как обобщение тетралогики и тетракодов применительно к многомерным логическим пространствам.

В целом в результате проведенных к настоящему времени исследований можно сформулировать следующие закономерности развития кодо-логического базиса и основанных на нем форм компьютеринга:

Приоритет логической эволюции. Эволюция логической составляющей является первичной и определяет наиболее фундаментальные изменения в развитии различных форм представления количественной информации и основанных на них средств и методов компьютеринга. Именно с ранних изменений в логических основах начинается постепенный переход к новому кодо-логическому базису: бинарная логика сформировалась в основных чертах

еще в эпоху античности, задолго до перехода к сугубо позиционным системам счисления, в основе которых лежит использование символа нуля, а троичная логика впервые была предложена в 20-х годах прошлого века, когда о постбинарном компьютеринге еще не могло быть и речи. При этом основное, наиболее значимое для эволюции средств и методов компьютеринга, развитие логики идет по пути возрастания размерности и полноты использования логического пространства: нульмерного для монологики, одномерного для бинарной логики и многомерного — для постбинарной. При этом именно свойства единичного логического значения определяют потенциальную информационную емкость и адекватность одного разряда соответствующего числового кода.

Разрядная эволюция кода. Повышение информативности и адекватности кодирования количественной информации за счет эволюции соответствующих показателей одного разряда кода. В простейшем монокоде каждый разряд представлен целым единичным значением, сумма которых и определяет итоговое значение кода, являющееся исключительно натуральным числом. Введение элементов позиционности и многозначности для отдельных разрядов кода (как, например, это имеет место для римских чисел) не меняет существенно сути монокода, лишь несколько увеличивая информативность отдельных разрядов. Глубинные изменения происходят только при переходе к современным позиционным системам счисления, однозначно сводимым к простейшему двоичному представлению, характерному для современных компьютерных технологий.

На сегодняшний день аппаратное обеспечение компьютеров ориентировано на операции с числами, представленными всего в двух форматах: целые числа и числа с плавающей запятой. Целочисленная арифметика опери-

рует конечным подмножеством множества целых чисел. Если аппаратная часть исправна, а программы не содержат ошибок, то целочисленная арифметика и программные надстройки над ней работают без погрешностей. Например, на основе целочисленной арифметики можно построить арифметику систематических дробей произвольной разрядности и обыкновенных дробей. Для точного представления результатов целочисленных арифметических операций современный компьютер имеет возможность сохранять необходимое количество разрядов для точного представления достаточно больших целых чисел. Но при решении большинства практических задач главным недостатком целочисленной арифметики является крайняя ограниченность диапазона представления чисел, что определило переход на использование в современных компьютерных технологиях преимущественно вещественных чисел.

Произвольное вещественное число представляется бесконечной десятичной или двоичной дробью. На практике в научных и инженерных вычислениях вещественные числа приходится представлять в компьютере конечными дробями, чаще всего числами с плавающей запятой.

В настоящее время общепринятым средством для выполнения научных и инженерных вычислений на компьютерах является арифметика чисел, представленных в формате с плавающей запятой. Данный формат разработан ассоциацией IEEE и используется для представления действительных чисел в двоичном коде [8]. На большинстве компьютеров каждая отдельная операция с плавающей точкой обеспечивает результат максимальной точности в том смысле, что получаемый округленный результат отличается от точного вещественного значения не более чем на единицу последнего разряда мантиссы. При этом

арифметика чисел с плавающей запятой поддерживается аппаратным обеспечением современных компьютеров и поэтому выполняется очень быстро. Но каждая операция с плавающей запятой может вносить погрешность, поскольку числа, представленные в плавающем формате, представляют конечное множество, на котором отображается бесконечное множество вещественных чисел. Следовательно, результат нескольких последовательных операций при определенных условиях может не содержать уже ни одной верной цифры (как в примере Румпа, описанном в разделе 4.4). Поскольку современные суперкомпьютеры преодолели «петафлопсный» рубеж [9], то достоверность вычисляемых результатов становится предметом особого внимания.

Стремительное развитие вычислительной техники явилось стимулом к возрастанию масштабов как самих численных задач, так и требований к надежности получаемых решений. В последние годы для многих задач и приложений численного анализа были разработаны методы решения, обеспечивающие необходимую арифметическую и вычислительную надежность, что дает возможность, во-первых, получения высокой точности результатов вычислений и, во-вторых, автоматической проверки их корректности. Поиск возможностей, направленных на повышение надежности вычислений, положил начало развитию обширной области исследований — интервальных вычислений (см. главу 3).

Приведем простой пример (аналогичный представленному в работе [10]). Пусть даны два целочисленных (!!)-вектора x и y :

$$\begin{aligned}x &= (10^{15}, 1500, -10^{18}, 10^{20}, 2, -10^{15}), \\y &= (10^{15}, 3, 10^{12}, 10^{13}, 222, 10^{18}).\end{aligned}$$

Обозначим скалярное произведение x и y через $x \times y$, при выполнении которого соответственные элементы векторов перемножаются и эти произведения складываются. В точной целочисленной арифметике имеем:

$$x \times y = 10^{30} + 4500 - 10^{30} + 10^{33} + 444 - 10^{33} = 4944.$$

Ниже приведен пример реализации данного целочисленного выражения в системе компьютерной алгебры Mathematica:

```
In[5] := 1030 + 4500 - 1030 + 1033 + 444 - 1033
```

```
Out[5] = 4944
```

Однако арифметика чисел с плавающей запятой на любом современном компьютере (включая те компьютеры, на которых арифметика реализована в соответствии с обновленным в 2008 году стандартом формата представления чисел с плавающей запятой IEEE 754-2008) выдаст для такого скалярного произведения нулевое значение. Причина этого в столь большой разнице порядков слагаемых, что обычное представление чисел с плавающей запятой не позволяет корректно выполнить вычисление. Ниже приведен пример реализации данного выражения в системе компьютерной алгебры Mathematica при представлении элементов векторов x и y вещественными числами:

```
In[6] := 10.30 + 4500. - 10.30 + 10.33 + 444. - 10.33
```

```
Out[6] = 0.
```

Эта катастрофическая погрешность возникает несмотря на то, что данные (в данном случае элементы векторов) используют менее 5% диапазона значений порядка, доступного на большинстве компьютеров!

Ниже показаны результаты символьного (признак знака « \rightarrow ») и численного (признак знака « \Rightarrow ») процессоров

математического пакета Mathcad Professional при вычислении данного примера:

$$10^{30} + 4500 - 10^{30} + 10^{33} + 444 - 10^{33} \rightarrow 4944 =$$

$$= 4.944 \times 10^3$$

$$10^{30} + 4500 - 10^{30} + 10^{33} + 444 - 10^{33} = 0$$

$$10^{30} + 4500 - 10^{30} + 10^{33} + 444 - 10^{33} \rightarrow 0 = 0$$

В последнем варианте вычислений (элементы векторов — вещественные числа) символьный и численный процессоры Mathcad оказались бессильны.

Состояние современной компьютерной арифметики можно существенно улучшить, т. е. сделать ее более «интеллектуальной», путем новых средств и методов аппаратно-программной поддержки. Например, в работе [10, с. 9–10] высказана возможность создания в арифметическом устройстве (или моделирования с помощью программных средств) специального регистра с фиксированной точкой, покрывающего своей разрядностью весь диапазон чисел с плавающей точкой. Причем возникающая при этом потеря скорости вычислений вполне компенсируется существенным возрастанием их надежности. Однако для обеспечения автоматического контроля погрешностей в существующих компьютерных архитектурах машинную арифметику необходимо как минимум обеспечить возможностью направленного округления результата, т. е. округления до ближайшего машинного числа с недостатком или избытком. Данная концепция представления действительных чисел, а также арифметические операции с составленными из них векторами и матрицами позволяют построить машинную интервальную арифметику [11], в которой интервалы представляются в виде компьютерных континуальных объектов и открывают для численного анализа совершенно новую перспективу. В памяти ЭВМ интервал записывается

парой чисел в формате с плавающей точкой, определяемых как границы этого интервала. Такой интервал фиксирует все множество вещественных чисел, заключенных между двумя хранимыми в ЭВМ машинными числами. Арифметические операции с такими парами чисел (т. е. с границами интервалов-операндов) являются составляющими арифметических операций над интервалами, результатом которых также является пара машинных чисел, представляющая границы результирующего интервала.

Подмножество x множества всех действительных (вещественных) чисел \mathbf{R} , такое что

$$\mathbf{x} = [x_1; x_2] = \{r \mid x_1 \leq r \leq x_2, \ x_1, x_2 \in \mathbf{R}\}, \quad (4.1)$$

называют **замкнутым вещественным интервалом** (или, аналогично (3.1), просто интервалом), причем каждый такой интервал является элементом множества всех замкнутых вещественных интервалов $\mathbf{I}(\mathbf{R})$. Однако всякое вещественное число a из \mathbf{R} также является элементом $\mathbf{I}(\mathbf{R})$, поскольку может быть представлено в виде интервала $[a; a]$, который называют **вырожденным (точечным)**. Интервальная математика предполагает выполнение унарных и бинарных операций над интервалами. Если $\bullet \in \{+, -, \cdot, : \}$ — бинарная операция на множестве \mathbf{R} и $\mathbf{x}, \mathbf{y} \in \mathbf{I}(\mathbf{R})$, то (см. раздел 3.5)

$$\mathbf{x} \bullet \mathbf{y} = \{x \bullet y \mid x \in \mathbf{x}, y \in \mathbf{y}\}. \quad (4.2)$$

Выражение (4.2) определяет бинарную операцию на множестве $\mathbf{I}(\mathbf{R})$. Если $s(x)$ — непрерывная унарная операция на \mathbf{R} , то выражение

$$s(\mathbf{X}) = [\min_{x \in \mathbf{X}} s(x); \max_{x \in \mathbf{X}} s(x)] \quad (4.3)$$

определяет соответствующую ей бинарную операцию на $\mathbf{I}(\mathbf{R})$.

Руководствуясь (3.2), при вычислении нижней (левой, минимальной) границы интервала-результата часто прибегают к округлению с недостатком, а при вычислении верхней (правой, максимальной) границы — с избытком. Таким образом, в результате выполнения интервальных операций получается результирующий интервал, который гарантированно содержит все результаты применения данной арифметической операции к любым парам чисел, взятых из любого интервала-операнда. Однако стоит заметить, что интервальное вычисление арифметического выражения требует примерно вдвое большего числа операций по сравнению с вычислением такого же выражения в стандартной арифметике с плавающей точкой. Очевидно, что границы возможных значений результирующего интервала и его ширина является естественной мерой неопределенности (см. раздел 3.4).

В рамках разделов этой главы (в частности, 4.3 и 4.4) рассматривается идея представления данных с помощью тетракодов (см. раздел 1.7), в частности при представлении вещественных чисел в виде интервалов (см. раздел 3.3). Идея такого представления интервала заключается в кодировании его границ одним «тетракодовым» числом, что способствует, во-первых, возможности представления этих границ в памяти ЭВМ не двумя, а одним машинным кодом, и, во-вторых, созданию предпосылок для разработки концептуально нового математического аппарата, работающего с «тетракодовыми» интервалами».

4.2. Особенности постбинарного кодирования на примере интервального представления результатов вычислений по формуле Бэйли-Боруэйна-Плаффа

В главе 3 упоминалось, что интервальная арифметика является основным инструментом интервального анализа, который в качестве математической дисциплины изучает задачи с интервальными неопределенностями в данных, а также методы их решения. Простейшей и наиболее распространенной ситуацией описания неизвестной точно величины является задание множества ее возможных значений [12]. Например, величина φ , представляющая неопределенность значений на числовой прямой между 1 и 5, может быть задана как $\varphi \in \{1, 2, 3, 4, 5\}$ или $1 \leq \varphi \leq 5$ или, в более обобщенном виде, как $\varphi \in R$.

Потребность такого представления неопределенности возникает во множестве самых разнообразных ситуаций, в частности при представлении в машинных кодах бесконечных десятичных дробей. Например, число π — бесконечную десятичную дробь можно представить как некоторое приближение вещественного числа, имеющее конечное число десятичных или двоичных знаков, например $\pi \approx 3,14159$. Однако при оперировании таким приближением числа π уже в начале вычислений допускается неизбежная ошибка, поскольку неизвестна информация о том, каким именно приближением, с недостатком или с избытком, является указываемое значение. Очевидно, что при более корректном представлении данных нужно явным образом указывать границу этой ошибки, например, путем уточнения того, что ошибка представления не превосходит половины единицы последнего разряда. Существует более

предпочтительный способ указания ошибки представления, который состоит в том, чтобы предоставить наиболее узкие точечно-представимые границы (нижнюю и верхнюю) для задания какой-либо неоднозначной величины. Так, для числа π :

$$\pi \in [3.14159, 3.14160].$$

Частым случаем ошибок представления, имеющим инженерный характер, являются ошибки перевода из одной системы счисления в другую. Многие конечные десятичные дроби не имеют точного конечного представления среди двоичных чисел, с которыми оперируют современные компьютерные системы. Но при введении подобных чисел в ЭВМ они заменяются некоторым конечным рядом по степеням двойки, что, как следствие, вносит ошибку в машинное представление числа.

Подход к решению различных типов задач, при котором использование машинной интервальной арифметики уменьшает (или вовсе устраняет) проблемы соотнесения вычисленного приближенного результата с истинным (абстрактным) решением, получил название «интервальный», поскольку интервал, представляемый парой рациональных чисел-границ, является простейшим видом конечно-представимого множества, локализирующего простейший абстрактный объект — вещественное число. Основная идея интервального подхода состоит в том, что вещественное число представляется в памяти вычислительной машины не одним, а двумя машинными числами — нижней и верхней оценкой, образующими интервальное число. Таким образом, в рамках интервального подхода исходные данные и промежуточные результаты представляются граничными значениями, над которыми и проводятся все операции. При этом сами операции (прежде всего арифметические) определяются таким образом, чтобы результат со-

ответствующей операции обязательно располагался внутри вычисляемых границ. Это в конечном счете приводит к возможности автоматически учитывать погрешности в задании исходных чисел и погрешности, вызываемые машинным округлением.

В интервальных вычислениях переход к тетралогике (см. разделы 1.2, 1.7) прежде всего обусловлен возможностью построения различных вариантов логических значений, включающих, кроме классических 1 («истина») и 0 («ложь»), также и различные парные комбинации, в частности, такие как А («неопределенность», «неоднозначность») и М («множественность», «многозначность»). Кроме того, система кодирования количественной информации, построенная на тетралогике, обладает по сравнению с традиционными рядом качественных преимуществ [6].

Рассмотрим, например, процесс перехода от десятичного интервала к «тетракодовому» на примере вычисления иррационального числа π с помощью **формулы Бэйли-Боруэйна-Плаффа** [13]:

$$\pi = f(n) = \sum_{k=0}^n \frac{1}{16^k} \cdot \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right). \quad (4.4)$$

Десятичный интервал $x = [x_-; x_+]$ ($x \in I(R)$, $x_-, x_+ \in I(R)$) составим из чисел, полученных на 1-м и 100-м шагах вычислений (8 значащих цифр):

$$x_- = f(1) = 3,1414225; \quad x_+ = f(100) = 3,1415927.$$

Поскольку полученные значения различаются, начиная с одной десятитысячной, то при использовании этого интервала в процессе вычислений, позволяющих проводить автоматический учет всех видов погрешностей, гарантиру-

ется точность четырех первых цифр самого результата, т. е. в данном случае числа π .

Эти числа, представленные в двоичном формате (для наглядности использован 32-битный двоичный формат для представления дробной части чисел), позволяют выразить данный интервал одним тетракодом $T \in \{t \mid t \in \Phi\}$, где $\Phi = \{0, 1, A, M\}$:

$$\begin{aligned} &+3,1414225_{10} = \\ &= +11.00100100001101000100001111010101_2, \\ &+3,1415927_{10} = \\ &= +11.0010010000111110110101101001111_2, \end{aligned}$$

откуда

$$T = +11.001001000011MMMMMAAAAAAAAAAAAAA.$$

Количество разрядов «многозначности» M обусловлено обеспечением необходимой плотности «отсчетов» внутри границ интервала, обеспечивающее приближение минимум одного из множества вычисляемых значений к исходному числу с заданной точностью. В частности, k -я позиция последнего разряда «многозначности» дробной части тетракодового числа может быть определена по формуле

$$k = \left\lceil \frac{\ln(10^n)}{\ln(2)} + 1 \right\rceil, \text{ или } k = \lceil 3,322n + 1 \rceil, \quad (4.5)$$

где n — количество цифр дробной части исходного числа в его десятичном представлении. Для двух исходных чисел с неравным количеством цифр дробной части n_1 и n_2 результирующее значение n выбирается из условия

$$n = \max(n_1, n_2).$$

Так, например, для нашего случая $n = 5$ (оба десятичных числа имеют 5 цифр после запятой). Следовательно,

$k = 18$, т. е. в тетракодовом числе разряды дробной части, начиная с 19-го, являются несущественными и принимают значение неопределенности A .

Обратный же переход к десятичным числам достигается абсолютной минимизацией/максимизацией M для получения левой/правой (нижней/верхней) границ интервала.

Получим двоичные значения границ интервала, заданных тетракодом T , причем рассмотрим его наихудшее значение с позиции неопределенности A . Очевидно, что для нижней границы интервала наихудшим будет тот случай, когда все разряды A примут значение 1; для верхней границы интервала наихудшим значением является то, в котором все разряды A примут значение 0:

$$t_{\min} = 11.00100100001100000011111111111111;$$

$$t_{\max} = 11.0010010000111111110000000000000000.$$

Полученные десятичные значения

$$d_1 = (t_{\min})_{10} = 3,1413612|363...$$

$$\text{и } d_2 = (t_{\max})_{10} = 3,1415977|478...$$

фиксируют все значения функции $f(x)$. В этом легко убедиться, поскольку истинное значение $\pi = 3,1415926|536....$

На рис. 4.1 показано поведение функции $f(x)$ на первых 10 шагах вычисления, а также ее «вхождение» в полученный интервал с границами d_1 и d_2 .

Представление границ интервалов в виде тетракодов дает возможность гибкого задания практически всех наборов числовых значений, включенных в этот интервал. Хотя по сравнению с обычным бинарным кодом количество бит, требуемых для кодирования тетракодов, увеличивается в 2 раза, кодирование числового интервала одним тетракодовым словом позволяет представить все значения числовой оси, лежащие между границами интервала. Повышение

степени информативности получаемых за счет этого кодов вполне оправдывает увеличение затрат на кодирование.

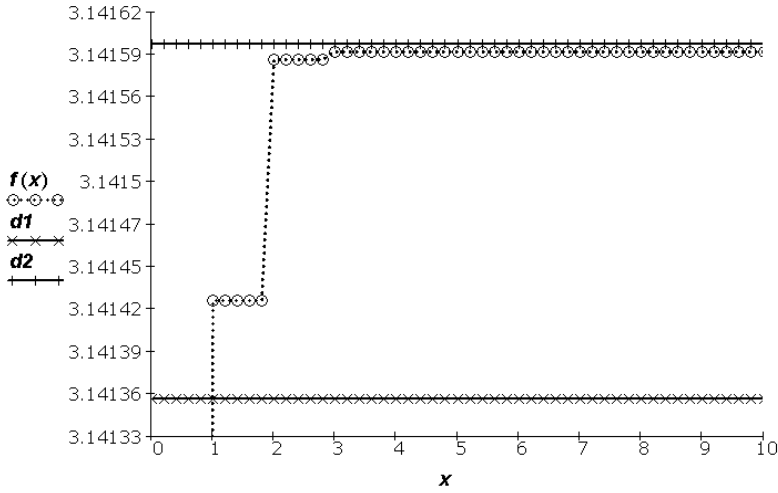


Рисунок 4.1 — Вычисление числа π с помощью формулы Бэйли-Боруэйна-Плаффа

Таким образом, с помощью тетракода можно закодировать границы интервала произвольной точности в одном поле данных. Однако следует учесть, что данный способ эффективен при незначительной ширине интервала $w = \text{wid}(x) = x_2 - x_1$.

4.3. Отображение тетракодов на множестве интервальных чисел

Тетракод как система кодирования количественной информации может быть представлен различными наборами «тетракодовых» разрядов — тетритов (см. раздел 2.1), кодирующими множество состояний двумерного

логического пространства [14, 15]. Однако тетракод $T = \{t\}$ с набором тетритов $t \in \{0, 1, A, M\}$ получил более широкое применение в ряде исследований (см. разделы 2.2–2.5, 4.2 или [16–18]). Поэтому в рамках данного раздела будет рассматриваться тетракод, состоящий из такого набора тетритов. Рассматривая тетракод в качестве носителя количественной информации, можно высказать следующие предположения:

- тетриты 0 и 1 сводятся к битам 0 и 1 соответственно, так как они кодируют аналогичные логические состояния;
- тетрит М сводится к битам 0 и 1 одновременно, поэтому представляется двумя точками на числовой оси;
- тетрит А сводится к битам 0 или 1 (в момент сведения его значение неизвестно), поэтому представляется одной из двух возможных точек на числовой оси.

На рис. 4.2 показано, как тетрит t сводится к биту b на бинарной оси x_b по событиям s_1 и s_2 — выборкам двоичных значений из тетракода в разные моменты времени. При этом тетриты 0 и 1 (кодирующие состояния «ложь» и «истина» тетралогики) можно назвать **однозначно представимыми** на числовой оси, поскольку их значения сводятся к аналогичным двоичным значениям 0 и 1 в любой момент времени. Это подтверждено идентичностью состояний «истина» и «ложь» тетралогики и классической бинарной логики.

Тетриты А и М (кодирующие состояния «неопределенности» и «множественности» тетралогики) относятся к классу **неоднозначно представимых**, поскольку не могут быть однозначно сведены к одному двоичному значению.

Тетрит М сводится к паре значений 0 и 1 во всех событиях s_i , т. е. он всегда позиционируется двумя точками на числовой оси. Тетрит А занимает одно случайное двоичное значение 0 или 1 в каждом событии, т. е. он всегда позиционируется одной из двух возможных точек на числовой оси. На рис. 4.2 тетрит А при наступлении события s_1 был случайным образом сведен к двоичной 1 (с такой же вероятностью он мог бы занять позицию двоичного 0), а при наступлении события s_2 принял значение двоичного нуля (с такой же вероятностью он мог бы сохранить предыдущее значение от события s_1). Данный аспект представления тетрита А позволяет эффективно кодировать состояния тетралогики, обладающие свойствами равновероятности, а при совокупности множества выборов — свойствами случайности.

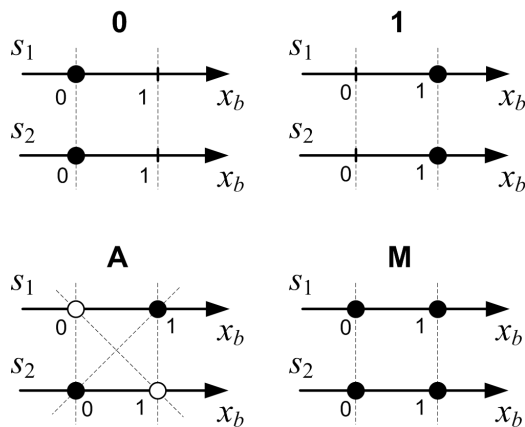


Рисунок 4.2 — Позиционирование тетритов на числовой оси

Наличие тетритов М в тетракоде предполагает его сведение к множеству двоичных значений, т. е. фактически к

замкнутому числовому промежутку. Такой числовой промежуток называется интервалом, который представляет собой множество всех значений, лежащих внутри его границ [11]. Очевидно, что кодирование границ вещественных интервалов в виде тетракода возможно лишь в том случае, если в нем присутствуют разряды М, являющиеся определяющим фактором выявления интервальных границ. В таком тетракоде допускается наличие тетритов А, которые являются уточняющим фактором при выявлении интервальных границ и занимают младшие разряды тетракода, уменьшая тем самым погрешность представления закодированных в тетракоде количественных значений.

Описанная выше структура тетракода с позиции расположения тетритов М и А наделяет его так называемой **нормированностью**, которая определена совокупностью следующих критериев:

- Тетриты 0 и 1 являются основной информационной составляющей и фактически определяют позицию числа на числовой оси.
- Наличие «группы М» — наличие тетрита или группы тетритов М обязательно для представления границ интервала: группа М неразрывна и располагается в младших разрядах тетракода.
- Наличие «группы А» — наличие тетрита или группы тетритов А не обязательно, однако носит уточняющий характер для представления границ интервала: группа А неразрывна и занимает более младшие разряды тетракода, чем группа М.
- Между группами М и А не допускается появления однозначно представимых тетритов 0 и 1.

Все представленные в работе результаты будут справедливы только для нормированных тетракодов, причем

тетракод, содержащий в себе только группу M , будем считать **нормированностью первого рода**, а содержащий в себе группы M и A , — **нормированностью второго рода**.

В данном разделе рассматривается реализация возможности использования нормированного тетракода для кодирования значений границ интервальных чисел в одном поле данных, т. е. фактически решается задача, обратная задаче предложенной в п. 4.2, в которой рассматриваются особенности перехода от интервального представления результатов вычислений к постбинарным. При этом предполагается, что определение диапазонов «плавающих» границ и ширины интервального числа, представленного нормированным тетракодом первого и второго рода, является достаточным условием для оценки эффективности и точности позиционирования границ при постбинарном кодировании интервалов. Рассмотренная концепция может послужить началом развития средств и методов постбинарных вычислений, способных позиционироваться как математическая дисциплина, предметом которой является решение задач с постбинарными интервальными неопределенностями и неоднозначностями в данных, возникающими в постановке задачи или на промежуточных стадиях процесса решения.

Пусть X — **целочисленный интервал** (интервальное число), в котором x_- и x_+ — его левая и правая границы, причем все значения x интервала принадлежат области целых чисел: $x \in \mathbb{Z}$. «Размах» границ интервала определяет его ширину w , которая выражается следующим соотношением:

$$w = x_+ - x_- \geq 0. \quad (4.6)$$

Ширина интервала — важнейшая его характеристика, поскольку является естественной мерой неопределенности

(неоднозначности) величины, выраженной интервальным числом.

Рассмотрим нормированный тетракод первого рода, т. е. содержащий группу M в младших разрядах. При сведении такого тетракода к множеству двоичных значений границы интервала определены крайними позициями этого множества: левая граница — при всех значениях M , сведенных в двоичный ноль; правая граница — при всех значениях M , сведенных в двоичную единицу. При этом ширина полученного интервала может быть вычислена по формуле (4.6), но поскольку однозначно представимые в исходном тетракоде тетриты 0 и 1 самоуничтожатся операцией вычитания, оценку ширины закодированного интервала можно получить, не используя численные значения x_- и x_+ (рис. 4.3, *а*). Так как значение тетракода сводится к отображению двоичных чисел, а двоичная система счисления является позиционной, то ширина полученного интервала зависит прежде всего от позиции группы M в тетракоде, а точнее — от позиции старшего тетрита поля M в тетракоде. Поэтому

$$w = 2^{k+1} - 1, \quad (4.7)$$

где k — позиция старшего тетрита $t = M$ в n -разрядном тетракоде: $n-1 \leq k \leq 0$.

Тетракод с нормированностью второго рода при сведении его к границам интервального числа предполагает определение расстояния q между значениями минимально и максимально возможных границ интервалов (x_-^{\min} , x_+^{\min} и x_-^{\max} , x_+^{\max}), имеющих ширину w_1 и w_2 соответственно (рис. 4.3, *б*):

$$q = x_-^{\max} - x_-^{\min} = x_+^{\max} - x_+^{\min}. \quad (4.8)$$

Фактически величина q является расстоянием между двумя интервалами шириной w_1 и w_2 , т. е. расстоянием на множестве $I(\mathbf{R})$ целочисленных и вещественных интервалов (поскольку $\mathbf{Z} \subset \mathbf{R}$, то множество целочисленных интервалов $I(\mathbf{Z})$ также является подмножеством множества $I(\mathbf{R})$, т. е. $I(\mathbf{Z}) \subset I(\mathbf{R})$). Отображение q на множестве $I(\mathbf{R})$ задает метрику, которая является **хаусдорфовой** [11, с. 23]. Хаусдорфова метрика обобщает понятие расстояния между двумя точками в замкнутом метрическом пространстве (в данном контексте таким пространством является множество \mathbf{R}) на случай всех компактных непустых подмножеств данного пространства. При введении на множестве $I(\mathbf{R})$ метрики оно становится топологическим пространством с сохранением понятий сходимости и непрерывности, как и в случае метрического пространства.

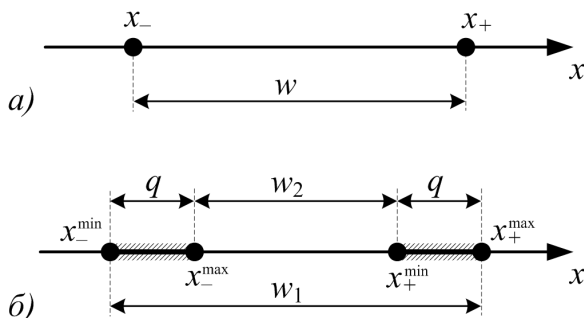


Рисунок 4.3 — Формирование целочисленных интервальных границ при декодировании нормированного тетракода первого (а) и второго (б) рода

При сведении к интервальным границам тетракода, обладающего свойствами нормированности второго рода, расстояние w_1 между крайними точками x_-^{\min} и x_+^{\max} опре-

деляется аналогично (4.8), поскольку поле A сводится к двоичным значениям, аналогичным полю M (все нули для левой и все единицы для правой границ интервала): $w_1 = w$. Расстояние w_2 между точками x_-^{\max} и x_+^{\min} связано с w_1 соотношением $w_1 = w_2 - 2q$. Точки x_-^{\min} и x_+^{\max} сформированы с позиции учета погрешности, вносимого в значения границ полем A тетракода. При этом максимальное значение q достигается для границ следующим образом:

- для левой границы: при минимальном значении поля M (все тетриты M сводятся к двоичному 0) значение поля A максимально (все тетриты A принимают значения двоичной 1);
- для правой границы: при максимальном значении поля M (все тетриты M сводятся к двоичной 1) значение поля A минимально (все тетриты A принимают значения двоичного 0).

Такая концепция предполагает определение параметров w_1 и q , при которых представленные тетракодом границы интервала будут находиться в выделенных на рис. 4.3, б участках числовой оси.

Поскольку на величину q оказывает влияние только значение, возвращаемое полем A тетракода, то длину q можно выразить через позицию старшего тетрита поля A :

$$q = 2^{l+1} - 1, \quad (4.9)$$

где l — позиция старшего тетрита $t = A$ в n -разрядном тетракоде: $k - 1 \leq l \leq 0$ (k — позиция старшего тетрита $t = M$)

При этом значение w_2 можно определить следующим образом: $w_2 = w_1 - 2q = 2^{k+1} - 1 - 2 \cdot (2^{l+1} - 1) = 2^{k+1} - 2^{l+2} + 1$.

При этом возможный диапазон «плавания» границ интервала, обусловленный заполнением поля A случайными

двоичными значениями, определяется следующим образом:

• $[x_-^{\min}, x_-^{\max}] = [x_-^{\max} - q, x_-^{\max}] = [x_-^{\min}, x_-^{\min} + q]$ — для левой границы;

• $[x_+^{\min}, x_+^{\max}] = [x_+^{\max} - q, x_+^{\max}] = [x_+^{\min}, x_+^{\min} + q]$ — для правой границы.

В качестве примера можно выполнить сведение 8-разрядного тетракода T с нормированностью второго рода к интервалу X : $T \rightarrow X$. Пусть $T = 101\text{ММААА}$. Тогда $k = 4$, $l = 2$. Получим возможный разброс границ $q = 2^3 - 1 = 7$ и максимально возможную ширину $w_1 = 2^5 - 1 = 31$. Таким образом, при любой выборке (наступлении события s_i) с учетом возвращения полем A случайного набора двоичных чисел ширина интервала будет не больше $w_1 = 31$ и не меньше $w_2 = 31 - 2 \cdot 7 = 17$.

Выполним проверку (\downarrow — сведение к двоичному 0, \uparrow — сведение к двоичной 1):

$$\begin{aligned} X^{\max} &= [x_-^{\min}, x_+^{\max}] = \\ &= [101 \underbrace{00}_{all\ M\downarrow\ all\ A\downarrow} \underbrace{000}_{all\ M\uparrow\ all\ A\uparrow}, 101 \underbrace{11}_{all\ M\uparrow\ all\ A\uparrow} \underbrace{111}_{all\ M\uparrow\ all\ A\uparrow}]_2 = [160, 191]_{10}. \\ X^{\min} &= [x_-^{\max}, x_+^{\min}] = \\ &= [101 \underbrace{00}_{all\ M\downarrow\ all\ A\downarrow} \underbrace{111}_{all\ M\uparrow\ all\ A\uparrow}, 101 \underbrace{11}_{all\ M\uparrow\ all\ A\uparrow} \underbrace{000}_{all\ M\downarrow\ all\ A\downarrow}]_2 = [167, 184]_{10}. \end{aligned}$$

Из (4.8) получаем $q = 160 - 167 = 191 - 184 = 7$. Диапазон «плавания» границ интервала X , закодированного в T : $[160, 167]$ — для левой границы; $[184, 191]$ — для правой границы

При сведении тетритов *n*-разрядного «вещественного» тетракода T' (условно разделенного на поля знака, порядка и мантиссы) к вещественному интервалу X' все оценки возможного размаха интервальных границ определяются аналогично описанным выше целочисленным представлениям. Однако сведение тетракода T' к двоичным значениям определяет формат последних как формат с плавающей запятой. При этом для расчета ширины предполагаемых диапазонов, а также расстояния между ними, следует учитывать некоторые дополнительные параметры: значение поля и смещение порядка, а также количество разрядов мантиссы. В таком случае ширина максимально и минимально представимого интервала из T' — w' , w'_1 и w'_2 — определена следующими соотношениями (F — коэффициент, учитывающий особенности формата представления чисел с плавающей запятой):

$$w' = F \cdot w; \quad (4.10)$$

$$w'_1 = F \cdot w_1; \quad (4.11)$$

$$w'_2 = F \cdot w_2. \quad (4.12)$$

Зависимость (4.10) применима для тетракода с нормированностью первого рода, а зависимости (4.11, 4.12) — для тетракода с нормированностью второго рода. Все правила нормированности для «вещественного» тетракода относятся только к полю мантиссы и не предполагают появления неоднозначно представимых тетритов ($t \in \{A, M\}$) в поле порядка.

С учетом специфичности форматов чисел с плавающей запятой «вещественный» тетракод может сводиться к границам интервала входящих в области нормализованных (имеющих ненулевое значение в поле порядка) и денормализованных (имеющих нулевое значение в поле порядка)

чисел с плавающей запятой [19]. Рассматривать иные особые случаи представления чисел плавающих форматов (не число: NaN, положительная и отрицательная бесконечности: $+\infty$ и $-\infty$) не имеет смысла, так как параметры таких интервалов изначально известны: $[\text{NaN}; \text{NaN}]$, $[\text{NaN}; -\infty]$ и $[+\infty; \text{NaN}]$ имеют ширину $w = \text{NaN}$; $[-\infty; +\infty]$ имеет ширину $w = +\infty$. Интервалы $[+\infty; +\infty]$ и $[-\infty; -\infty]$ задать с помощью нормированного тетракода невозможно, так как данные интервалы точечные (границы интервалов совпадают), т. е. в плавающих форматах представляют точку на числовой прямой. Таким образом, коэффициент F может быть представленным только в двух вариантах: для нормализованных (F_1) и денормализованных (F_2) чисел:

$$F_1 = 2^{E-\text{offset}-m}, \quad (4.13)$$

$$F_2 = 2^{1-\text{offset}-m}, \quad (4.14)$$

где E — десятичное значение экспоненты двоичного числа, m — количество двоичных разрядов мантииссы (в 32-битном IEEE754 $m = 23$, в 64-битном — $m = 52$), $\text{offset} = 2^{b-1} - 1$ — заданное смещение экспоненты, имеющей b двоичных разрядов (в 32-битном IEEE754 оно равно +127, в 64-битном — +1023).

На основании (4.10–4.14) получаем основные зависимости (k и l — номера старших разрядов полей М и А соответственно):

- для нормализованных чисел:

$$w'_1 = w' = 2^{E-\text{offset}-m} \cdot (2^{k+1} - 1); \quad (4.15)$$

$$w'_2 = 2^{E-\text{offset}-m} \cdot (2^{k+1} - 2^{l+2} + 1); \quad (4.16)$$

$$q' = F_1 \cdot q = \frac{1}{2} \cdot (w'_1 - w'_2) = 2^{E-\text{offset}-m} \cdot (2^{l+1} - 1). \quad (4.17)$$

- для денормализованных чисел:

$$w'_1 = w' = 2^{1-\text{offset}-m} \cdot (2^{k+1} - 1); \quad (4.18)$$

$$w'_2 = 2^{1-\text{offset}-m} \cdot (2^{k+1} - 2^{l+2} + 1); \quad (4.19)$$

$$q' = 2^{1-\text{offset}-m} \cdot (2^{l+1} - 1). \quad (4.20)$$

В качестве примера можно выполнить сведение 32-разрядного «вещественного» тетракода T' с нормированностью второго рода к интервалу $X' : T' \rightarrow X'$.

Пусть

$$T' = 1 \ 01111100 \ 10110111001\text{MMMMAAAAAAA}.$$

Все числа, сводимые данным тетракодом, представляют множество нормализованных чисел с плавающей запятой. Получаем исходные данные: $k = 11$, $l = 7$, $m = 23$, $\text{offset} = 127$, $E = 01111100_2 = 124_{10}$. Используя зависимости (4.15–4.17), рассчитаем возможный разброс границ интервала:

$$\begin{aligned} q' &= 2^{124-127-23} \cdot (2^8 - 1) = 2^{-26} \cdot 255 = \\ &= \frac{255}{67108864} = 3,7997961... \cdot 10^{-6} \approx 3,8 \cdot 10^{-6}; \\ w'_1 &= 2^{-26} \cdot (2^{12} - 1) = \frac{4095}{67108864} = 6,10202550... \cdot 10^{-5}; \\ w'_2 &= 2^{-26} \cdot (2^{12} - 2^9 + 1) = \frac{3585}{67108864} = 5,3420663... \cdot 10^{-5}. \end{aligned}$$

Таким образом, при любой выборке (наступлении события s_i) с учетом возвращения полем А случайного набора двоичных чисел ширина интервала не превысит значение $6,10202550 \cdot 10^{-5}$ и не станет меньше, чем $5,3420663 \cdot 10^{-5}$.

Выполним проверку с учетом того, что данный T' сводится к отрицательным числам. Следовательно, числа с большим модулем находятся на отрицательной части числовой оси левее, чем числа с меньшим модулем. Учитывая данную особенность отрицательных чисел, границы максимально и минимально представимых диапазонов формируются следующим образом (\downarrow — сведение к двоичному нулю, \uparrow — сведение к двоичной единице):

$$\begin{aligned}
 X'^{\max} &= [x'^{\min}_-, x'^{\max}_+] = \\
 &= [1 \ 01111100 \ 10110111001 \underbrace{11111}_{all\ M\uparrow} \underbrace{11111111}_{all\ A\uparrow}], \\
 &\quad 1 \ 01111100 \ 10110111001 \underbrace{0000}_{all\ M\downarrow} \underbrace{00000000}_{all\ A\downarrow}]_2 = \\
 &= [-0.21447752, -0.21441650]_{10}.
 \end{aligned}$$

$$\begin{aligned}
 X'^{\min} &= [x'^{\max}_-, x'^{\min}_+] = \\
 &= [1 \ 01111100 \ 10110111001 \underbrace{1111}_{all\ M\uparrow} \underbrace{00000000}_{all\ A\downarrow}], \\
 &\quad 1 \ 01111100 \ 10110111001 \underbrace{0000}_{all\ M\downarrow} \underbrace{11111111}_{all\ A\uparrow}]_2 = \\
 &= [-0.21447372, -0.21442030]_{10}.
 \end{aligned}$$

Из (4.8) получаем

$$\begin{aligned}
 q' &= -0,21447372 + 0,21447752 = \\
 &= -0,21441650 + 0,21442030 = 3,8 \cdot 10^{-5}.
 \end{aligned}$$

Диапазон «плавания» границ интервала X' , извлеченного из T' :

$$\begin{aligned}
 [-0.21447752, -0.21447372] &\text{ — для левой границы;} \\
 [-0.21442030, -0.21441650] &\text{ — для правой границы}
 \end{aligned}$$

Описанная в данном разделе методика сведения n -разрядного нормированного «вещественного» тетракода к вещественному интервалу приводит к извлечению границ интервала одного знака, т. е. к таким интервальным числам, которые могут занимать позиции либо на положительной, либо на отрицательной полуосях вещественных чисел. Следовательно, данная техника кодирования не является достаточной, поскольку исключает представление интервала, содержащего нулевое значение. Таким образом, для представления интервала X'' ($x \in \mathbf{R}$, $0 \in X''$) формируется нормированный «вещественный» тетракод T'' , содержащий в поле знака тетрит $t = M$. Такой тетракод отвечает обозначенным требованиям к нормированности (относящийся прежде всего к полю мантиссы) и сводится к интервальным границам, аналогично описанной выше методике, однако при формировании левой границы тетрит M в поле знака сводится к двоичной 1 (формируется отрицательное значение), а при формировании правой границы — к двоичному 0 (формируется положительное значение).

На рис. 4.4 представлены соотношения интервалов X'' и X' , отличающихся тем, что декодированы из идентичных «вещественных» тетракодов T'' (в поле знака однозначно представимые значения тетритов 0 ($X' > 0$) или 1 ($X' < 0$)) и T'' (в поле знака значение M). Поскольку интервал X'' объединяет зеркальное отображение интервала X' на отрицательной и положительной полуосях, то интервальные оценки данных интервалов связаны, и основная оценка — расстояние d'' — определима следующим образом: $d'' = w'_1$.

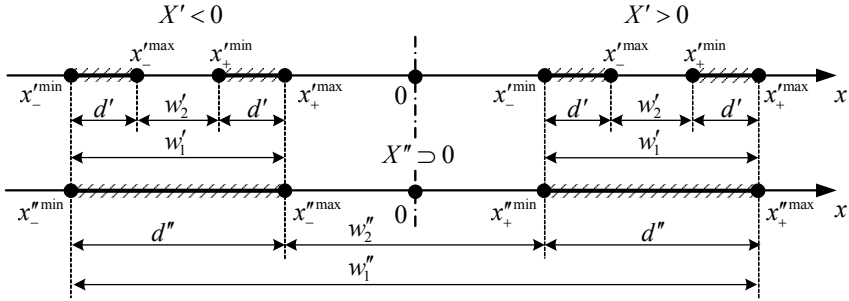


Рисунок 4.4 — Формирование вещественных интервальных границ для интервала, содержащего нулевое значение

Следовательно

- для нормализованных чисел:

$$d'' = 2^{E-\text{offset}-m} \cdot (2^{k+1} - 1); \quad (4.21)$$

- для денормализованных чисел:

$$d'' = 2^{1-\text{offset}-m} \cdot (2^{k+1} - 1). \quad (4.22)$$

Значения ширины w'_1 и w'_2 зависят от всех разрядов поля мантиссы в тетракоде (в первую очередь от однозначно представимых значений тетритов, поскольку при нахождении расстояния между границами интервала с противоположными знаками их значения складываются) и изменяют свое значение в каждом отдельном случае. И, наконец, если интервал X'' представлен тетракодом T'' с нормированностью первого рода, то середина такого интервала будет всегда равна нулевому значению, что обусловлено симметричным расположением границ интервала относительно нуля. При использовании «вещественного» тетракода T'' с нормированностью второго рода такая симметрия будет нарушена, поскольку для каждой из гра-

ниц все тетриты поля А будут сводиться к различным наборам двоичных бит. Появление тетрита М в знаковом разряде переопределяет принципы получения минимально и максимально допустимых значений границ интервала. Однако стоит отметить одинаковость поведения полей М и А, поэтому в формулах (4.21, 4.22) не используется значение l — позиция старшего тетрита в поле А.

Для наглядного представления описанных выше положений рассмотрим сведение 32-разрядного «вещественного» тетракода T'' (аналогичного T' из предыдущего примера) к интервалу X'' : $T'' \rightarrow X''$.

Пусть

$$T'' = \text{М } 01111100 \ 10110111001\text{MMMMAAAAAAAAA}.$$

Получаем исходные данные: $k = 11$, $m = 23$, $\text{offset} = 127$, $E = 01111100_2 = 124_{10}$. Используя зависимости (4.15–4.17), рассчитаем возможный разброс границ интервала:

$$\begin{aligned} q'' &= 2^{124-127-23} \cdot (2^{12} - 1) = \frac{4095}{2^{-26}} = \\ &= 6,10202550... \cdot 10^{-5} \approx 6,102 \cdot 10^{-5}. \end{aligned}$$

Выполним проверку (\downarrow — сведение к двоичному нулю, \uparrow — сведение к двоичной единице):

$$\begin{aligned} X''^{\max} &= [x''^{\min}_-, x''^{\max}_+] = \\ &= [\underbrace{1}_{\text{М}\uparrow} \ 01111100 \ 10110111001 \underbrace{1111111111111111}_{\text{all М}\uparrow} \underbrace{11111111111}_{\text{all А}\uparrow}], \\ &\quad \underbrace{0}_{\text{М}\downarrow} \ 01111100 \ 10110111001 \underbrace{1111111111111111}_{\text{all М}\uparrow} \underbrace{11111111111}_{\text{all А}\uparrow}]_2 = \\ &= [-0,21447752, 0,21447752]_{10}. \end{aligned}$$

$$\begin{aligned}
X''^{\min} &= [x''^{\max}_-, x''^{\min}_+] = \\
&= [\underbrace{1}_{M\uparrow} \ 01111100 \ 10110111001 \underbrace{0000}_{all\ M\downarrow} \underbrace{00000000}_{all\ A\downarrow}, \\
&\quad \underbrace{0}_{M\downarrow} \ 01111100 \ 10110111001 \underbrace{0000}_{all\ M\downarrow} \underbrace{00000000}_{all\ A\downarrow}]_2 = \\
&= [-0,21441650, 0,21441650]_{10}.
\end{aligned}$$

Из (4.8) получаем

$$\begin{aligned}
q'' &= -0,21441650 + 0,21447752 = \\
&= 0,21447752 - 0,21441650 = 6,102 \cdot 10^{-5}.
\end{aligned}$$

Диапазон «плавания» границ интервала X' , извлеченного из T' :

$$\begin{aligned}
[-0.21447752, -0.21441650] &\text{ — для левой границы;} \\
[0.21441650, 0.21447752] &\text{ — для правой границы}
\end{aligned}$$

При декодировании тетракода, т. е. при сведении значений тетритов t_i (i — позиция тетрита в n -разрядном тетракоде, $n-1 \leq i \leq 0$) тетракода к интервалу, можно прибегнуть к предложенным в разделе 2.2 **унарным функциям тетралогикки** (таблица 2.1): MIN_A (MAX_A) — минимизация (максимизация) неопределенности; MIN_M (MAX_M) — минимизация (максимизация) множественности.

Используя сочетания приведенных в таблице функций, можно все значения тетритов А и М привести к значениям 0 и 1. Тетракод, состоящий из тетритов 0 и 1, фактически является двоичным числом — значением интервальной границы. При сведении n -разрядного тетракода T_n к значениям интервальных границ предлагается использовать следующие четыре сочетания:

$$\text{MIN_M}(\text{MIN_A}(t_i)) = \begin{cases} 0, & \text{if } t_i = 0 \vee t_i = A \vee t_i = M, \\ 1, & \text{if } t_i = 1; \end{cases} \quad (4.23)$$

$$\text{MIN_M}(\text{MAX_A}(t_i)) = \begin{cases} 0, & \text{if } t_i = 0 \vee t_i = M, \\ 1, & \text{if } t_i = 1 \vee t_i = A; \end{cases} \quad (4.24)$$

$$\text{MAX_M}(\text{MIN_A}(t_i)) = \begin{cases} 0, & \text{if } t_i = 0 \vee t_i = A, \\ 1, & \text{if } t_i = 1 \vee t_i = M; \end{cases} \quad (4.25)$$

$$\text{MAX_M}(\text{MAX_A}(t_i)) = \begin{cases} 0, & \text{if } t_i = 0, \\ 1, & \text{if } t_i = 1 \vee t_i = A \vee t_i = M. \end{cases} \quad (4.26)$$

Таким образом, в качестве оценки предельных колебаний интервальных границ можно рассматривать выражения (4.23–4.26), которые фактически заменяют использованные в примерах условные выражения $\text{all } M(A) \downarrow(\uparrow)$ следующим образом:

$$\text{all } M \downarrow \vee \text{all } A \downarrow \equiv \text{MIN_M}(\text{MIN_A}(T_n));$$

$$\text{all } M \downarrow \vee \text{all } A \uparrow \equiv \text{MIN_M}(\text{MAX_A}(T_n));$$

$$\text{all } M \uparrow \vee \text{all } A \downarrow \equiv \text{MAX_M}(\text{MIN_A}(T_n));$$

$$\text{all } M \uparrow \vee \text{all } M \uparrow \equiv \text{MAX_M}(\text{MAX_A}(T_n));$$

$$M \downarrow \equiv \text{MIN_M}(t_{n-1}); \quad M \uparrow \equiv \text{MAX_M}(t_{n-1}).$$

Равенства в последней строке выражают поочередное сведение к нулю и к единице знакового тетрита t_{n-1} «вещественного» n -разрядного тетракода.

Таким образом, используя соотношения (4.7, 4.9, 4.15–4.26), можно провести интервальное оценивание и получить необходимые метрики интервального числа, границы которого представлены тетракодом, не прибегая к его декодированию в двоичные/десятичные значения. При этом

возможно эффективное хранение интервальных чисел в виде тетракодов с их дальнейшим использованием в постбинарных компьютерных системах.

Актуальность приведенной концепции нацелена на недалекое будущее, в котором внедрение средств постбинарного компьютеринга несомненно приведет к развитию множества математических структур, в числе которых может оказаться и **постбинарная интервальная арифметика** как структура, определившая арифметические интервальные операции для нормированных тетракодов. Такая структура сможет использовать тетракоды в качестве операндов и применять к ним постбинарные логические и арифметические операции, реализация которых уже сегодня заложена в основу разработки алгебры тетралогии [17, 18].

4.4. Пример Румпа в контексте традиционных и интервальных вычислений

Лавинообразное нарастание объемов вычислений в процессе исследования, моделирования и проектирования сложных систем и динамических процессов, как никогда ранее, актуализирует контроль достоверности и точности вычислительных процессов. Обоснованно предполагается, например, что многие техногенные катастрофы последних десятилетий, причины которых традиционно объяснили преимущественно «человеческим фактором», были в первую очередь обусловлены разного рода вычислительными ошибками [20]. В большинстве же случаев ошибки в вычислениях просто остаются незамеченными, существенно искажая полученные результаты.

Одним из простейших путей минимизации ошибок, связанных с округлением и потерей точности ввиду огра-

ничений, обусловленных особенностью представления чисел в современных цифровых компьютерах, является дальнейший рост разрядности вычислений. В августе 2008 года в этом направлении был сделан важный шаг, связанный с публикацией стандарта IEEE 754–2008, который заменил ранее действовавший стандарт вычислений с плавающей запятой IEEE 754–1985. Стандарт 1985 года предусматривал 2 типа чисел с плавающей запятой: одинарной (32-разрядные) и двойной (64-разрядные) точности. Одной из первых его аппаратных реализаций стал арифметический сопроцессор Intel 8087, в котором дополнительно в качестве внутреннего формата использовался «расширенный» 80-разрядный формат с 64-разрядной мантиссой и 16-разрядным порядком. Основным нововведением в стандарте 2008 года стал 128-разрядный формат «квадро», т. е. формат «учетверенной точности», который, по идее, должен обеспечить соответствующие потребности по части точности вычислений еще на ближайшие 20 лет.

Но один довольно простой пример показывает, насколько эти надежды могут оказаться иллюзорными. Речь идет о так называемом примере Румпа, впервые опубликованном еще в 1988 году [21], и названном по фамилии его автора, который являясь в то время сотрудником немецкого отделения фирмы IBM и исследуя алгоритмы вычислений с гарантированными границами интервалов, заведомо включающими правильный результат, получил полином, дающий при определенном сочетании значений переменных заведомо неправильный результат:

$$f = 333,75 b^6 + a^2 (11a^2 b^2 - b^6 - 121b^4 - 2) + 5,5b^8 + a/(2b), \quad (4.27)$$

где $a = 77617$, $b = 33096$.

При различной разрядности вычислений с плавающей запятой на практически стандартной для того времени большой вычислительной системе ИБМ 370 для данного полинома были получены примерно одинаковые результаты:

32-bit: $f = 1,172604$;

64-bit: $f = 1,1726039400531786$;

128-bit: $f = 1,1726039400531786318588349045201838$.

Но во всех случаях этот результат весьма существенно (даже своим знаком!) отличался от правильного (полученного с помощью специального алгоритма):

$$f = -0,827396059946821368141165095479816...$$

В дальнейшем данный пример стал классической иллюстрацией тех проблем, которые присущи современным вычислениям с плавающей запятой. В частности, в 2001 году были опубликованы результаты исследования примера Румпа с применением различных современных вычислительных платформ и инструментов [22]. Вычисление полинома Румпа на компьютерах с процессорами Intel Pentium и Sun Sparc дало различные результаты в зависимости от того, какая система компьютерной математики использовалась. При этом, например, использование системы Matlab давало классический результат 1.1726... практически независимо от разрядности.

В системе Maple V: $f = -1,18059... \cdot 10^{21}$. В системе Mathematica v.4.0.2: $f = 1,1726...$ или $-1,18059... \cdot 10^{21}$ в зависимости от различных условий.

При этом делается вывод, что для получения **правильного результата необходимо минимум 122 двоичных разряда, т. е. 36 десятичных разрядов** [3]. Стандартное же представление чисел с плавающей запятой эквивалентно фактически **8-ми десятичным разрядам при**

одинарной точности, 17-ти — при двойной, 34-м — при учетверенной.

В 2002 году этот пример исследовался сотрудниками фирмы Sun Microsystems, которые показали, что в некоторых случаях современные стандартные средства вычислений дают еще более непредсказуемые результаты. Например, соответствующие вычисления полинома Румпа на базе программы, полученной с помощью компилятора Fortran 95 фирмы Sun Microsystems Inc., дали следующие результаты [23]:

$$32\text{-bit: } f = -6,338253E + 29,$$

$$64\text{-bit: } f = -1,1805916207174113E + 21,$$

$$128\text{-bit: } f = +1,1726039400531786318588349045201838.$$

Позднее 3. Румп показал [24], что аналогичные проблемы при использовании приведенных выше значений a и b возникают и в случае вычисления более простого полинома вида

$$f = 21 b^2 - 2 a^2 + 55 b^4 - 10 a^2 b^2 + a/(2b). \quad (4.28)$$

Особую обеспокоенность такого рода результаты вызывают в связи с широким распространением в последнее время вычислений на базе графических процессоров (GPU), которые для достижения максимальной производительности чаще всего оперируют с числами только одинарной точности, что резко повышает вероятность получения неверных результатов [25]. Это не столь критично при расчетах, связанных исключительно с визуализацией, но весьма опасно при использовании GPU для высокопроизводительных вычислений общего назначения.

При этом самым тревожным фактом является то, что при проверке примера Румпа практически во всех современных математических пакетах не только результат оказывается неверным, но и **отсутствуют какие-либо при-**

знаки того, что при вычислениях возникли проблемы! А это означает, что такого рода незамечаемых ошибок в современных вычислениях и вычислительном моделировании может быть непредсказуемое множество.

Все вышеизложенное побудило провести более детальное исследование примера Румпа с использованием последних версий современного вычислительного программного обеспечения типа электронных таблиц и систем компьютерной математики (СКМ) Wolfram Mathematica, Mathsoft MathCAD и SciLab [26]. Как и следовало ожидать, в стандартном режиме все СКМ вычислили полином неверно. На рис 4.5, в частности, приведен пример получения традиционного неверного результата в довольно популярной свободно распространяемой СКМ SciLab. Аналогичный результат получен и СКМ MathCAD.

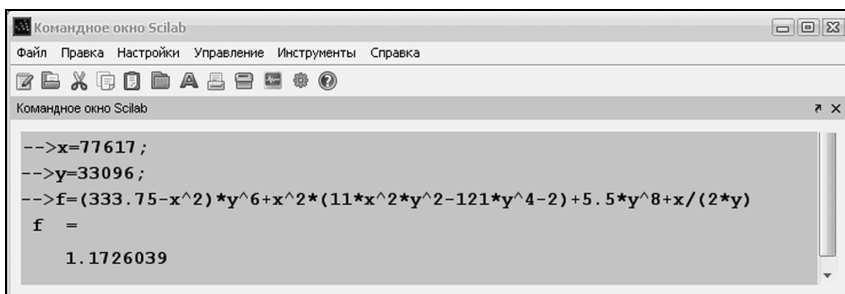


Рисунок 4.5 — Пример получения неверного результата в СКМ SciLab

На рис. 4.6 и 4.7 показаны результаты поэтапного вычисления полиномов (4.27) и (4.28) в MS Excel 2007. При этом не только не получен правильный результат, но наглядно продемонстрировано, что для полинома (4.27) разброс неверных значений существенно зависит от после-

довательности выполняемых вычислений, но при этом даже порядок результата весьма значительно отличается от правильного.

В современной версии СКМ Mathematica 7 попытка вычисления полинома стандартным способом также дает те же результаты, которые были получены для версии 4 практически 10 лет назад и представлены в работе [22]. Таким образом, проблема в общем случае остается нерешенной.

Одним из способов получения правильного результата для примера Румпа является использование символьных вычислений. В данном контексте это означает, что в таком режиме СКМ при вычислении полинома представляет вещественные числа не в традиционном формате с плавающей точкой, а в виде рациональных дробей.

В СКМ MathCAD данное преобразование необходимо выполнять вручную, т. е. непосредственно при вводе формулы полинома с клавиатуры. В этом случае полином Румпа принимает следующий вид:

$$\begin{aligned} f &:= \frac{33375}{100} y^6 + x^2 (11x^2 y^2 - y^6 - 121y^4 - 2) + \frac{55}{10} y^8 + \frac{x}{2y} = \\ &= \frac{1335}{4} y^6 + x^2 (11x^2 y^2 - y^6 - 121y^4 - 2) + \frac{11}{2} y^8 + \frac{x}{2y}. \end{aligned}$$

Для получения правильного результата вычисления необходимо проводить, используя символьный процессор MathCAD [27, с.139–143], обеспечивающий необходимую точность вычислений. Полученный результат в виде обыкновенной дроби при ее численном вычислении дает, наконец, искомое правильное значение

$$f \rightarrow \frac{-54767}{66192} = -0,827396059946821.$$

a	b	
77617	33096	
	Полином 1	
	$333.75 b^6$	4,3860575084639300000000000000000E+29
$a^2 (11a^2 b^2 - b^6 - 121b^4 - 2)$		-7,9171117792747100000000000000000E+36
	$5.5b^8$	7,9171113406689600000000000000000E+36
	$a/(2b)$	1,1726039400531800000000000000000E+00
	Σ	-1,1805916207174100000000000000000E+21
	Полином 2	
	$21 b^2$	2,3002249536000000000000000000000E+10
	$- 2 a^2$	-1,2048797378000000000000000000000E+10
	$+ 55 b^4$	6,5987962821774800000000000000000E+19
	$- 10 a^2 b^2$	-6,5987962832728200000000000000000E+19
	$+ a/(2b)$	1,1726039400531800000000000000000E+00
	Σ	1,1726039400531800000000000000000E+00

Рисунок 4.6 — Результаты вычисления классического (4.27) и сокращенного (4.28) полиномов Румпа (Полином 1 и Полином 2 соответственно) в MS Excel (для представления чисел в экспоненциальном формате установлено максимально возможное значение количества десятичных знаков, равное 30-ти)

В СКМ Mathematica можно осуществить ввод рациональных дробей не только ручным способом, но и воспользоваться функцией `Rationalize[]`, которая возвращает аргумент, преобразованный в рациональную дробь. В результате вычисления примера Румпа со всеми числами с плавающей точкой, преобразованными в обыкновенные дроби, СКМ Mathematica также позволяет получить правильное значение в виде дроби:

$$F = -\frac{54767}{66192}.$$

Однако символьные вычисления не всегда позволяют решить вычислительную проблему типа полинома Румпа. Например, если бы в выражение Румпа входила функция,

возвращающая иррациональный результат (синус, логарифм, радикал и др.), то ее преобразование к дробному виду гарантированно привело бы к погрешности и такая нефиксируемая потеря точности стала бы причиной получения неверного результата.

a	b	
77617	33096	
Полином 1		
333.75 b ⁶	438 605 750 846 393 000 000 000 000 000,00000000000000000000	
x = 11a ² b ²	72 586 759 116 001 000 000,00000000000000000000	
y = - b ⁶ - 121b ⁴ - 2	-1 314 174 679 544 730 000 000 000 000,00000000000000000000	
a ² (x+y)	-7 917 111 779 274 710 000 000 000 000 000 000,00000000000000000000	
5.5b ⁸	7 917 111 340 668 960 000 000 000 000 000 000,00000000000000000000	
a/(2b)	1,17260394005318000000	
Σ	-1 314 175 562 601 590 000 000 000 000,00000000000000000000	
Полином 2		
21 b ²	23 002 249 536,00000000000000000000	
- 2 a ²	-12 048 797 378,00000000000000000000	
+ 55 b ⁴	65 987 962 821 774 800 000,00000000000000000000	
- 10 a ² b ²	-65 987 962 832 728 200 000,00000000000000000000	
+ a/(2b)	1,17260394005318000000	
Σ	1,17260394005318000000	

Рисунок 4.7 — Результаты вычисления классического (4.27) и сокращенного (4.28) полиномов Румпа (Полином 1 и Полином 2 соответственно) в MS Excel (для наглядности представления чисел выбран числовой формат с количеством десятичных знаков 20): изменение порядка операций существенно меняет значение результата при вычислении Полинома 1

Еще одним способом получения правильного результата является использование при вычислениях разрядности, существенно превышающей стандартную. Для примера Румпа это составляет минимум 122 двоичных разряда, т. е. 36 десятичных разрядов.

Высокой эффективности вычислений при этом можно добиться, используя встроенные средства относительно низкоуровневого языка программирования. Традиционно для этих целей используется либо программирование на уровне ассемблера, отличающееся чрезвычайной трудоемкостью и жесткой зависимостью от конкретной аппаратной платформы, либо принимается более компромиссное решение, заключающееся в использовании специальных библиотек для такого широко распространенного языка системного программирования как C++ [28]. Для платформы UNIX/Linux наиболее популярной является библиотека GMP (GNU Multiple-Precision Library) [29]. Для платформы MS Windows наиболее типичным примером является свободно распространяемый класс `cBigNumber`, который реализует целые числа неограниченной разрядности для C++ [30]. В нем предусмотрены все стандартные операции языка C++, включая арифметические, логические и битовые операции, а также операции сравнения, сдвиги и др. Класс оптимизирован для работы с числами от 500 до 20 000 двоичных разрядов. Испытания проведены для чисел, содержащих до 12 000 000 двоичных разрядов [30].

В отличие от C++ в большинстве более современных языков программирования для обеспечения вычислений повышенной точности стандартно поддерживаются целые числа произвольной длины. Например, в C# это обеспечивается использованием типа `System.Numerics.BigInteger`, включенного в базовую библиотеку классов начиная с версии 4.0. В Java это обеспечивается с помощью классов `BigInteger` и `BigDecimal`. Еще одним типичным примером такого рода является Python (Питон) — объектно-ориентированный язык сверхвысокого уровня. Целое в Питоне соответствует типу `long`. Длинное целое — это целое бесконечной длины. С помощью таких чисел можно

производить вычисления неограниченной разрядности, этот тип эмулируется библиотекой, встроенной в интерпретатор.

Однако следует отметить, что использование всех перечисленных программных средств оправдано лишь тогда, когда заведомо требуется оперировать с числами повышенной разрядности. В случае примера Румпа эта необходимость не столь очевидна и, более того, целесообразность более всестороннего исследования такого рода проблем требует использования более гибкого инструмента, который, обладая меньшей вычислительной производительностью, обеспечивал бы существенное повышение суммарной производительности исследований. С этой точки зрения, на сегодня практически оптимальным является выбор СКМ Mathematica.

В СКМ Mathematica начиная с версии 2.0 (в нынешнем виде — начиная с версии 3.0) используется два типа внутреннего представления вещественных чисел — числа с машинной точностью (*machine-precision numbers*) и числа с произвольной точностью (*arbitrary-precision numbers*). Числа с машинной точностью содержат фиксированное количество десятичных знаков, в типичном случае — 16, так что значение *Precision* для них всегда равно 16. В отличие от них числа с произвольной точностью могут содержать произвольное, практически сколь угодно большое число знаков, и значение *Precision* для них может быть любым. Операции над числами с машинной точностью выполняются быстрее, чем для чисел с произвольной точностью, однако использование чисел с произвольной точностью позволяет получить практически сколь угодно точный результат. В общем случае числа с произвольной точностью представляются в следующем формате [31]:

$$\begin{array}{c}
 \text{Precision} \\
 \hline
 \underbrace{x_1 \ x_2 \ \cdots \ x_s}_{\text{Scale}} \cdot \underbrace{x_{s+1} \ x_{s+2} \ \cdots \ x_{s+a}}_{\text{Accuracy}} \underbrace{z_1 \ z_2 \ \cdots \ z_n}_{\text{Guard digits}}
 \end{array}$$

где **Precision** (общая точность) — общее количество десятичных знаков в числе, **Accuracy** (точность дробной части) — количество цифр после десятичной точки, **Scale** — количество цифр перед десятичной точкой, **Guard digits** — вспомогательные цифры для сохранения точности вычислений.

Для того чтобы обеспечить заданную точность вычислений, необходима фиксация соответствующих параметров \$MaxPrecision и \$MinPrecision с помощью функции Block[] на время вычисления выражения. Для удобства можно написать собственную функцию, осуществляющую вычисление таким образом:

```

SetAttributes[FixedPrecisionEvaluate, HoldFirst];
FixedPrecisionEvaluate[input_, digits_]:=
Block[{$MaxPrecision=digits, $MinPrecision=digits}, input];

```

Аргумент input — это вычисляемое выражение; digits — точность, которую необходимо зафиксировать (количество десятичных значащих цифр, которые может иметь число). Необходимо также, чтобы все числа, входящие в выражение, имели ту же фиксированную точность digits.

Для исследования полинома Румпа его необходимо представить в форме, позволяющей задавать необходимую точность:

$$\begin{aligned}
 f3[x_, y_, digits_] := & \text{SetPrecision}[333.75, digits] \ y^6 \\
 & + x^2 * (11 * x^2 y^2 - y^6 - 121 * y^4 - 2) \\
 & + \text{SetPrecision}[5.5, digits] * y^8 + \text{SetPrecision}[0.5, digits] \frac{x}{y}
 \end{aligned}$$

Для вычисления полинома Румпа с различными значениями точности в диапазоне, например, от 1 до 25, необходимо сформировать следующее выражение:

```
Table[FixedPrecisionEvaluate[f3[SetPrecision[77617,i],SetPrecision[33096,i],i],{i,25}]
```

Полученный при этом результат будет представлен в виде следующего пакета данных:

```
{-1.,-1.0,-1.00,-1.000,-1.0000,-1.00000,-1.000000,-1.0000000,-1.00000000,-1.000000000,-1.0000000000,-1.00000000000,-1.000000000000,-1.0000000000000,-1.00000000000000,-1.000000000000000,-0.827396059946821368,-0.8273960599468213682,-0.82739605994682136818,-0.827396059946821368176,-0.8273960599468213681761,-0.82739605994682136817613,-0.827396059946821368176126,-0.8273960599468213681761258}
```

Проведенные исследования показали, что, начиная с определенного значения общей точности (в данном случае начиная уже с 18-ти десятичных цифр, т.е. при незначительном превышении двойной точности), мы получаем практически правильный результат, который на каждом последующем шаге лишь несколько уточняется.

К сожалению, у данного метода тот же недостаток, что и у метода символьных вычислений: заранее неизвестна погрешность результата. В общем случае совершенно неясно, до каких пор стоит увеличивать точность, чтобы получить правильный результат с необходимой точностью.

Зависимость времени вычисления полинома Румпа от задаваемой точности в пакете Wolfram Mathematica 7 (в виде измеренных точечных значений для компьютера Apple MacBook с процессором класса Intel Core 2 Duo — 2 GHz), а также ее линейная аппроксимация приведены на рис. 4.8. Из рисунка видно, что исследуемая зависимость является практически линейной.

В процессе исследований было также установлено, что резкое нарастание ошибки вычислений при недостаточной заданной разрядности имеет место не только для значений $a = 77617$ и $b = 33096$, но и для практически неограничен-

ного количества сочетаний значений исходных параметров в случае, если они соотносятся как $a/b = 2,345\dots$ (рис. 4.9).

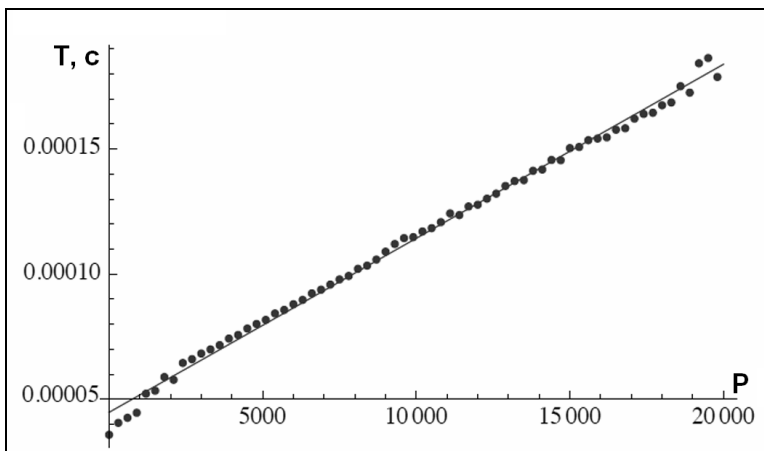


Рисунок 4.8 — Зависимость времени вычисления функции Румпа (T , секунд) от задаваемой фиксированной точности (P — Precision, количество десятичных разрядов)

Целым рядом авторов в контексте проблемы, связанной с примером Румпа, в качестве своеобразной панацеи предлагается **использование интервальной арифметики** [23]. В этом случае для расчетов вместо «точечных» чисел используются так называемые интервалы — диапазоны числовых значений, определенных левой/нижней и правой/верхней границами, что позволяет, как правило, решить целый ряд проблем, связанных с неточностью представления исходных данных и пр. (см. раздел 3.3, а также [32]).

В СКМ Mathematica 7 поддержка интервальной арифметики реализована достаточно эффективно и просто. Для

того чтобы ею пользоваться, достаточно при расчетах задавать исходные значения в виде интервалов. При этом ответ также выдается в виде интервала, в котором должно находиться искомое правильное значение. Например, выражение `Sin[Interval[{0, 0.1}]]` дает в результате `Interval[{0, 0.0998334}]`.

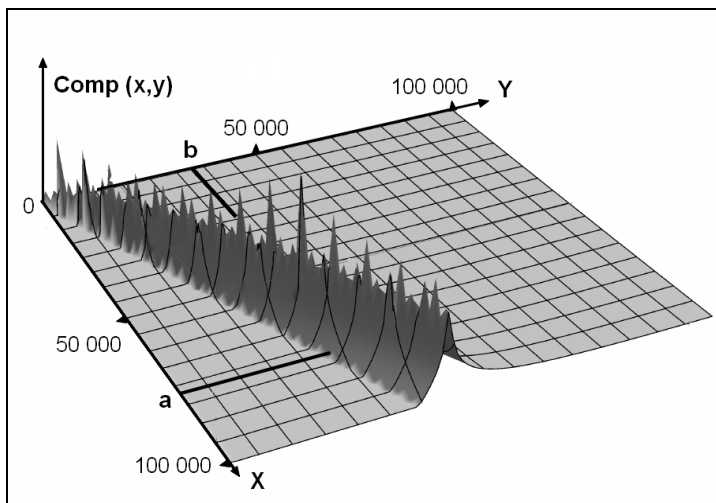


Рисунок 4.9 — Резкое нарастание ошибки вычисления полинома Румпа ($\text{Comp}(x, y)$) при значениях исходных данных, приближающихся к соотношению $a/b = 77617/33096 = 2.345\dots$

Вычисление примера Румпа с использованием интервальной арифметики осуществляется при помощи интервалов, имеющих ширину, обеспечивающую необходимую точность (такие числа представляются в виде набора `Interval[SetPrecision[a, digits]]`). Чтобы вычислить значение полинома Румпа с использованием интервальной арифметики с точностью, например, в 20 цифр можно, воспользоваться командой:

$f[\text{Interval}[\text{SetPrecision}[77617, 20]], \text{Interval}[\text{SetPrecision}[33096, 20]]]$,

где f — полином Румпа (4.27).

С помощью интервальной арифметики можно с заданной точностью получить интервал, в котором заведомо находится искомый точный результат, и определить максимально возможную погрешность вычислений (определяется из ширины интервала). Так, в частности, при заданной точности в 60 цифр был получен достаточно узкий интервал, локализирующий правильный результат вычисления полинома Румпа с точностью до 20-го десятичного знака после запятой:

```
f[Interval[SetPrecision[a, 60]], Interval[SetPrecision[b, 60]]]  
Interval[{-0.827396059946821368142, -0.827396059946821368140}]
```

Зависимость ширины получаемого интервала от задаваемой точности вычислений показана на рис. 4.10, где по оси ординат представлена ширина интервала, а по оси абсцисс — задаваемая точность (количество десятичных цифр). Ось ординат имеет логарифмический масштаб.

В ходе исследований была также оценена зависимость времени вычислений с использованием интервальной арифметики от точности в СКМ Wolfram Mathematica 7. Эта зависимость приведена на рис. 4.11 в виде измеренных точечных значений (компьютер Apple MacBook с процессором класса Intel Core 2 Duo — 2 GHz) и аппроксимации результатов функцией $y = a\sqrt{x^3}$.

Таким образом, использование традиционных интервальных вычислений также позволяет получить правильный результат в случае вычисления полинома Румпа, но при этом требуются существенно повышенная разрядность вычислений (примерно в 2 раза) и значительно увеличенные затраты времени (более чем на порядок).

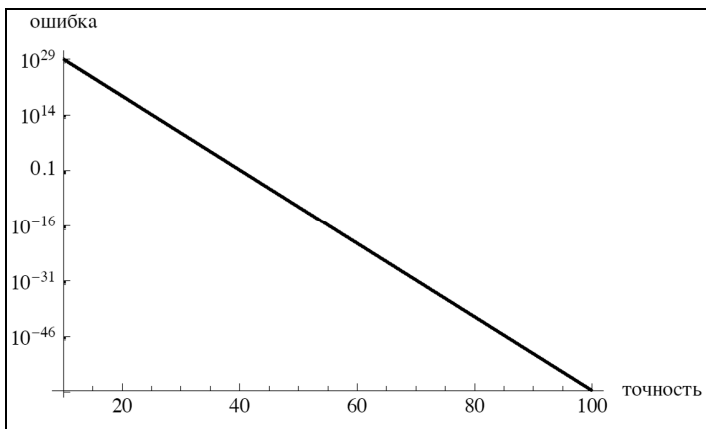


Рисунок 4.10 — Зависимость ошибки (итоговая ширина интервала) от точности (значение Precision, разрядов) при использовании интервальной арифметики для вычисления полинома Румпа

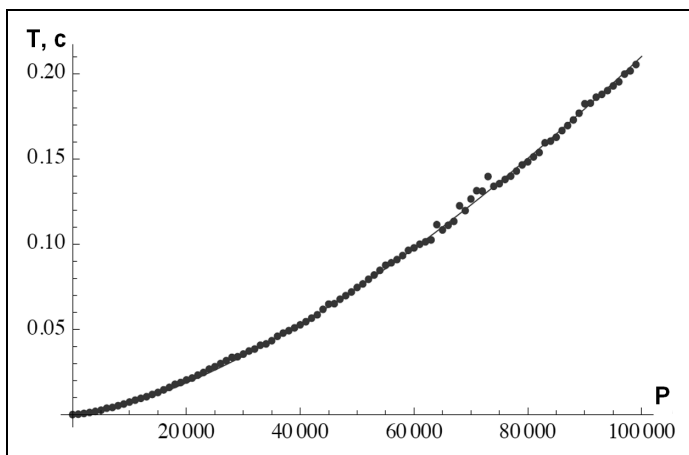


Рисунок 4.11 — Зависимость времени вычисления полинома Румпа (Т, секунд) от задаваемой фиксированной точности (Р — Precision, количество десятичных разрядов) при использовании интервальной арифметики

4.5. Постбинарные вычисления и преодоление ограничений разрядности

Полином Румпа является всего лишь моделью, демонстрирующей тот печальный факт, что современные вычисления весьма уязвимы и практически полностью не защищены от появления такого рода грубейших ошибок, risking остаться в большинстве случаев просто незамеченными вплоть до момента катастрофического проявления некорректных результатов вычислений. Другие примеры такого рода приведены и в работах [33–35]. Самое тревожное во всем этом то, что вероятность таких ошибок пока еще растет практически прямо пропорционально росту вычислительных мощностей современных компьютерных систем.

Как показали исследования, корректный результат в такого рода ситуациях может быть получен как минимум одним из 3-х следующих способов:

- увеличением разрядности до требуемого для корректных вычислений значения (для полинома Румпа в зависимости от способа вычислений это должна быть или учетверенная или превышающая ее точность).
- вычислением отдельно числителя и знаменателя с получением результата их деления только на самом последнем шаге;
- использованием интервальных вычислений.

Однако вычислительная производительность при использовании перечисленных способов на современных компьютерных системах в среднем в разы ниже, чем при выполнении обычных вычислений, ввиду преимущественно программной их реализации. Поэтому требуется поиск кардинально новых решений, позволяющих без существенного падения производительности добиться доста-

точно надежного и эффективного контроля за корректностью вычислений. Наиболее перспективным направлением при этом представляется разработка и реализация постбинарных методов вычислений, основанных на постбинарном представлении количественных значений [14, 16].

Следует отметить, что и в рамках традиционных вычислений уже делались достаточно успешные попытки преодоления проблем, связанных с ограничением разрядности традиционных компьютерных вычислений. Например, разрабатывавшиеся под руководством академика В.М. Глушкова в 1961–1981 гг. в Институте кибернетики НАН Украины ЭВМ серии МИР (серийный выпуск с 1965 года, всего выпущено свыше 3-х тысяч) обеспечивали достаточно эффективную реализацию работы с вещественными числами произвольной разрядности и целыми числами неограниченной разрядности. Кроме этого, была реализована поддержка точных операций над дробными рациональными числами и др. Компьютеры серии МИР, не имевшие аналогов в мире, патентно чистые и защищенные многочисленными авторскими свидетельствами СССР и других стран, были отмечены в 1968 году Государственной премией СССР. Это был первый случай в стране, когда такого рода наградой была отмечена работа в области вычислительной техники. Но существенного продолжения в дальнейшем эти разработки не получили. Одной из причин было то, что такого рода проекты существенно опережали свое время и плохо вписывались в парадигму традиционных бинарных вычислений.

В настоящее время уже созрели все предпосылки для существенной модификации всей системы компьютерных вычислений с целью повышения ее надежности и адекватности современным требованиям.

В современных условиях речь может идти о дальнейшем развитии как логической, так и вычислительной составляющей современного компьютеринга [14]. Особо актуальной представляется при этом задача такой модификации вычислений с плавающей запятой, которая позволила бы исключить рассмотренные выше ситуации. Наиболее перспективный вариант модификации — расширение существующих стандартных форматов представления чисел с плавающей запятой и алгоритмов работы с ними путем введения следующих изменений:

1. Для обеспечения совместного гибкого использования различных форматов чисел с плавающей запятой, кроме стандартных полей мантиссы и порядка со знаками, **вводится поле идентификатора формата**, для которого в стандартных форматах могут быть задействованы части младших разрядов мантиссы (рис. 4.12) и **которое в общем случае состоит из кода и модификатора формата** (таблицы 4.1 и 4.2).

2. Идентификаторы формата определяют разрядность и форму представления чисел с плавающей запятой в диапазоне от 16-ти двоичных разрядов (binary16) до неопределенно больших значений. При этом вплоть до разрядности 128 (binary128) **обеспечивается совместимость с существующими форматами** (таблица 4.1), а начиная с разрядности binary256 обеспечивается **последовательное удвоение разрядности формата по мере необходимости** с соблюдением следующих правил роста разрядности отдельных элементов формата (при каждом удвоении): код и модификатор формата увеличиваются в совокупности на 2 разряда, порядок (экспонента) — на 4 разряда, под мантиссу используются все оставшиеся разряды формата, что обеспечит постепенное увеличение удельной разрядности

мантииссы по мере роста совокупной разрядности представления чисел.

3. Кроме традиционного бинарного (точечного) представления численных значений, дополнительно **вводятся «парные форматы»**, в которых путем удвоения каждого из разрядов в виде единого «постбинарного значения» представляются пары традиционных точечных значений, интерпретируемые **либо как числитель и знаменатель обыкновенной дроби, либо как граничные значения интервалов** (для поддержки интервальных вычислений).

4. В качестве основного постбинарного формата рассматривается тетракод [14], на базе которого может быть реализовано представление «нормированных интервалов» (равных весовым коэффициентам двоичных разрядов) и своеобразное маскирование реально значимых разрядов (за счет использования значения «неопределенности» для разрядов не заданных и не полученных в явном виде, а появившихся, например, в результате дополнения разрядности числа до стандартной или при выравнивании порядков).

5. В алгоритмы выполнения арифметических операций вносятся такие изменения, которые позволят автоматически наращивать (и сокращать) разрядность по мере необходимости, что позволит **реализовать арифметику «без округлений»**, а, следовательно, и без потери точности.

6. Основаниями для изменения разрядности могут быть как возрастание требуемой разрядности мантииссы (например, при умножениях и возведениях в степень), так и возрастание требуемой разрядности порядка, что позволит исключить такие характерные для вычислений с плавающей запятой явления как исчезновение и переполнение порядка.

Таблица 4.1 — Предлагаемая последовательность форматов чисел с плавающей запятой: жирным шрифтом выделены форматы стандарта IEEE 754:2008, в столбце «Разрядность мантиссы» вторые строки представляют модифицированную разрядность мантиссы с учетом бит, выделяемых для модификатора и кода формата

Формат Код формата CF	Точность	Разрядность мантиссы со знаком (модифицированная мантисса + код CF и модификатор MF формата)	Разрядность порядка	Смещение порядка
pbinary16* —	Half (половинная точность)	11 (10+0+0)*	5	+16
pbinary32 0	Single (одинарная точность)	24 (22+1+1)	8	+127
pbinary64 01	Double (двойная точность)	53 (49+2+2)	11	+1023
pbinary128 011	Quadruple (квадроточность)	113 (105+3+5)	15	+16383
pbinary256 0111	Octuple (октоточность)	236 (220+4+12)	20	+524287

* — формат pbinary16 используется только как часть формата pbinary32, в котором он занимает 15-разрядное поле (формат binary16 без младшего разряда мантиссы).

Соответствующая модификация форматов, алгоритмов и арифметико-логических устройств позволит существенно повысить надежность вычислений в целом, в том числе в случаях, аналогичных примеру Румпа.



Рисунок 4.12 — Предлагаемая модификация формата чисел с плавающей запятой (на примере числа половинной точности): слева – традиционное представление, справа – формат с поддержкой постби-нарного кодирования, содержащий на месте младших разрядов мантииссы код формата

Детальный анализ основных проблем, связанных с традиционными вычислениями с плавающей запятой, приведен в фундаментальном издании «Искусство программирования» Дональда Кнута [36]. В частности, он отмечает следующее: «Вычисления над числами в формате с плавающей точкой неточны по самой своей природе, и программисту нетрудно столь неудачно организовать их выполнение, что полученные результаты будут почти полностью состоять из «шума». Одна из главных проблем численного анализа состоит в анализе точности результатов тех или иных численных методов; сюда же относится и проблема «степени доверия»: мы не знаем, насколько правильны результаты вычислений на компьютере... Многие из серьезных математиков пытались строго проанализировать последовательность операций с плавающей точкой, но, обнаружив, что задача слишком сложна, удовлетворялись правдоподобными рассуждениями» [36, с.265]. Слова эти, впервые написанные достаточно давно, актуальны и в настоящее время. Проведенное в современных условиях

исследование примера Румпа весьма наглядное тому подтверждение.

С другой стороны, возможности формального доказательства корректности получаемых результатов также имеют тенденцию к резкому сужению, о чем профессор факультета математики Лондонского королевского колледжа Брайан Дэвис в своей довольно резонансной статье «Куда идет математика?» [37] пишет следующее: «Будущее чистой математики должно разительно отличаться от ее прошлого. В 1875 году любой грамотный математик мог полностью усвоить доказательства всех существовавших на тот период теорем за несколько месяцев. В 1975 году, за год до того как была доказана теорема о четырех цветах, об этом уже не могло быть и речи, однако отдельные математики еще могли теоретически разобраться с доказательством любой известной теоремы. К 2075 году многие области чистой математики будут построены на использовании теорем, доказательства которых не сможет полностью понять ни один из живущих на Земле математиков — ни в одиночку, ни коллективными усилиями» (цитируется по работе [38]). А это фактически означает, что в условиях широкого распространения сугубо компьютерных методов не только для сложных и сверхсложных вычислений, но и для математических доказательств различного уровня сложности требования к уровню достоверности вычислительных процессов существенно возрастают.

В. М. Юровицкий в своих выводах еще более категоричен: «Цивилизационное развитие уже упирается в барьер нынешнего понятия числа (рационального числа). Задачей номер 1 является создание новой концепции числа и методов его обработки. Это будет величайший историко-цивилизационный поворот, переход в **новую числовую эпоху**. Такой переход будет не менее значим, чем переход

от римской системы счисления к арабской, создавшей базу промышленной революции, чем переход от ручного счета к компьютерному» [39]. Но концепция так называемых метрологических чисел, предлагаемая В.М. Юровицким, являясь шагом в правильном направлении, в современных условиях не может рассматриваться в качестве достаточного и окончательного решения.

Речь должна идти о переходе к постбинарным вычислениям в самом широком контексте: начиная от постбинарной компьютерной логики [14] и заканчивая существенным расширением форм и форматов компьютерного представления чисел, на базе чего должен в конечном счете сформироваться **постбинарный компьютеринг**, включающий существующие сегодня формы компьютеринга лишь в качестве частного случая.

4.6. Реализация постбинарных форматов чисел с плавающей запятой

В современных условиях в качестве одного из решений проблем, связанных с форматом представления чисел с плавающей запятой, может выступать дальнейшее развитие как логической, так и вычислительной составляющей современного компьютеринга [14]. В работе [40] были предложены способы для преодоления проблем, связанных с ограничением разрядности чисел, поскольку использование при вычислениях разрядности, существенно превышающей стандартную, приводит к получению правильных результатов. К таким способам можно отнести операции, приводящие к увеличению (или выравниванию) разрядности во избежание переполнения порядка результата и выполнения корректных вычислений, выполнение отложенного деления (числитель и знаменатель вычисляются от-

дельно, а деление откладывается на заключительный этап вычислений), а также использование интервальных вычислений (см. главу 3, а также [32, 41]).

Реализация подобных операций была заложена в разработку постбинарных методов вычислений, основанных на постбинарном представлении количественных значений [14, 16]. Поэтому на основании форматов чисел различной точности стандарта IEEE 754–2008 было предложено пять постбинарных форматов, общая схема полей которых представлена на рис. 4.13, а размерность полей — в таблице 4.2.

Для обеспечения совместного гибкого использования различных форматов чисел с плавающей запятой вводится поле идентификатора формата, для которого в стандартных форматах может быть задействована часть младших разрядов мантиссы M и которое в общем случае состоит из кода (определителя формата числа) CF и модификатора MF формата. Мантисса, уступившая часть младших разрядов для значений полей CF и MF , становится модифицированной M^m . По значениям поля MF формат числа уточняется и может быть представлен как «парный формат», в котором путем удвоения каждого из разрядов в виде единого «постбинарного значения» представляются пары чисел, интерпретируемые либо как числитель и знаменатель обыкновенной дроби, либо как граничные значения интервалов (для поддержки интервальных вычислений) [40].

На основании разработанных форматов представления вещественных чисел средствами языка VHDL [42] описана модель устройства, выполняющего преобразование чисел в условный постбинарный тип данных (одно значение сформированного тетракода (см. раздел 1.7) условно кодируется одним двоичным разрядом) с дальнейшим использованием полученного формата в интервальных вычислениях.

Фактически разработан алгоритм преобразования одного постбинарного числа, которое, в конечном счете, представляет собой значения границ интервала. Структура модели представлена на рис. 4.14.

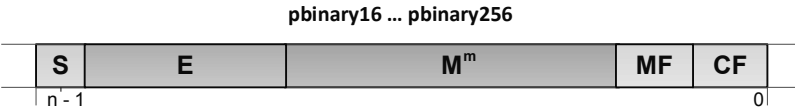


Рисунок 4.13 — Предлагаемые варианты модифицированных форматов чисел с плавающей запятой (S, E, M^m — знак, порядок и мантисса числа, MF, CF — модификатор и формат числа, (n – 1) — номер старшего бита формата)

Таблица 4.2 — Разрядность полей постбинарных форматов чисел (в скобках возле количества разрядов модифицированной мантиссы M^m приведено количество разрядов традиционной мантиссы M числа соответствующей точности)

Формат	Количество двоичных разрядов полей формата					
	n	S	E	M ^m (M)	MF	CF
pbinary16	16	1	5	10 (10)	0	0
pbinary32	32	1	8	21 (23)	1	1
pbinary64	64	1	11	48 (52)	2	2
pbinary128	128	1	15	104 (112)	5	3
pbinary256	256	1	20	219 (235)	12	4

Рассмотрим работу модели. Пусть на входную 32-разрядную шину inDATA поступило, например, двоичное значение десятичной дроби 0,123456789. Однако данное число не может быть точно представлено в формате одинарной точности (Single), соответствующего стандарту IEEE 754–2008, и после обратного преобразования в десятичный формат, в окне Windows будет возвращено число

0,1234568 (по результатам работы программы IEEE 754 v.1.00 [43]), а действительное представление полученного числа будет следующим: +0,12345679104328155517578125.

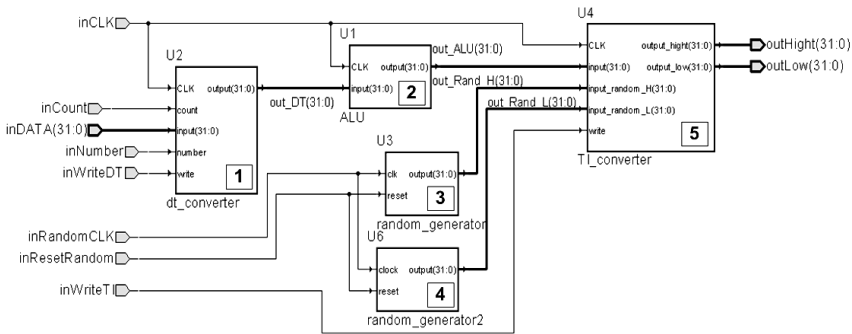


Рисунок 4.14 — Схема преобразователя (dt_converter (1) – блок преобразования входного вещественного числа в постбинарное, ALU (2) – арифметико-логическое устройство, random_generator (3 и 4) – генераторы псевдослучайных чисел, TI_converter (5) – блок разложения постбинарного числа на границы интервала)

Полученное число несет в себе ошибку точности представления вещественного числа формата Single. Во избежание такого рода ошибок в блоке dt_converter происходит преобразование бинарного числа в постбинарное, в котором каждый бит заменяется значением тетракода, отображающего одно из четырех состояний тетралогии: логические ноль и единица, состояния множественности (многозначности) и неопределенности (см. раздел 1.2). Значения множественности заносятся в те разряды мантиссы, начиная с которых возникает возможная ошибка округления (номер бита, с которого начинаются разряды множественности, и количество таких разрядов поступают на соответствующие входы inNumber и inCount блока dt_converter).

Младшие разряды, начиная от последнего разряда множественности, заполняются значениями неопределенности, поскольку уже не являются значащими. Так, двоичное представление числа 0,123456789 в 32-разрядном формате с плавающей запятой выглядит следующим образом (значение на шине inDATA):

0 01111011 11111001101011011101010.

После обработки в блоке dt_converter на шину outDT поступает постбинарное число, которое имеет следующий вид: 0 01111011 11111001101011011ММАААА.

В блоке TI_converter происходит разбиение полученного постбинарного числа на два бинарных, представляющих собой границы интервального результирующего числа. Процедура перехода постбинарного числа в интервальное происходит по принципу, описанному в разделе 4.3 (а также в [16]). При этом все разряды множественности заменяются либо двоичным «0» для формирования левой (нижней) границы интервала, либо двоичной «1» для формирования правой (верхней) границы интервала. Значения неопределенности для каждой границы заполняются случайными двоичными значениями, которые формируются в соответствующих генераторах псевдослучайных чисел. Полученные значения границ интервала поступают на соответствующие 32-разрядные выходные шины outLow и outHight.

На рис. 4.15 представлена временная диаграмма работы модели данного устройства, из которой видны все промежуточные значения и значения результирующего интервала (подчеркнутые поля чисел могут быть заполнены случайным набором двоичных значений):

[0 01111011 11111001101011011001111;
0 01111011 11111001101011011110111].

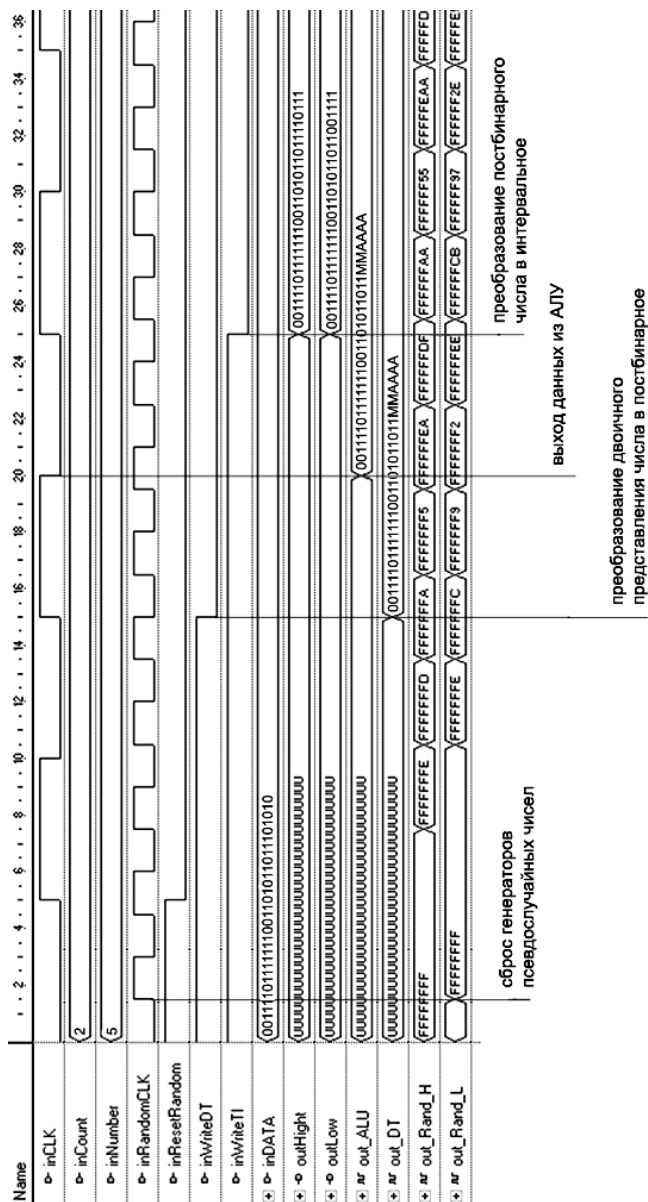


Рисунок 4.15 — Временная диаграмма работы преобразователя

Преобразовав данный интервал в 10 с/с, получим следующую пару значений: $[0.1234566, 0.1234569]$, представляющих собой границы интервала, имеющего достаточно узкую ширину $wid = 2,458692 \cdot 10^{-7}$.

При этом выполняется следующее условие:

$$0,123456789 \in [0.1234566, 0.1234569],$$

свидетельствующее о том, что полученный интервал представляет исходное число, являющееся элементом множества R действительных чисел, на множестве $I(R)$ интервальных чисел.

4.7. Способы представления вещественных чисел в постбинарных форматах

В разделе 4.9 (а также в работе [40]) был предложен ряд способов для преодоления проблем, связанных с ограничением разрядности чисел, поскольку использование при вычислениях разрядности, существенно превышающей стандартную, является одним из способов получения правильных результатов. Все эти способы в совокупности позволяют в процессе вычислений выполнять следующие операции:

- **увеличение (или выравнивание) разрядности** во избежание переполнения разрядов результата и выполнения корректных вычислений;
- выполнение так называемого **отложенного деления**, когда отдельно вычисляются числитель и знаменатель, а деление производится на последнем шаге вычисления;
- использование **интервальных вычислений**.

В рамках реализации вышеперечисленных операций возможно достижение достаточно надежного и эффективного контроля за корректностью вычислений. Сами же

операции можно эффективно использовать при разработке и реализации **постбинарных методов вычислений**, основанных на постбинарном представлении количественных значений [14, 16]. Однако введение данных операций в постбинарный вычислительный процесс невозможно без незначительных модификаций самих форматов чисел с плавающей запятой. Поэтому на основании форматов чисел (binary32, binary64, binary128) стандарта IEEE754-2008 было предложено 5 модифицированных (постбинарных) форматов чисел различной точности (рис. 4.16): от одинарной (pbinary32) до увеличенной в 8 раз по сравнению со стандартной (pbinary256).

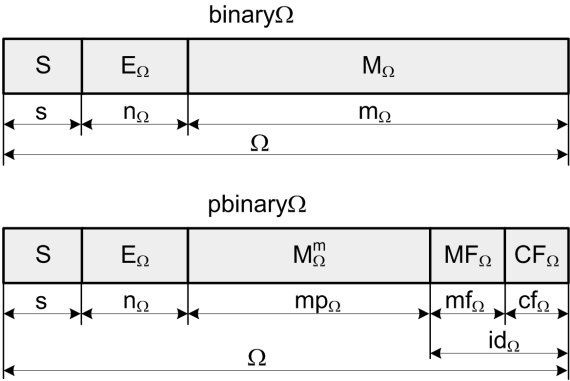


Рисунок 4.16 — Структура числа в формате binary Ω и в эквивалентном ему формате pbinary Ω (Ω — количество разрядов формата числа, S — знак, E $_{\Omega}$ — смещенная экспонента, M $_{\Omega}$ и M $^m_{\Omega}$ — остаток мантиссы соответствующих форматов, MF $_{\Omega}$ — модификатор формата, CF $_{\Omega}$ — код формата)

В предложенных постбинарных форматах поле мантиссы числа претерпело модификацию путем выделения

необходимого количества бит для **идентификатора** формата (поле id_Ω), состоящего из **модификатора** MF_Ω (внутренняя модификация формата rbinary_Ω) и **кода формата** CF_Ω (признак принадлежности к формату rbinary_Ω).

Формирование кода идентификатора формата CF_Ω определено таким образом (см. табл. 4.4), что по положению младшего нуля относительно указателя p на начальный (первый младший) бит числа можно точно определить как сам формат числа rbinary_Ω , так и его границы в диапазоне $[p + \Omega - 1 : p]$.

В то же время в зависимости от модификатора MF_Ω , формат rbinary_Ω может содержать в себе другие подформаты представления данных. Для таких форматов приняты обозначения в виде $\text{rbinary}_\Omega/\Psi\phi$, где Ψ — точность формата, в 2 (для $\phi \in \{f, i, p\}$) или 4 (для $\phi \in \{fp, ip\}$) раз меньшая Ω и показывающая формат составной части чисел, заключенных в определяемую разрядность Ω ; ϕ — указатель типа числа с плавающей запятой (представлены следующие типы: f — дробное число, i — интервальное, p — постбинарное) или типа постбинарного числа (представлены fp — постбинарное дробное и ip — постбинарное интервальное). В таблицах 4.3 и 4.4 приведены все разработанные модификации форматов чисел различной точности в зависимости от значения модификатора MF_Ω .

Рассмотрим структуру, особенности формирования и назначение каждой из представленных модификаций постбинарного формата rbinary_Ω (в качестве примера на рис. 4.17 представлены все модификации формата rbinary128):

- **pbinary_Ω** — фактически число формата binary_Ω , имеющее модифицированную (уменьшенную на id_Ω разрядов) мантиссу. Для постбинарного формата rbinary_Ω спра-

ведливы следующие соотношения, связанные с определением количества разрядов полей после модификации:

$id_{\Omega} = \frac{\Omega}{16}$ — разрядность идентификатора формата;

$cf_{\Omega} = \log_2 id_{\Omega} = \log_2 \frac{\Omega}{16}$ — разрядность кода формата;

$mf_{\Omega} = id_{\Omega} - cf_{\Omega} = \frac{\Omega}{16} - \log_2 \frac{\Omega}{16}$ — разрядность модификатора формата;

$mp_{\Omega} = m_{\Omega} - id_{\Omega}$ — разрядность модифицированной мантиссы.

Таблица 4.3 — Предлагаемые модификации формата $pbinary_{\Omega}$ (при $\Omega = 32$ и $\Omega = 64$)

Модификатор MF[1:0]	pbinary32 MF[0]	pbinary64 MF[1:0]
00	pbinary32	pbinary64
01	pbinary32/16p	pbinary64/32f
10		pbinary64/32i
11		pbinary64/32p

Таблица 4.4 — Предлагаемые модификации формата $pbinary_{\Omega}$ (при $\Omega = 128$ и $\Omega = 256$)

Модификатор MF[11:0]	pbinary128 MF[4:0]	pbinary256 MF[11:0]
0000 ... 0000	pbinary128	pbinary256
0000 ... 0001	pbinary128/64f	pbinary256/128f
0000 ... 0010	pbinary128/64i	pbinary256/128i
0000 ... 0011	pbinary128/64p	pbinary256/128p
0000 ... 0100	pbinary128/32fp	pbinary256/64fp
0000 ... 0101	pbinary128/32ip	pbinary256/64ip
...	резерв	резерв

Таблица 4.5 — Разрядность $\text{rbinary}\Omega$
с указанием значения кода формата

Ω	CF_Ω	s	n_Ω	mp_Ω	id_Ω	mf_Ω	cf_Ω
32	0	1	8	21	2	1	1
64	01	1	11	48	4	2	2
128	011	1	15	104	8	5	3
256	0111	1	20	219	16	12	4

Поля знака (S) и смещенной экспоненты (E_Ω) переносятся в модифицированный формат без изменения. В таблице 4.5 показаны разрядности форматов $\text{rbinary}\Omega$ по отношению к $\text{binary}\Omega$.

Каждое число, представленное в формате $\text{rbinary}\Omega$, имеет знаковый бит S (рис. 4.16): $S = 0$ для положительного и $S = 1$ для отрицательного числа. Значение смещенной экспоненты E_Ω формируется следующим образом:

$$E_\Omega = \text{exp}_2 + \text{offset}_2, \quad (4.29)$$

где exp_2 — значение экспоненты двоичной дроби в нормализованном экспоненциальном виде, offset — заданное смещение экспоненты в Ω -битном формате: $\text{offset}_{10} = 2^{(n_\Omega-1)} - 1 \Rightarrow \text{offset}_2 = (\text{offset}_{10})_2$.

В поле M_Ω записывается остаток мантиссы двоичного нормализованного числа с плавающей точкой (отброшена старшая 1 от нормализованного двоичного числа вида $\pm 1, \text{xx} \dots \text{x} \cdot \text{exp}_2$). В формате полей $\text{rbinary}\Omega$ область нормализованных чисел лежит в пределах: $S = x$ (любое значение); $E_\Omega = 000 \dots 01 \div 111 \dots 10$; $M_\Omega^m = 000 \dots 00 \div 111 \dots 11$.

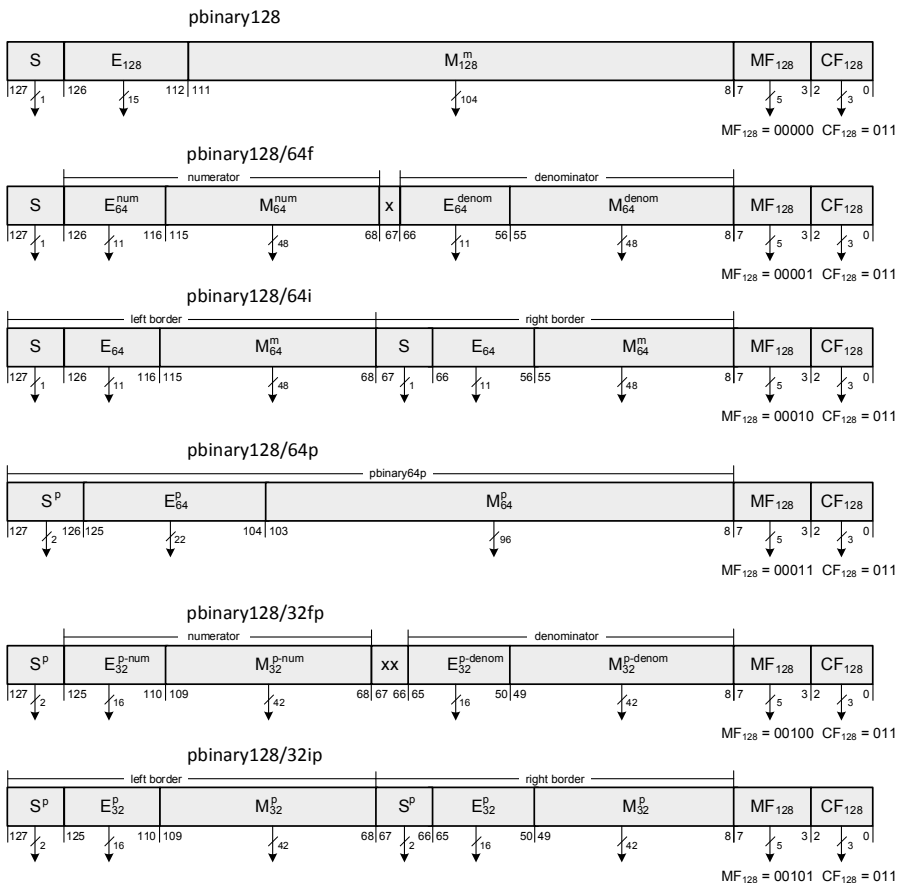


Рисунок 4.17 — Модификации формата pbinary64
(numerator – числитель дроби; denominator – знаменатель дроби; left border – левая граница интервала;
right border – правая граница интервала)

Таким образом, для формата `rbinaryΩ` справедливы следующие соотношения:

– двоичное нормализованное число dn :

$$dn = (-1)^S \cdot 1, M_{\Omega}^m \cdot \exp_2^{(E_{\Omega} - \text{offset}_2)}; \quad (4.30)$$

– десятичное нормализованное число xn :

$$xn = (-1)^S \cdot 2^{((E_{\Omega})_{10} - \text{offset}_{10})} \cdot \left(1 + \frac{(M_{\Omega}^m)_{10}}{2^{\text{mp}_{\Omega}}} \right); \quad (4.31)$$

– абсолютная максимально возможная погрешность Δ_{\max} представления числа равна половине шага числа, т. е. половине величины наименьшего разряда:

$$\Delta_{\max} = 2^{((E_{\Omega})_{10} - \text{offset}_{10} - \text{mp}_{\Omega} + 1)}; \quad (4.32)$$

– относительная максимально возможная погрешность $\delta_{N_{\max}}$ представления нормализованного числа:

$$\delta_{N_{\max}} = \frac{1}{2^{\text{mp}_{\Omega}} + (M_{\Omega}^m)_{10}} \cdot 100\%. \quad (4.33)$$

Аналогично `binaryΩ`, формат `rbinaryΩ` имеет ряд исключительных чисел, к которым нельзя применять формулы (4.30–4.33):

1) Положительный и отрицательный ноль: $+0$ ($S = 0$; $E_{\Omega} = 0$; $M_{\Omega}^m = 0$) и -0 ($S = 1$; $E_{\Omega} = 0$; $M_{\Omega}^m = 0$). Большинство программных средств эти нули не различает (поскольку в этом нет необходимости), считая их просто нулевыми значениями.

2) Положительная и отрицательная бесконечности ($+\text{Infinity}$, $-\text{Infinity}$): $+\infty$ ($S = 0$; $E_{\Omega} = 111\dots 11$; $M_{\Omega}^m = 0$) и $-\infty$ ($S = 1$; $E_{\Omega} = 111\dots 11$; $M_{\Omega}^m = 0$). Это числа, которые больше границ диапазона представления чисел.

3) Не числа — NaN (No a Numbers), к которым относятся символы, или результаты недопустимых операций. При $E_{\Omega} = 111\dots 11$ и M_{Ω}^m — любое ненулевое значение, различают +NaN ($S = 1$) и -NaN ($S = 0$).

4) Ненормализованные числа — числа, мантиссы которых лежат в диапазоне $0,1 \leq M_{\Omega}^m < 1$. Ненормализованные числа находятся ближе к нулю, чем нормализованные, и разбивают минимальный разряд нормализованного числа на некоторое подмножество. Для ненормализованных чисел справедливы следующие равенства:

– двоичное ненормализованное число dd :

$$dd = (-1)^S \cdot 0, M_{\Omega}^m \cdot \exp_2^{-\text{offset}_2}; \quad (4.34)$$

– десятичное ненормализованное число xd :

$$xd = (-1)^S \cdot 2^{(1-\text{offset}_{10})} \cdot \frac{(M_{\Omega}^m)_{10}}{2^{\text{mp}_{\Omega}}}; \quad (4.35)$$

– относительная максимально возможная погрешность $\delta_{D\max}$ представления ненормализованного числа:

$$\delta_{D\max} = \frac{1}{2 \cdot (M_{\Omega}^m)_{10}} \cdot 100\%; \quad (4.36)$$

Формула (4) справедлива также для ненормализованных чисел.

На рис. 4.18 представлено отображение чисел форматов binary_{Ω} и rbinary_{Ω} на числовой оси.

Совокупности полей $\pm\text{Norm}$ и $\pm\text{Denorm}$ ($[-xn_{\max}^{\text{pb}}; -xd_{\min}^{\text{pb}}] \cup [+xd_{\min}^{\text{pb}}; +xn_{\max}^{\text{pb}}]$) являются диапазоном представления положительных или отрицательных чисел формата rbinary_{Ω} на всей числовой оси. Используя соотношения (4.31) и (4.35), можно получить десятичные зна-

чения модулей минимальной xd_{\min}^{pb} (значение по модулю наименьшего ненормализованного числа) и максимальной xn_{\max}^{pb} (значение по модулю наибольшего нормализованного числа) границ диапазона представления чисел в формате $\text{rbinary}\Omega$ (табл. 4.6, 4.7). Модификация (т. е. уменьшение разрядности) мантииссы постбинарного формата приводит прежде всего к сужению диапазона представления близких к нулю чисел (область ненормализованных чисел) по отношению к эквивалентному двоичному формату $\text{binary}\Omega$. Используя формулы (4.35, 4.31), получим абсолютные погрешности границ диапазона представления чисел по отношению к формату $\text{binary}\Omega$ (xn_{\max} , xd_{\max} и xd_{\min} — значения модулей соответствующих границ диапазона чисел формата $\text{binary}\Omega$):

— абсолютная погрешность представления минимального (минимального ненормализованного) Ω -разрядного числа:

$$\Delta d_{\min} = \left| xd_{\min} - xd_{\min}^{\text{pb}} \right| = 2^{(1 - \text{offset}_{10} - \text{mp}_{\Omega})} \cdot (1 - 2^{-\text{id}_{\Omega}}); \quad (4.37)$$

— абсолютная погрешность представления максимального (максимального нормализованного) Ω -разрядного числа:

$$\Delta n_{\max} = \left| xn_{\max} - xn_{\max}^{\text{pb}} \right| = 2^{(\text{offset}_{10} - \text{mp}_{\Omega})} \cdot (1 - 2^{-\text{id}_{\Omega}}). \quad (4.38)$$

Абсолютная погрешность представления максимального ненормализованного Ω -разрядного числа Δd_{\max} соответствует формуле (4.37), следовательно $\Delta d_{\max} = \Delta d_{\min}$. Последнее равенство можно объяснить тем, что разница в представлении ненормализованных чисел $\text{rbinary}\Omega$ и $\text{binary}\Omega$ обусловлена одинаковой погрешностью представления значений мантиисс постбинарного и бинарного фор-

матов (все ненормализованные числа имеют нулевое значение порядка) на протяжении всего диапазона $\pm\text{Denorm}$. Таким образом,

$$\begin{aligned} [xd_{\min}^{\text{pb}}; xd_{\max}^{\text{pb}}] &= [xd_{\min} + \Delta d_{\min}; xd_{\max} - \Delta d_{\max}] = \\ &= [xd_{\min} + \Delta d_{\min}; xd_{\max} - \Delta d_{\min}] = [2^{\text{id}_{\Omega}} \cdot xd_{\min}; 2^{-\text{id}_{\Omega}} \cdot xd_{\max}]. \end{aligned} \quad (4.39)$$

Равенство (4.39) показывает, что модуль минимального ненормализованного числа формата rbinary_{Ω} больше модуля аналогичного числа формата binary_{Ω} в $2^{\text{id}_{\Omega}}$ раза, и наоборот, модуль максимального ненормализованного числа формата rbinary_{Ω} меньше модуля аналогичного числа формата binary_{Ω} в $2^{\text{id}_{\Omega}}$ раза. Таким образом, диапазон $\pm\text{Denorm}$ формата rbinary_{Ω} уже соответствующего диапазона формата binary_{Ω} в $2^{\text{id}_{\Omega}+1}$ раза.

Отношение максимальных границ диапазона нормализованных чисел для форматов rbinary_{Ω} и binary_{Ω} близко к единице, однако следует учитывать величину абсолютной погрешности представления числа в постбинарном и бинарном форматах. Десятичное значение минимального нормализованного числа rbinary_{Ω} соответствует аналогичному числу binary_{Ω} ($\pm xn_{\min}^{\text{pb}} = \pm xn_{\min}, \Delta n_{\min} = 0$), поскольку в формировании этих чисел участвуют только значения полей знака и порядка, которые идентичны в соответствующих постбинарных и бинарных форматах (рис. 4.16). Таким образом,

$$\begin{aligned} [xn_{\min}^{\text{pb}}, xn_{\max}^{\text{pb}}] &= [xn_{\min}, xn_{\max} - \Delta n_{\max}] = \\ &= \left[xn_{\min}, \frac{2^{-(\text{mp}_{\Omega}+1)-\text{id}_{\Omega}} - 1}{2^{-(\text{mp}_{\Omega}+1)} - 1} \cdot xn_{\max} \right]. \end{aligned} \quad (4.40)$$

Рассмотрим функцию $f(a, b) = \frac{2^{-(a+1)-b} - 1}{2^{-(a+1)} - 1}$. Имеем $\lim_{a \rightarrow \infty} f(a, b) = 1$, откуда следует, что $f(a, b) \approx 1$ при возрастающем аргументе a . Соответственно, для значений mp_Ω и id_Ω представленных форматов справедливо соотношение

$$\frac{2^{-(\text{mp}_\Omega+1)-\text{id}_\Omega} - 1}{2^{-(\text{mp}_\Omega+1)} - 1} \approx 1, \quad (4.41)$$

причем данное приближенное равенство стремится к строгому равенству при возрастании значения mp_Ω .

Исходя из (4.41), выражение (4.40) можно записать в виде

$$\begin{aligned} [xn_{\min}^{\text{pb}}, xn_{\max}^{\text{pb}}] &= [xn_{\min}, \approx xn_{\max}] \Rightarrow \\ \Rightarrow [xn_{\min}^{\text{pb}}, xn_{\max}^{\text{pb}}] &\approx [xn_{\min}, xn_{\max}]. \end{aligned} \quad (4.42)$$

Десятичные значения максимальных и минимальных чисел формата rbinary_Ω , их абсолютной погрешности по отношению к формату binary_Ω представлены в таблицах 4.6 и 4.7.

- **$\text{rbinary}_\Omega/\Psi\text{f}$** — дробное Ω -разрядное число, состоящее из числителя (numerator), знаменателя (denominator) Ψ -разрядной точности (т. е. двух чисел формата rbinary_Ψ) и общего знака для дроби. Значения Ω и Ψ связаны следующим соотношением: $\Psi = \Omega/2$ при $\Omega = \{64, 128, 256, \dots\}$.

Пусть a и b — числа формата binary_Ψ , т. е. фактически числа стандарта IEEE 754, над которыми нужно выполнить операцию деления a/b .

В ряде случаев, обусловленных прежде всего сложностью общего цикла вычислений, данную операцию можно отложить, т. е. сохранить результат в виде дроби a/b до заключительного этапа вычислений.

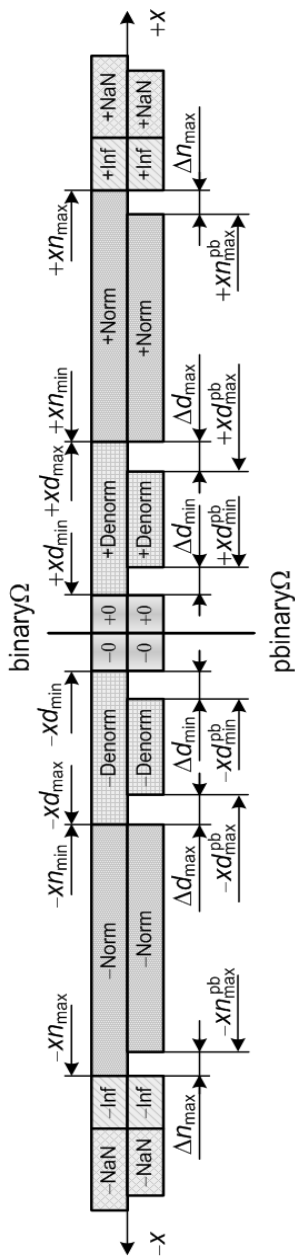


Рисунок 4.18 — Отображение чисел форматов $\text{binary}\Omega$ и $\text{pbinary}\Omega$ на числовой оси (Norm, Denorm — области нормализованных и ненормализованных чисел; pb — признак постбинарного формата; $x\eta$ и $x\eta^{\text{pb}}$ — ненормализованные и нормализованные числа)

Таблица 4.6 — Значения границ диапазона ненормализованных чисел ($\pm\text{Denorm}$) формата $\text{rbinary}\Omega$ (в скобках указаны значения абсолютной погрешности представления чисел в сопоставлении со стандартным форматом Ω -разрядной точности)

Ω	$xd_{\min}^{\text{pb}} (\Delta d_{\min})$	xd_{\max}^{pb}
32	$\pm 2^{-147} \approx \pm 5,60519386 \cdot 10^{-45}$ ($\approx 4,204 \cdot 10^{-45}$)	$\pm 2^{-126} \cdot (1 - 2^{-21}) \approx$ $\approx \pm 1,17549379 \cdot 10^{-38}$
64	$\pm 2^{-1070} \approx \pm 7,90505033 \cdot 10^{-323}$ ($\approx 7,411 \cdot 10^{-323}$)	$\pm 2^{-1022} \cdot (1 - 2^{-48}) \approx$ $\approx \pm 2,22507386 \cdot 10^{-308}$
128	$\pm 2^{-16486} \approx \pm 1,65764483 \cdot 10^{-4963}$ ($\approx 1,651 \cdot e^{-4963}$)	$\pm 2^{-16382} \cdot (1 - 2^{-104}) \approx$ $\approx \pm 3,36210314 \cdot 10^{-4932}$
256	$\pm 2^{-524505} \approx \pm 1,8286336 \cdot 10^{-157892}$ ($\approx 1,829 \cdot 10^{-157892}$)	$\pm 2^{-524286} \cdot (1 - 2^{-219}) \approx$ $\approx \pm 1,54061213 \cdot 10^{-157826}$

Таблица 4.7 — Значения границ диапазона нормализованных чисел ($\pm\text{Norm}$) формата $\text{rbinary}\Omega$ (в скобках указаны значения абсолютной погрешности представления чисел в сопоставлении со стандартным форматом Ω -разрядной точности)

Ω	xn_{\min}^{pb}	$xn_{\max}^{\text{pb}} (\Delta n_{\max})$
32	$\pm 2^{-126} \approx$ $\approx \pm 1,17549379 \cdot 10^{-38}$	$\pm 2^{127} \cdot (2 - 2^{-21}) \approx \pm 3,40282286 \cdot 10^{38}$ ($\approx 6,085 \cdot 10^{31}$)
64	$\pm 2^{-1022} \approx$ $\approx \pm 2,22507386 \cdot 10^{-308}$	$\pm 2^{1023} \cdot (2 - 2^{-48}) \approx \pm 1,79769313 \cdot 10^{308}$ ($\approx 2,994 \cdot 10^{293}$)
128	$\pm 2^{-16382} \approx$ $\approx \pm 3,36210314 \cdot 10^{-4932}$	$\pm 2^{16383} \cdot (2 - 2^{-104}) \approx \pm 1,18973150 \cdot 10^{4932}$ ($\approx 2,921 \cdot 10^{4900}$)
256	$\pm 2^{-524286} \approx$ $\approx \pm 1,54061213 \cdot 10^{-157826}$	$\pm 2^{524287} \cdot (2 - 2^{-219}) \approx \pm 2,5963757 \cdot 10^{157826}$ ($\approx 1,541 \cdot 10^{157760}$)

В таком случае справедливы следующие равенства:

$S = S_{\Psi a} \oplus S_{\Psi b}$ — знак дроби;

$P_{\Psi}^{\text{num}} = P_{\Psi a}$, $M_{\Psi}^{\text{num}} = M_{\Psi a}^m$ — порядок и мантисса числителя дроби;

$P_{\Psi}^{\text{denom}} = P_{\Psi b}$, $M_{\Psi}^{\text{denom}} = M_{\Psi b}^m$ — порядок и мантисса знаменателя дроби.

- **rbinary $\Omega/\Psi i$** — интервальный Ω -разрядный формат числа, содержащий два полноценных числа формата **rbinary Ψ** ($\Psi = \Omega/2$, $\Omega = \{64, 128, 256, \dots\}$), которые являются левой (left border) и правой (right border) границами интервала (см. раздел 3.3). Заполнение полей данного формата аналогично дробному, за исключением наличия знаковых полей для каждого значения границ интервала. Числа в формате **rbinary $\Omega/\Psi i$** могут быть использованы для решения задач в рамках интервального анализа или интервальной математики, где они будут представлять собой интервальный тип данных, вычислительные операции с которыми исключают возможные ошибки округления [41].

- **rbinary $\Omega/\Psi p$** — постбинарное Ψ -разрядное число ($\Psi = \Omega/2$, $\Omega = \{32, 64, 128, 256, \dots\}$). В таком числе в качестве основного постбинарного формата рассматривается тетракод, каждый разряд которого представлен тетритом (см. раздел 2.1), кодирующим одно из четырех состояний: 0, 1, а также состояния «неопределенности» (A) и «множественности» (M).

Поскольку тетрит должен кодировать четыре возможных значения разряда, для его хранения и использования в современных компьютерных системах потребуется два бита, поэтому расширение числа формата **rbinary Ψ** до постбинарного Ω -разрядного формата **rbinary $\Omega/\Psi p$** происходит

за счет удвоения разрядности знака, порядка и модифицированной мантиссы. При этом каждая пара бит постбинарных значений знака, мантиссы и порядка имеет одно из значений множества $\{0, 1, A, M\}$ [15, с.30], кодируемые следующим образом: 00 — состояние неопределенности (A); 01 — ноль (0); 10 — единица (1); 11 — состояние множественности (M).

При формировании разрядности знака и порядка и мантиссы постбинарного числа формата $\text{pbinary}\Omega/\Psi p$ справедливы следующие соотношения: $s_\Omega = s_\Psi^p = 2 \cdot s_\Psi$, $n_\Omega = n_\Psi^p = 2 \cdot n_\Psi$, $mp_\Omega = mp_\Psi^p = 2 \cdot mp_\Psi$.

- **$\text{pbinary}\Omega/\Psi f p$** — представляет формат постбинарного дробного Ψ -разрядного числа ($\Psi = \Omega/4$, $\Omega = \{128, 256, \dots\}$). Формируется объединением описанных выше преобразований $\text{pbinary}\Omega/\Psi f \cup \text{pbinary}\Omega/\Psi p$.

- **$\text{pbinary}\Omega/\Psi i p$** — представляет формат постбинарного интервального Ψ -разрядного числа ($\Psi = \Omega/4$, $\Omega = \{128, 256, \dots\}$). Формирование полей данного числа происходит по схеме: $\text{pbinary}\Omega/\Psi i \cup \text{pbinary}\Omega/\Psi p$.

Стоит отметить, что реализация вычислительных алгоритмов на базе предложенных форматов позволит обеспечить существенное расширение функциональных возможностей перспективных процессоров, в том числе за счет реализации текущего контроля требуемой разрядности и обеспечения на этой базе «гибкого форматирования» и «гибкой разрядности» при работе с компьютерным представлением численной информации. Важнейшим результатом этого должно стать кардинальное повышение надежности вычислений и оптимизация их разрядности.

На первом этапе речь, естественно, может идти преимущественно о программном моделировании новых фор-

матов, алгоритмов и архитектур на базе существующих компьютерных систем. На следующем этапе предполагается реализация соответствующих схемных решений на базе FPGA вплоть до реализации экспериментальных прототипов постбинарных процессоров.

4.8. Выводы

В контексте перехода к постбинарному компьютерингу многообразие представления логических и численных значений будет неизбежно возрастать в связи с необходимостью развития как логических и алгоритмических основ компьютерных технологий, так и самого понятия числа, в том числе на уровне базовых форматов представления информации. Предложенная в работе система компьютерных форматов данных и соответствующая система обозначений могут рассматриваться как прототипы базовых элементов для нового поколения компьютерных архитектур, соответствующих начальному этапу постбинарного компьютеринга.

В перспективе переход от бинарного компьютеринга к постбинарному представляется вполне закономерным (см. раздел 4.1). Во времени этот процесс может растянуться на десятилетия, но начинать его необходимо уже сейчас.

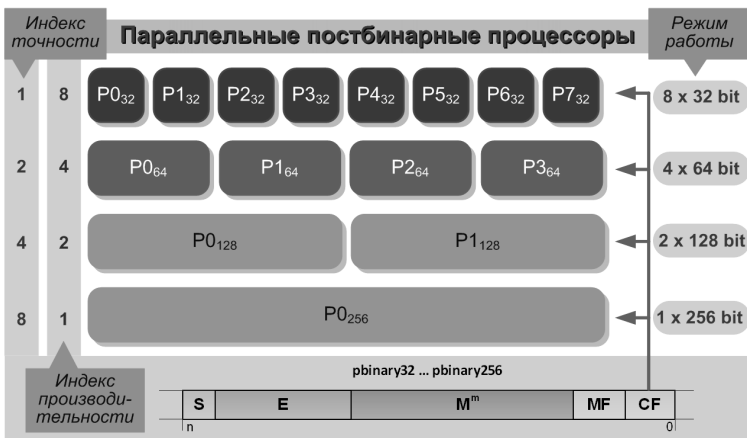
Стоит также отметить, что при вычислении относительно простого полинома Румпа [40] с определенными входными значениями большинство существующих на сегодняшний день математических пакетов не в состоянии получить верный результат. Способы получения верного результата при вычислении полинома Румпа, в принципе, существуют (см. разделы 4.4, 4.5), однако принятие решения о выборе тех или иных способов вычислений полностью возлагается на пользователя, причем такой пользова-

тель должен хорошо понимать суть вычислительных процессов и иметь достаточную математическую подготовку. Естественно, что при нарастании объемов вычислений в процессе исследования, моделирования и проектирования сложных систем и динамических процессов становится невозможным буквально «вручную» отслеживать проблемные участки в вычислениях и выявлять подобные примеру Румпа вычислительные аномалии. При этом в большинстве случаев ошибки в вычислениях остаются просто незамеченными, существенно искажая полученные результаты. Это позволяет предположить, например, что многие техногенные катастрофы последних десятилетий были в первую очередь обусловлены не «человеческим фактором», а разного рода вычислительными ошибками [34].

В данной главе рассмотрена актуальность перехода к постбинарному компьютерингу и «гибкой разрядности» при кодировании количественных значений в современных компьютерных системах и предложены постбинарные форматы кодирования вещественных чисел.

Заключение

Рассмотренные в монографии варианты постбинарной логики, постбинарного кодирования и постбинарных форматов чисел с плавающей запятой могут найти применение в самых различных компьютерных системах и приложениях. Но наиболее эффективно и комплексно они могут быть реализованы в виде параллельных постбинарных процессоров (соответствующая архитектура представлена на рисунке ниже).



Как известно, в современных условиях наиболее экономичным способом достижения максимальной производительности является реализация многопроцессорных систем на 32-разрядных графических процессорах (GPU). Однако во многих случаях такое ограничение разрядности является неприемлемым. Точно определить, какая именно разрядность является достаточной в реальных условиях, довольно сложно. Если же разрядность вычислений заранее выбирать повышенной (64 или 128 в соответствии с имеющимися на сегодня стандартами вычислений с пла-

вающей запятой), то это практически всегда означает снижение производительности соответственно в 2 или 4 раза. Поэтому просто повышение разрядности «с запасом» является довольно неэффективным. При этом и повышенная разрядность (до 128-ми включительно) не всегда является достаточной, что наглядно демонстрирует пример Румпа.

Наиболее эффективным выходом в данном случае является переход к гибкой разрядности, которая может изменяться в процессе вычислений по мере необходимости, что обеспечивается использованием постбинарных форматов и алгоритмов. При этом предполагается модульная организация архитектуры параллельного постбинарного вычислителя, состоящего из 32-разрядных процессорных модулей, «индекс точности» которых принимается за единицу. Для 8-ми параллельно работающих таких модулей «индекс производительности» будет равен 8. Но при необходимости повышения точности вычислений модули могут объединяться вплоть до достижения разрядности 256. На первом этапе реализация соответствующих схемных решений вплоть до реализации экспериментальных прототипов постбинарных процессоров осуществляется на базе FPGA. В дальнейшем постбинарная архитектура вполне может стать основой организации серийно выпускаемых микропроцессоров.

Успешная реализация данного проекта позволит перейти к постбинарным вычислениям в самом широком контексте: начиная от постбинарной компьютерной логики и заканчивая существенным расширением вычислительных возможностей компьютерных систем различных классов, что позволит в конечном счете перейти к постбинарному компьютерингу, позволяющему достичь максимальной эффективности и достоверности массовых параллельных вычислений.

СПИСОК ЛИТЕРАТУРЫ

К главе 1

1. Аноприенко А.Я. Обобщенный кодо-логический базис в вычислительном моделировании и представлении знаний: эволюция идеи и перспективы развития // Научные труды Донецкого национального технического университета. Серия «Информатика, кибернетика и вычислительная техника» (ИКВТ-2005), выпуск 93: — Донецк: ДонНТУ, 2005. 289–316 с.
2. Аноприенко А.Я. Рекурсивное разбиение пространства при описании трехмерных объектов для визуализации в реальном времени // Материалы 4-го научно-технического семинара «Математическое обеспечение систем с машинной графикой». — Устинов. — 1986. — 5–6 с.
3. Аноприенко А.Я. Повышение производительности систем генерации изображений: структуры и алгоритмы на уровне регенерационной памяти. Автореферат диссертации на соискание ученой степени кандидата технических наук по специальности 05.13.13. — Киев, 1987. — 20 с.
4. Аноприенко А.Я., Башков Е.А. Запоминающее устройство с многоформатным доступом к данным. А.с. 1336109 (СССР) / Оpubл. 1987, БИ № 33.
5. Аноприенко А.Я., Башков Е.А. Запоминающее устройство с многоформатным доступом к данным. А.с. 1355997 (СССР) / Оpubл. 1987, БИ № 44.
6. Моисеев Н.Н. Алгоритмы развития (серия «Академические чтения»). — М: Наука, 1987. — 304 с.
7. Аноприенко А.Я., Кухтин А.А. О некоторых возможностях расширения логического базиса информатики. / В кн. Тези доповідей міжнародної науково-практичної конфе-

- ренції “Інформатизація в умовах переходу до ринку”, Київ, 5–6 листопада 1992 р., с. 30–32.
8. Аноприенко А.Я. Тетралогики и тетракоды. / В кн. “Сборник трудов факультета вычислительной техники и информатики”. Вып.1. Донецк, ДонГТУ, 1996, с.32–43.
 9. Аноприенко А.Я. Расширенный кодо-логический базис компьютерного моделирования / В кн. “Информатика, кибернетика и вычислительная техника (ИКВТ-97). Сборник научных трудов ДонГТУ.” Выпуск 1. Донецк, ДонГТУ, 1997, с. 59–64.
 10. Аноприєнко О., Кривошеев С. Тетракоди: новий метод кодування сигналів і зображень. / В кн. “Оброблення сигналів і зображень та розпізнавання образів. Праці всеукраїнської міжнародної конференції УкрОБРАЗ’96. Київ, 1996, с. 15–17.
 11. Аноприенко А. Я., Кривошеев С. В., Приходько Т. А. Тетракоды в кодировании и распознавании образов // Сборник научных трудов ДонГТУ. Серия “Информатика, кибернетика и вычислительная техника”. Выпуск 1 (ИКВТ-97). — Донецк: ДонГТУ. — 1997. — 99–104 с.
 12. Аноприенко А.Я. О некоторых приложениях стохастической геометрии к анализу и синтезу вычислительных систем и алгоритмов // Сборник трудов факультета вычислительной техники и информатики. Вып.1. — Донецк: ДонГТУ. — 1996. — 129–137 с.
 13. Святный В.А., Цайтц М., Аноприенко А.Я. Реализация системы моделирования динамических процессов на параллельной ЭВМ в среде сетевого графического интерфейса // Вопросы радиоэлектроники, серия “ЭВТ”, вып. 2. — 1991. — 85–94 с.
 14. Аноприенко А.Я., Святный В.А. Универсальные моделирующие среды // Сборник трудов факультета вычислительной техники и информатики. Вып.1. — Донецк: ДонГТУ. — 1996. — 8–23 с.

15. Аноприенко А.Я., Святный В.А. Высокопроизводительные инфомационно-моделирующие среды для исследования, разработки и сопровождения сложных динамических систем // Научные труды Донецкого государственного технического университета. Выпуск 29. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» — Севастополь: «Вебер». — 2001. — 346–367 с.
16. Аноприенко А.Я. Эволюция алгоритмического базиса вычислительного моделирования и сложность реального мира // Научные труды Донецкого национального технического университета. Вып. 52. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-2002): Донецк: ДонНТУ, 2002. — 6–9 с.
17. Anoprienko A., Svjatnyi V., Reuter A. Extended logical and numerical basis for computer simulation / "Short Papers Proceedings of the 1997 European Simulation Multiconference ESM'97. Istanbul, June 1–4, 1997" — Istanbul, SCS, 1997, p. 23–26.
18. Anoprienko A. Interpretation of some artefacts as special simulation tools and environments / "Short Papers Proceedings of the 1997 European Simulation Multiconference ESM'97. Istanbul, June 1–4, 1997" — Istanbul, SCS, 1997, p. 23–26.
19. Anoprienko A. Tetralogic and tetracodes: an effective method for information coding // 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics. Berlin, August 24–29, 1997. Vol. 4. Artificial Intelligence and Computer Science. — Berlin: Wissenschaft und Technik Verlag. — 1997. — p. 751–754.
20. Аноприенко А.Я. Компьютерное исследование феноменов астроморфного моделирования в контексте когнитивно-культурной эволюции // Научные труды Донецкого государственного технического университета. Выпуск 29. Серия "Проблемы моделирования и автоматизации проекти-

- рования динамических систем" — Севастополь: «Вебер». — 2001. — С. 327–345.
21. Аноприенко А.Я. От вычислений к пониманию: когнитивное компьютерное моделирование и его практическое применение на примере решения проблемы Фестского диска / В кн. "Информатика, кибернетика и вычислительная техника (ИКВТ-99). Сборник научных трудов ДонГТУ." Выпуск 6. Донецк, ДонГТУ, 1999, с. 36–47.
 22. Аноприенко А.Я. Восхождение интеллекта: эволюция монокодовых вычислительных моделей // Научные труды Донецкого государственного технического университета. Выпуск 15. Серия "Информатика, кибернетика и вычислительная техника" (ИКВТ-2000). — Донецк: ДонГТУ. — 2000. — С. 87–107.
 23. Anoprienko A. Archaeosimulation: new sight on ancient society and lessons for computer era / Problems of Simulation and Computer Aided Design of Dynamic Systems. Scientific Papers of Donetsk State Technical University. Vol. 29. — Sevastopol: Weber, 2001. P. 320–326.
 24. Anoprienko A. The early history of simulation in Europe: scale planetariums and astromorphic models // EUROSIM 2004: 5th EUROSIM Congress on Modeling and Simulation. 06–10 September 2004. ESIEE Paris, Marne la Vallée, France. Book of abstracts. p. 146–147.
 25. Аноприенко А.Я. Модельная и компьютерная поддержка принятия решений в ситуации когнитивного конфликта: рассмотрение на примере сравнительного анализа гипотез о локализации Атлантиды Платона // Научные труды Донецкого национального технического университета. Выпуск 52. Серия "Проблемы моделирования и автоматизации проектирования динамических систем" (МАП-2002): Донецк: ДонНТУ, 2002. — 177–243 с.
 26. Gilles E.D., Kienle A., Waschler R., Sviatnyi V., Anoprienko A., Potapenko V. Zur Entwicklung des Trainingssimulators einer großchemischen Anlage // Problems of Simulation and

- Computer Aided Design of Dynamic Systems (SCAD-2002). Scientific Papers of Donetsk National Technical University. Volume 52. Donetsk, 2002, pages 23–26.
27. Аноприенко А.Я., Забровский С.В., Потапенко В.А. Использование технологии CORBA в распределенном моделировании сложных технологических систем // Наукові праці Донецького державного технічного університету. Серія “Обчислювальна техніка та автоматизація”. Випуск 38. — Донецьк, ДонДТУ, 2002, с. 186–190.
 28. Аноприенко А.Я., Потапенко В.А. Опыт создания распределенных моделирующих сред // Труды международной научно-технической конференции «Современные средства автоматизации и компьютерно-интегрированные технологии». — Краматорск, 2003.
 29. Святный В.А., Аноприенко А.Я., Потапенко В.А. Модульные среды для сетевого распределенного моделирования сложных динамических систем // Труды международной конференции «Современные проблемы информатизации в технике и технологиях»: Выпуск 8. — Воронеж, 2003. — 122–123 с.
 30. Anoprienko A., Potapenko V. Web-basierte Simulationsumgebung mit DIVA-Serverkomponente für komplexe verfahrenstechnische Produktionsanlagen // 17. Symposium “Simulationstechnik” ASIM 2003, Magdeburg, 16.09 bis 19.09.2003. — SCS-Europe, 2003. p. 205–208.
 31. Карпенко А.С. Логика на рубеже тысячелетий // Online Journal “Logical Studies”, N5 (2000). [Электронный ресурс] — Режим доступа: http://ihtik.lib.ru/philosoph/ ihtik_131.htm.
 32. Gabbay D. M. (ed.) What is a logical system? Oxford: Clarendon Press (and New York), 1994.
 33. Wang Hao. What is logic? // The Monist. 1994. Vol. 77. N 3. p. 261–277.
 34. Hintikka J. What is true elementary logic? // Gavroglu K., Stachel J., Wartofsky M. (eds.), Physics, philosophy and the scientific community. Dordrecht: Kluwer. 1994. P. 301–326.

35. Batens D. Inconsistency-adaptive logics and the foundation of non-monotonic logic // *Logique et Analyse*. 1994. N 145. p. 57–94.
36. Zadeh L. A. Soft computing and Fuzzy Logic. *Software Engineering Journal*, November, 1994.
37. Bezdek E. Fuzzy models — what are they and why? *IEEE Trans. on Fuzzy systems*. Vol. 1. No. 1, February, 1993.
38. Яценко А. NexGen History [Электронный ресурс] — Режим доступа: <http://www.3dnews.ru/editorial/nexgen-history>.
39. Виджаян Дж. В ожидании Itanium // *Еженедельник "Computerworld"*, #05, 2001 год // Издательство «Открытые системы». [Электронный ресурс] — Режим доступа: http://www.osp.ru/cw/2001/05/022_0.htm.
40. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. — СПб.: БХВ-Петербург, 2002. — 608 с.
41. Куроуз Дж., Росс К. Компьютерные сети: Многоуровневая архитектура Интернет. — СПб.: Питер, 2004. — 765 с.
42. Кастельс М. Галактика Интернет. — Екатеринбург: У-Фактория, 2004. 328 с.
43. Немнюгин С., Стесик О. Параллельное программирование для многопроцессорных вычислительных систем. — СПб.: БХВ-Петербург, 2002. 400 с.
44. Орфали Р., Харки Д., Эдвардс Д. Основы CORBA. — М.: МАЛИП, Горячая Линия — Телеком, 1999. — 318 с.
45. Аноприенко А.Я. Археомоделирование: модели и инструменты докомпьютерной эпохи. — Донецк: УНИТЕХ, 2007. — 318 с.
46. Аверкин А.Н. Мягкие вычисления — основа новых информационных технологий / В кн. «КИИ-96», Сборник научных трудов 5-й национальной конференции с международным участием «Искусственный интеллект — 96», т. 2, Казань, 1996, с. 237–239.
47. Ильин В.В. Высокие информационно-вычислительные технологии. Вестник РАН, №6, 1996.

48. Ульянов С.В. Нечеткие модели интеллектуальных систем управления: теоретические и прикладные аспекты (обзор). Техническая кибернетика, № 3, 1991, с. 3–28.
49. Маковельский А.О. История логики. — М.: Наука, 1967. — 502 с.
50. Заде Л.А. Основы нового подхода к анализу сложных систем и процессов принятия решений. / В кн. «Математика сегодня», М., 1974.
51. Zadeh L.A., Fuzzy Sets. Information and Control, June 1965, p. 338–353.
52. Zadeh L. A. Fuzzy Logic = Computing with Words. IEEE Transactions on Fuzzy Systems, Vol. 4, No. 2, May 1996, p. 103–111.
53. Аноприенко А.Я., Башков Е.А. Устройство для отображения графической информации на экране телевизионного индикатора. А.с.1403091 (СССР) / Опубл.1988, БИ № 22.
54. Аноприенко А.Я., Гриза В.А. Запоминающее устройство с многоформатным доступом к данным. А.с. 1624526 (СССР) / Опубл. 1991, БИ № 4.
55. Стахов А.П. Коды золотой пропорции. — М.: Радио и связь, 1984. — 152 с.
56. Post E. L. Introduction to General Theory of Elementary Propositions. American Journal of Mathematics. 1921. Vol. 43. № 3.
57. Маковский Н.Н. Лингвистическая генетика: проблемы онтогенеза слов в индоевропейских языках. — М.: Наука, 1992, 189 с.
58. Ларичев В.Е. Мудрость змеи: первобытный человек, Луна и Солнце. — Новосибирск: Наука, 1989. — 272 с.
59. Ригведа. Мандалы I — IV. — М.: Наука, 1989. — 767 с.
60. Словарь по кибернетике. Под ред. В.С. Михалевича. — К.: Гл. Ред. УСЭ им. М.П. Бажана, 1989, 751 с.

61. Кликс Ф. Пробуждающееся мышление. У истоков человеческого интеллекта. — М.: Прогресс, 1983. — 302 с.
62. Санжаров С.Н. Погребения донецкой катакомбной культуры с игральными костями. Советская археология, № 1, 1988. — с.140–158.
63. Фоли Дж. Энциклопедия знаков и символов. — М.: Вече, АСТ, 1996. — 432 с.
64. Бонгард-Левин Г.М. Древнеиндийская цивилизация. Философия, наука, религия. — М.: Наука, 1980, 333 с.
65. Да услышат меня земля и небо: Из ведийской поэзии: Пер. с ведийск. — М.: Худож. лит., 1984, 270 с.
66. Стройк Д. Я. Краткий очерк истории математики. — М.: Наука, 1984, 282 с.
67. Апокин И.А., Майстров Е.М. Развитие вычислительных машин. — М.: Наука, 1974. — 399 с.
68. Пипуныров В.Н. История часов с древнейших времен до наших дней. — М.: Наука, 1982, 496 с.
69. Reuter A. Grenzen der Parallelitaet (Limitations of Parallelism). Informationstechnik, 34 (1992) 1, z. 62–74.
70. Lukasiewicz J. O pojeciu mozliwosci. — Ruch Filozoficzny. Lwow. 1920. R. 5. № 9.
71. Бочвар Д.А. Об одном трехзначном исчислении и его применении к анализу парадоксов классического расширенного функционального исчисления. Математический сборник. 1938. Т. 4 (46). № 2.
72. Reichenbach H. Philosophical Foundations of Quantum Mechanics. Berkeley — Los Angeles, 1946.
73. Зверев Г.Н. Точные и аппроксимационные логики в машинных рассуждениях / В кн. «КИИ-96», Сборник научных трудов 5-й национальной конференции с международным участием «Искусственный интеллект 96», т. 1, Казань, 1996, с. 46–49.

74. Брусенцов Н.П., Румянцев Д. Долой биты! (Интервью с конструктором троичной ЭВМ) // «Академия Тринитаризма», М., Эл № 77–6567, публ.11503, 2004.
75. Lukasiewicz J. Aristotle's Syllogistic from the Standpoint of Modern Formal Logic. Clarendon Press. Oxford, 1957.
76. Раушенбах Б.В. Логика троичности. / В кн. «Пристрастие». — М.: Издательство «Аграф», 1997, с. 117–129.
77. Белнап Н., Стил Т. Логика вопросов и ответов. — М., 1981. — 214 с.

К главе 2

1. Иваница С.В., Аноприенко А.Я. Особенности реализации операций тетралогики // Научные труды Донецкого национального технического университета. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2011). Выпуск 13 (185). — Донецк: ДонНТУ, 2011. С. 134–140.
2. Boole G. The mathematical analysis and logic / G. Boole — Cambridge: Macmillan, Barclay, & Macmillan; London: George Bell, 1847. — p. 82.
3. Белнап Н. Как нужно рассуждать компьютеру // Белнап Н., Стил Т. Логика вопросов и ответов. — М.: «Прогресс», 1981. — 288 с.
4. Аверкин А.Н., Гаазе-Рапопорт М.Г., Поспелов Д.А. Толковый словарь по искусственному интеллекту / А.Н. Аверкин и др. — М.: Радио и связь, 1992. — 256 с.
5. Аноприенко А.Я. Обобщенный кодо-логический базис в вычислительном моделировании и представлении знаний: эволюция идеи и перспективы развития // Научные труды Донецкого национального технического университета. Серия «Информатика, кибернетика и вычислительная техника» (ИКВТ-2005), выпуск 93: — Донецк: ДонНТУ, 2005, С. 289–316.

6. Аноприенко А.Я. Археомоделирование: Модели и инструменты докомпьютерной эпохи. / А.Я. Аноприенко — Донецк: УНИТЕХ, 2007. — 317 с., ил.
7. Аноприенко А.Я., Коноплева А.П., Хасан Аль Абабех. Постбинарный компьютинг, Grid и «облачные вычисления»: новые реальности компьютерного моделирования // Материалы третьей международной научно-технической конференции «Моделирование и компьютерная графика» 7-9 октября 2009 года, — Донецк, ДонНТУ, 2009, — С. 6.
8. История троичной логики. Материал из свободной русской энциклопедии «Традиция». — [Электронный ресурс] — Режим доступа: http://traditio.ru/wiki/Троичная_логика.
9. Hayes N.T. Trits to Tetrts. — [Электронный ресурс] — Режим доступа: <http://grouper.ieee.org/groups/1788/PositionPapers/Tetrts.pdf>.
10. Яблонский С.В. Введение в дискретную математику: учебное пособие для вузов. — 2-е изд., перераб. и доп. / С.В. Яблонский. — М.: Наука, 1986. — 384 с.
11. Троичная логика. Материал из Википедии — свободной энциклопедии. [Электронный ресурс] — Режим доступа: http://ru.wikipedia.org/wiki/Троичная_логика.
12. Ивин А.А. Логика. Учебное пособие; 2-е изд. / А.А. Ивин. — М.: Знание, 1998. — 228 с.
13. Марченков С.С. Замкнутые классы булевых функций / С.С. Марченков. — М.: ФизМатЛит, 2000. — 128 с.
14. Аноприенко А.Я., Иваница С.В. Особенности реализации постбинарных логических операций // Научно-теоретический журнал «Искусственный интеллект», №2, 2011. С. 110–121.
15. Колмогоров А.Н. Математическая логика, изд 3-е, стереотипное / А.Н. Колмогоров, А.Г. Драгалин — М.: КомКнига, 2006., — 256 с.
16. Великий предел — материал из Википедии. [Электронный ресурс] — Режим доступа:

http://ru.wikipedia.org/wiki/Великий_предел.

17. Владимиров Д.А. Булевы алгебры / Д.А. Владимиров — М.: Наука, 1969 — 319 с.
18. Гиндикин С.Г. Алгебра логики в задачах / С.Г. Гиндикин — М.: Наука, 1972 — 288 с.
19. Аноприенко А.Я. Интервальные вычисления и перспективы их развития в контексте кодо-логической эволюции / А.Я. Аноприенко, С.В. Иваница // Научные труды Донецкого национального технического университета. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-2010). — 2010. — Вып. 8(168), С. 150–160.
20. Аноприенко А.Я. Особенности постбинарного кодирования на примере интервального представления результатов вычислений по формуле Бэйли-Боруэйна-Плаффа / А.Я. Аноприенко, С.В. Иваница // Научные труды Донецкого национального технического университета. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2010). — 2010. — Выпуск 11(164). — С. 19–23.

К главе 3

1. Аноприенко А.Я. Обобщенный кодо-логический базис в вычислительном моделировании и представлении знаний: эволюция идеи и перспективы развития // Научные труды Донецкого национального технического университета. Серия «Информатика, кибернетика и вычислительная техника» (ИКВТ-2005), выпуск 93: — Донецк: ДонНТУ, 2005, С. 289–316.
2. Кнут Д.Э. Искусство программирования, томи 2. Получисленные алгоритмы. — М.: Издательский дом «Вильямс», 2000. — 832 с.
3. Аноприенко А.Я. Восхождение интеллекта: эволюция монокодовых вычислительных моделей // Научные труды Донецкого государственного технического университета.

- Выпуск 15. Серия «Информатика, кибернетика и вычислительная техника» (ИКВТ-2000). — Донецк: ДонГТУ. — 2000. — 87–107 с.
4. Курант Р., Роббинс Г. Что такое математика? — М.: Просвещение, 1967. — 558 с.
 5. Даан-Дальмедико А., Пейффер Ж. Пути и лабиринты. Очерки по истории математики. — М.: Мир, 1986. — 432 с.
 6. Сигорский В.П. Математический аппарат инженера. — К.: «Техника», 1975. — 768 с.
 7. Новиков Ф.А. Дискретная математика для программистов. — СПб: Питер, 2000. — 304 с.
 8. Белов В.М., Евстигнеев В.В., Королькова С.М., Лагуткина Е.В., Суханов В.А. Интервальная кинетика химических реакций. Обратимые реакции первого порядка // Химия растительного сырья. Т.1 (1997), N3, с. 32–35. [Электронный ресурс] — Режим доступа: http://www.nioch.nsc.ru/mirrors/press/chemwood/volume1/n3/stat_6.html.
 9. Аноприенко А.Я. От вычислений к пониманию: когнитивное компьютерное моделирование и его практическое применение на примере решения проблемы Фестского диска / В кн. «Информатика, кибернетика и вычислительная техника (ИКВТ-99). Сборник научных трудов ДонГТУ». Выпуск 6. Донецк, ДонГТУ, 1999, с. 36–47.
 10. Аноприенко А.Я., Святный В.А. Универсальные моделирующие среды // Сборник трудов факультета вычислительной техники и информатики. Вып. 1. — Донецк: ДонГТУ. — 1996. — 8–23 с.
 11. Аноприенко А.Я., Святный В.А. Высокопроизводительные информационно-моделирующие среды для исследования, разработки и сопровождения сложных динамических систем // Научные труды Донецкого государственного технического университета. Выпуск 29. Серия «Проблемы моделирования и автоматизации проектирования

- динамических систем» — Севастополь: «Вебер». — 2001. — 346–367 с.
12. Аноприенко А.Я. Эволюция алгоритмического базиса вычислительного моделирования и сложность реального мира // Научные труды Донецкого национального технического университета. Выпуск 52. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-2002): Донецк: ДонНТУ, 2002. — 6–9 с.
 13. Буч Г., Рамбо Дж., Джекобсон А. Язык UML. Руководство пользователя. — М.: ДМК, 2000. — 366 с.
 14. Нариньяни А.С. Модель или алгоритм: новая парадигма информационной технологии // Информационные Технологии, 1997, № 4, стр.11–16. [Электронный ресурс] — Режим доступа: <http://www.artint.ru/articles/narin/PARAD-R1.htm>.
 15. Нариньяни А.С. Информационные технологии 21 века: на пороге революции. [Электронный ресурс] — Режим доступа: http://www.artint.ru/articles/narin/it_rev_s.htm.
 16. Марков Е. Архитектура, управляемая моделью // CitCity, 15 декабря 2005 г. [Электронный ресурс] — Режим доступа: <http://citcity.ru/integration/articles/11353>.
 17. Заде Л.А. Понятие лингвистической переменной и его применение к понятию приближенного решения, М.: Мир, 1976.
 18. Нариньяни А.С., Недоопределенность в системе представления и обработки знаний, Изв. АН СССР, «Техническая Кибернетика», № 5, 1986.
 19. Hyvonen E. Constraint reasoning based on interval Arithmetic: the tolerance propagation approach, Artificial Intelligence, v.58, 1992.
 20. Интервальный анализ и его приложения. Краткий неформальный очерк. [Электронный ресурс] — Режим доступа: <http://www.nsc.ru/interval/index.php>.

21. Аноприенко А.Я., Иваница С.В. Интервальные вычисления и перспективы их развития в контексте кодологической эволюции // Научные труды Донецкого национального технического университета. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-2010). Выпуск 8 (168): Донецк: ДонНТУ, 2010. с. 150–160.
22. Алефельд Г., Херцбергер Ю. Введение и интервальные вычисления. М.: Мир, 1987. — 360 с.
23. Interval Computations. [Электронный ресурс] — Режим доступа: <http://www.cs.utep.edu/interval-comp/main.html>.
24. Вощинин А.П., Бочков А.Ф., Сотиров Г.Р. Интервальный анализ данных как альтернатива регрессионному анализу. Заводская лаборатория, 1990, № 7, — с. 76–81.
25. Gorsky V., Shvetzova-Shilovskaya T., Voschnin A. Risk assessment of accident involving environmental high-toxicity substances, Journal of Hazardous Materials, № 78, 2000.
26. Шарый С.П. Интервальные алгебраические задачи и их численное решение. Диссертация на соискание ученой степени доктора физико-математических наук, 2000, — 322 с.
27. Добронез Б.С. Интервальная математика: Учеб. пособие. Краснояр. гос. ун-т. — Красноярск. 2004. — 216 с.
28. Марчук Г.И. Методы вычислительной математики. — М.: Наука. 1977. — 456 с.
29. Самарский А.А. Введение в теорию разностных схем. — М.: Наука, 1971. — 552 с.
30. Hansen E. Topics in Interval Analysis. — London: Oxford University Press. — 1969.
31. Moore R.E. Interval analysis. Eiiglewood Cliffs. — N.J.:Prentic-e-llall. — 1966.
32. Шокин Ю.И. Интервальный анализ. — Новосибирск: Наука, 1981. — 111 с.

33. Young R.C. Algebra of many-valued quantities // *Mathematische Annalen* / R. C. Young, 1931. S. 260–290.
34. Dwyer P.S. *Linear Computations* / P. S. Dwyer — New York: John Wiley & Sons, 1951. — 36 p.
35. Warmus M. Calculus of approximations // *Bull. Acad. Polon. Sci.* / M. Warmus — 1956, Cl. III, vol. IV, № 5.
36. Sunaga T. Theory of an interval algebra and its application to numerical analysis // *RAAG Memoirs*. — Vol. 2, Misc. II, 1958.
37. Markov S., Okumura K. The contribution of T.Sunaga to interval analysis and reliable computing // *Developments in Reliable Computing* / Cendes T., ed. — Dordrecht: Kluwer Academic Publishers, 1998.
38. Брадис В.М. Опыт обоснования некоторых практических правил действий над приближенными числами // *Известия Тверского педагогического института*. 1927. — Вып. 3.
39. Брадис В.М. Теория и практика вычислений. Пособие для высших педагогических учебных заведений. — Москва: Учпедгиз, 1937.
40. Брадис В.М. Средства и способы элементарных вычислений. — Москва: Издательство Академии педагогических наук РСФСР, 1948.
41. Канторович Л.В. О некоторых новых подходах к вычислительным методам и обработке наблюдений // *Сибирский Математический Журнал*. — 1962. — Т. 3, № 5.
42. Назаренко Т.И., Марченко Л.В. Введение в интервальные методы вычислительной математики. / Т.И. Назаренко и др. — Иркутск: Изд-во Иркут. ун-та. 1982.
43. Калмыков С.Л., Шокин Ю.И., Юлдашев З.Х. Методы интервального анализа. / О.Л. Калмыкин и др. — Новосибирск: Наука. 1986. — 224 с.
44. Шарый С.П. Конечномерный интервальный анализ. — Новосибирск, Институт вычислительных технологий СО РАН, 2009. — 569 с.

45. Beierbaum, F., Schwierz. K.P. A bibliography on interval mathematics // J. Comput. Appl. Math. V. 4, N 1. P.59–86.
46. Moore R.E. Methods and Applications of Interval Analysis. SIAM. Philadelphia 1979.
47. Клатте Р., Кулиш У., Неага М., Рац Д., Ульрих Х. PASCAL-XSC Язык численного программирования. / Р. Клатте и др. — М.: ДМК Пресс, 2000.
48. Агеев М.П., Алик И.П., Марков Ю.И. Алгоритм 616. Процедуры интервальной математики // Библиотека алгоритмов 516–11/06. — М.: Сов. Радио, 1976.
49. Interval arithmetic. From Wikipedia, the free encyclopedia. [Электронный ресурс] — Режим доступа: http://en.wikipedia.org/wiki/Interval_arithmetic.
50. IEEE Interval Standard Working Group — P1788. [Электронный ресурс] Страница доступа: <http://grouper.ieee.org/groups/1788/>
51. Аноприенко А.Я., Иваница С.В. Особенности постбинарного кодирования на примере интервального представления результатов вычислений по формуле Бэйли-Боруэйна-Плаффа // Научные труды Донецкого национального технического университета. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2010). Выпуск 11 (164). — Донецк: ДонНТУ, 2010, с. 19–23.
52. Аноприенко А.Я., Гранковский В.А., Иваница С.В. Пример Румпа в контексте традиционных, интервальных и постбинарных вычислений // Научные труды Донецкого национального технического университета. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-2011). Выпуск 9 (179): Донецк: ДонНТУ, 2011, с. 324–343.
53. Аноприенко А.Я., Иваница С.В. Реализация интервальных вычислений средствами математического пакета SciLab с использованием интервального расширения Int4Sci // «Донбасс-2020: перспективы развития глазами молодых ученых»: Материалы V научно-практической конферен-

- ции. Донецк, 25–27 мая 2010 г. — Донецк, ДонНТУ Министерства образования и науки, 2010, с. 629–633.
54. Иваница С.В., Меркулов А.В., Аноприенко А.Я. Интервальные вычисления в математических пакетах Scilab и Mathematica // Информатика и компьютерные технологии / Материалы VI международной научно-технической конференции студентов, аспирантов и молодых ученых — 23–25 ноября 2010 г. Т.1. Донецк, ДонНТУ. — 2010, с. 240–246.
55. Иваница С.В., Меркулов А.В. Особенности вычисления интервальных полиномов с контролем точности результирующих интервалов // Моделирование и компьютерная графика / Материалы IV международной научно-технической конференции — 5–8 октября 2011 г. Донецк, ДонНТУ. — 2011, с. 119–126.
56. Аноприенко А.Я., Иваница С.В., Кулибаба С.В. Особенности представления постбинарных вещественных чисел в контексте интервальных вычислений и развития аппаратного обеспечения средств компьютерного моделирования // Моделирование и компьютерная графика / Материалы IV международной научно-технической конференции — 5–8 октября 2011 г. Донецк, ДонНТУ. — 2011. С. 13–19.
57. Hayes B. A Lucid Interval. A reprint from American Scientist the magazine of Sigma Xi, the Scientific Research Society, Volume 91, Number 6, November-December, 2003, p. 484–488.
58. Строгий учет ошибок округлений на цифровых ЭВМ. [Электронный ресурс] — Режим доступа: <http://www.sbras.ru/interval/introduction/RusIntro>.
59. Кулиш У., Рац Д., Хаммер Р., Хокс М. Достоверные вычисления. Базовые численные методы. — М.: «Регулярная и хаотическая динамика». — 2005.
60. Wolfram Mathematica home page. [Электронный ресурс] — Режим доступа:

<http://www.wolfram.com/products/mathematica/index.html>.

61. Scilab official website. [Электронный ресурс] — Режим доступа: <http://www.scilab.org>.
62. Долгов Ю.Г. Метод глобальной оптимизации на основе метода ветвей и границ / Ю.Г. Долгов. Интервальная математика и распространение ограничений МКВМ-2004, с. 184–192.
63. IEEE754-2008 — IEEE Standard for Floating-Point Arithmetic (IEEE 754), From Wikipedia, the free encyclopedia. [Электронный ресурс] — Режим доступа: http://en.wikipedia.org/wiki/IEEE_754-2008.
64. Марчук Г.И. Методы вычислительной математики. / Г.И. Марчук — М.: Наука. 1977. — 456 с.
65. Иваница С.В., Меркулов А.В. Методы контроля точности результирующих интервалов при вычислении интервальных полиномиальных функций // Научные труды Донецкого национального технического университета. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2012). Выпуск 14 (186). — Донецк: ДонНТУ, 2011.
66. Кнут Д.Э. Искусство программирования, том 3. Сортировка и поиск, 3-е изд. — Спб.: Диалектика, — 2005.
67. Аноприенко А.Я. Тетралогики и тетракоды. // Сборник трудов факультета вычислительной техники и информатики. Вып.1. — Донецк: ДонГТУ. — 1996. С. 32–43.
68. Аноприенко А.Я. Расширенный кодо-логический базис компьютерного моделирования / В кн. «Информатика, кибернетика и вычислительная техника» (ИКВТ-97). Сборник научных трудов ДонГТУ. Выпуск 1. Донецк, ДонГТУ, 1997, с. 59–64.

К главе 4

1. Computing – Wikipedia, the free encyclopedia. [Электронный ресурс] — Режим доступа: <http://en.wikipedia.org/wiki/Computing>.

2. Аноприенко А.Я., Башков Е.А. Запоминающее устройство с многоформатным доступом к данным. А.с. 1336109 (СССР) / Опубл. 1987, БИ № 33.
3. Аноприенко А.Я., Башков Е.А. Запоминающее устройство с многоформатным доступом к данным. А.с. 1355997 (СССР) / Опубл. 1987, БИ № 44.
4. Моисеев Н.Н. Алгоритмы развития (серия «Академические чтения»). — М: Наука, 1987. — 304 с.
5. Аноприенко А.Я., Кухтин А.А. О некоторых возможностях расширения логического базиса информатики. / В кн. "Тези доповідей міжнародної науково-практичної конференції "Інформатизація в умовах переходу до ринку", Київ, 5-6 листопада 1992 р., с. 30–32.
6. Аноприенко А.Я. Тетралогика и тетракоды. / В кн. «Сборник трудов факультета вычислительной техники и информатики». Вып.1. Донецк, ДонГТУ, 1996, с.32–43.
7. Аноприенко А.Я. Расширенный кодо-логический базис компьютерного моделирования / В кн. "Информатика, кибернетика и вычислительная техника (ИКВТ-97). Сборник научных трудов ДонГТУ." Выпуск 1. Донецк, ДонГТУ, 1997, с. 59-64.
8. Hough D. Applications of the Proposed IEEE 754 Standard for Floating-Point Arithmetic — Computer, 1981. [Электронный ресурс] — Режим доступа: <http://www.ieeexplore.ieee.org>.
9. Аноприенко А.Я. Вызовы времени и постбинарный компьютинг // Информатика и компьютерные технологии / Материалы VI международной научно-технической конференции — 23–25 ноября 2010 г. Т. 1. Донецк, ДонНТУ. — 2010. С. 13–31.
10. Hammer R., Hocks M., Kulisch U., Ratz D. — Numerical Toolbox for Verified Computing (Pascal-XSC Programs) Springer-Verlag Berlin Heidelberg, — 1993, — с. 8.

11. Алефельд Г., Херцбергер Ю. Введение и интервальные вычисления. — М.: Мир, 1987. — 360 с.
12. Шарый С.П. Конечномерный интервальный анализ. — [Электронный ресурс] — Режим доступа: www.nsc.ru/interval/Library/InteBooks/SharyBook.pdf
13. Bailey David H., Borwein Peter B., Plouffe Simon On the Rapid Computation of Various Polylogarithmic Constants // Mathematics of Computation. — 1997. — В. 218. Т. 66. — р. 903–913.
14. Аноприенко А.Я. Обобщенный кодо-логический базис в вычислительном моделировании и представлении знаний: эволюция идеи и перспективы развития // Научные труды Донецкого национального технического университета. Серия «Информатика, кибернетика и вычислительная техника» (ИКВТ-2005), выпуск 93: — Донецк: ДонНТУ, 2005. С. 289–316.
15. Аноприенко А.Я. Археомоделирование: модели и инструменты докомпьютерной эпохи. — Донецк: УНИТЕХ, 2007. — 318 с.
16. Аноприенко А.Я., Иваница С.В. Особенности постбинарного кодирования на примере интервального представления результатов вычислений по формуле Бэйли-Боруэйна-Плаффа // Научные труды Донецкого национального технического университета. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2010). Выпуск 11 (164). — Донецк: ДонНТУ, 2010. С. 19–23.
17. Иваница С.В., Аноприенко А.Я. Особенности реализации операций тетралогии // Научные труды Донецкого национального технического университета. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2011). Выпуск 13 (185). — Донецк: ДонНТУ, 2011. С. 134–140.
18. Аноприенко А.Я., Иваница С.В. Особенности реализации постбинарных логических операций // Научно-теорети-

- ческий журнал «Искусственный интеллект», № 2, 2011. С. 110–121.
19. Фролов А.В., Фролов Г.В. Аппаратное обеспечение персонального компьютера. // Библиотека системного программиста, том 33. — М.: Диалог-МИФИ, 1997, 304 с.
 20. Петров Ю.П. Обеспечение надежности и достоверности компьютерных расчетов. — СПб: БХВ-Петербург, 2008. — 160 с.
 21. Rump S.M. Algorithm for verified inclusions: Theory and practice // Reliability in Computing, R. E. Moore ed., Academic press, San Diego, 1988, p. 109–126.
 22. Cuyt A., Verdonk B., Becuve S., Kuterna P. A Remarkable Example of Catastrophic Cancellation Unraveled // Computing, 66, 2001, p. 309–320.
 23. Loh E., Walster G. Rump's Example Revisited // Reliable Computing 8: 2002, p. 245–248.
 24. Rump S.M. Rigorous results using floating-point arithmetic // Acta Numerica, 19, 2010, p. 287–449.
 25. Farber R. Numerical Precision: How Much is Enough? As we approach ever-larger and more complex problems, scientists will need to consider this question. 2009. [Электронный ресурс] — Режим доступа: <http://www.scientificcomputing.com/article-hpc-Numerical-Precision-How-Much-is-Enough-063009.aspx>.
 26. Алексеев Е.Р., Чеснокова Е.А., Рудченко Е.А. Решение инженерных и математических задач в пакете Scilab. — М.: ALT Linux, 2008. — 257 с.
 27. Кирьянов Д.В. Самоучитель MathCAD 2001. — СПб.: БХВ-Петербург, 2002. — 544 с.
 28. Arbitrary-precision arithmetic // From Wikipedia, the free encyclopedia. [Электронный ресурс] — Режим доступа: http://en.wikipedia.org/wiki/Arbitrary-precision_arithmetic.
 29. The GNU Multiple Precision Arithmetic Library. [Электронный ресурс] Страница доступа: <http://gmplib.org/>.

30. Шакиров Р.Н. Класс `cBigNumber` для целочисленной арифметики неограниченной разрядности в языке C++ // Программные продукты и системы. 2009. 1. С. 7–11.
31. Sofroniou M., Spaletta G. Precise Numerical Computation, 2000. [Электронный ресурс] — Режим доступа: www.sop.inria.fr/lemme/AOC/workshop/.
32. Аноприенко А.Я., Иваница С.В. Интервальные вычисления и перспективы их развития в контексте кодологической эволюции // Научные труды Донецкого национального технического университета. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-2010). Выпуск 8 (168): Донецк: ДонНТУ, 2010. 150–160 с.
33. Яшкардин В. IEEE 754 — стандарт двоичной арифметики с плавающей точкой, 2010. [Электронный ресурс] — Режим доступа: <http://www.softelectro.ru/ieee754.html>.
34. Юровицкий В.М. IEEE754-тика угрожает человечеству. [Электронный ресурс] — Режим доступа: <http://www.yur.ru/science/computer/IEEE754.htm>.
35. Пехотин Е.В. Методы выполнения операций высокой точности над многоразрядными числами в формате с плавающей точкой // Портал магистров ДонНТУ, 2010. [Электронный ресурс] — Режим доступа: <http://masters.donntu.edu.ua/2010/fknt/pehotin/diss/index.htm>.
36. Кнут Д.Э. Искусство программирования, том 2. Получисленные алгоритмы. — М.: Издательский дом «Вильямс», 2000. — 832 с.
37. Davies E.B. Whither Mathematics? // Notices of the American Mathematical Society. December 2005. Volume 52, Number 11. — 1350–1356 p.
38. Дэвис Б. Куда идет математика? // Элементы большой науки, 14.11.2005. [Электронный ресурс] — Режим доступа: <http://www.elementy.ru/news/164970>.

39. Юровицкий В.М. Общая теория чисел и числовых эпох. Мир на пороге новой числовой эпохи // Материалы международной конференции «Диалог-2008», Кипр, 12–15 мая 2008 г. [Электронный ресурс] — Режим доступа: <http://www.yur.ru/Conference/Cipros/Theory-of-number.htm>.
40. Аноприенко А.Я., Гранковский В.А., Иваница С.В. Пример Румпа в контексте традиционных, интервальных и постбинарных вычислений // Научные труды Донецкого национального технического университета. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-2011). Выпуск 9 (179): Донецк: ДонНТУ, 2011. 324–343 с.
41. Moore R.E. Interval analysis. Eiiglewood Cliffs / R.E. Moore — N.J.:Prentice-Hall, 1966.
42. Суворова Е., Шейнин Ю. Проектирование цифровых систем на VHDL. — СПб.: BHV, 2003. — 576 с.
43. Бесплатные программы SoftElectro. Программа IEEE754 — конвертор чисел формата IEEE754 с абсолютной точностью представления результата. [Электронный ресурс] — Режим доступа: <http://softelectro.ru/program.html>.

Для заметок

Для заметок

Для заметок

*Авторы выражают свою глубокую
благодарность профессорам ДонНТУ
Баикову Евгению Александровичу и
Лапко Владимиру Васильевичу, внима-
тельно ознакомившимся с рукописью мо-
нографии и внесшим ряд существенных
замечаний и рекомендаций, позволивших
ее заметно улучшить.*

Научное издание

Аноприенко Александр Яковлевич

Иваница Сергей Васильевич

Постбинарный компьютеринг и интервальные вычисления в контексте кодо-логической эволюции. — Донецк, ДонНТУ, УНИТЕХ, 2011. — 248 с.

ISBN 978-966-8248-20-7

© Аноприенко А.Я., 2011

© Иваница С.В., 2011

© ДонНТУ, 2011

Издательство: ООО «Технопарк ДонГТУ УНИТЕХ»

Свидетельство о внесении субъекта издательского дела в государственный реестр издателей, изготовителей и распространителей издательской продукции: ДК № 1017 от 21.08.2002.

83000, г. Донецк, ул. Артема, 58, к.1.311

Тел.: (062) 304–90–19

Подписано к печати 24.12.2011. Формат 60х84 1/32

Усл. печ. л. 10. Печать лазерная.

Тираж 500 экз.

Отпечатано в типографии «Цифровая типография»

Адрес: г. Донецк, ул. Челюскинцев, 291а

Тел.: +38 062 388 07 30



Аноприенко Александр Яковлевич

профессор кафедры
компьютерной инженерии
Донецкого национального технического
университета



Иваница Сергей Васильевич

ассистент кафедры
компьютерной инженерии
Донецкого национального технического
университета