РОССИЙСКАЯ АКАДЕМИЯ НАУК

Институт проблем управления им. В.А. Трапезникова

На правах рукописи

Выхованец Валерий Святославович

СИНТЕЗ ЭФФЕКТИВНЫХ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ ДИСКРЕТНОЙ ОБРАБОТКИ ДАННЫХ НА ОСНОВЕ АЛГЕБРАИЧЕСКОЙ И ПОНЯТИЙНОЙ ДЕКОМПОЗИЦИИ ПРЕДМЕТНОЙ ОБЛАСТИ

Специальность 05.13.11
Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Специальность 05.13.15
Вычислительные машины и системы

Автореферат диссертации
на соискание ученой степени
доктора технических наук

Москва 2007

Работа выполнена в Институте проблем управления им. В. А. Трапезникова РАН

Научный консультант:
доктор технический наук В. Д. Малюгин

Официальные оппоненты:
доктор технических наук М. Ф. Каравай
доктор физико-математических наук Н. Н. Непейвода
доктор технических наук И. Ф. Чебурахин

Ведущая организация: Московский государственный технический университет им. Н. Э. Баумана.

Защита состоится 8 сентября 2008 г. в 11 часов на заседании Диссертационного совета Д002.226.03 Института проблем управления им. В. А. Трапезникова РАН по адресу: 117997, г. Москва, ул. Профсоюзная, 65. Тел. +7(495) 335-9329.

С диссертацией можно ознакомиться в библиотеке Института проблем управления им. В. А. Трапезникова РАН и в сети Интернет по адресу http://valery.vykhovanets.ru.

Автореферат разослан 7 февраля 2007 г.

Ученый секретарь
диссертационного совета,
д-р. техн. наук Е. В. Юркевич

Общая характеристика работы

Диссертация содержит изложение основных результатов, полученных автором при исследовании синтеза эффективных математических моделей дискретной обработки данных в рамках формально-логического и концептуально-онтологического подходов.

Актуальность темы исследования определяется тем, что дискретная обработка – самый распространенный метод вычислений, лежащий в основе современных вычислительных средств. На аппаратурном уровне дискретная обработка реализуется устройствами, состоящими из блоков, каждый из которых выполняет преобразование входных данных в выходные. Трассировка данных от выходов одних блоков ко входам других осуществляется их соединениями. Каждый такой блок, в свою очередь, может быть представлен как отдельное устройство, состоящее из других, более мелких блоков. В пределе, необходимом для практической реализации устройства, в качестве элементарных блоков используются логические элементы, имеющие физическую природу преобразования входных данных в выходные.

На программно-аппаратном уровне преобразование данных осуществляется в виде смены состояния информационной среды под управлением программы, сама информационная среда рассматривается как совокупность носителей данных, а программа представляется формализованным описанием этого процесса. Элементарными компонентами программ являются команды, выполняемые техническими средствами информационной системы. Более сложные компоненты – подпрограммы, являются элементарными единицами вызова (адресации) и рассматриваются как именованные совокупности команд. Одна или несколько подпрограмм объединяются в модули, представляющие собой единицы загрузки и хранения программ. В свою очередь совокупность модулей образуют следующий уровень иерархии – программные средства, предназначенные для выполнения той или иной задачи по обработке данных. И, наконец, программы средства объединяются в комплексы и служат для решения целого класса задач.

Как в первом, так и во втором случае эффективность обработки данных определяется количеством операций (логических элементов, команд), которые необходимо выполнить. Однако, как при высокоуровневом моделировании – в рамках концептуально-онтологического подхода, так и при низкоуровневом моделировании – в рамках формально-логического подхода, не решена проблема, связанная с выбором методологии анализа и технологии декомпозиции предметной области, которые позволяют получить формальное описание дискретной обработки данных, обеспечивающее эффективное решение стоящих прикладных задач. Более того, не существует единой теории, позволяющей выработать критерии и оценить эффективность произвольной дискретной обработки данных не прибегая к сравнению с другими ее реализациями.

Цель работы состоит в решении важной прикладной задачи автомати-

зации преобразования высокоуровневого (первичного) описания предметной области в терминах содержательной постановки задачи в ее эффективное низкоуровневое представление, состоящее из последовательности команд (операций) вычислительного средства.

Объектом исследования является процесс обработки данных, реализуемый вычислительными средствами дискретного действия.

Предмет исследования – математические модели предметной области, полученные на основе концептуальной, объектной, структурной, функциональной и логической декомпозиции.

Решаемые задачи. Общей задачей, решаемой в диссертации, является получение эффективных математических моделей дискретной обработки данных на основе формальной спецификации предметной области и решаемых в ней прикладных задач. Частными задачами, вытекающими из общей, являются:

– разработка методологии анализа предметной области, позволяющей строить ее эффективные декомпозиционные схемы в виде синтаксически и семантически замкнутых (прозрачных) формальных спецификаций;

– создание технологии обработки данных, основанной на отражении декомпозиционных схем предметной области в конструкциях специализированного предметного (проблемного) языка;

– разработка методов описания проблемного языка, обеспечивающих решение заданных прикладных задач путем дискретной обработки данных;

– обоснование методики определения эффективности дискретной обработки данных и получение точных, приближенных и асимптотических оценок сложности синтезируемых математических моделей;

– развитие общей теории дискретных функций на основе аппарата алгебраической декомпозиции и его использования для синтеза эффективных низкоуровневых описаний дискретной обработки данных.

Методика исследования основана на формально-логическом и концептуально-онтологическом моделировании. Концептуально-онтологические модели строятся путем формальной спецификации результатов понятийного анализа предметной области, а формально-логические – при алгебраической декомпозиции дискретных функций. Как в первом, так и во втором случае ищутся декомпозиционные схемы, позволяющие получать эффективные математические модели дискретной обработки данных.

Научная новизна. Основной результат диссертационной работы состоит в теоретической разработке и практическом решении проблемы синтеза эффективных математических моделей дискретной обработки данных. При этом впервые получены следующие результаты:

– предложена методология понятийного анализа, позволяющая на основе четырех видов отображений понятий получать синтаксически и семантически замкнутые формальные спецификации предметной области;

– разработана технология контекстной обработки данных, предназначенная для сокращения семантического разрыва между содержательными

представлениями относительно предметной области и языком моделирования (программирования);

– решена задача описания семантики формальных языков на основе метода математической индукции путем определения семантических категорий в процессе описания языка и описанными ранее средствами;

– обоснована методика алгебраического синтеза дискретных функций и найдены точные верхние оценки сложности синтезируемых формул в асимптотической области и при конечной размерности задачи;

– обобщена теория алгебраической декомпозиции дискретных функций в широком классе образующих алгебр, различающихся требованиями к алгебраическим операциям.

Обоснованность результатов. Научные положения, теоремы и выводы диссертации обоснованы с использованием теории формальных языков и грамматик, теоретических основ программирования, общей теории формальных систем, методов дискретной математики, теории групп и полей, аппарата линейной алгебры, а также проверены в процессе вычислительного эксперимента.

Практическая значимость работы заключается в разработке контекстной технологии программирования, при использовании которой получаются более эффективные и качественные программы, а также в обосновании методик алгебраического синтеза формул и оценки эффективности математических моделей дискретной обработки данных.

Реализация результатов. Результаты и положения диссертации приняты к использованию в ЗАО «ЛАНИТ» (понятийный анализ, элементы системы контекстного программирования) и внедрены в учебный процесс «МАТИ» – РГТУ им. К. Э. Циолковского (контекстная технология, алгебраический синтез), о чем имеются акты: ЗАО «ЛАНИТ» – акт об использовании результатов диссертационного исследования при выполнении государственного контракта 698ДК-МФ-03 по разработке проектно-сметной документации для создания систем инженерного обеспечения территориальных органов Федерального казначейства путем автоматической генерации документов на основе понятийного моделирования; «МАТИ» – РГТУ им. К. Э. Циолковского – акт о внедрении результатов диссертационной работы в учебный процесс на кафедре «Испытание летательных аппаратов» по дисциплинам «Организация ЭВМ и систем» и «Программные средства автоматизации».

Апробация работы. Результаты диссертационного исследования докладывались на международных конференциях «Современные методы цифровой обработки сигналов в системах измерения, контроля, диагностики и управления» (Минск, 1998), «Математические методы в образовании, науке и промышленности» (Тирасполь, 1999, 2001), конференции по телекоммуникациям (Одесса, 1999), по проблемам управления (Москва, 1999, 2003 и 2006), на конференции «Цифровая обработка сигналов и ее применение» (Москва, 1999, 2002, 2003), «Идентификация систем и задачи управления»

(Москва, 2000, 2003), «Проблемы управления безопасностью сложных систем» (Москва, 2000), «Распознавание образов и обработка информации» (Минск, 2001), «Параллельные вычисления и задачи управления» (Москва, 2001), «Автоматизация проектирования дискретных устройств» (Минск, 2001, 2003), на конференции, посвященной 100-летию со дня рождения члена-корреспондента АН СССР М. А. Гаврилова (Москва, 2003); на семинарах «Логическое моделирование» (Москва, 2005) и «Проблемы искусственного интеллекта» (Москва, 2006).

Публикации. По теме диссертации опубликовано 54 работы общим объемом 26 печатных листов, в том числе 8 статей в журналах из перечня ВАК. Под руководством автора и по тематике исследования защищена одна кандидатская диссертация.

Структура и объем работы. Диссертация состоит из введения, 6 глав, заключения и 5 приложений. Общий объем диссертации – 469 страниц.

Содержание работы

В **Главе 1** приводится критический анализ известных результатов в области математического моделирования и обработки данных. Данные при дискретной обработке представляются в виде конечных последовательностей знаков, а сама обработка осуществляется путем преобразования входных данных в выходные посредством разделения входных данных на части (переменные) и выполнения над ними некоторой последовательности операций. В связи с этим возникает задача представления функций большой размерности в виде композиции функций меньшей размерности. Постановка задачи в управлении базируется на автоматных моделях с памятью. Описание автомата осуществляется в виде распределенных во времени элементарных действий, каждое их которых является результатом вычисления дискретной функции. Таким образом, главной задачей дискретной обработки данных является дискретная декомпозиция.

История декомпозиции связана, в основном, с декомпозицией булевых функций. Установлено, что почти все функции в асимптотической области реализуются со сложностью, близкой к максимальной. Нерешенной осталась задача определения максимальной сложности функций при конечной размерности задач и нахождения методик синтеза, удовлетворяющих этой оценке.

Формально-логическая декомпозиция оказалась практически нереализуемой, ибо приводит к серьезным комбинаторным трудностям, связанным с предельно общей постановкой задачи. В связи с этим дискретную обработку данных представляют в алгоритмических формах, на основе поиска и формализации частных декомпозиционных схем. Теоретической базой для этого служат функциональная, структурная и объектная методологии анализа предметной области, реализуемые одноименными технологиями программирования.

Несмотря на развитость и завершенность перечисленных методологий,

трудно преодолимым остается семантический разрыв между содержательными представлениями относительно предметной области и теми средствами, которые служат для выражения этих представлений в виде формальных спецификаций. Для сокращения семантического разрыва используют повышение уровня абстракции языков моделирования. Однако, последнее не затрагивает существенным образом выразительность языков программирования, что связано с проблемой описания семантики таких языков.

В итоге показано, что описание дискретной обработки данных до сих пор выполняется в основном на интуитивном уровне с применением неформальных методов, основанных на искусстве разработчиков, их практическом опыте, экспертных оценках и экспериментальных проверках получаемых результатов.

Глава 2 посвящена концептуально-онтологическому подходу к анализу, декомпозиции и описанию предметной области – понятийному анализу. Основная цель понятийного анализа состоит в получении таких декомпозиционных схем предметной области, которые хотя и сформулированы в рамках содержательных представлений, однако обладают формальной строгостью и точностью, достаточной для прямого использования полученного высокоуровневого описания для низкоуровневой реализации дискретной обработки данных.

Суть подхода заключается в том, что для формальной спецификации предметной области используются две формальные системы. Первая формальная система – исчисление понятий, применена для выражения результатов понятийной декомпозиции предметной области. Вторая формальная система – специализированный предметный язык, или проблемный язык, строится для каждого класса решаемых задач и используется для описания решения.

Понятия, выявленные в процессе анализа предметной области, условно разделены на две группы: терминальные, или сигнификативные, выражаемые последовательностью знаков терминального алфавита проблемного языка, и нетерминальные, или денотационные, соответствующие нетерминальным знаками порождающей грамматики этого языка. Разделение понятий на денотационные и сигнификативные осуществляется с учетом некоторой фиксированной проблематики, задающей класс решаемых задач.

На основе выявления способов абстрагирования денотационных понятий строится понятийная структура предметной области, где под абстракцией понимается одно из четырех видов отображений одних понятий в другие, которые соответствуют четырем фундаментальным способам их образования: обобщению, типизации, агрегации и ассоциации. Для каждой такой абстракции дано формальное и семантически прозрачное определение, не требующее предметной интерпретации, как это имеет место в других концептуальных моделях, где используется множество связей между понятиями, несущими различную семантическую нагрузку.

Выявленные в процессе анализа предметной области денотационные

понятия включаются в множество понятий проблемного языка, а найденные декомпозиционные схемы преобразуются в его языковые конструкции, которые, в свою очередь, рассматриваются как формы выражения денотационных понятий в тексте и задаются последовательностью денотационных и сигнификативных понятий.

Таким образом, исследуемый подход основан на допущении, что уже в процессе изучения предметной области, еще до начала формализации, создается система понятий и декомпозиционные схемы, наиболее приспособленные для решения стоящих прикладных задач.

Сущности и понятия. *Проблемную область* будем рассматривать как совокупность предметной области и решаемых на ней задач (проблем), где под *предметной областью* понимается фрагмент реальной (виртуальной) действительности, представляемый некоторой совокупностью принадлежащих ему сущностей.

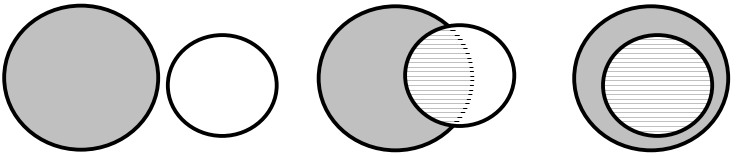
Сущность определим как устойчивое и уникальное представление предметной области, воспринимаемое некоторой совокупностью признаков. *Признак*, как именованная сущность, характеризуется множеством своих проявлений (значений) и имеет некоторую проблемную интерпретацию (семантическую роль). Признаки будем воспринимать как элементарные сущности, с точностью до которых осуществляется описание предметной области.

Понятие представим множеством сущностей, объединенных на основе общности своих признаков. Понятия будем именовать и задавать схемой, интенсионалом и экстенсионалом. *Имя*, или знаковое представление понятия, будем рассматривать как языковую единицу, отражающую некоторый смысл в семантическом плане, и некоторую конкретную сущность – в плане синтаксическом. *Схему понятия* зададим набором признаков, на которых понятие определено. Признаки будем интерпретировать как понятия, на которых определяются схемы. *Интенсионал*, или содержание понятия, представим набором значений взаимосвязанных признаков, позволяющим отличать сущности, принадлежащие понятию, от других сущностей предметной области. *Экстенсионал*, или объем понятия, будем рассматривать как множество сущностей, принадлежащих понятию.

Абстрагирование понятий. *Абстрагирование* – форма мышления, при которой происходит образование понятия. При абстрагировании между понятиями выявляется независимость, дифференциация и интеграция (рис. 1). Понятия *независимы*, если их признаки не пересекаются. Если у двух понятий имеются общие признаки, то наблюдается *дифференциация* понятий. Если все признаки одного понятия являются признаками другого понятия, то происходит их *интеграция*.

Известны следующие *абстракции* [Л7]: обобщение (специализация), типизация (конкретизация), агрегация (декомпозиция) и ассоциация (индивидуализация). Обобщение и типизация, и обратные им специализация и конкретизация, выражают общность понятий, проявляющуюся при диффе-

ренциации. Агрегация и ассоциация, и обратные им декомпозиция и индивидуализация раскрывают интеграцию понятий.



а) независимость б) дифференциация в) интеграция

Рис. 1. Пространство признаков

При обобщении происходит порождение нового понятия на основе одного или нескольких подобных понятий, когда порождаемое понятие сохраняет общие признаки исходных понятий, но игнорирует их различия. *Обобщение* – порождение понятия на основе пересечения схем обобщаемых понятий и расширенного объединения их экстенсионалов. При *специализации,* наоборот, из понятия-обобщения выделяется одно из обобщенных в нем понятий.

Типизация является частным случаем обобщения. В отличие от обобщения при типизации имеется возможность для каждой сущности из экстенсионала понятия-типа узнать ее исходное понятие. Для этого используется множество признаков, называемое *ключом*. Таким образом, *типизация* – порождение понятия на основе пересечения схем типизируемых понятий и объединения их экстенсионалов. При *конкретизации* понятия-типа фиксируется одно из типизированных в нем понятий, для чего необходим ключ.

При ассоциации устанавливается взаимосвязь между сущностями одного и того же или разных понятий. Ассоциация выражает специфическое соединение сущностей. Это соединение позволяет от сущности одного понятия перейти к одной или нескольким сущностям других понятий. *Ассоциация* – порождение понятия на основе объединения схем ассоциируемых понятий и ограниченного декартового произведения их экстенсионалов. При *индивидуализации* из понятия-ассоциации выделяются ассоциированные в нем понятия. Для перехода между сущностями этих понятий используется набор признаков, называемый *связью*.

При агрегации понятие строится как совокупность других понятий. Процесс, противоположный агрегации называется декомпозицией. *Агрегация* – порождение понятия на основе объединения схем агрегируемых понятий и декартового произведения их экстенсионалов. При *декомпозиции* понятие-агрегат разделяется на входящие в него агрегированные понятия.

Агрегация является предельным случаем ассоциации. В отличие от ассоциации, где между сущностями устанавливаются только часть связей, при агрегации присутствуют все возможные связи: на одном и том же множестве понятий можно задать несколько ассоциаций, в то время как их агрегация единственна.

Понятийная структура. Под *понятийной структурой* будем понимать совокупность понятий, для которых заданы способы их образования (абстрагирования). Носителем понятийной структуры является множество понятий, а ее сигнатурой – множество отображений обобщения, типизации, агрегации и ассоциации.

В отличие от семантических сетей и концептуальных схем, где на понятиях задаются различные виды отношений,[1] понятийная структура определена множеством понятий с четырьмя типами отображений, единственное назначение которых – показать способы образования понятий. Понятийная структура близка расширенной модели данных «сущность-связь» (EER-модель), однако в EER-модели элементами являются не понятия, а типы данных, причем для детализации модели используется трудно формализуемая семантическая разметка в виде дополнительных нотаций и ограничений.

Пример понятийной структуры приведен на рис. 2. Заметим, что даже для такой широко известной предметной области как «Числа», в зависимости от активной проблематики, можно определить несколько понятийных структур.

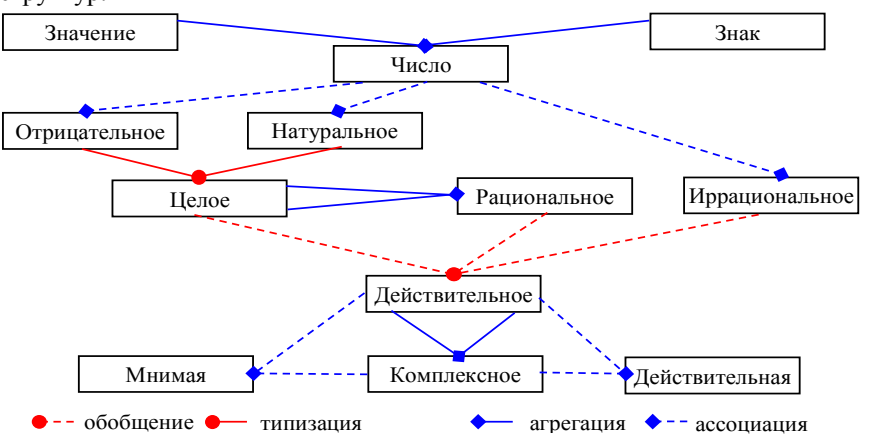


Рис. 2. Пример понятийной структуры

Исчисление понятий. *Исчисление понятий* определим как исчисление понятийных структур, а саму понятийную структуру будем задавать множеством формул, каждая из которых выражает образование одного из ее понятий. *Алфавит* исчисления включает знаки понятий, знак отсутствия определения понятия $\neg$, знаки операций объединения, пересечения и включения множеств, круглых скобок. Исчисления понятий строится на основе теории множеств с одной дополнительной аксиомой – *аксиомой* существования

[1] Количество различных типов отношений между понятиями в семантических сетях может превышать 200 [Л6].

пустого понятия:

$$\frac{()()}{()},$$

где использована следующая нотация: в числителе задается имя понятия и способы его абстрагирования (слева – список обобщения, справа – список ассоциации), а в знаменателе – схема понятия (shm). Для порождения формул исчисления используются четыре *правила вывода*:

$$\frac{()()}{()} \overset{\neg N}{\mapsto} \frac{()N()}{(N)};$$

$$\frac{(\ldots)N_1(\ldots)}{\mathrm{shm}\,N_1}\ldots\frac{(\ldots)N_m(\ldots)}{\mathrm{shm}\,N_m} \quad \underset{\bigcap_{i=1}^{m}\mathrm{shm}\,N_i\supset\{\}}{\overset{\neg N}{\mapsto}} \quad \frac{(N_1\ldots N_m)N()}{\bigcap_{i=1}^{m}\mathrm{shm}\,N_i};$$

$$\frac{(\ldots)N_1(\ldots)}{\mathrm{shm}\,N_1}\ldots\frac{(\ldots)N_m(\ldots)}{\mathrm{shm}\,N_m} \quad \underset{\bigcup_{i=1}^{m}\mathrm{shm}\,N_i\supset\{\}}{\overset{\neg N}{\mapsto}} \quad \frac{()N(N_1\ldots N_m)}{\bigcup_{i=1}^{m}\mathrm{shm}\,N_i};$$

$$\frac{(\ldots)N_1(\ldots)}{\mathrm{shm}\,N_1}\ldots\frac{(\ldots)N_m(\ldots)}{\mathrm{shm}\,N_m} \quad \underset{\bigcap_{i=1}^{t}\mathrm{shm}\,N_i\supseteq\bigcup_{j=t+1}^{m}\mathrm{shm}\,N_j}{\overset{\neg N}{\mapsto}} \quad \frac{(N_1\ldots N_t)N(N_{t+1}\ldots N_m)}{\bigcup_{j=t+1}^{m}\mathrm{shm}\,N_j},$$

где в левой части правил (до знака $\mapsto$) задаются посылки, а в правой части – заключение; над и под знаком вывода указаны условия применения правил.

Понятийный анализ. *Понятийный анализ* определим как методику построения понятийной структуры предметной области. Основными этапами понятийного анализа являются:

– разделение сущностей предметной области на сигнификативные и денотационные;

– означивание сигнификативных сущностей и выявление существенных признаков у денотационных сущностей;

– сопоставление денотационных сущностей и определение их общих и различающихся признаков;

– образование новых или определение уже существующих понятий на основе интеграции и дифференциации признаков;

– создание понятийной структуры предметной области путем описания отображений одних понятий на другие;

– уточнение способа абстрагирования понятий (обобщения или типизации, ассоциации или агрегации);

– вычисление схем понятий и задание ключей – для типизации, связей – для ассоциации.

Схемы понятий вычисляются по следующей рекуррентной процедуре:

– схема простого понятия состоит из этого понятия;

– схема понятия, полученного в результате дифференциации признаков, равна пересечению схем дифференцируемых понятий;

– схема понятия, полученного в результате интеграции признаков, равна объединению схем интегрируемых понятий;

– схема понятия, полученного в результате дифференциации и интеграции признаков, равна объединению схем интегрируемых понятий, принадлежащая пересечению схем дифференцируемых понятий.

Из табл. 1 видно, что понятийный анализ является некоторым обобщением объектного. В отличие от объектного, результаты понятийного анализа легко формализуются и могут быть подвергнуты элементарной проверке на корректность.

Глава 3 посвящена контекстной технологии обработки данных, основанной на понятийном анализе предметной области, контекстной интерпретации текстов и определения семантики проблемного языка, создаваемого для решения задачи. Отличительной особенностью контекстной технологии является то, что для формализации знаний о некоторой предметной области, данные, выражающие эти знания, сопровождаются описанием их структуры (синтаксиса) и содержания (семантики).

Суть подхода заключается в реализации возможности выражать необходимые для описания предметной области понятия и найденные в процессе анализа декомпозиционные схемы в виде специализированной формальной системы, определять семантику этой системы и, в конечном итоге, выполнять содержательное описание предметной области и решение стоящих прикладных задач на созданном для этого проблемном языке.

Контекстная технология. Формализованное описание предметной области получим путем построения ее понятийной модели. В качестве программы будем использовать понятийную модель, дополненную описанием решения одной или нескольких прикладных задач:

Программа = Понятийная модель + Решение задачи.

В понятийной модели выделим три части: понятийную структуру, синтаксис форм выражения понятий и семантику каждой такой формы:

Понятийная модель = Структура + Синтаксис + Семантика.

Понятийную структуру и синтаксис понятий опишем на декларируемом в контекстной технологии протоязыке, а описание семантики и решаемых задач выполним на проблемном языке, определяемом в понятийной модели.

Привязку модели к целевой вычислительной платформе осуществим описанием семантики, как на определяемом языке, так и с использованием некоторого множества базовых примитивов. *Базовые примитивы* непосредственно реализуются целевой платформой и являются элементарными семантическими категориями предметной области.

В качестве *целевых платформ* используются такие аппаратурные и программно-аппаратные платформы как микроконтроллеры, не имеющие операционных систем, вычислительные системы с развитой операционной сре-

дой, кросс-платформенные виртуальные машины, другие системы программирования и т.п.

Таблица 1. Сравнение понятийного и объектного анализа

Понятийный анализ	**Объектный анализ**
Сущность (единичное понятие):	***Объект***:
– обладает уникальностью; – различается признаками; – выражает смысл.	– обладает идентичностью; – имеет состояние; – проявляет поведение.
Признак (элементарное понятие):	***Свойство*** (атрибут):
отличает одну сущность от другой; имеет имя, домен и семантическую роль	принимает различные значения и характеризует состояние объекта.
Понятие:	***Класс***:
множество сущностей, образованное на основе абстрагирования.	множество объектов, имеющих общую структуру и общее поведение.
Спецификация понятия:	***Спецификация класса***:
– имя (уникальность), схема (признаки); – интенсионал (содержание); – экстенсионал (состав).	– имя (идентичность); – свойства (состояние); – методы (поведение).
Взаимосвязь понятий:	***Взаимосвязь классов***:
– обобщение (есть некоторый); – агрегация (есть часть); – ассоциация (есть участник); – типизация (есть экземпляр).	– общее и частное (наследование); – целое и часть (агрегация); – зависимость (ассоциация).
Обобщение (расширенная типизация):	***Наследование***:
объединение сущностей дифференцируемых понятий.	наследуемый класс повторяет структуру и поведение базового класса.
Агрегация:	***Агрегация***:
соединение сущностей интегрируемых понятий.	отношения целого и части, приводящие к иерархии объектов.
Ассоциация (ограниченная агрегация):	***Ассоциация***:
связывание сущностей интегрируемых понятий.	зависимость классов, обеспечивающая переход между объектами этих классов.
Типизация:	***Виртуальные классы***:
идентификация сущностей дифференцируемых понятий.	объект базового класса замещается объектами различных классов.
Понятийная структура:	***Диаграммы***:
множество понятий с отображениями абстрагирования.	иерархия классов, диаграммы состояний и поведения объектов.

Протоязык. Протоязык необходим для выражения высказываний о проблемном языке. В отличие от базовых примитивов, которые декларируются в каждой модели перед использованием, протоязык является универсальным (общим для всех моделей).

Терм будем рассматривать как элементарную синтаксическую единицу текста, состоящую из знаков терминального алфавита. *Лексему* определим

как элементарную семантическую единицу текста, представленную одним или несколькими термами. *Текстом* будем называть последовательность лексем, предназначенную для выражения сложного смысла. Текст, в отличие от лексем, предполагает свое деление на смысловые части, в то время как лексема такого деления не допускает.

Протоязык контекстной технологии зададим грамматикой:

1	program	→	cognition [situation] [program]
2	cognition	→	essences [cognition]
3	essences	→	differenciation ***notion*** [integration] [intension]
4	differenciation	→	'(' [notions] ')'
5	integration	→	'(' [notions] ')'
6	notions	→	notion [alias] [notions]
7	intension	→	sentence [intension]
8	sentence	→	syntax semantic
9	syntax	→	item [parse] [compile] [syntax]
10	item	→	notion [alias] | lexeme [alias]
11	alias	→	'`' ***terms*** '`'
12	lexeme	→	term | pattern
13	term	→	'’' [***terms***] '’'
14	pattern	→	'”' [***terms***] '”'
15	semantic	→	pragmatic [semantic]
16	pragmatic	→	[***aspect***] '{' [text] '}'
17	parse	→	'<' [text] '>'
18	compile	→	[aspect] '[' [text] ']'
19	situation	→	[aspect] '<' [text] '>'
20	text	→	phrase [text]
21	phrase	→	terms | [aspect] '{' text '}'

где нетерминальные понятия обозначены строками над терминальным алфавитом, а терминальные понятия заключены в одинарные кавычки. Грамматика задана с точностью до пробелов и обозначенных курсивом нетерминальных знаков ***notion***, ***aspect***, ***terms***, служащих для выражения имен определяемых понятий, аспектов, лексем и алиасов. В квадратных скобках заданы нетерминальные знаки, которые могут быть опущены, а альтернативные правые части правил грамматики разделены вертикальной чертой.

Текст программы. Программа program состоит из понятийной модели cognition и ситуационной части situation (правило 1). В модели определяется язык, на котором в ситуационной части описывается решение прикладной задачи. Понятийная модель cognition является законченным фрагментом определения проблемного языка, а ситуация situation может содержать полное решение задачи, если в приведенных ранее фрагментах модели проблемный язык задан полностью, или часть решения, когда язык определен в объеме некоторого подъязыка, достаточного для описания только этой части.

Понятийная модель. Понятийная модель cognition состоит из описаний сущностей предметной области essences (правило 2). Сущностям при-

сваивается имя ***notion*** нетерминального понятия определяемого языка, а само понятие задается как дифференциация differenciation и интеграция integration ранее определенных понятий notions (правила 3-6).

Для указания на отсутствие у понятия абстракций используются пустые круглые скобки (правила 4, 5). В этом случае понятие считается обобщенным от «пустого» понятия empty или агрегирующим это понятие. Для выражения перекрестных связей между понятиями возможно их предварительное объявление, которое не включает описание содержания понятия intension (правило 3).

Выражение абстракций. Конструкция differentiation служит для выражения обобщения или типизации понятия (правило 4). Если схемы дифференцируемых понятий одинаковы, то имеем случай типизации. В противном случае, differentiation выражает обобщение. Определение исходного понятия для сущности из экстенсионала понятия-типа осуществляется через ключ и описывается в конструкциях sentence типизируемых понятий.

Конструкция integration позволяет выразить агрегацию или ассоциацию (правило 5). Если интенсионал понятия, задаваемый описанием его содержания intension, ограничивает вхождение сущностей интегрируемых понятий в экстенсионал образуемого понятия, то реализуется ассоциация. В противном случае интеграция выражает агрегацию. Переход от сущностей одного из ассоциированных понятий к другим таким сущностям осуществляется через связь и описывается предложениями sentence ассоциированных понятий.

Для ссылок на понятия в списках notions могут использоваться алиасы alias (правило 6). Алиасы служат для реализации обратных абстракций: декомпозиции (индивидуализации) – для интегрированных понятий, и конкретизации (специализации) – для дифференцированных. Необходимость использования обратных абстракции возникает при описании семантики предложений проблемного языка.

Интенсионалы понятий. Содержание понятия intension состоит из предложений sentence (правило 7). Предложения служат для определения интенсионала понятия и содержат как формы его выражения syntax, так и семантику semantic (правило 8). С каждым предложением связывается понятие-результат именем ***notion*** которого названы описываемые сущности (правило 3), определяемые предложением полностью – когда больше нет предложений с тем же понятием-результатом, или частично – если такие предложения имеются (правило 7).

Синтаксис предложения syntax выражается последовательностью элементов item: понятий notion и лексем lexeme (правило 9). Лексема является терминальным понятием определяемого языка. Для выражения лексем используются как терм term, так и множества термов, задаваемых на языке регулярных выражений в виде шаблонов pattern (правила 12-14). Для ссылок на отдельные элементы item применяются алиасы (правила 6, 11).

Средства расширения. Предложение sentence является правилом вы-

вода грамматики определяемого языка с тем отличием, что в его состав введены конструкции parse и compile (правило 9). Конструкция parse (правило 17) служит для расширения протоязыка и подлежит компиляции и исполнению при протоязыковом разборе предложения. Конструкция compile (правило 18) используется для реализации компилятора проблемного языка, компилируется и сохраняется в структуре предложения для исполнения во время распознавания этого предложения в тексте.

Прагматика. Семантику понятий будем задавать в виде текста, построенного по правилам определяемого языка. Для этого каждое предложение sentence дополним описанием его семантики semantic, которую выразим множеством прагматик pragmatic (правила 8, 15). Текст прагматики pragmatic компилируется в *императив* – некоторую последовательность действий, которую система программирования выполняет всякий раз, когда понятие выражается этим предложением (правило 16). По своей сути императивы – *единицы вызова* целевой вычислительной платформы, в то время как синтаксис предложения – описание структуры таких вызовов.

Аспекты. Понятийная структура может быть общей для нескольких задач. Предусмотрим возможность задания для каждого предложения одной или нескольких прагматик (правило 15), которые будем именовать (правило 16) и называть *аспектами* aspect. В итоге, любой текст на специализированном предметной языке может задавать решение целого класса задач, а его конкретизация осуществляться указанием на одну из определенных прагматик (правила 16, 19, 21).

Контекстные условия. Контекстные условия в протоязыке задаются для понятий notion, аспектов aspect и лексем lexeme, а места их задания в грамматике протоязыка выделены курсивом (правила 3, 13, 14, 16).

Имена понятий ***notion*** как терминальных понятий протоязыка объявляются при описании сущностей essences (правило 3). После такого объявления эти имена могут появляться в списке notions (правила 4, 5) и как элементы item при определении синтаксиса syntax (правила 9, 10).

Аспекты ***aspect*** являются именами прагматик pragmatic (правило 16), назначение которых – определение именованных семантических интерпретации фрагментов текста phrase (правило 21) и текстов в описании семантики semantic, грамматического разбора parse, компиляции compile и ситуации situation (правила 16-19).

Лексемы lexeme задаются в виде термов term и шаблонов pattern (правило 12) и используются для выражения терминальных понятий ***terms*** определяемого языка (правила 13, 14).

Демонстрационный пример. Рассмотрим понятийную модель предметной области «Исчисление высказываний»:

```
() Variable ()
    ″[A-Za-z][A-Za-z0-9]*″ [...] { }
() Constant ()
    'false' [ asm{ mov eax, 0; push eax } ] { }
```

```
        'true' [ asm{ mov eax, -1; push eax } ] { }
            bit[ asm{mov eax, 1; push eax} ] { }
    (Variable) Logic ()
        Variable [ asm{ pop ebx; mov eax, [ebx]; push eax } ] { }
        Integer [ asm{ pop eax; cmp eax, 0; je m; mov eax, -1; m: push eax } ]
            bit[ asm{ pop eax; and eax, 1; push eax } ] { }
        '(' Boolean ')' { }
    (Constant Logic) Negation  ()
        'not' Logic [ asm{ pop eax; not eax; push eax } ] { }
    (Negation) Conjunction ()
        Negation 'and' Negation
            [ asm{ pop eax; pop edx; and eax, edx; push eax } ] { }
    (Conjunction) Disjunction ()
        Conjunction `a` 'or' Conjunction `b` { not a and not b }
    (Disjunction) Boolean ()
        Disjunction `a` 'imp' Disjunction `b` { not a or b },
```

где Constant именует простое понятие «логическая константа», а Variable – «пропозиционная переменная». Другие понятия являются сложными: Logic обозначает понятие «логическое значение», Negation – «отрицание», Conjunction – «конъюнкция», Disjunction – «дизъюнкция», Boolean – «высказывание». Constant, Logic, Negation, Conjunction и Disjunction являются частными случаями Boolean. Однако выражение этих частных случаев осуществляется по-разному. Например, Negation получено обобщением Logic и Constant, т.е. Logic и Constant является конкретизацией Negation.

В демонстрационных целях семантика языка исчисления высказываний определена низкоуровневыми средствами. Для доступа к данным и организации их временного хранения использован аппаратный стек, а для хранения переменных – память произвольного доступа с линейной организацией. В предложении ″[A-Za-z][A-Za-z0-9]*″ текст компиляции (не показан) описывает создание переменной путем включения ее имени в некоторую таблицу идентификаторов и выделения памяти требуемого объема. Логические константы 'false' и 'true' реализованы как занесение нуля и минус единицы на вершину стека. Переменная определена адресом ячейки памяти. Для получения значения переменной Variable ее адрес извлекается из стека, а содержимое адресуемой ячейки заносится в стек. Для преобразования целого числа в булево значение использовано предложение Integer. Предложение '(' Boolean ')' { } не нуждается в императиве, так как его роль – задание приоритета выражений, заключенных в круглые скобки.

В примере определены две прагматики: по умолчанию (без имени) и с именем аспекта bit. В первом случае логический ноль кодируется арифметическим нулем, а логическая единица – минус единицей. Во втором случае логическая единица кодируется арифметической единицей. Для порождения кода используется прагматика, определенная аспектом asm (в примере не определена): текст в фигурных скобках, помеченных аспектом asm, переда-

ется целевой системе программирования – ассемблеру аппаратной платформы.

Семантика последних двух предложений выражена на уже определенном к тому времени специализированном предметном подъязыке, для чего использованы известные тождества: $a \vee b = \overline{a}\ \&\ \overline{b}$ и $a \to b = \overline{a} \vee b$, где a – алиас первого понятия предложения, b – второго.

Текст ситуационной части <(not x or y) and z> приведет к вычислению этого выражения при арифметическом кодировании логических значений, а bit < (not x or y) and z > – при битовом. В рассмотренной понятийной модели возможно задание других прагматик, например, для вычислений в нечеткой логике. Тогда текст fuzzy < (not x or y) and z >, где fuzzy – аспект нечетких вычислений, будет интерпретирован как нечеткое высказывание.

Семантическое замыкание. Предметный язык служит для описания не только ситуационной части (text в правиле 19), но и для задания семантики предложений и описания самого процесса грамматического разбора и компиляции (text в правилах 7, 17, 18). Тем самым достигается синтаксическая и семантическая замкнутость понятийной модели (рис. 3), позволяющая использовать для обработки перечисленных выше текстов одни и те же средства.

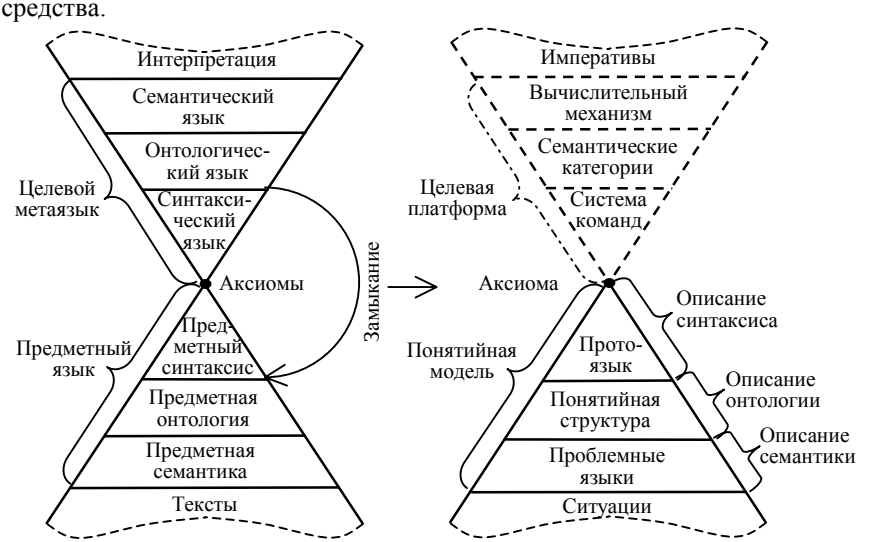


Рис. 3. Семантическое замыкание

Семантическая индукция. Описание синтаксиса задается на протоязыке, а семантика описывается текстами, состоящими из фраз phrase (правило 20), которые построены по правилам, задаваемым синтаксисом. Описание семантики осуществляется на основе метода математической индукции, заключающегося в том, что семантические категории модели определяются по мере необходимости, в процессе определения проблемного языка

и средствами этого языка, т.е. проблемный язык одновременно является и семантическим языком, служащим для своего описания.

В итоге, семантика понятий задается путем декларации первичных семантических категорий, которые непосредственно реализуются целевой платформой (база индукции), и описания новых семантических категорий текстами на проблемном языке, для чего используются ранее определенные семантические категории (индуктивный переход).

Синтаксически-управляемые схемы. Рассмотрим в качестве примера понятийную модель для дифференцирования выражений, включающих целочисленные константы, переменную x, функции sin и cos, а также алгебраические операции: изменение знака –, сложение +, вычитание –, умножение *.

Свяжем с каждым предложением два перевода, формируемые прагматикой по умолчанию и именованной прагматикой dif. Прагматика по умолчанию указывает на то, что выражение не дифференцируется, а именованная – что выражение необходимо продифференцировать. Таким образом, формальная производная некоторого текста s – это dif{s}. Процесс перевода опишем так:

```
() Expression (String)
    ″ [0-9]+ ″ `n`
        { n }
        dif { ‘0’ }
    'x'
        { 'x' }
        dif { '1' }
    '(' Expression ')' `exp`
        { '(' & exp & ')' }
        dif { '(' & dif { exp } & ')' }
    'sin' '(' Expression `exp` ')'
        { 'sin(' & exp & ')' }
        dif { 'cos(' & exp & ')*(' & dif { exp } & ')' }
    'cos' '(' Expression `exp` ')'
        { 'cos(' & exp & ')' }
        dif { '–sin(' & exp & ')*(' & dif { exp } & ')' }
    ‘–’ Expression `exp`
        { '–' & exp }
        dif { '–' & dif { exp } }
    Expression `exp1` '*' Expression `exp2`
        { exp1 & '*' & exp2 }
        dif {'('& exp1 & '*' & dif { exp2 } &'+'& dif { exp1 } & '*' & exp2 & ')'}
    Expression `exp1` ″+ | –″ `oper` Expression `exp2`
        { exp1 & oper & exp2 }
        dif { '(' & dif { exp1 } & oper & dif { exp2 } & ')' },
```

где знаком & обозначена конкатенация строк, определенная при описании

понятия String (в примере не показано). Текст ситуационной части dif { sin(5*cos(x)) – x*x } будет переведен так:

(cos(5*cos(x))*(5* – sin(x)*(1) + 0*cos(x)*(1))–(x*1 + 1*x).

Для тождественных преобразований выражений, получаемых после дифференцирования, возможно определение прагматики с именем equ. В результате ее применения к рассматриваемой строке имеем:

-5*(cos(5*cos(x))*sin(x))–2*x.

Атрибутные грамматики. Реализуем трансляцию текстового представления числа, заданного в формате с фиксированной запятой, в его двоичный эквивалент. Семантика такой задачи традиционно описывается атрибутной грамматикой. Для выражения числа Fixed введем два дополнительных понятия: Integer (целая часть) и Fraction (дробная). Понятию Integer припишем атрибут int, равный значению целой части числа, а понятию Fraction – атрибут frac, равный его дробной части. Атрибуты сопоставим сущностям выражаемых понятий. В итоге имеем следующую понятийную модель:

```
(Number) Integer ()
    ″[0-9]″ `digit` { digit }
    Integer `int` ″[0-9]″ `digit` { int * 10 + digit }
(Float) Fraction ()
    ″[0-9]″ `digit` { digit }
    ″[0-9]″ `digit` Fraction `frac` { digit + frac / 10 }
() Fixed (Float)
    Integer `int` '.' Fraction `frac` { int + frac / 10 },
```

где использованы понятия Number (целое число) и Float (число с плавающей запятой), при описании которых определяются необходимые арифметические операции. В отличие от атрибутных грамматик, не имеющих средств для задания порядка вычисления атрибутов, в рассматриваемом примере порядок вычисления атрибутов определен выразительными средствами протоязыка. Так для вычисления атрибута int понятие Integer определено как подлежащее грамматическому разбору слева направо, а для вычисления атрибута frac понятие Fraction подвергается разбору справа налево.

Глава 4 посвящена исследованию системы контекстного программирования, которая является одной из реализаций технологии контекстной обработки данных на основе спецификации предметной области в форме языка-письма. Однако, не видится препятствий для построения на тех же принципах системы обработки, например, речевых данных, реализующей другой способ обработки – в форме языка-речи.

Суть подхода заключается в использовании в качестве программы понятийной модели предметной области, которая дополнена решением прикладной задачи, выраженном на создаваемом в процессе описания проблемном языке и предваряющем это решение. Отсюда, в частности, следует, что понятийная модель и решение задачи должны быть подвергнуты грамматическому разбору и компиляции таким образом, чтобы откомпилированные

ранее предложения могли быть использованы при грамматическом разборе и компиляции следующих.

Контекстная интерпретация. При контекстной обработке определение семантического значения терма осуществляется по его месту в тексте, т.е. одна и та же последовательность терминальных знаков может быть разбита на термы различным образом. Следовательно, разделение анализа текста на фазы лексического и синтаксического анализа не представляется возможным. Применим специальный метод грамматического разбора, позволяющий выполнить контекстную интерпретацию фрагментов текста.

Разнесенный разбор. *Разнесенный* грамматический разбор заключается в разделении определения применимости предложений на две части: контекстное сопоставление, осуществляемое при просмотре текста назад, и структурное распознавание, выполняемое при просмотре вперед. Для этого разделим описание синтаксиса предложений на четыре области: контекст, лексему, область разбора и результат.

Контекстом предложения назовем последовательность понятий, предшествующих первой лексеме. Под *лексемой* предложения будем понимать его первую лексему. Область *разбора* определим как часть предложения, которая непосредственно следует за первой лексемой. И, наконец, *результат* предложения – это понятие, выражаемое этим предложением.

Контекстное сопоставление. Пусть имеется структура данных, которую назовем *текущим контекстом* и реализуем в виде стека контекста (рис. 4). Текущий контекст содержит последовательность понятий, которые уже распознаны, а соответствующий этим понятиям текст извлечен из входного потока. Таким образом, в каждый момент времени контекстный анализатор имеет некоторое текущее состояние, определяемое состоянием стека контекста и текущей позицией во входном потоке.

Поиск предложений, применимых в текущем состоянии анализатора, осуществим путем сравнения контекста предложений с текущим контекстом. Только предложения, имеющие сопоставимый контекст, могут использоваться в текущем состоянии анализатора. Здесь под *сопоставимостью* понимается соответствие понятий на вершине стека контекста понятиям из контекста предложений. Из сопоставимых предложений следует выбрать только *применимые*, лексема которых представляется текущим термом из входного потока. Таким образом, текст, подлежащий просмотру назад, представлен текущим контекстом, а его грамматический разбор выполняется до сопоставления предложений.

Структурное распознавание. После выявления применимого предложения наступает этап разбора оставшейся его части, которая еще не сопоставлена входному потоку. Для этого контекстный анализатор запоминает свое состояние и выполняет просмотр вперед при пустом контексте.

Если элементом разбора является лексема, то выполняется ее сравнение с текущим термом из входного потока. При успешном сравнении терм извлекается из входного потока, а в случае неудачи – состояние анализатора

восстанавливается, а анализируемое предложение считается нераспознанным. Если требуется распознать понятие, то для анализа выбираются только те предложения, которые имеют это понятие или его специализацию (конкретизацию) в качестве результата. При удачном распознавании анализатор переходит к следующему элементу из области разбора. В противном случае анализируемое предложение считается нераспознанным, состояние анализатора восстанавливается, и он переходит к разбору следующего применимого предложения.

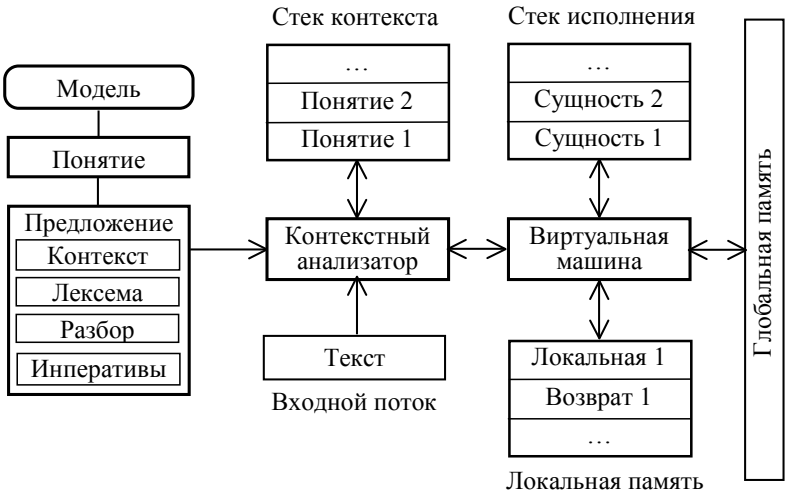


Рис. 4. Архитектура системы программирования

Если все элементы области разбора сопоставлены входному потоку, то предложение считается *распознанным*, контекст предложения в стеке контекста замещается на его понятие-результат, а полученное состояние входного потока объявляется текущим.

Приоритеты. Для упорядочивания грамматического разбора установим приоритетность предложений и понятий. Приоритет предложений, задаваемый порядком их перечисления в понятийной модели, назовем *естественным* или абсолютным. Проверку применимости и сопоставление предложений осуществим в обратном порядке, т.е. в соответствии с их естественным приоритетом.

При структурном распознавании внутри группы предложений с одним и тем же понятием-результатом приоритеты предложений, как и ранее, зададим их абсолютным приоритетом. Однако приоритет самого понятия определим его относительным местом в понятийной структуре: из двух предложений, выражающих сопоставимые понятия, выбирается то, понятие которого определено позже или получено при дифференциации понятия другого предложения. Тем самым учитываются *относительные* приоритеты

понятий, задаваемые понятийной структурой.

Конверторы. Протоязык контекстной технологии позволяет определять предложения вида sentence → notions, не имеющие лексем и состоящие из одного контекста. Такие предложения будем называть *конверторами*. Конверторы задают денотационную форму выражения понятия, в то время как предложения, состоящие только из лексем, – сигнификативную. Контекстное сопоставление и структурное распознавание следует осуществлять с учетом конверторов, устанавливающих эквивалентность одного или нескольких понятий другому.

Неоднозначные грамматики. При разнесенном грамматическом разборе возможен анализ текста, порождаемого неоднозначными грамматиками. Для учета неоднозначности контекстный анализатор перед началом структурного распознавания сохраняет свое состояние (создает точку отката назад) и начинает разбор предложения. Если в некотором состоянии анализатора не найдено применимых предложений, то восстанавливается сохраненное последним состояние (производится откат назад) и анализируется следующее предложение.

Описанная процедура осуществляется рекурсивным вызовом анализатора всякий раз, когда начинается распознавание нового предложения. Учитывая то, что понятийная модель предназначена для описания предметной области естественным образом, количество неоднозначностей в тексте не должно быть большим. Если неоднозначностей будет много, то предусмотрим некоторое критическое количество точек отката назад. Тогда, при достижении этого количества, делается вывод о недостаточной проработке понятийной модели.

Специальные лексемы. Введем в использование лексемы специального вида, инициирующие грамматический разбор «пустого» понятия empty и произвольного понятия, ранее объявленного в понятийной модели. Грамматический разбор empty задается лексемой вида " (две одинарные кавычки) и используется в том случае, когда из входного потока необходимо извлечь текст, не выражающий никакого понятия. Когда необходимо выражать суждения о понятиях, разбор и извлечение из входного потока текста, выражающего некоторое ранее описанное понятие, задается лексемой ″″ (две двойные кавычки). В этом случае алиас этой лексемы принимает значение имени распознанного понятия.

Аксиома. Аксиома необходима для машинной интерпретации текстов на проблемном языке и реализует привязку понятийной модели к целевой вычислительной платформе. Рассмотрим предложение для записи в область кода числа, например, байта: () () '#' "[0-9A-F][0-9A-F]" {}, где декларируется понятие empty и определяется предложение, выражающее это понятие в виде двух лексем: терма '#' и шаблона "[0-9A-F][0-9A-F]", описывающего две шестнадцатеричные цифры. Так как рассматриваемое предложение первое, то не существует средств для задания его семантики. Для разрешения этой проблемы будем использовать следующее соглашение (аксиому): *«пустые»*

квадратные скобки реализуют запись в область кода того значения, которое задается элементом предложения, после которого эти скобки указаны.

С учетом аксиомы переопределим первое предложение так: '#' "[0-9A-F][0-9A-F]" [ ] { }. Этого уже достаточно для задания семантики всех других предложений. Заметим, что запись в область кода двоичных данных является частной интерпретацией аксиомы. Для генерации некоторого промежуточного (текстового) представления возможно, например, такое определение: '`' "[^`]+" [ ] '`' { }, которое реализует запись в область кода мнемонического обозначения команды виртуальной машины или строки целевого языка программирования. В последнем примере текст, подлежащий записи, заключается в обратные одинарные кавычки и может содержать произвольные знаки.

Организация системы. Реализована система контекстного программирования, состоящая из следующих частей (рис. 5): входной поток Stream, лексический анализатор Token с препроцессором Text и словарем макроопределений Vocabulary, синтаксический анализатор Parser, семантический анализатор Analyzer, виртуальные машины Engine и понятийная модель Cognition.

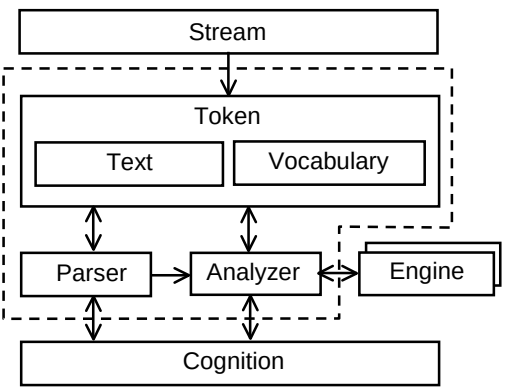


Рис. 5. Структура системы программирования

Входной поток Stream организован в виде файла и содержит текст программы. Результатом обработки текста является понятийная модель Cognition. Если во входном файле встретится текст ситуационной части, то после компиляции происходит его выполнение. Лексический анализатор Token предназначен для контекстного выделения лексем. При его реализации предусматривается сохранение (восстановление) состояния входного потока, препроцессора Text и словаря макроопределений Vocabulary.

Основное назначение синтаксического анализатора Parser – грамматический разбор той части текста, которая описывается грамматикой протоязыка. Другая часть текста, выраженная на определяемом языке, обрабатывается семантическим анализатором Analyzer.

Для выполнения императивов используется целевая вычислительная платформа. Со стороны системы программирования целевая платформа представляется одной или несколькими виртуальными машинами Engine.

Фазы работы. Система программирования в каждый момент времени находится в одной из следующих фаз: в фазе разбора, в фазе компиляция ил в фазе выполнение. В *фазе разбора* выполняется разбор текста, заданного на протоязыке. В *фазе компиляции* происходит преобразование текста на проблемном языке в его императив. В *фазе выполнения* исполняется императив, порожденный в фазе компиляции.

Императивы. На вход Engine подаются императивы, являющиеся ее *единицами вызова*. Каждая единица вызова может представляться командами непосредственного исполнения, интерпретируемыми операторами, текстом на входном языке другой системы программирования и т.п. В последних двух случаях виртуальная машина из *промежуточного* кода, создаваемого во время компиляции, генерирует его *исполняемый* код – императив, который в виде идентификатора образа памяти возвращается для сохранения в понятийной модели Cognition и использования при повторном вызове того же промежуточного кода без повторной генерации исполняемого. Допускается существование нескольких императивов у каждого промежуточного кода, по одному для каждого типа виртуальных машин.

Динамическая семантика. Сущности представляются данными, которые выражают соответствующие этим сущностям понятия. Во время описания семантики предложений возникает необходимость в создании, использовании и уничтожении временных сущностей, для чего используется специальный механизм добавления и удаления временных предложений. В простейшем случае такие предложения вводят имя сущности и связывают это имя с некоторым понятием. Таким способом реализуется, например, таблица контекстных и бесконтекстных идентификаторов.

Системные сервисы. Ряд функций системы программирования, вытекающих из особенностей понятийного анализа и требований контекстной технологии обработки данных реализованы в виде системных сервисов.

Сервисы обратного вызова служат для выполнения действий, которые активируются из области кода специальными командами:

– *сервис компиляции* – для записи в текущий компилируемый императив данных различных типов;

– *информационный сервис* – для доступа к данным, сохраненным в понятийной модели;

– *сервис состояний* – для управления состоянием синтаксического и семантического анализаторов;

– *сервис контекстных условий* – для извлечения контекстов и определения предложений динамической семантики;

– *сервис временной памяти* – для организации взаимосвязи различных императивов, задаваемых в пределах одного предложения.

Сервисы прямого вызова предназначены для переопределения стан-

дартных действий, выполняемых системой в интерфейсных точках:

– *жизненного цикла* сущности (декларация, создание, извлечение, передача, возврат, копирование, уничтожение);

– *генерации кода* (запись вызова императива, определение эпилога и пролога таких вызовов, доступ к сущностям-аргументам);

– *активации* (перед вызовом и после вызова виртуальных машин).

Контекстность языка. Проверка правого и левого контекста нетерминального понятия проблемного языка осуществляется в текстах компиляции, для чего используются вызов соответствующего сервиса системы программирования. Последнее делает выразительные возможности проблемных языков эквивалентными контекстным языкам.

Глава 5 посвящена исследованию формально-логического подхода к синтезу математических моделей дискретной обработки данных. Необходимость использования формально-логического подхода вызвана тем, что существует класс задач, для эффективного решения которых неприменимы известные методологии анализа предметной области и соответствующие им технологии программирования, в том числе и описанный ранее понятийный подход и основанная на этом подходе контекстная технология.

При решении таких задач использование высокоуровневых моделей не представляется возможным по причине отсутствия у них необходимой содержательной интерпретации. Например, такие задачи возникают в цифровой обработке сигналов, проектировании дискретных устройств, первичной обработке измерительных данных, и в других областях, характерной особенностью которых является представление исходных и результирующих данных в виде таблиц (векторов, двумерных и многомерных массивов и других однородных структур).

Суть подхода заключается в том, что дискретная обработка данных, заданная в табличном виде, представляется как одна или несколько дискретных функций. Для синтеза формального описания дискретная функция декомпозируется и представляется в виде композиции функций меньшей размерности. Особенностью описываемого подхода является использование алгебраической декомпозиции дискретных функций в наиболее общей постановке задачи, когда с целью синтеза эффективных математических моделей дискретной обработки данных переменные и функции принимают значения на произвольных конечных множествах, а выбор алгебраических операций не ограничен каким-либо их подмножеством.

Дискретная обработка. Для кодирования данных выберем в качестве универсального множество целых чисел $N_0 = \{..., -1, 0, 1, ...\}$. Когда необходимо задать конечное множество из k элементов будем использовать подмножество N_k этого множества, $N_k = \{0, 1, ..., k-1\}$.

Обработку данных представим дискретной функцией f, заданной на множестве $N_{k_0} \times N_{k_1} \times \ldots \times N_{k_{n-1}}$ и принимающей значения на множестве N_{k_f}, где $\times$ – знак декартова произведения. Будем говорить, что функция имеет значность k_f и зависит от n переменных x_0, x_1, …, x_{n-1} со знач-

ностями k_0, k_1, …, k_{n-1} соответственно. При дискретной обработке преобразование входных данных в выходные происходит путем выполнения дискретных операций. *Операцией* будем называть функцию, существенно зависящую от своих переменных.

Рассмотрим две крайние формы описания дискретной обработки: регулярную и нерегулярную. *Нерегулярные* формы получают в результате декомпозиции общего вида, когда дискретная функция f, зависящая от переменных X, на каждом шаге представляется как композиция функций g и h,

$$f(X) = g(h(X'), X''),$$

где X', X'' – подмножества X. Декомпозиция завершается, когда g и h непосредственно реализуются вычислительным средством. Синтез нерегулярных форм без заранее заданной декомпозиционной схемы[2] практически нереализуем, ибо связан с перебором большого числа вариантов.

Регулярные формы основаны на алгебраической декомпозиции [48], осуществляемой в алгебре из двух операций $+$ и $\times$,

$$f(X) = \sum \theta_i(X') \times a_i(X''),$$

где θ_i, a_i – некоторые функции, которые будем называть *частичными*. Частным видом алгебраической декомпозиции является решающее разложение функции, применяемое при построении диаграмм решений и логических программ [Л1]:

$$f(X) = \sum \theta_i(x) \times a_i(X \setminus x),$$

где θ_i – система решающих функций, a_i – остаточные функции, \ – знак разности множеств. В случае, когда $X' = X$ и $X'' = \varnothing$, имеем широко известную спектральную форму,

$$f(X) = \sum \theta_i(X) \times a_i,$$

где θ_i – система спектральных функций, a_i – коэффициенты разложения.

Алгебраическая декомпозиция. Найдем условия, при которых алгебраическая декомпозиция существует. Разделим множество переменных функции $X = \{x_0, x_1, \ldots, x_{n-1}\}$ на два непересекающихся множества $X' = \{x'_0, x'_1, \ldots, x'_{n'-1}\}$ и $X'' = \{x''_0, x''_1, \ldots, x''_{n''-1}\}$ с числом переменных n' и n'' соответственно. Представим функцию f, заданную вектором значений длины m, зависящей от двух переменных x' и x'' со значностями k' и k'',

$$k' = \prod_{(x_j \in X')} k_j = \prod_{j=0}^{n'-1} k'_j, \quad k'' = \prod_{(x_j \in X'')} k_j = \prod_{j=0}^{n''-1} k''_j, \quad k'k'' = m.$$

Заметим, что разделение переменных порождает прямоугольную таблицу (матрицу) функции с числом строк k' и числом столбцов k'', причем изменение порядка переменных в множествах X' и X'' приводит, соответ-

[2] Здесь под декомпозиционной схемой понимается совокупность ограничений на вид функций g, h и множества переменных X', X'', которые сокращают перебор вариантов.

ственно, к перестановке строк и столбцов этой таблицы.

Разложим f по некоторой системе функций θ_i ($i = \overline{0, k'-1}$),

$$f(x', x'') = \sum_{i=0}^{k'-1} \theta_i(x') \times a_i(x''). \qquad (1)$$

При подстановке в (1) значений переменной $x' = \overline{0, k'-1}$ получаем систему алгебраических уравнений

$$\begin{cases} f(0, x'') = \theta_0(0) \times a_0(x'') + \ldots + \theta_{k'-1}(0) \times a_{k'-1}(x''); \\ f(1, x'') = \theta_0(1) \times a_0(x'') + \ldots + \theta_{k'-1}(1) \times a_{k'-1}(x''); \\ \vdots \quad \vdots \quad \vdots \quad \ddots \quad \vdots \\ f(k'-1, x'') = \theta_0(k'-1) \times a_0(x'') + \ldots + \theta_{k'-1}(k'-1) \times a_{k'-1}(x''), \end{cases} \qquad (2)$$

из которой требуется найти a_i ($i = \overline{0, k'-1}$). В матричном виде система (2) представляется так:

$$\mathbf{F} = \mathbf{D} \times \mathbf{A} \quad (\mathbf{A} = \mathbf{Q} \times \mathbf{F}), \qquad (3)$$

где $\mathbf{F}$ и $\mathbf{A}$ – матрицы функции и коэффициентов размерности $k' \times k''$, $\mathbf{D}$ и $\mathbf{Q}$ – квадратные матрицы прямого и обратного преобразования размерности $k' \times k'$ с элементами $d_{ij} = \theta_i(j)$ и $q_{ij} = \vartheta_i(j)$ соответственно. Столбцы матрицы $\mathbf{D}$ являются векторами частичных функций θ_i, а строки $\mathbf{A}$ – векторами частичных функций a_i.

Существует несколько алгебр, позволяющих разрешить систему (2) относительно неизвестных функций $a_i(x'')$. Такие алгебры будем называть *образующими*[3].

Алгебра логики. Рассмотрим алгебру $R_L = \langle N_k, +, \times \rangle$ и потребуем существование таких элементов σ и $\tau \neq \sigma$, что

$$a + \sigma = a, \quad \sigma + a = a, \quad \sigma \times a = \sigma, \quad \tau \times a = a$$

для всех $a \in N_k$. Элемент σ назовем *нулем*, а τ *– единицей* алгебры R_L.

Определение 1. *Матрицей перестановок* $\mathbf{P}_{k'}$ *порядка* k' *называется квадратная матрица, состоящая из нулей* σ *и единиц* τ *и имеющая в каждой строке (столбце) ровно по одной единице.*

Теорема 1. *Произвольная функция* f *представима в* R_L *разложением (1), если* $\mathbf{D}$ *является матрицей перестановок. Тогда* $\mathbf{Q} = \mathbf{D}^T$ *и* $\mathbf{Q} \times \mathbf{D} = \mathbf{E}$, *где* $\mathbf{E}$ *– единичная матрица,* T *– операция транспонирования.*

Мультипликативная алгебра. Определим образующую алгебру $R_M = \langle N_k, +, \times \rangle$, у которой операции удовлетворяют условиям алгебры R_L, а умножение дополнительно образует группу $G_M = \langle N_k \setminus \{\sigma\}, \times \rangle$ на множестве N_k без нулевого элемента σ. Последнее означает, что для любого элемента $a \in G_M$ существует обратный ему элемент a^{-1}, такой, что $a \times a^{-1} = \tau$.

Определение 2. *Мономиальной матрицей* $\mathbf{M}_{k'}$ *порядка* k' *называется*

[3] Известно [Л2], что разложение (1) возможно в полях и целостных кольцах (коммутативных кольцах с единицей и без делителей нуля). Следовательно, перечисленные алгебры являются образующими.

квадратная матрица, получаемая перестановкой строк (столбцов) диагональной матрицы с элементами $d_i \neq \sigma$.

Теорема 2. *Произвольная функция* f *представима в* R_M *разложением (1), если* $\mathbf{D}$ *является мономиальной. Тогда* $\mathbf{Q} = \tilde{\mathbf{D}}^T$ *и* $\mathbf{Q} \times \mathbf{D} = \mathbf{E}$, *где* $\mathbf{E}$ *– единичная матрица, а* $\tilde{\mathbf{D}}$ *получается из* $\mathbf{D}$ *заменой ненулевых элементов на обратные в группе* G_M.

Аддитивная алгебра. Пусть $R_A = \langle N_k, +, \times \rangle$, где операции сложения и умножения удовлетворяют условиям алгебры R_L, а сложение таково, что образует коммутативную группу $G_A = \langle N_k, + \rangle$, т.е. для любого элемента $a \in G_A$ существует противоположный ему элемент $-a$, такой, что $a + (-a) = \sigma$.

Определение 3. *Порядком элемента* a *группы* G_A *называется минимальное натуральное число* $c_a > 0$, *для которого*

$$c_a \cdot a = \underbrace{a + a + \ldots + a}_{c_a} = \sigma,$$

где σ *– нейтральный (нулевой) элемент группы* G_A.

Для определенности положим $0 \cdot a = \sigma$ и $-\lambda \cdot a = \lambda \cdot (-a)$ для любого $\lambda \in Z$, где Z – множество целых чисел.

Определение 4. *Циклическим порядком группы* G_A *будем называть минимальное значение порядков всех ее элементов, кроме нейтрального.*

Лемма 3. *В произвольной группе* G_A *уравнение вида* $\lambda \cdot a = b$, *где* $\lambda \in Z$, $a, b \in G_A$, *имеет единственное решение тогда и только тогда, когда циклический порядок группы больше* $|\lambda|$.

Определение 5. *Биполярной матрицей* $\mathbf{B}_{k'}$ *порядка* k' *называется квадратная матрица, состоящая из нулей* σ, *единиц* τ *и минус единиц* $-\tau$ *алгебры* R_A.

Определение 6. *Матрицей, сопряженной биполярной матрице* $\mathbf{B}_{k'}$, *называется такая матрица* $\mathbf{B}_{k'}$, *которая получена из* $\mathbf{B}_{k'}$ *заменой нулей, единиц и минус единиц* R_A *на нули, единицы и минус единицы кольца целых чисел* $R_Z = \langle Z, +, \times \rangle$.

Теорема 4. *Произвольная функция* f *представима в* R_A *разложением (1), если* $\mathbf{D}$ *является биполярной, и модуль определителя* Δ *сопряженной ей матрицы* $\tilde{\mathbf{D}}$ *не равен нулю и меньше циклического порядка группы* G_A. *Тогда* $\Delta \cdot \mathbf{A} = \bar{\mathbf{D}}^T \cdot \mathbf{F}$, *где* $\bar{\mathbf{D}}$ *– матрица алгебраических дополнений* $\tilde{\mathbf{D}}$, *вычисленная в кольце целых чисел* R_Z.

Заметим, что умножение из R_A применяется только при вычислении разлагаемых функции, для нахождения частичных функций используется циклическая сумма.

Фундаментальная алгебра. Пусть $R_F = \langle N_k, +, \times \rangle$, где операции сложения и умножения образуют поле и, тем самым, удовлетворяют условиям алгебр R_L, R_M и R_A. Известна следующая теорема [Л2].

Теорема 5. *Произвольная функция* f *представима в* R_F *разложением (1), если определитель* $\mathbf{D}$ *в* R_F *отличен от нуля. Тогда* $\mathbf{Q} = \mathbf{D}^{-1}$ *и*

$\mathbf{Q}\times\mathbf{D}=\mathbf{E}$, *где* $\mathbf{D}^{-1}$ *– матрица, обратная* $\mathbf{D}$ [4].

Сигнатуры образующих алгебр. Элемент a называется *инвертирующим* (слева и справа) на множестве $N^a \subseteq N_k$, если уравнения $a+x=b$ и $x+a=b$ имеют единственные решения относительно $x\in N_k$ для всех $b\in N^a$. Элемент a называется *обращающим* (слева) на множестве N^a, если уравнение $a\times x=b$ имеет единственное решение относительно x для всех $b\in N^a$. Элемент a называется *вырожденным* (слева) на множестве N^a, если элемент $x=a\times b$ может быть найден (выражен, определен) при любом наперед неизвестном $b\in N^a$. Такие элементы b будем называть *свободными*. Вырожденный элемент a называется *нейтральным*, если $b=a\times b$, *константным* – если $c=a\times b$ и c не зависит от b, *нулеобразующим* – если $\sigma=a\times b$ для всех b. Элемент a называется *неделимым* (слева) на множестве N^a, если уравнение $b\times x=a$ не имеет решений относительно x при любом $b\in N^a$.

В табл. 2 приведены свойства специальных элементов образующих алгебр, а отношения между алгебрами показаны на рис. 6а.

Таблица2. Сигнатуры образующих алгебр

Алгебра	Сложение	Умножение
Логики R_L	$\sigma+\sigma=\sigma$, σ – инвертирующий	ϕ – вырожденный ($\phi\times a=\sigma$), τ – обращающий
Мультипликативная R_M		ϕ – вырожденный ($\phi\times a=\sigma$), $N_k\setminus\{\sigma\}$ – обращающие
Аддитивная R_A	N_k – инвертирующие, ассоциативность, коммутативность	ϕ – вырожденный, τ – обращающий
Фундаментальная R_F		ϕ – неделимый на $N_k\setminus\{\sigma\}$, ассоциативность, коммутативность, дистрибутивность по сложению

Из таблицы находим, что операция сложения в аддитивной и фундаментальной алгебре образует коммутативную (абелеву) группу на множестве N_k с нейтральным элементом σ. Фундаментальная алгебра включает поле ($k>0$) или коммутативное кольцо без делителей нуля ($k=0$). В фундаментальной алгебре элемент τ становится нейтральным по умножению, а нулеобразующий элемент ϕ вырождается в σ. В остальных алгебрах элементы ϕ и σ могут различаться.

[4] Теорема 5 справедлива на целостном кольце. При конечном числе элементов целостное кольцо изоморфно конечному полю [Л2, с. 25], следовательно, носитель кольца – счетное множество N_0.

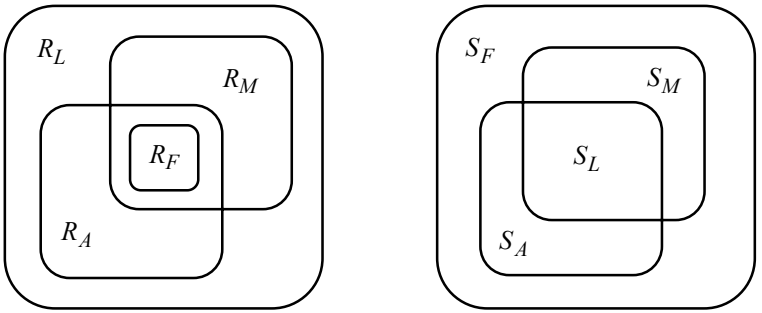


Рис. 6. Классы образующих алгебр и частичных функций

В аддитивной алгебре возможно различное вырождение ϕ и τ, но в любом случае требуется, чтобы хотя бы одно вырождение существенным образом зависело от свободного элемента. Частными случаями аддитивной алгебры являются униполярная и биполярная алгебры. *Униполярная* алгебра имеет традиционное вырождение элементов: $\phi \times x = \sigma$, $\tau \times x = x$. В *биполярной* алгебре элемент ϕ вырождается иначе: $\phi \times x = -x$, где $-x$ – элемент, противоположный x. В общем случае возможно существование еще одного вырожденного элемента – минус единицы: $-\tau \times x = -x$.

Функциональная полнота. По своей сути разложение (1) позволяет свести вычисление произвольной функции f к вычислению функций θ_i и a_i меньшей размерности, а теоремы разложения определяют условия, при которых базис $\{+, \times, \{\theta_i\}, < a >\}$ обладает функциональной полнотой, где $\{\theta_i\}$ – фиксированная система k'-функций, а $< a >$ – класс, включающий все k''-функции. Здесь под *k*-функцией понимается произвольная дискретная функция, задаваемая вектором значений длины *k*.

Классы функций. Из отсутствия ограничений на a_i следует, что четыре основных типа образующих алгебр порождают четыре класса частичных функций θ_i (табл. 3).

Таблица 3. Классы частичных функций

Класс функций	Двузначные	Многозначные
Унимодальные	Алгебры логики	Мультипликативные алгебры
Мультимодальные	Аддитивные алгебры	Фундаментальные алгебры

Унимодальными называются функции, принимающие значения, отличные от нулеобразующего элемента ϕ только в одной точке области определения. Если эти значения одинаковые, то система функций двузначная, в противном случае – многозначная.

К *мультимодальным* относятся функции, принимающие произвольные значения. Если множество значений включает только два элемента, то такие

функции называются двузначными, если более двух – то многозначными. Мультимодальные функции включают два подкласса – *униполярные* и *биполярные*, получаемые при униполярном и биполярном вырождении элементов. Классы частичных функций находятся в отношении включения друг в друга (рис. 6б).

Синтез формул. Покажем на примере синтез формульного представления дискретной функции в аддитивной алгебре R_A. Операции сложения и умножения зададим матрицами

$$+ = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 3 & 0 \\ 2 & 3 & 0 & 1 \\ 3 & 0 & 1 & 2 \end{bmatrix}, \quad \times = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

где + – сумма по модулю 4, образующая абелеву группу на множестве N_4 с нулем $\sigma = 0$ и циклическим порядком 2, а $\times$ – операция логического сдвига ($a \times b$ есть сдвиг b на a двоичных разрядов влево) с $\phi = 3$ и обращающим элементом $\tau = 0$.

Пусть функция определена векторами значений $\mathbf{F}$ и значностей $\mathbf{K}$,

$$\mathbf{F} = [3\,2\,2\,1\,0\,2\,3\,1\,2\,1\,0\,3\,1\,0\,3\,0\,2\,0\,0\,2\,0\,2\,2\,0], \quad \mathbf{K} = [2\,2\,3\,2],$$

т.е. зависит от переменных $X = \{x_0, x_1, x_2, x_3\}$ со значностями $k_0 = 2$, $k_1 = 2$, $k_2 = 3$ и $k_3 = 2$. Запишем (1) при $X' = \{x_1, x_3\}$ и $X'' = \{x_0, x_2\}$,

$$f = \theta_0(x') \times a_0(x'') + \theta_1(x') \times a_1(x'') + \theta_2(x') \times a_2(x'') + \theta_3(x') \times a_3(x'').$$

Зададим систему частичных функций биполярной матрицей $\mathbf{D}$ и найдем на кольце целых чисел определитель Δ и алгебраические дополнения $\underline{\mathbf{D}}$ сопряженной ей матрицы $\overline{\mathbf{D}}$:

$$\mathbf{D} = \begin{bmatrix} \tau & \phi & \phi & \phi \\ \tau & \tau & \phi & \phi \\ \tau & \phi & \tau & \phi \\ \tau & \phi & \phi & \tau \end{bmatrix}, \quad \widetilde{\mathbf{D}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad \Delta = 1, \quad \underline{\mathbf{D}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}.$$

Так как циклический порядок группы больше Δ, то в соответствии с теоремой 4 система функций $\mathbf{D}$ функционально полна. Тогда

$$\Delta \cdot \mathbf{A} = \underline{\mathbf{D}} \cdot \mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 & 0 & 2 & 2 & 0 \\ 2 & 1 & 3 & 1 & 1 & 3 \\ 1 & 0 & 2 & 0 & 0 & 2 \\ 3 & 2 & 0 & 2 & 2 & 0 \end{bmatrix},$$

где $\cdot$ – операция циклической суммы, $\mathbf{A}$ – матрица коэффициентов, $\mathbf{T}$ – матрица функции, найденная для заданного разделения переменных. Умножая $\underline{\mathbf{D}}$ и $\mathbf{T}$ относительно сложения и циклической суммы, находим $\Delta \cdot \mathbf{A}$, а затем, на основании леммы 3, и $\mathbf{A}$:

$$\Delta \cdot \mathbf{A} = \begin{bmatrix} 3 & 2 & 0 & 2 & 2 & 0 \\ 3 & 3 & 3 & 3 & 3 & 3 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 3 & 2 & 0 & 2 & 2 & 0 \\ 3 & 3 & 3 & 3 & 3 & 3 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

В итоге имеем искомую формулу функции,

$$f(X) = \begin{bmatrix} \tau \\ \tau \\ \tau \\ \tau \end{bmatrix} x' \times \begin{bmatrix} 3 \\ 2 \\ 0 \\ 2 \\ 2 \\ 0 \end{bmatrix} x'' + \begin{bmatrix} \phi \\ \tau \\ \phi \\ \phi \end{bmatrix} x' \times \begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{bmatrix} x'' + \begin{bmatrix} \phi \\ \phi \\ \tau \\ \phi \end{bmatrix} x' \times \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \end{bmatrix} x'' + \begin{bmatrix} \phi \\ \phi \\ \phi \\ \tau \end{bmatrix} x' \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} x'',$$

где $[y_i]x = y_x$ – унарная операция, задаваемая вектором $[y_i]$. После тождественных преобразований формулы с учетом свойств специальных элементов образующей алгебры, получим

$$f(X) = \begin{bmatrix} 3 \\ 2 \\ 0 \\ 2 \\ 2 \\ 0 \end{bmatrix} x'' + \begin{bmatrix} 3 \\ 0 \\ 3 \\ 3 \end{bmatrix} x' \times 3 + \begin{bmatrix} 3 \\ 3 \\ 0 \\ 3 \end{bmatrix} x' \times 2 + \begin{bmatrix} 3 \\ 3 \\ 3 \\ 0 \end{bmatrix} x' \times 0 = \begin{bmatrix} 3 \\ 2 \\ 0 \\ 2 \\ 2 \\ 0 \end{bmatrix} (x_0, x_2) + \begin{bmatrix} 0 \\ 3 \\ 2 \\ 0 \end{bmatrix} (x_1, x_3)$$

и, представив частичные функции от двух переменных операциями $\circ_0$ и $\circ_1$, имеем

$$f(X) = x_0 \begin{bmatrix} 3 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix} x_2 + x_1 \begin{bmatrix} 0 & 2 \\ 3 & 0 \end{bmatrix} x_3 = x_0 \circ_0 x_2 + x_1 \circ_1 x_3,$$

где $x[y_{ij}]z = y_{xz}$ – бинарная операция, задаваемая матрицей $[y_{ij}]$.

Глава 6 посвящена алгебраическому синтезу эффективных математических моделей дискретной обработки, где *эффективным* называется такой синтез, при котором представление функции содержит количество операций, не превосходящее некоторого максимального их числа, достаточного для реализации любой функции той же размерности.

Для сравнения различных декомпозиционных схем введем количественные характеристики. Для этого рассмотрим алгебраическую декомпозицию (1), у которой исключены выражения вида $\phi \times a_i(X'') = \sigma$,

$$f(X) = \sum_{i=0}^{M-1} \theta_i(X') \times a_i(X''), \qquad (4)$$

где M – количество слагаемых, $M \le k'$. *Сложность разложения* опреде-

лим количеством слагаемых M. Под *сложностью представления* L будем понимать количество операций, необходимых для вычисления функции по формуле (4).

Алгебраическая декомпозиция близка преобразованию Карунена-Лоэва [Л3, с. 182], где разложение осуществляется по собственным векторам ковариационной матрицы функции и, тем самым, обеспечивается наилучшее (оптимальное) приближение функции в среднеквадратическом смысле, а также достигается наименьшая сложность разложения. В этом случае частичные функции вычисляются с некоторой степенью свободы, а их количество равно числу ненулевых собственных значений ковариационной матрицы. Воспользуемся оставшимися степенями свободы для достижения не только наименьшая сложность разложения, но и наименьшей сложности представления функции.

Конструкции формул. При алгебраической декомпозиции имеется возможность использования частичных функций с несущественной зависимостью от части переменных, что позволяет синтезировать формулы, как с повторными вхождениями переменных, так и с их отсутствием. Однако, вхождения в аналитическую конструкцию частичной функции одной и той же переменной более одного раза приводят к ее принципиальной неотличимости от выражений, получаемых в результате алгебраической декомпозиции. Следовательно, не нарушая общности, можно потребовать отсутствия повторного вхождения переменных в конструкцию частичных функций.

Поэтому рассмотрим бесповторную аналитическую конструкцию, заданную с точностью до расстановки скобок и порядка вхождения переменных,

$$\theta(X) = \triangleleft_0\, x_0 \circ_0 \triangleleft_1\, x_1 \circ_1 \dots \circ_{n-2} \triangleleft_{n-1}\, x_{n-1}, \qquad (5)$$

где $\triangleleft_j$ ($\circ_j$) – произвольные унарные (бинарные) операции. После тождественных преобразований (5) имеем ее *каноническую* форму

$$\theta(X) = x_{i_0} \circ_0 x_{i_1} \circ_1 x_{i_2} \circ_2 \dots \circ_{n'-2} x_{i_{n'-1}}, \qquad (6)$$

также заданную с точностью до расстановки скобок, но уже не содержащую унарных операций и несущественных переменных ($n' \le n$).

Бесскобочная конструкция. Подсчитаем количество различных функций N_θ^*, порождаемых бесскобочным вариантом конструкции (6). Для этого рассмотрим бинарные операции от двух переменных со значностями k_0, k_1 и найдем количество порождаемых ими функций $N_o(k,k_0,k_1)$ значности k. Общее число функций равно $k^{k_0 k_1}$ и

$$\sum_{j=0}^{k} C_k^j N_o(j,k_0,k_1) = k^{k_0 k_1},$$

где множитель C_k^j – число сочетаний из k элементов по j, который учитывает функции значности j, принимающие значения не на всем множестве N_k, а на различных его подмножествах из j элементов. Отсюда получаем рекуррентное выражение для $N_o(k,k_0,k_1)$,

$$N_o(0,k_0,k_1)=0\,,\quad N_o(k,k_0,k_1)=k^{k_0k_1}-\sum_{j=0}^{k-1}C_k^jN_o(j,k_0,k_1)\,. \qquad (7)$$

Аналогично находим $N_o^{\rightarrow}(k,k_0,k_1)$ – количество функций значности k, порождаемых всевозможными бинарными операциями при их более чем существенной зависимости от левой переменной, т.е. когда матрицы операций не содержат одинаковых строк:

$$N_o^{\rightarrow}(1,k_0,k_1)=1\,,\quad N_o^{\rightarrow}(k,k_0,k_1)=\prod_{i=0}^{k_0-1}(k^{k_1}-i)-\sum_{j=1}^{k-1}C_k^jN_o^{\rightarrow}(j,k_0,k_1)\,. \qquad (8)$$

Для подсчета количества функций N_θ^*, порождаемых конструкцией (6), используем систему рекуррентных уравнений,

$$\begin{cases} N_\theta(k_f,k_0,k_1)=N_o(k_f,k_0,k_1), \\ N_\theta(k_f,k_0,k_1,\ldots,k_t)=\sum\limits_{i=1}^{k_f}\dfrac{C_{k_f}^i}{i!}N_\theta(i,k_0,k_1,\ldots,k_{t-1})N_o^{\rightarrow}(i,i,k_t), \\ N_\theta^*(k,k_0,k_1,\ldots,k_{n-1})=\sum\limits_{j=1}^{k}C_k^jN_\theta(j,k_0,k_1,\ldots,k_{n-1}), \end{cases} \qquad (9)$$

Из (7) и (8) следует

$$k^{k_0(k_1-1)}<N_o^{\rightarrow}(k,k_0,k_1)<N_o(k,k_0,k_1)<k^{k_0k_1}\,,$$

а с учетом (9) получаем оценку для N_θ^* при $n>2$,

$$k^{k_0k_1+(k-1)\sum_{i=2}^{n-1}k_i}<N_\theta^*(k,k_0,k_1,\ldots,k_{n-1})<k^{k_0k_1+k\sum_{i=2}^{n-1}k_i}\,. \qquad (10)$$

Оценка (10) допускает достаточно простую содержательную интерпретацию: верхняя граница степеней свободы аналитической конструкции меньше суммарной площади операций, а нижняя граница определяется при более чем существенной зависимости операций от правого операнда, что эквивалентно вычеркиванию одной строки у каждой операции, кроме нулевой (рис. 7).

Вместо неравенств (10) будем использовать равенство

$$N_\theta^*(k,k_0,k_1,\ldots,k_{n-1})=k^{k_0k_1+(k-\alpha)\sum_{i=2}^{n-1}k_i}\,, \qquad (11)$$

где α – *порождающая способность* аналитической конструкции, $0<\alpha<1$.

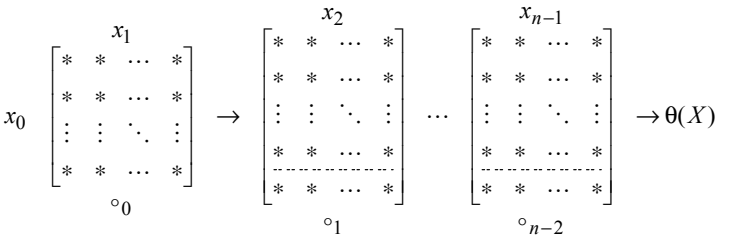


Рис. 7. Бесскобочная аналитическая конструкция

Вычисление α осуществим по формуле

$$\alpha = k - \frac{\log_k N_\theta^* - k_0 k_1}{\sum_{j=2}^{n-1} k_j}, \qquad (12)$$

где k_{i_j} – значности переменных в порядке их вхождения в формулу, N_θ^* – количество порождаемых функций, найденное путем решения (9).

На рис. 8 показана зависимость порождающей способности α от количества переменных n.

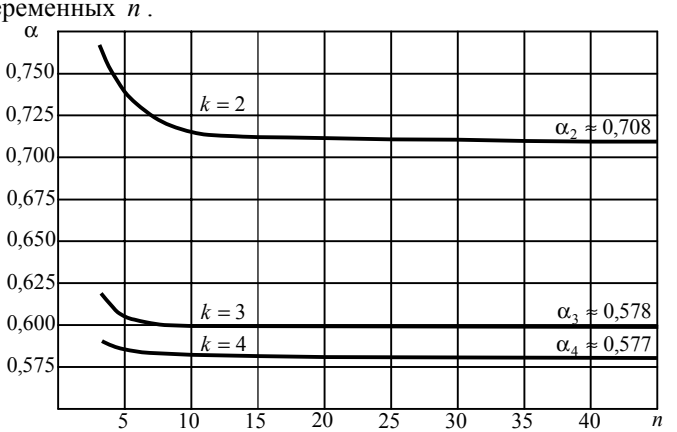


Рис. 8. Порождающая способность бесповторных формул

С ростом n параметр α асимптотически стремиться к некоторой величине α_k, которую назовем *предельной порождающей способностью*.

Из (9) с учетом (12) находим

$$\alpha_k = k - \frac{1}{k} \log_k \prod_{i=0}^{k-1} \frac{k^k - i}{i+1}. \qquad (13)$$

Зависимость α_k от k показана на рис. 9.

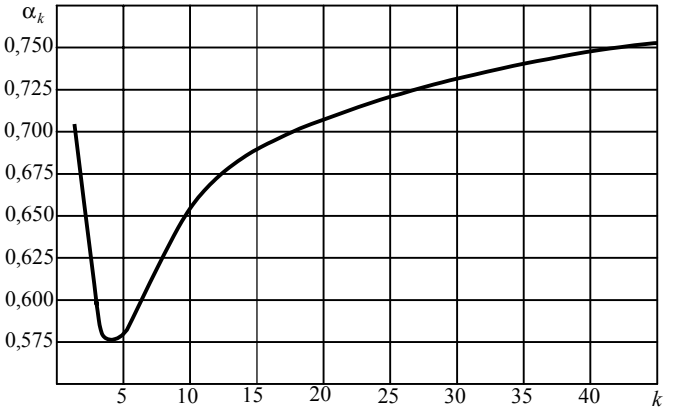


Рис. 9. Предельная порождающая способность

Скобочная конструкция. Если конструкция (6) является скобочной, то для подсчета количества функций на каждом уровне вложенности скобок применим представление:

$$\theta(X) = \theta_0(X_0) \circ_0 \theta_1(X_1) \circ_1 \ldots \circ_{t-2} \theta_{s-1}(X_{s-1}),$$

где θ_i – некоторые подформулы в скобках, зависящие от переменных из подмножеств X_i со значностями K_i (рис. 10).

$$\theta_0(X_0) \quad \overset{\theta_1(X_1)}{\begin{bmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & * \end{bmatrix}} \quad \underset{\circ_0}{\rightarrow} \quad \overset{\theta_2(X_2)}{\begin{bmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & * \end{bmatrix}} \quad \underset{\circ_1}{\cdots} \quad \overset{\theta_{s-1}(X_{s-1})}{\begin{bmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & * \end{bmatrix}} \quad \underset{\circ_{s-2}}{\rightarrow} \theta(X)$$

Рис. 10. Скобочная аналитическая конструкция

Если $N_\theta(k, K_i)$ – количество функций, порождаемых подформулой θ_i, а $N_o^{\leftrightarrow}(k, k_0, k_1)$ – количество операций при их более чем существенной зависимости от операндов, т.е. когда матрицы операций не содержат одинаковых строк и одинаковых столбцов, то получаем следующую систему рекуррентных уравнений,

$$\begin{cases} N_\theta(k_f, K_0, \ldots, K_t) = \sum\limits_{i=1}^{k} \dfrac{C_{k_f}^i}{i!i!} N_\theta(i, K_0, \ldots, K_{t-1}) N_o^{\leftrightarrow}(i,i,i) N_\theta(i, K_t), \\ N_\theta^*(k, K_0, \ldots, K_{s-1}) = \sum\limits_{j=1}^{k} C_k^j N_\theta(j, K_0, \ldots, K_{s-1}). \end{cases}$$

Так как $N_o^{\leftrightarrow}(k, k_0, k_1) < N_o^{\rightarrow}(k, k_0, k_1)$, то

$$N^*_{\theta}(k,K_0,K_1,\ldots,K_{s-1}) < N^*_{\theta}(k,k_0,k_1,\ldots,k_{n-1}) .$$

Следовательно, скобочная аналитическая конструкция является менее выразительной, так как по сравнению с бесскобочной порождает меньшее число различных функций.

Порядок переменных. Возможно, произвольная расстановка скобок или изменение порядка переменных порождает новые функции. Однако, представить формулой большее число функций, чем это определено степенями свободы аналитической конструкции, не представляется возможным:

$$N^*_{\theta}(k,k_0,k_1,\ldots,k_{n-1}) < k^{k_0k_1+k\sum_{i=2}^{n-1}k_i} .$$

Таким образом, при алгебраической декомпозиции достаточно использовать бесповторную бесскобочную аналитическую конструкцию формул с фиксированным порядком вхождения переменных.

Спектральная декомпозиция. Синтезируем формульное представление функции путем ее разложения в спектральном базисе $\{\theta_i\}$:

$$f(X) = \sum_{i=0}^{M-1}\theta_i(X)\times a_i . \qquad (14)$$

Функции, порождаемые бесповторной аналитической конструкцией, образуют множество линейно-независимых классов. Внутри класса функции преобразуются друг в друга путем умножения на ненулевые константы. Следовательно, в один класс включаются функции, которые могут быть преобразованы друг в друга умножением элементов матрицы последней операции на ненулевую константу. В фундаментальной алгебре таких констант $k-1$, в аддитивной алгебре – одна или две[5].

Пусть в разложении (14) при $M < m$, где m – длина вектора значений f, различные комбинации спектральных функций θ_i, взятые из различных классов, порождают различные функции[6]. Тогда для порождения всех функций необходимо, чтобы

$$k^m = \sum_{i=1}^{M} C^i_{N_c}(k-1)^i , \qquad (15)$$

где $N_c = N_{\theta}/(k-1)$ – число классов функций, порождаемых конструкцией (6). Из (15) получаем

---

[5] Полное использование комбинаторных возможностей бесповторной аналитической конструкции возможно только в аддитивной и фундаментальной алгебрах. В других алгебрах конструкция является избыточной и не влияет на эффективность синтеза [40, 42].

[6] Допущение о том, что различные линейные комбинации частичных функций порождают различные функции основано на следующих соображениях. Известно [Л4], что количество невырожденных матриц размерности $m\times m$ значности k равно $k^{m(m-1)/2}\Pi^m_{i=1}(k^i-1)$. Отсюда следует, что доля невырожденных матриц с различающимися столбцами от общего их числа всегда больше некоторого параметра χ, который легко вычисляется.

$$2^{-M}(k-1)^M N_c^M < k^m < (k-1)^M N_c^M ,$$

а после логарифмирования с учетом (10) имеем оценку M_a и M_f при спектральной декомпозиции в аддитивной и фундаментальной алгебре:

$$M_a = \frac{1}{\log_k 2}\frac{1}{k-\alpha}\frac{\prod_{i=0}^{n-1} k_i}{\sum_{i=0}^{n-1} k_i - \beta}, \qquad M_f = \frac{1}{k-\alpha}\frac{\prod_{i=0}^{n-1} k_i}{\sum_{i=0}^{n-1} k_i - \beta}, \tag{16}$$

где β – поправочный коэффициент для учета начальных условий,

$$\beta = k_0 + k_1 - \frac{k_0 k_1 + \log_k \chi}{k-\alpha} .$$

Тогда для максимального количества операций L, необходимых для представления произвольной функции от n переменных со значностями k_0, k_1, …, k_{n-1}, из (16) следует

$$L(k, k_0, k_1, \ldots, k_{n-1}) = \frac{n}{k-\alpha}\frac{\prod_{i=0}^{n-1} k_i}{\sum_{i=0}^{n-1} k_i - \beta} = \frac{C}{2} m , \tag{17}$$

где C – константа, характеризующая аналитическую конструкцию,

$$C = \frac{1}{k-\alpha}\frac{2}{\frac{1}{n}\left(\sum_{i=0}^{n-1} k_i - \beta\right)} . \tag{18}$$

Из (15) получаем точное и оценочное значение доли функций δ, имеющих меньшую, чем L сложность,

$$\delta(m) = \frac{\sum_{i=1}^{M(1-\varepsilon)} C_{N_c}^i (k-1)^i}{k^m} < k^{-\varepsilon m} , \tag{19}$$

где ε – доля слагаемых, на которую уменьшается сложность разложения.

Из (17) и (19) следует, что при спектральной декомпозиции количество ненулевых коэффициентов M и число операций L в синтезируемой формуле удовлетворяют асимптотическим оценкам:

$$M \sim \frac{C_\infty}{2}\frac{m}{n}, \quad L \sim \frac{C_\infty}{2} m , \quad C_\infty = \frac{1}{k-\alpha_k}\frac{2}{\bar{k}}, \tag{20}$$

где ~ – знак асимптотического равенства, α_k – предельная порождающая способность аналитической конструкции, $\bar{k}$ – средняя значность переменных. Более того, для любого $\varepsilon > 0$ доля функций δ, для которых

$$M < (1-\varepsilon)\frac{C_\infty}{2}\frac{m}{n}, \quad L < (1-\varepsilon)\frac{C_\infty}{2}m$$

меньше чем $k^{-\varepsilon m}$ и стремится к нулю с ростом m.

Алгебраическая декомпозиция. Найдем сложность алгебраического разложения (4) при произвольном разделении переменных. Для этого в качестве первого приближения системы частичных функций $\{\theta_i\}$ выберем столбцы матрицы функции $\mathbf{T}$ размерности $k' \times k''$ (рис. 11).

$$\left[\begin{array}{cccc:cc} \theta'_0 & \theta'_1 & \theta'_2 & \theta'_3 & \phi & \phi \\ \theta'_0 & \theta'_1 & \theta'_2 & \theta'_3 & \phi & \phi \\ \theta'_0 & \theta'_1 & \theta'_2 & \theta'_3 & \phi & \phi \\ \theta'_0 & \theta'_1 & \theta'_2 & \theta'_3 & \phi & \phi \\ \theta'_0 & \theta'_1 & \theta'_2 & \theta'_3 & \phi & \phi \\ \theta'_0 & \theta'_1 & \theta'_2 & \theta'_3 & \phi & \phi \end{array}\right] \times \left[\begin{array}{cccc} \tau & \phi & \phi & \phi \\ \phi & \tau & \phi & \phi \\ \phi & \phi & \tau & \phi \\ \phi & \phi & \phi & \tau \\ \hdashline \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \end{array}\right] = \left[\begin{array}{cccc} \theta'_0 & \theta'_1 & \theta'_2 & \theta'_3 \\ \theta'_0 & \theta'_1 & \theta'_2 & \theta'_3 \\ \theta'_0 & \theta'_1 & \theta'_2 & \theta'_3 \\ \theta'_0 & \theta'_1 & \theta'_2 & \theta'_3 \\ \theta'_0 & \theta'_1 & \theta'_2 & \theta'_3 \\ \theta'_0 & \theta'_1 & \theta'_2 & \theta'_3 \end{array}\right]$$

Рис. 11. Алгебраический синтез: θ'_i – элементы линейно-независимых столбцов функции; ϕ – нулеобразующий элемент; τ – единичный элемент

Для формирования $\{\theta_i\}$ возможно использование различных линейных комбинаций столбцов $\mathbf{T}$. Количество таких комбинаций равно $k^{k''} - 1$, что при $k'' \approx k'$ позволяет сформировать практически произвольную систему функций, в том числе и представимую формулами с наименьшим числом операций. При уменьшении k'' количество слагаемых в разложении уменьшается. В пределе имеем спектральное разложение. Однако, в общем случае, разлагаемая функция не имеет бесповторного представления. Следовательно, существует некоторое минимальное k'', при котором система $\{\theta_i\}$ все еще представима бесповторными формулами.

Так как при достаточно большом $k'' \le k'$ частичные функции могут быть фактически произвольными, представим их как спектральное разложение некоторой неизвестной функции. Тогда из (16) следует условие, при котором гарантируется представимость системы из k'' частичных функций длины k' бесповторными формулами, а сложность решающего разложения M станет равна k'':

$$\prod_{j=0}^{n''-1} k''_j = \frac{1}{k-\alpha} \frac{\prod_{i=0}^{n'-1} k'_i}{\sum_{i=0}^{n'-1} k'_i - \beta'}. \qquad (21)$$

Двухступенчатая декомпозиция. Рассмотрим алгебраическую декомпозицию, у которой частичные функции a_i, в свою очередь, подвергнуты спектральному разложению,

$$f(X)=\sum_{i=0}^{M'-1}\theta'_i(X')\times\sum_{j=0}^{M''-1}\theta''_{ij}(X'')=\sum_{i=0}^{M-1}\theta_i(X',X''),$$

где $M=M'M''$, $M'=k''$, θ'_i и θ''_{ij} – бесповторные бесскобочные конструкции, θ_i – бесповторные скобочные формулы, состоящие из двух бесскобочных подформул. Из (16) и (21) получаем

$$M=\frac{1}{(k-\alpha)^2}\frac{\prod_{i=0}^{n-1}k_i}{(\sum_{i=0}^{n'-1}k'_i-\beta')(\sum_{j=0}^{n''-1}k''_j-\beta'')}. \qquad (22)$$

Найдем разделение переменных, которое обеспечивает наименьшую сложность разложения. Для этого сформулируем задачу поиска условного экстремума (22):

$$\begin{cases} s=\sum_{i=0}^{n'-1}k'_i+\sum_{j=0}^{n''-1}k''_j, \\ m=(k-\alpha)(\sum_{i=0}^{n'-1}k'_i-\beta')\prod_{j=0}^{n''-1}k''^2_j \\ (\sum_{i=0}^{n'-1}k'_i-\beta')(\sum_{j=0}^{n''-1}k''_j-\beta'')\to\max, \end{cases} \qquad (23)$$

где s и m – константы, первое и второе тождество задает зависимость переменных, а последнее выражение определяет целевую функцию. Решение (23) методом неопределенных множителей Лагранжа выявляет следующую связь между значностями переменных[7]:

$$\sum_{i=0}^{n'-1}k'_i-\sum_{j=0}^{n''-1}k''_j=\mu(2n''-1),$$

где μ – некоторая константа, не зависящая от разделения переменных. Полагая $\mu(2n''-1)=\rho n$, получаем

$$M=\frac{4}{(k-\alpha)^2}\frac{\prod_{i=0}^{n-1}k_i}{(\sum_{i=0}^{n-1}k_i-\beta'-\beta'')^2-\rho^2n^2}, \quad \rho=\frac{1}{n}(\sum_{i=0}^{n'-1}k'_i-\sum_{j=0}^{n''-1}k''_j),$$

где ρ – параметр, характеризующий разделение переменных, для вычисления оптимального значения которого (рис. 12) воспользуемся условием (21), заданным для переменных со средней арифметической и средней геометрической значностью $\bar{k}$ и $\tilde{k}$,

[7] Решение (23) получено при условии $\sum k'_i >> \beta'$.

$$\rho n = \bar{k}\log_{\tilde{k}}\left(\frac{1}{2}(k-\alpha)(\bar{k}n - 2\beta + \rho n)\right).$$

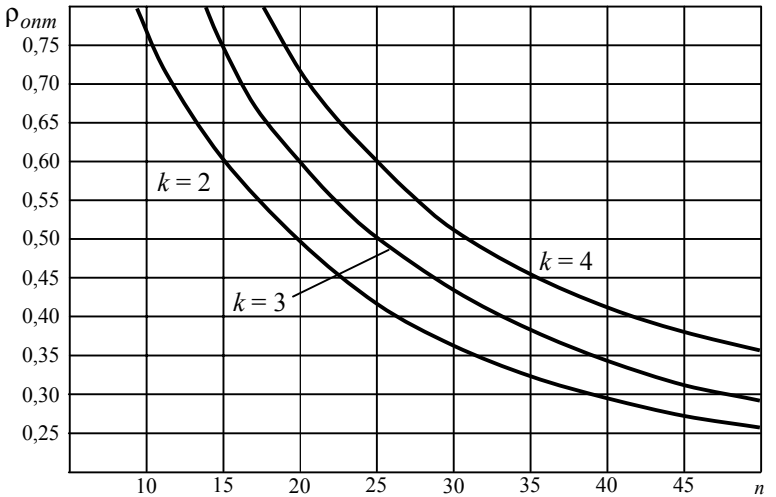


Рис. 12. Оптимальное разделение переменных

Многоступенчатая декомпозиция. Обобщим полученные результаты на случай относительно произвольной расстановки скобок. В этом случае для максимального количества операций имеем:

$$L = C^2\frac{m}{n}, \qquad C = \frac{1}{k-\alpha}\frac{2}{\sqrt{\left(\frac{1}{n}\left(\sum_{i=0}^{n-1}k_i - \sum_{j=0}^{\gamma-1}\beta_j\right)\right)^2 - \rho^2}}, \qquad (24)$$

где C – константа, характеризующая аналитическую конструкцию, α – порождающая способность, γ – количество скобок или ступеней декомпозиции, $2 \le \gamma < \log_2 n - 1$, β_j – поправки начальной порождающей способности подформул, ρ – параметр разделения переменных.

С увеличением глубины вложенности скобок оценка (24) ухудшается, а наименьшая сложность достигается при $\gamma = 2$. Из (24) находим:

$$M \sim C_\infty^2\frac{m}{n^2}, \quad L \sim C_\infty^2\frac{m}{n}, \quad C_\infty = \frac{1}{k-\alpha_k}\frac{2}{\bar{k}}, \qquad (25)$$

причем для любого $\varepsilon > 0$ доля функций δ, для которых[8]

[8] Сложность алгебраического разложения может быть уменьшена только за счет спектрального разложения коэффициентов. Полагая $k^* < \sqrt{m}$, из (19) получаем искомую долю функций.

$$M < (1-\varepsilon)C_\infty^2 \frac{m}{n^2}, \quad L < (1-\varepsilon)C_\infty^2 \frac{m}{n}$$

стремится к нулю с ростом m как $k^{-\varepsilon\sqrt{m}}$. Зависимость C_∞^2 от значности образующей алгебры k показана на рис. 13.

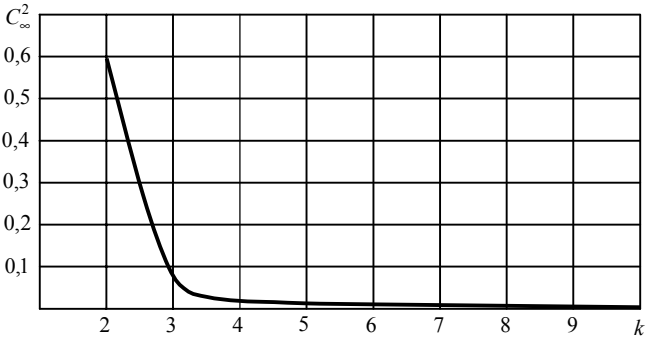


Рис. 13. Зависимость C_∞ от значности алгебры k

В частности, из (25) следует асимптотическая оценка количества операций, необходимых для представления произвольной булевой функции,

$$L(n) \sim \frac{4}{(1+\log_2 3)^2}\frac{2^n}{n} \approx 0{,}5986\frac{2^n}{n},$$

которая согласуется с оценкой количества команд (операторов и условных переходов), полученной при программной реализации логической обработки данных [Л5]. Заметим, что программная реализация имеет ту же природу, что и аппаратурная [Л1].

Эффективность модели. Установлено, что почти все дискретные функции реализуются со сложностью

$$L = \frac{4}{n(k-\alpha)^2}\frac{\prod_{i=0}^{n-1} k_i}{\frac{1}{n^2}(\sum_{i=0}^{n-1} k_i - 2\beta)^2 - \rho^2}, \qquad (26)$$

где k – значность образующей алгебры ($k \geq k_f$), α – порождающая способность аналитической конструкции, β – начальная порождающая способность, ρ – параметр разделения переменных.

Если функция принадлежит множеству функций, требующих для своей реализации меньшее число операций, то возможна ее минимизация. При этом целесообразно использовать и скобочные аналитические конструкций с произвольным порядком вхождения переменных. Если же функция принадлежит множеству функций с максимальной сложностью (а таких функций большинство), то следует говорить о ее эффективном синтезе, для реа-

лизации которого достаточно бесповторных бесскобочных формул с фиксированным порядком вхождения переменных. Последнее достижимо при алгебраической декомпозиции булевых функций (возможно, с не булевыми переменными) в аддитивной алгебре, осуществляемой по системам мультимодальных двузначных функций; а многозначных функций – только в фундаментальной алгебре по системам мультимодальных многозначных функций.

Формула (26) позволяет по объему обрабатываемых данных и количеству операций L' оценить эффективность произвольной дискретной обработки данных, а в случае, если количество операций L' будет меньше L, определить и степень минимизации математической модели, равную отношению L к L'.

Алгебраический синтез. Алгебраический синтез осуществим при алгебраической декомпозиции функции путем оптимального разделения переменных X на два непересекающихся множества X' и X'' таких, что

$$\frac{1}{n}\left(\sum_{i=0}^{n'-1} k_i' - \sum_{j=0}^{n''-1} k_j''\right) \approx \rho_{onm},$$

а искомую формулу будем искать в виде

$$f(X) = \sum_{i=0}^{M-1} g_i(X') \times h_i(X''),$$

где g_i и h_i – частичные функции, задаваемые бесповторными бесскобочными формулами, а M таково, что обеспечивается выполнение условия (26).

При алгебраическом синтезе не требуется проверять все варианты разделения переменных: минимизация числа ненулевых коэффициентов обеспечивается оптимальным их разделением. В отличие от алгебраического подхода, основанного на получении формулы функции и тождественных ее преобразованиях, при алгебраической декомпозиции возможен прямой синтез эффективного представления, а фундаментальность синтезируемых частичных функций позволяет применить хорошо разработанный аппарат линейной алгебры.

Однако трудоемкость алгоритмов алгебраического синтеза, как это имеет место и при разложении по Карунену-Лоэву, достаточно высока. Это объясняется отсутствием ограничений на степени свободы декомпозируемых функций и является следствием абстрагирования общей теории декомпозиции от содержательных представлений о решаемой задаче.

В **Приложениях** приведен пример решения средствами контекстной технологии задачи управления лифтом, описана реализация языка исчисления предикатов первого порядка, определена виртуальная машина, использованная для задания первичных семантических категорий исчисления предикатов, дана методика синтеза полиномиальных и неполиномиальных

форм частичных функций при алгебраической декомпозиции, рассмотрен алгоритм полиномиальной факторизации частичных функций и его применение для сжатия изображений.

Полученные результаты

1) Предложена методология понятийного анализа, позволяющая получать эффективные декомпозиционные схемы предметной области в виде синтаксически и семантически замкнутых формальных спецификаций. Для этого:

– формализованы четыре основные абстракции понятий и разработано исчисление понятий, предназначенное онтологического описания предметных областей с помощью четырех видов связей между понятиями;

– разработан протоязык для спецификации результатов понятийного анализа, предусматривающий выразительные средства для определения понятийной структуры, синтаксиса выражения понятий и многоаспектного описания семантики;

– найден метод описания семантики формальных языков на основе математической индукции, осуществляемый путем определения семантических категорий по мере необходимости, в процессе описания и описанными ранее средствами.

2) Создана контекстная технология обработки данных на основе отражения понятийной структуры предметной области в понятиях создаваемого проблемного языка, а получаемые при понятийном анализе декомпозиционные схемы – в его языковых конструкциях. Для реализации системы контекстного программирования:

– доказано, что выразительные возможности языковых средств контекстной технологии обработки данных эквивалентны формализму контекстных грамматик;

– разработан метод разнесенного грамматического разбора, позволяющий выполнять эффективный анализ текста, порожденного, в том числе и неоднозначными и контекстными грамматиками;

– предложен метод определения прагматик понятий в виде многоаспектного описания их семантики, позволяющий, в том числе, для такого описания использовать различные интерпретаторы: от процессоров целевых вычислительных платформ до текстов на целевых языках программирования.

3) Обобщена теория алгебраической декомпозиции дискретных функций на основе объединения алгебраических методов, разделительной декомпозиции и ортогональных разложений в широком спектре алгебраических систем. В частности:

– показано существование четырех типов образующих алгебр (алгебры логики, мультипликативной, аддитивной и фундаментальной алгебр),

позволяющих решать системы линейных алгебраических уравнений, изучены их свойства и доказаны условия существования решений;

– найдены четыре класса частичных функций (унимодальные и мультимодальные, двузначные и многозначные), имеющие эффективную реализацию на вычислительных средствах с различными операционными возможностями;

– предложен метод многошагового алгебраического синтеза формул, позволяющий учесть специфические свойства декомпозируемой функции и, на основе этого, получить ее эффективное представление.

4) Разработан метод синтеза эффективных моделей дискретной обработки данных при двухступенчатой алгебраической декомпозиции дискретных функций, базирующаяся на следующих результатах:

– выполнен количественный и качественный анализ порождающей способности аналитических конструкций формул и установлено, что для эффективного синтеза функций достаточно использовать наиболее простую аналитическую конструкцию бесповторных бесскобочных формул с фиксированным порядком вхождения переменных;

– доказано, что при алгебраическом синтезе не требуется проверять все варианты разделения переменных, минимизация сложности представления обеспечивается оптимальным их разделением, гарантирующим бесповторное бесскобочное представление частичных функций;

– получены точные, приближенные и асимптотические оценки сложности синтезируемых формул и показано, что количество операций, необходимое для представления дискретных функций, не хуже известных наилучших оценок;

– найдена методика оценки эффективности и степени минимизации математических моделей дискретной обработки данных при конечной размерности решаемых задач, что позволяет сравнивать по эффективности различные модели.

Список публикаций по теме диссертации

1. *Выхованец В. С.* Многократные параллельные логические вычисления // Информационно-управляющие системы и специализированные вычислительные устройства для обработки и передачи данных: Тез. докл. всероссийской конф. Махачкала, 1996. С. 105-106.

2. *Выхованец В. С.* Вычислительные средства и комплексы // Учебно-методическое пособие. Тирасполь, 1996. Ч. 2. 90 с.

3. *Выхованец В. С.* Многократные параллельные логические вычисления // Вестник Приднестровского университета. Тирасполь, 1997. № 2. С. 64-74.

4. *Выхованец В. С.* Дискретное преобразование Фурье и его применение при логических вычислениях / Приднестровский гос.-корпорат. унив. им. Т.Г. Шевченко. Тирасполь, 1997. 18 с. Деп. в ВИНИТИ 05.06.97, № 1852-В97.

5. *Выхованец В. С., Выхованец О. Д.* Параллельная парадигма в программировании и ее использование в языках высокого уровня // Вестник Приднестровского университета. Тирасполь, 1997. № 2. С. 74-79.

6. *Выхованец В. С.* Кратные логические вычисления и их применение в управлении, обработке информации и других областях / Приднестровский гос.-корпорат. унив. им. Т.Г. Шевченко. Тирасполь, 1997. 18 с. Деп. в ВИНИТИ 05.06.97, № 1851-В97.

7. *Выхованец В. С., Малюгин В. Д.* Спектральные методы в логическом управлении // Тр. 2-й Межд. науч.-техн. конф. «Современные методы цифровой обработки сигналов в системах измерения, контроля, диагностики и управления». Минск, 1998. С. 56-59.

8. *Выхованец В. С.* Кратные логические вычисления и их применение при моделировании дискретных объектов / Автореферат дис. на соиск. уч. ст. канд. техн. наук. М., 1998. 23 с.

9. *Выхованец В. С., Малюгин В. Д.* Кратные логические вычисления // Автоматика и телемеханика. 1998. № 6. С. 163-171.

10. *Выхованец В. С.* Булевы мультипликативные формы // Тез. докл. Межд. науч.-практ. конф. «Математические методы в образовании, науке и промышленности». Тирасполь, 1999. С. 52-53.

11. *Выхованец В. С.* Контекстная технология программирования // Тр. 4-ой Межд. науч.-техн. конф. по телекоммуникациям (Телеком-99). Одесса, 1999. С. 116-119.

12. *Vykhovanets V. S.* The generalized multiplicative forms // Тез. докл. Межд. конф. по проблемам управления. М., 1999. Т. 3. С. 319-321.

13. *Выхованец В. С.* Обработка сигналов в дискретных базисах на основе обобщенных полиномиальных форм //Докл. 2-й Межд. конф. «Цифровая обработка сигналов и ее применение». М., 1999. Т. 2. С. 372-377.

14. *Vykhovanets V. S.* Digital signal processing on basis of generalized polynomial forms // Proc. 2nd Int. Conf. «Digital signal processing and its applications». Moscow, 1999. Vol. 2. PP. 378-379.

15. *Выхованец В. С.* Обобщенные полиномиальные формы // Радиоэлектроника. Информатика. Управление. 1999. № 2. С. 55-59.

16. *Выхованец В. С.* Параллельные вычисления во времени // Автоматика и телемеханика. 1999. № 12. С. 155-165.

17. *Выхованец В. С.* Структурная организация кратных вычислений // Матер. межвузовской науч.-техн. конф. «Управляющие и вычислительные системы. Новые технологии». Вологда, 2000. С. 139-140.

18. *Выхованец В. С.* Асимптотические оценки при идентификации дискретных объектов // Матер. Межд. конф. «Идентификация систем и задачи управления» М., 2000. Электрон. опт. диск. ISBN 5-201-09605-0.

19. *Выхованец В. С.* Асимптотические оценки в многозначной логике // Матер. юбилейной конф. профессорско-преподавательского состава, посвященной 70-летию Приднестровского государственного университета им. Т.Г. Шевченко. Тирасполь, 2000. С. 264-270.

20. *Выхованец В. С.* Технология безопасного программирования // Тез. докл. 8-й Межд. конф. «Проблемы управления безопасностью сложных систем». М., 2000. Т. 2. С. 89-91.

21. *Выхованец В. С.* Кратные вычисления в современных высокопроизводительных системах обработки данных // Вестник Приднестровского университета. 2000. № 1-2 (12). С. 99-110.

22. *Иосенкин В. Я., Выхованец В. С.* Технология контекстного программирования в экономике и бизнесе // Матер. межвузовской науч.-техн. конф. «Управляющие и вычислительные системы. Новые технологии». Вологда, 2001. С. 180-181.

23. *Иосенкин В. Я., Выхованец В. С.* Контекстное программирование в моделировании // Матер. межд. науч.-практ. конф. «Моделирование. Теория, методы и средства». Новочеркасск, 2001. Ч. 7. С. 49-51.

24. *Выхованец В. С.* О вычислимости конечных полей // Матер. межд. науч.-прак. конф. «Математическое моделирование в образовании, науке и производстве». Тирасполь, 2001. С. 475-477.

25. *Иосенкин В. Я., Выхованец В. С.* Формализация семантики искусственных языков // Матер. межд. науч.-практ. конф. «Математическое моделирование в образовании, науке и производстве». Тирасполь, 2001. С. 477-479.

26. *Иосенкин В. Я., Выхованец В. С.* Технология контекстного программирования в телекоммуникационных системах // Тр. 2-ой Межд. науч.-практ. конф. «Современные информационные и электронные технологии». Одесса: Друк, 2001. С. 118-119.

27. *Vykhovanets V. S.* Algebraic Systems for Digital Signal Processing // Proc. 6th Int. Conf. “Pattern Recognition and Information Processing”. Mink, 2001. Vol. 2. PP. 93-99.

28. *Иосенкин В. Я., Выхованец В. С.* Применение технологии контекстного программирования для решения больших прикладных задач // Тр. Межд. конф. «Параллельные вычисления и задачи управления». М., 2001. С. (4)-121-139. Электрон. опт. диск. ISBN 5-201-09559-3.

29. *Vykhovanets V. S.* Fundamental Theorems for Polynomial Representation of Discrete Functions // Proc. 4th Int. Conf. “Computer-Aided Design of Discrete Devices”. Mink, 2001. Vol. 1. PP. 69-76.

30. *Выхованец В. С.* Спектральные методы в логической обработке данных // Автоматика и телемеханика. 2001. № 10. С. 28-53.

31. *Выхованец В. С.* Аддитивная алгебра в цифровой обработке сигналов // Докл. 4-й Межд. конф. и выставки «Цифровая обработка сигналов и ее применение». М., 2002. Т. 2. С. 255-258.

32. *Vykhovanets V. S.* Additive Algebra for Digital Signal Processing // Proc. 4th Int. Conf. and Exhib. on Digital Signal Processing and its Applications. Moscow, 2002. Vol. 2. PP. 258-260.

33. *Иосенкин В.Я., Выхованец В. С.* Контекстная модель технологического процесса предприятия // Тр. 2-ой Межд. конф. «Идентификация сис-

тем и задачи управления». М., 2003. С. 859-871. Электрон. опт. диск. ISBN 5-201-14948-0.

34. *Выхованец В. С.* Хааро-подобные системы сигналов // Докл. 5-й Межд. конф. и выставки «Цифровая обработка сигналов и ее применение». М., 2003. Т. 2. С. 279-283.

35. *Vykhovanets V. S.* Haar-Like Systems of Signals // Proc. 5th Int. Conf. and Exhib. on Digital Signal Processing and its Applications. Moscow, 2003. Vol. 2. PP. 283-285.

36. *Выхованец В. С.* Факторизация полиномиальных форм // Тез. докл. 2-ой Межд. конф. по проблемам управления. М., 2003. Т. 2. С. 108.

37. *Выхованец В. С., Иосенкин В. Я.* Высокоуровневая форма синтеза систем управления // Тез. докл. 2-ой Межд. конф. по проблемам управления. М., 2003. Т. 2. С. 109.

38. *Выхованец В. С., Малюгин В. Д.* Мультипликативные формы и их применение при логических вычислениях // Тез. докл. 2-ой Межд. конф. по проблемам управления. М., 2003. Т. 2. С. 110-111.

39. *Выхованец В. С., Иосенкин В. Я.* Компиляция знаний, представленных на языке ESSE // Тез. докл. 2-ой Межд. конф. по проблемам управления. М., 2003. Т. 2. С. 165.

40. *Выхованец В. С., Малюгин В. Д.* Аппаратная и программная реализация мультипликативных форм // Теория и практика логического управления: Тез. докл. Межд. конф., посвященной 100-летию со дня рождения чл.-кор. АН СССР М. А. Гаврилова. М., 2003. С. 79-81.

41. *Выхованец В. С.* Алгебраическая декомпозиция дискретных функций в аддитивной алгебре // Теория и практика логического управления: Тез. докл. Межд. конф., посвященной 100-летию со дня рождения чл.-кор. АН СССР М. А. Гаврилова. М, 2003. С. 81-84.

42. *Выхованец В. С., Малюгин В. Д.* Мультипликативная алгебра и ее применение в логической обработке данных // Проблемы управления. 2004. № 3. С. 67-77.

43. *Выхованец В. С.* Полиномиальная факторизация спектральных базисов // Тр. 5-й Межд. конф. «Автоматизация проектирования дискретных устройств». Минск, 2004. Т. 2. С. 71-79.

44. *Выхованец В. С.* Полиномиальная факторизация спектральных базисов // Автоматика и телемеханика. 2004. № 12. С. 3-14.

45. *Vykhovanets V. S.* Additive algebra for signal and image processing // Proc. of SPIE. 2005. Vol. 5822. PP. 94-97.

46. *Выхованец В. С., Иосенкин В. Я.* Понятийный анализ и контекстная технология программирования // Проблемы управления. 2005. № 4. С. 2-11.

47. *Выхованец В. С.* Разнесенный грамматический разбор // Проблемы управления. 2006. № 3. С. 32-43.

48. *Выхованец В. С.* Алгебраическая декомпозиция дискретных функций // Автоматика и телемеханика. 2006. № 3. С. 20-56.

49. *Выхованец В. С.* Оптимальный синтез логического управления //

Тез. докл. 3-й Межд. конф. по проблемам управления. М., 2006. Т. 2. С. 106.

50. *Выхованец В. С.* Верификация результатов понятийного анализа //Тез. докл. 3-й Межд. конф. по пробл. управления. М., 2006. Т. 1. С. 105.

51. *Выхованец В. С.* Репрезентация знаний в системах управления крупномасштабными производствами // Тез. докл. Межд. конф. «Управление развитием крупномасштабных систем». М., 2007. С. 124-125.

52. *Выхованец В. С.* Что истино во всех мирах // Матер. межд. науч.-практ. конф. «Математическое моделирование в образовании, науке и производстве». Тирасполь, 2007. С. 36.

53. *Выхованец В. С.* Репрезентация знаний // Матер. Межд. конф. «Системы проектирования, технологической подготовки производства и управления этапами жизненного цикла промышленного продукта». М., 2007. С. 49-53.

54. *Выхованец В. С.* Исчисление понятий // Тез. докл. межд. конф. «Когнитивный анализ и управление развитием ситуаций». М., 2007. С. 87-88.

В журналах из перечня ВАК опубликовано 8 работ: [9], [16], [30], [42], [44], [46], [47], [48].

Личный вклад автора в совместные публикации

[5] – зависимость эффективности программ от парадигмы программирования, [7, 9] – кратные вычисления в форме совместного описания систем функций на множестве наборов, [22, 23, 25, 26, 28, 33, 37, 39, 46] – методология понятийного анализа и контекстная технология обработки данных, [38, 40] – алгебраическая декомпозиция.

Диссертации, выполненные под научным руководством автора

Д1. Иосенкин В.Я. Технология контекстного программирования и ее применение. М.: Институт проблем управления, 2004.

Цитированная литература

Л1. *Кузнецов О. П.* О программной реализации логических функций и автоматов // Автоматика и телемеханика. 1977. № 7. С. 163-174. № 9. С. 137-149.

Л2. *Лидл Л., Нидеррайтер Г.* Конечные поля: Т. 1. М.: Мир, 1988.

Л3. *Ахмед Н., Рао К. Р.* Ортогональные преобразования при обработке цифровых сигналов. М.: Связь, 1980.

Л4. *Артин Э.* Геометрическая алгебра. М.: Наука, 1969.

Л5. *Кузьмин В.А.* Оценка сложности реализации функций алгебры логики простейшими видами бинарных программ // Методы дискретного анализа в теории кодов и схем. Новосибирск, 1976. Вып. 29. С. 24-36.

Л6. *Поспелов Д. А.* Ситуационное управление: теория и практика. М.: Наука, 1986.

Л7. *Brodie M. L., Mylopoulos J., Schmidt J. W.* On Conceptual Modeling. New York: Springer-Verlag, 1984.

Подписано в печать 22.01.2007 г.
Формат 60×90 1/16. Объем 3,0 п.л. Тираж 150 экз. Заказ № 1403071

Оттиражировано в ООО «Полиграф-Сервис»
Св. о регистрации № 1057748169159 от 12 сентября 2005 г.
ИНН 7725548326