На правах рукописи

Кручинин Владимир Викторович

МЕТОДЫ, АЛГОРИТМЫ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМБИНАТОРНОЙ ГЕНЕРАЦИИ

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

АВТОРЕФЕРАТ

диссертации на соискание ученой степени

доктора технических наук

Томск – 2010

Работа выполнена в *Томском государственном университете систем управления и радиоэлектроники.*

Научный консультант: *д.т.н., профессор,*
Шелупанов Александр Александрович.

Официальные оппоненты: *д.т.н., профессор,*
Костюк Юрий Леонидович
д.т.н.,профессор,
Рябко Борис Яковлевич.
д.т.н. профессор,
Марков Николай Григорьевич.

Ведущая организация: *Московский государственный университет приборостроения и информатики*

Защита состоится 24 ноября 2010 г. в 15 часов на заседании совета по защите докторских и кандидатских диссертаций *Д 212.269.06* при *Национальном исследовательском Томском политехническом университете*, расположенном по адресу: *Россия, 634050, г. Томск, проспект Ленина 30.*

С диссертацией можно ознакомиться в библиотеке *Национального исследовательского Томского политехнического университета.*

Автореферат разослан «____»_____ 2010 г.

Ученый секретарь
совета, *к.т.н.* *Сонькин Михаил Аркадьевич*

Общая характеристика работы

Актуальность. Комбинаторная генерация – это научное направление, находящееся на стыке комбинаторики, информатики и программирования. Объектом исследования являются алгоритмы генерации и нумерации элементов комбинаторных множеств. Комбинаторное множество – это конечное множество, элементы которого имеют некоторую структуру и имеется процедура построения элементов этого множества. Примерами таких множеств являются перестановки, сочетания, размещения, композиции, разложения, разбиения, графы и деревья, выражения языков, заданных контекстно-свободными грамматиками и т.д. Исторический обзор развития этого направления дает Д. Кнут в 4 томе «Искусство программирования». Однако, классификация алгоритмов генерации и определения понятий здесь не приводятся. Такую классификацию приводит Ф. Раски в работе «Комбинаторная генерация». Он выделяет 4 класса алгоритмов генерации: последовательная генерация всех элементов комбинаторного множества; нумерация всех элементов комбинаторного множества; генерация элементов комбинаторного множества в соответствии с процессом нумерации; случайная генерация элемента комбинаторного множества (random selection). Алгоритмы последовательной генерации комбинаторных множеств наиболее изучены. Обобщенная схема их работы следующая: установка начального элемента комбинаторного множества; организация цикла, в котором производится вывод следующего элемента комбинаторного множества или его конструирование из данного. Этот процесс можно реализовать двумя способами. Первый способ предполагает реализацию процедуры генерации как контейнера, т.е. для всех элементов данного комбинаторного множества вызывается некоторая заданная процедура. Обычно такой механизм реализуется в виде процедуры под названием ForEach. Второй способ подразумевает реализацию процедур генерации First и Next. Процедура First обеспечивает установку начального элемента. Процедура Next генерирует следующий элемент. Нумерация рассматривается как процесс упорядочения множества комбинаторных объектов. Если некоторое комбинаторное множество упорядочено, то номер позиции некоторого элемента в этом упорядочении будет являться порядковым номером (rank). Соответствующий алгоритм или процедуру обычно называют Rank. Процесс генерации комбинаторного объекта (unranking) по его номеру (рангу) является обратным процессу нумерации. Соответствующий алгоритм или процедуру называют Generate. В некоторых случаях желательно наличие алгоритма генерации со случайным выбором (random selection), когда последовательность объектов не упорядочена или не требуется все комбинаторное множество. Методы и алгоритмы комбинаторной генерации приобретают важное научное и практическое значение в следующих отраслях: в программировании, в про-

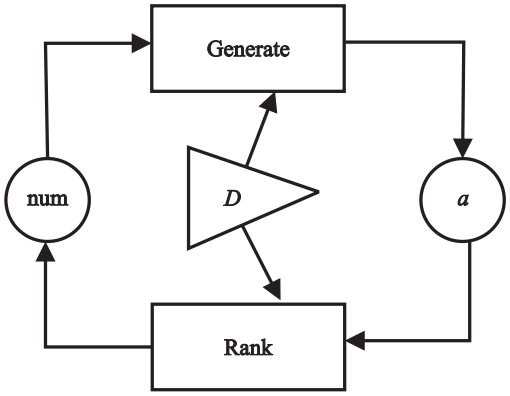


Рис. 1. Основная схема алгоритмов генерации и нумерации элементов комбинаторных множеств

ектировании информационных систем и баз данных, в теории кодирования и сжатия информации, в химии, при исследовании различных химических структур, в биологии при моделировании ДНК-структур. В общем случае, для получения процедур генерации и нумерации комбинаторного множества необходимо задать биективное отображение

$$Generate : N_m \to A_m,$$

где N_m – конечное подмножество натуральных чисел; A_m – комбинаторное множество. Тогда обратное отображение задает процедуру нумерации:

$$Rank : A_m \to N_m.$$

На рис. 1 показаны основные элементы и схема взаимодействия алгоритмов Generate и Rank. Алгоритм Generate на основании числа $num \in N_m$ и некоторого описания D создает элемент $a \in A_m$. Алгоритм Rank производит обратное действие на основании $a \in A_m$ и некоторого описания D, формирует номер $num \in N_m$.

Методы построения алгоритмов генерации и нумерации комбинаторных множеств самые разнообразные и зависят от конкретного рассматриваемого комбинаторного объекта. Следует отметить, что сравнительно недавно появились методы, претендующие на некоторую универсальность применения. Так, Э.Ренгольд, Ю.Нивергельт, Н.Део для генерации комбинаторных объектов предложили использовать метод, основанный на поиске с возвращением (backtracking), дальнейшее развитие этого метода предложено Д.Крехером и Д.Стинсоном в работе «Combinatorial Algorithm: Generation, Enumeration and Search». Однако во многих случаях этот метод используется только для последовательной генерации всех объектов.

Другим методом, претендующим на некоторую универсальность, является метод ECO (Enumeration Combinatorial Object), предложенный Баргутччи,

Дел Лунго, Пергола и Пинзани, в основе которого лежит использование следующих рекурсивных правил:
1) комбинаторные объекты размера $n+1$ строятся из объекта n-го размера;
2) каждый объект $n+1$ размера имеет одного родителя n-го размера.
По этим правилам строится генерирующее дерево, в узлах которого записано число объектов. Данный метод хорошо зарекомендовал себя для перечисления большого числа комбинаторных объектов. Для генерации комбинаторных объектов используется идея перехода от объекта размером n к объекту размером $n+1$. Примером может служить алгоритм последовательной генерации некоторых классов полимино. Число узлов в дереве пропорционально числу комбинаторных объектов рассматриваемого класса.

Следует отметить также метод, предложенный Ф. Флажолетом, Б. Катсемем и П. Циммераманом для построения алгоритмов генерации элементов комбинаторных множеств со случайным выбором, который был развит К. Мартинец и Х. Мулинеро для построения алгоритмов последовательной генерации, нумерации и генерации по номеру. Этот метод базируется на построении соответствий между производящими функциями, описывающими мощности комбинаторных множеств и некоторыми операциями, определенными над этими множествами. Кроме известных операций над множествами $+$ и $\times$ вводятся специальные операции Seq, Set, $PSet$, $Circle$. Например, операция Seq представляет собой следующее выражение:

$$Seq(A) = \epsilon + A + (A \times A) + (A \times A \times A) + \ldots$$

Если множество A конечно для всех конечных множеств $Seq(A)$, то можно задать алгоритмы комбинаторной генерации. Основные недостатки данного метода:
1. Невозможно воспользоваться методом для построения алгоритмов комбинаторной генерации, если нет производящей функции.
2. Если производящая функция имеется, то ее надо выразить в терминах операций $\{+, \times, Seq, Set, Pset\}$, что является довольно сложной задачей.
3. Не учитывается рекурсивный характер комбинаторного множества.

Во многих алгоритмах генерации и нумерации комбинаторных множеств описание D носит эвристический характер, построение алгоритма осуществляется на основе некоторых допущений, характерных для данного комбинаторного множества или класса комбинаторных множеств и после построения алгоритма доказательство его эффективности. Поэтому актуальной является проблема получения методов построения и анализа алгоритмов комбинаторной генерации.

Цели и задачи. Целью данной диссертационной работы является разработка методологии проектирования и анализа алгоритмов генерации и нумерации комбинаторных множеств, применение ее для разработки и исследо-

вания широкого класса алгоритмов, создание универсального программного обеспечения и применения его в различных прикладных программных системах.

Объектом исследования являются модели и методы проектирования и анализа алгоритмов.

Предметом исследований являются модели и методы проектирования и анализа алгоритмов комбинаторной генерации, основанных на применении деревьев И/ИЛИ.

Основные задачи:

1. Обосновать и создать методологию проектирования и анализа алгоритмов комбинаторной генерации с применением деревьев И/ИЛИ.

2. Разработать методы комбинаторной генерации для построения алгоритмов последовательной генерации, нумерации и генерации по номеру элементов комбинаторных множеств.

3. Применить предложенные методы для классических комбинаторных множеств и получить новые алгоритмы генерации для сочетаний, перестановок, разбиений, композиций, разложений.

4. Построить новые алгоритмы комбинаторной генерации для множеств, описываемых формулами Фибоначчи, Сильвестра, Стирлинга, Каталана.

5. Создать новые алгоритмы комбинаторной генерации и нумерации деревьев и выражений языков, заданных контекстно-свободными грамматиками.

6. Разработать универсальное программное обеспечение для исследования и проектирования алгоритмов комбинаторной генерации и библиотеку шаблонов классов для полученных алгоритмов комбинаторной генерации.

7. Создать и внедрить прикладное программное обеспечение для:
– информационных систем, основанных на реляционных базах данных;
– систем идентификации и прослеживаемости изделий;
– систем построения и использования генераторов тестовых заданий;
– автоматизированных систем управления технологическими процессами и безналичными расчетами за нефтепродукты с применением магнитных, электронных и смарт-карт.

Методы исследования. Методы теории рекурсивных функций и алгоритмов, комбинаторного анализа и теории чисел, теории графов, теории синтаксического анализа. Методы и средства создания системного программного обеспечения и объектно-ориентированного программирования.

Научная новизна

1. Разработана новая методология построения алгоритмов комбинаторной генерации, основанная на применении деревьев И/ИЛИ.

2. Впервые получено:

1) Если для некоторого множества комбинаторных объектов имеется схема рекурсивной композиции построения дерева И/ИЛИ, то существует ре-

курсивная функция алгебры $\{N, +, \times, R\}$, вычисляющая мощность данного множества.

2) Если мощность комбинаторного множества задана в виде функции $f(n)$ алгебры $\{N, +, \times, R\}$, то для такого множества можно однозначно построить алгоритм последовательной генерации и алгоритмы генерации элемента по номеру и нумерации элемента.

3. Для построения алгоритмов последовательной генерации элементов комбинаторных множеств разработан новый формализм в виде автомата, представляющего четверку $\{B, N, P_{First}, P_{Next}\}$, где B- начальный магазинный символ, N- множество магазинных символов; правила P_{First} для операции First, правила P_{Next} для операции Next. Разработан алгоритм построения автомата по схеме рекурсивной композиции деревьев И/ИЛИ, получено выражение для определения временной сложности алгоритмов последовательной генерации.

4. Впервые получены рекуррентные выражения для композиций и разбиений натурального числа n с ограничениями на части, записаны оригинальные производящие функции и формулы для композиций и разбиений, построен и исследован треугольник разбиений, разработаны и исследованы алгоритмы генерации и нумерации композиций и разбиений.

5. Развит метод построения алгоритмов генерации корневых деревьев с заданным числом узлов, основанный на процедуре полного разбиения. Получены оригинальные функции и алгоритмы генерации t-арных деревьев, упорядоченных и неупорядоченных корневых деревьев, деревьев Кемпа.

6. Созданы оригинальные алгоритмы генерации и нумерации комбинаторных множеств, заданных формулой Сильвестра и выражений языков, заданных однозначными контекстно-свободными грамматиками.

Практическая значимость. Разработанная методология проектирования и анализа алгоритмов комбинаторной генерации с применением деревьев И/ИЛИ, полученные алгоритмы и программы комбинаторной генерации обеспечивают:

1. Решение задачи глобальной нумерации. Если для некоторой предметной области строится дерево И/ИЛИ, то каждый объект из данной предметной области нумеруется и по данному номеру получается однозначное описание.
2. Решение задач кодирования. Если для некоторого множества объектов строится дерево И/ИЛИ, то алгоритм Rank будет кодировать объект, а алгоритм Generate - декодировать.
3. Решение задач сжатия информации. Если для некоторого множества объектов построить дерево И/ИЛИ, то алгоритм Rank будет производить сжатие объекта, а алгоритм Generate - восстанавливать объект. Причем, если при передаче объектов по каналам связи само дерево И/ИЛИ не передавать (т.е. дерево И/ИЛИ имеется на стороне приемника и на стороне передатчика), то

можно получить уменьшение объемов передаваемой информации.
4. Решение задач тестирования. Если для множества тестовых примеров построить дерево И/ИЛИ, то можно, используя алгоритм Generate, строить тестовые выборки. Особенно это касается задач тестирования программ с некоторым входным проблемно-ориентированным языком.
5. Построение механизма распараллеливания переборных задач и перечисления элементов некоторого комбинаторного множества.
6. Решение задачи стандартизации и классификации представления алгоритмов генерации и нумерации различных классов объектов.
7. Для реляционных баз данных:
• Кодировать базу данных, разделив ее на две части. При этом записи таблицы реляционной базы данных будут представлять собой номера вариантов в дереве И/ИЛИ. Само дерево И/ИЛИ хранит домены соответствующих столбцов таблицы. Таким образом, такую базу данных можно безопасно переносить от одного источника к другому.
• Сжимать объем базы данных. При этом коэффициент сжатия будет иметь значение 2 и более. Еще больший эффект будет получен, если в компьютерной сети таблица базы данных хранится централизовано, а домены, представленные деревом И/ИЛИ, имеются у конечных потребителей.

Положения, выносимые на защиту

1. Для любой рекурсивно-примитивной функции можно взаимно-однозначно поставить в соответствие схему рекурсивной композиции дерева терма.
2. Если для некоторого комбинаторного множества построена схема рекурсивной композиции или получено фиксированное дерево И/ИЛИ, то однозначно задаются:
1) алгоритм вычисления мощности данного множества;
2) алгоритм последовательной генерации элементов этого множества;
3) алгоритм генерации элемента множества по номеру;
4) алгоритм нумерации элементов этого множества;
5) алгоритмы для исследования вычислительной сложности предложенных алгоритмов.
3. Рекуррентные и закрытые формулы для числа разбиений и композиций натурального числа n с ограничениями на части $\lambda_i \geqslant m$. Выражения производящих функций, треугольник разбиений.
4. Метод, основанный на процедуре полного разбиения для композиций, разбиений и разложений натурального числа n, обеспечивает построение алгоритмов комбинаторной генерации для корневых непомеченных деревьев.
5. Для любой контекстно-свободной грамматики с однозначным выводом взаимнооднозначно ставится в соответствие схема рекурсивной композиции деревьев И/ИЛИ.

Достоверность. Достоверность полученных результатов достигается до-

казательством теорем и аналитическими выкладками, сравнением с решениями других авторов, компьютерным моделированием, широким внедрением результатов диссертационной работы в практику.

Внедрение. Основные результаты диссертационной работы внедрены и используются:

1. На ОАО ТПО «Контур», при создании системы идентификации и прослеживаемости изделий телекоммуникационной механики и узлов структурных кабельных систем.
2. На ЗАО НПФ «Сибнефтекарт», при создании и тиражировании:

– Автоматизированной системы управления технологическими процессами и безналичными расчетами на автозаправочных станциях и комплексах «Сибнефтекарт-АЗС».

– Автоматизированной системы централизованного ведения и прогрузки на АЗС справочников товаров и контрагентов, приема, накопления и обработки сменных отчетов и протоколов с АЗС — «Сибнефтекарт-Офис».

– Программно-аппаратного комплекса эмиссии карт и учета транзакций процессингового центра «Сибнефтекарт-ПЦ».

3. На ООО НПФ «Информационные системы безопасности» при разработке архива удостоверяющего центра.
4. В дистанционной технологии обучения студентов: Томского государственного университета систем управления и радиоэлектроники по 38 специальностям, в Кемеровском технологическом институте пищевой промышленности по трем специальностям, в Югорском государственном университете по двум специальностям. Система проведения контрольных работ и экзаменов передана в отраслевой фонд алгоритмов и программ Министерства образования Российской Федерации (свидетельство регистрации №3524). Инструментальная система «Фея-3» передана в отраслевой фонд алгоритмов и программ Министерства образования Российской Федерации (свидетельство регистрации №3510). Пакет генераторов по дисциплине «Высшая математика» для проведения экзаменов передан в отраслевой фонд алгоритмов и программ Министерства образования Российской Федерации (свидетельство регистрации №3536).
5. В учебном процессе Томского университета систем управления и радиоэлектроники.

Личный вклад автора. Математический аппарат, аналитические выражения, алгоритмы, технологии и программное обеспечение, изложенные в диссертации, разработаны лично автором.

Апробация результатов диссертации. Основные результаты диссертации были доложены на международных, всесоюзных, всероссийских, региональных конференциях и семинарах. В том числе: на Международной конференции «Application of the Conversion Research Results for International

Cooperation, SIBCONVERS '99» Томск, 1999; на Международной конференции «Градоформирующие технологии XXI века», Москва, 2001; на второй Международной конференции «2-nd WBLE Conference» 2001, Lund University, Швеция; на Международной методической конференции «Новые информационные технологии в университетском образовании», Кемерово, 2002; на Всероссийской конференции «Современная образовательная среда», Москва, 2002; на Международной научно-методической конференции, Новосибирск, 2003; на Международной научно-методической конференции «Развитие системы образования в России XXI века», Красноярск, 2003; на Международном конгрессе «Информационные технологии в образовании», 2003; на Всероссийской конференции «Телематика», Санкт-Петербург, 2003; на III Всероссийской научно-практической конференции-выставки «Единая образовательная информационная среда: проблемы и пути развития», Омск, 2004; на Международной научно-методической конференции «Инновационные технологии организации обучения в техническом ВУЗе: на пути к новому качеству образования», Пенза, 2004. на Всероссийской научно-методической конференции «Телематика», Санкт-Петербург, 2007; на Международной конференции «Образование и виртуальность», Ялта, 2007; на Всероссийской конференции «Системы автоматизации в образовании, науке и производстве», Новокузнецк, 2007; на Всероссийской конференции «Телематика», Санкт-Петербург, 2008; на Международной научной конференции «Космос, астрономия и программирование[Лавровские чтения]»,Санкт-Петербург, 2008; на 9 Сибирской научной школе-семинаре с международным участием «Компьютерная безопасность и криптография», Омск, 2009; на городском алгебраическом семинаре Сибирского федерального университета, Красноярск, 2010.

Публикации. Результаты диссертационной работы опубликованы в 104 научных работах, в том числе в четырех монографиях, в пяти учебных пособиях и в 17 статьях журналов, рекомендованных ВАК РФ для опубликования научных результатов диссертаций на соискание ученой степени доктора наук.

Структура и объем работы. Диссертация состоит из введения, 6 глав, заключения, списка литературы, трех приложений. Общий объем диссертации составляет 387 страницы, включая 43 таблиц, 116 рисунков, библиографический список из 130 наименований, приложения (67 стр.)

Содержание работы

Во введении обоснована актуальность диссертационной работы, сформулирована цель и аргументирована научная новизна исследований, показана практическая значимость полученных результатов, представлены выносимые на защиту научные положения.

В первой главе введены и рассмотрены основные элементы предлага-

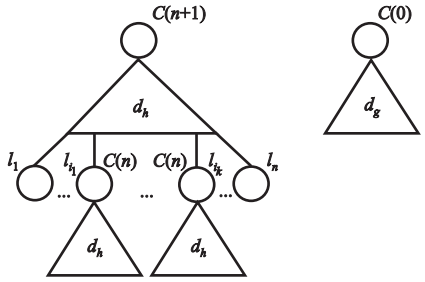


Рис. 2. Схема рекурсивной композиции деревьев

емой методологии: дерево И/ИЛИ, операция рекурсивной композиции деревьев, вариант дерева И/ИЛИ.

Определение 1. Пусть имеется множество помеченных деревьев $D = \{d_i\}_{i=1}^{n}$, в дереве $d_j \in D$ имеется m листьев, тогда операция композиции деревьев формулируется следующим образом: В дереве D_j k листьев ($k \leq m$) заменяются корнями деревьев из множества D. В результате получается новое дерево d_r. Эта операция обозначена как:

$$d_r = S_{i_1,i_2,...,i_k}(d_j, t_1, ..., t_k),$$

где $t_l \in D$ для всех $l = \overline{1,k}$ и $i_1, i_2, \ldots, i_k$ - метки листьев дерева d_j. Дерево d_r однозначно определяется фиксацией параметров $j, i_1, ..., i_k, t_1, ..., t_k$.

Определение 2. Пусть $D_0 = \{d_g, d_h\}$ и d_h имеет не менее m листьев, тогда схемой рекурсивной композиции деревьев будет

$$C(n) = \begin{cases} d_g, & n = 0, \\ S_{i_1,i_2,...,i_k}(d_h, t_1, \ldots, t_k), & n > 0, \end{cases} \qquad (1)$$

где $k \leq m$, $D_n = D_{n-1} \cup C(n-1)$, $t_l \in D_n$ для $l = \overline{1,k}$. Схему рекурсивной композиции $C(n)$ удобно представлять графически(см. рис. 2): на графическом изображении дерева d_h выделяются листья, к которым присоединяются деревья $C(n-1)$, изображенные треугольниками. $C(0)$ представляется отдельным изображением, в котором представлено дерево d_g. Если в схеме рекурсивной композиции отсутствует d_g, то подразумевается, что дерево d_g состоит из единственного узла. Математические выражения однозначно представляются в виде K-дерева (дерево Кантаровича, абстрактное синтаксическое дерево), во внутренних узлах которого записаны операции, а в листьях – операнды. Показано, что если d_g и d_h в схеме рекурсивной композиции $C(n)$ являются K-деревьями и соответствуют функциям $g(x_1, x_2, \ldots, x_m)$ – m-местная, $h(x_1, x_2, \ldots, x_{m+2})$ – $m+2$-местная в операторе примитивной рекурсии $R(g,h)$, то схема рекурсивной композиции $C(n)$ однозначно задает рекурсивную функцию $f(n)$.

Определение 3. Дерево И/ИЛИ – это корневое дерево, содержащее два типа

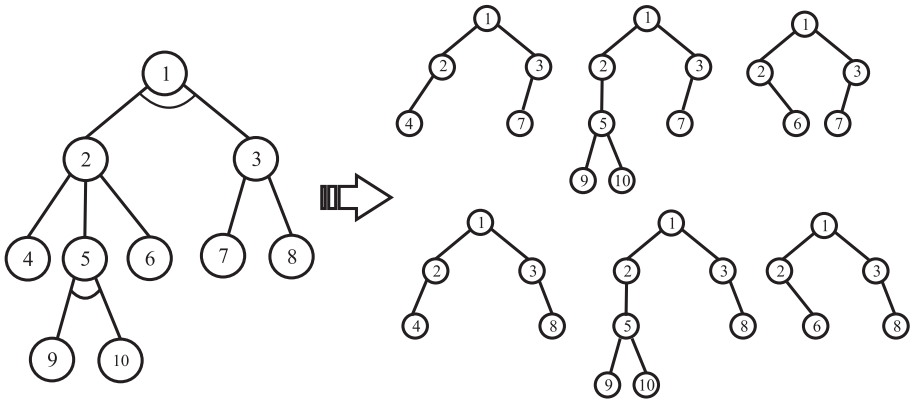


Рис. 3. Пример дерева И/ИЛИ и всех его вариантов

узла: И-узел и ИЛИ-узел.

Определение 4. Вариантом дерева И/ИЛИ назовем поддерево, которое получается из заданного путем отсечения выходных дуг кроме одной у всех ИЛИ-узлов. Вариантов в дереве И/ИЛИ может быть некоторое множество. На рис. 3 показан пример дерева И/ИЛИ и всех его вариантов. Получено следующее рекуррентное выражение для подсчета числа вариантов:

$$\omega_z = \begin{cases} \sum\limits_{i=1}^{n_z} \omega_i^z, & \text{для ИЛИ-узла;} \\ \prod\limits_{i=1}^{n_z} \omega_i^z, & \text{для И-узла;} \\ 1, & \text{для листа.} \end{cases} \tag{2}$$

где z - рассматриваемый узел дерева; $\{s_i^z\}_{i=1}^{n_z}$ - множество сыновей узла z; ω_z – число вариантов для узла z. Подсчитав значение функции для корня дерева, получим общее число вариантов, имеющихся в данном дереве. При этом будет подсчитано число вариантов для каждого узла всего дерева. Теперь, если записать '1' во все листья дерева И/ИЛИ, '+' – во все ИЛИ-узлы; '$\times$' – во все И-узлы, то получится абстрактное синтаксическое дерево, значение, которого будет равно числу вариантов в дереве И/ИЛИ. Таким образом, деревья И/ИЛИ однозначно строятся для выражений алгебры $\{1, +, \times\}$. Запишем алгебру $A = \{N, +, \times, R\}$, где N- множество натуральных чисел, R-оператор примитивной рекурсии. Покажем, что для любой функции $f \in A$ можно взаимно-однозначно поставить в соответствие схему рекурсивной композиции деревьев И/ИЛИ. Пусть f не рекурсивна, тогда ее выражение можно представить в виде K-дерева, где во внутренних узлах записаны операции '+' или '$\times$', а листья будут содержать переменные или константы. При фиксации аргументов функции f получим, все константы. Далее, каждую константу n_i представляем в виде ИЛИ-узла, у которого ровно n_i листьев, получим дерево И/ИЛИ. Если f рекурсивна, то для функций g и h строятся деревья И/ИЛИ

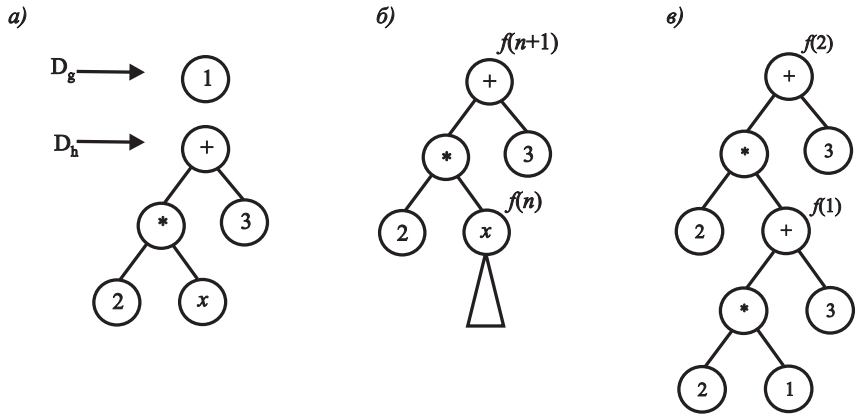


Рис. 4. Схема рекурсивной композиции деревьев И/ИЛИ для функции $f(n+1) = 2f(n)+3$

d_g и d_h и получаем схему рекурсивной композиции деревьев И/ИЛИ. При фиксации аргументов функции f, получается конкретное дерево И/ИЛИ, число вариантов которого равно значению рекурсивной функции. Пример схемы рекурсивной композиции показан на рис. 4.

Были рассмотрены алгоритмы First()/Next() для генерации вариантов дерева И/ИЛИ, для этого был разработан механизм стекового представления варианта(см. рис. 5). Первоначально создается главный стек, в который заносится корень дерева. Операция First() обеспечивает построение самого левого варианта и выполняется по следующим правилам:
1) рассматривается узел в текущем стеке;
2) если на вершине текущего стека находится ИЛИ-узел, то его самый левый сын заносится в стек и выполняется операция First для данного стека;
3) если на вершине - И-узел, то самый левый сын заносится в текущий стек, а для остальных сыновей создаются новые стеки, и в каждый из них заносятся сыновья И-узла, для всех стеков выполняется операция First();
4) если на вершине - лист, то для данного стека просмотр заканчивается.
Таким образом, в результате выполнения операци First() получается самый левый вариант в дереве И/ИЛИ, причем, в главном стеке находится самый левый путь от корня к листу. Для всех правых сыновей И-узлов созданы собственные стеки.

Были получены следующие правила для операции Next():
1) первоначально текущим становится главный стек;
2) рассматривается узел на вершине текущего стека;
3) если узел является листом, то удаляется из стека, переход на шаг 2;
4) если узел является ИЛИ-узлом, то, если у удаленного узла имеется соседний брат справа, то он помещается в стек и выполняется операция First(), выход из Next(). Если удаленный был последним, то из стека удаляется данный ИЛИ-узел, происходит переход на шаг 2;
5) если узел является И-узлом, то для стеков правых сыновей выполняется

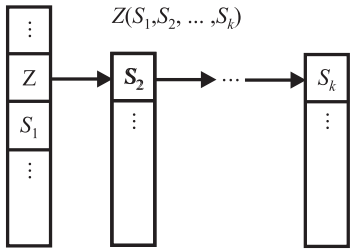


Рис. 5. Структура стеков для И-узла $Z(S_1, S_2, \ldots, S_k)$

операция Next, до тех пор пока не будет успешно выполнена эта операция, если операция Next выполнилась неудачно (стек пуст) для самого правого сына, то данный И-узел удаляется из стека и происходит переход на шаг 2, если операция завершилась удачно, то для всех левых сыновей относительно данного выполняется операция First(), выход из Next();
6) если в стеке пусто, то все варианты поддерева просмотрены и выполняется возврат, если это главный стек, то все варианты дерева просмотрены, выполняется завершение.
Данные правила формализованы в виде автомата последовательной генерации вариантов: $(B, N_D, P_{First}, P_{Next})$, где N_D- множество магазинных символов; B - начальный магазинный символ; P_{First} – правила для операции First; P_{Next}- правила для операции Next;

Множеством магазинных символов N_D является множество помеченных узлов дерева И/ИЛИ. Начальным магазинным символом $B \in N_d$ является корень дерева. Правила P_{First} описывают порождение первого варианта для каждого внутреннего узла дерева И/ИЛИ. Для ИЛИ узла правило гласит, что если на вершине магазина записан ИЛИ-узел $z(s_1, s_2, \ldots, s_k)$, то в магазин помещается левый сын узла s_1^z. Это правило будет записано следующим образом:

$$First(z) \to First(s_1^z).$$

Для И-узла правило гласит, что если на вершине магазина записан И-узел $z(s_1, s-2, .., s_k)$, то для всех сыновей, кроме самого левого создаются новые стеки. В первичный стек заносится левый сын, во вновь созданные стеки заносятся соответствующие сыновья узла z. Выполняется операция First для всех стеков. Это правило будет записано следующим образом: $First(z) \to First(s_1^z)First(s_2^z) \ldots First(s_k^z)$.

Рассмотрены правила P_{Next}. Для устранения неоднозначности в применении правил для выполнения операции Next задан предикат $Prev(s)$, который разрешает применение правила, если из текущего стека был удален символ s. Для обозначения операции удаления символа из стека вводится символ pop. Тогда:
1. Правила для листьев, если в стеке лист дерева, то он вынимается из стека.

$Next(z)Prev(Nil) \rightarrow pop$
2. Правила ИЛИ-узла, правила замены
$Next(z)Prev(s_1^z) \rightarrow First(s_2^z)$
$Next(z)Prev(s_2^z) \rightarrow First(s_3^z)$
...
$Next(z)Prev(s_{k-1}^z) \rightarrow First(s_k^z)$
$Next(z)Prev(s_k^z) \rightarrow pop$
3. Правила для И-узла:
$Next(z)Prev(s_1^z) \rightarrow First(s_1^z)Next(s_2^z)$
$Next(z)Prev(s_2^z) \rightarrow First(s_1^z)First(s_2^z)Next(s_3^z)$
...
$Next(z)Prev(s_{k-1}^z) \rightarrow First(s_1^z)First(s_2^z)\ldots First(s_{k-1}^z)Next(s_k^z)$
$Next(z)Prev(s_k^z) \rightarrow pop$
Операция First реализована в виде следующего алгоритма

```
algorithm First(stack) begin
    do begin
       z:=stack.top() { определяем символ на вершине стека}
       Rule(z,P_First) {выполняем правило для узла z}
      end
while(z<>leaf) {конец если на вершине находится лист }
   end
```

Операция Next реализована в виде следующего алгоритма

```
algorithm Next(stack)
do {
z:=stack.top() { определяем символ на вершине стека }
      if z=leaf then {  если это лист }
        begin
          stack.pop() {выталкиваем его}
          prev := z { устанавливаем значение prev }
        end
     else begin { если это не лист }
        iRule=TNext[z,prev] { ищем номер правила по таблице }
        if Run(iRule) then { выполняем это правило }
        return; { если нашли лист }
       end
   while(stack.size()>0)
end
```

Задача определения сложности алгоритма последовательной генерации вариантов была сведена к подсчету числа стековых операций при генерации всех вариантов для данного дерева И/ИЛИ. Причем операция First формирует стековое представление самого левого варианта. Операция Next производит

формирование следующего варианта путем замены одного поддерева варианта на другое. После генерации последнего варианта операция Next выталкивает все элементы из всех стеков. Таким образом, число операций push и pop при генерации всех вариантов будет одинаковое. Тогда общее число операций push для дерева И/ИЛИ будет равно:

$$Op(z) = \begin{cases} \sum_{i=1}^{k} Op(s_i) + 1, & \text{для ИЛИ-узла;} \\ \sum_{j=1}^{k-1} [Op(s_j) \dot{\prod}_{i=j+1}^{k} \omega(s_i)] + Op(s_k) + 1, & \text{для И-узла;} \\ 1, & \text{для листа.} \end{cases} \quad (3)$$

Если дерево И/ИЛИ задано в виде схемы рекурсивной композиции, то параметр рекурсии n позволяет различать узлы при построении конкретного дерева. Если в дереве d_h корень помечен как Z, то при фиксации параметра n, получим множество узлов $\{Z_i\}_{i=1}^{n}$. Таким образом, множество магазинных символов равно множеству $|d_h|n + |d_g|$. Начальный символ будет $Z[n]$. Правила First и Next будут иметь рекурсивную структуру.

Основная идея алгоритма генерации варианта по номеру заключается в том, что, зная номер варианта для заданного узла z, можно пересчитывать этот номер в номера вариантов для поддеревьев его сыновей, основываясь на значениях $\omega_{s_i^z}$. Алгоритм пересчета для И-узла основан на использовании выражения

$$\omega_z = \prod_{i=1}^{n_z} \omega_i^{(z)},$$

и для некоторого номера l_z , такого, что $0 \le l_z < \omega_z$, воспользовавшись алгоритмом преобразования числа l_z в число со смешанной системой счисления, в которой основаниями являются $\{\omega_i^z\}_{i=1}^{n_z}$, получим значения для номеров варианта l_i всех сыновей И-узла z. Отсюда, между числом l_z и множеством $\{l_i\}_{i=1}^{n_z}$ будет взаимно-однозначное соответствие. Для ИЛИ-узла определяется не только номер варианта для сына l_s, но и номер соответствующего сына. Основываясь на выражении подсчета вариантов для ИЛИ-узла, можно предложить следующий алгоритм: ищется интервал, в который попадает l_z.
Если $l_z < \omega_1^{(z)}$, то выбирается первый сын $l_s := l_z$;
Если $l_z < \omega_1^{(z)} + \omega_2^{(z)}$, то выбирается второй сын и $l_s := l_z - \omega_1^{(z)}$
...
Если $l_z < \sum_{i=1}^{k} \omega_i^{(z)}$, то выбирается k-й сын и $l_s := l_z - \sum_{i=1}^{k-1} \omega_i^{(z)}$.
Очевидно, что между l_z и парой (k, l_k) взаимно-однозначное соответствие. Тогда алгоритм генерации варианта следующий:
1. В стек помещается пара $(num, root)$: номер варианта и корень дерева.
2. Из стека вынимается очередная пара (l, z).
3. Если z это И-узел, то вычисляются l_i для всех сыновей s_i узла z, все пары

(l_i, s_i) заносятся в стек, переход на шаг 2.
4. Если z это ИЛИ-узел, то определяется пара (l_k, s_k), заносится в стек, переход на шаг 2.
5. Если z лист, то переход на шаг 2.
6. Если стек пуст, то останов.
Алгоритм генерации варианта задает биективное отображение N_n на множество вариантов W_d дерева И/ИЛИ d.

Для схемы рекурсивной композиции алгоритм генерации варианта носит рекурсивный характер и не требуется явного построения дерева И/ИЛИ. При этом:
1) число вариантов для всех узлов в схеме рекурсивной композиции выражается через рекурсивную функцию $\omega(n)$;
2) поскольку дерево И/ИЛИ строится из деревьев d_h и d_g, то необходимо построить алгоритм GenerateVariant для случаев d_h и d_g и записать рекурсивный алгоритм. Алгоритм нумерации варианта дерева И/ИЛИ следующий. Производится сопоставление варианта $v \in V$ в дереве d и вычисление соответствующего номера i. Алгоритм сопоставления следующий:
1. В стек s_v заносим корень варианта, в стек s_d заносим корень дерева.
2. Из стека s_v вынимаем узел варианта z и ищем этот узел в дереве И/ИЛИ.
3. Если это лист, то на шаг 2.
4. Если это ИЛИ-узел, то находим k-го сына в дереве И/ИЛИ, помещаем его в стек s_d, сына узла варианта z помещаем в стек s_v.
5. Если это И-узел, то всех сыновей дерева И/ИЛИ заносим в стек s_d, а сыновей варианта в s_v.
6. Если стек s_v пуст, то завершить работу алгоритма.
В результате работы алгоритма сопоставления в стеке s_d будут находиться все узлы дерева И/ИЛИ, входящие в вариант. Далее, используя стек s_d, вычисляются значения номеров вариантов для всех узлов, находящихся в стеке.

Вычисление номера начинается с рассмотрения листьев варианта v. Все листья варианта имеют значения 0.
1. Для каждого И-узла z вычисляется

$$l_z = l_1 + \omega_1(l_2 + \omega_2(...l_{n-1}\omega_n)...)),$$

где $\{l_i\}$ – соответствующие номера, полученные для сыновей.
2. Для каждого ИЛИ-узла вычисляется

$$l_z = \sum_{i=1}^{k-1} \omega_i + l_k,$$

где k- номер соответствия для узла ИЛИ в дереве d, l_k - номер варианта для этого сына.

Для определения временной сложности предложенных алгоритмов рассмотрено число операций для генерации всех вариантов в дереве И/ИЛИ. Основными операциями для генерации варианта выделены операции сравнения и деления. При этом все операции, которые необходимо использовать для получения варианта, будут иметь временную сложность равную единице. Таким образом, для дерева И/ИЛИ получено следующее выражение:

$$Op_z = \begin{cases} \sum_{i=1}^{n_z}[(n_z - i)\omega_i^z + Op_{s_i}], & \text{для ИЛИ-узла;} \\ (n_z - 1 + \sum_{i=1}^{n_z} \frac{Op_{s_i}}{\omega_i}) \cdot \omega_z, & \text{для И-узла;} \\ 0, & \text{для листа.} \end{cases} \quad (4)$$

Отношение Op_z/ω_z дает среднее число операций на один вариант, сгенерированный по номеру.

Предложенная методология базируется на возможности создания дерева И/ИЛИ с заданным множеством вариантов. Если мощность некоторого комбинаторного множества задана в виде числа n, то строится конкретное дерево И/ИЛИ, число вариантов которого равно n. Если мощность некоторого класса комбинаторных множеств задана в виде функции $f \in \{N, +, \times, R\}$, то можно задать схему рекурсивной композиции деревьев И/ИЛИ. При фиксации параметров этой функции и, соответственно, схемы рекурсивной композиции значение функции совпадает с числом вариантов в дереве И/ИЛИ, полученном по схеме рекурсивной композиции.

Кроме того, схему рекурсивной композиции деревьев И/ИЛИ можно получить не зная функции мощности класса комбинаторного множества, основываясь на некоторых правилах и свойствах порождающего алгоритма для данного класса. Например, множество перестановок из $(n+1)$ элементов может быть получено из множества перестановок n элементов. Или множество двоичных деревьев высотой не более $n+1$ может быть получено из множества двоичных деревьев высотой не более n. Отсюда, используя свойства деревьев И/ИЛИ, можно построить биективное отображение между комбинаторным множеством и множеством вариантов:

$$G : CM \to W_d,$$

где CM – комбинаторное множество, W_d – множество вариантов дерева d.

На основе полученной биекции предложены методы комбинаторной генерации:

1. Метод построения алгоритмов последовательной генерации элементов комбинаторного множества CM:

– Строится фиксированное дерево И/ИЛИ d или схема рекурсивной композиции $C_d(n)$.

– Записывается автомат $(B, N_D, P_{First}, P_{Next})$.

– Разрабатывается процедура получения элемента комбинаторного множества по стековому представлению варианта.
– Реализуются алгоритмы First()/Next().
– Производится анализ автомата генерации вариантов и процедуры получения элемента комбинаторного множества.
2.Метод построения алгоритмов генерации элементов комбинаторного множества CM по номеру:
– Строится фиксированное дерево И/ИЛИ d или схема рекурсивной композиции $C_d(n)$.
– Разрабатывается алгоритм генерации варианта по номеру, в случае, если задана схема рекурсивной композиции, то записывается рекурсивный алгоритм получения варианта.
– Записывается процедура получения элемента комбинаторного множества по варианту.
– Производится анализ алгоритма генерации варианта и процедуры получения элемента комбинаторного множества.
3. Метод построения алгоритмов нумерации элементов комбинаторного множества CM:
– Строится фиксированное дерево И/ИЛИ d или схема рекурсивной композиции $C_d(n)$.
– Разрабатывается алгоритм нумерации варианта, в случае, если задана схема рекурсивной композиции, то записывается рекурсивный алгоритм нумерации варианта.
– Записывается процедура получения варианта по элементу комбинаторного множества.
– Производится анализ алгоритма нумерации варианта и процедуры получения варианта по элементу комбинаторного множества.

Во второй главе рассмотрен класс деревьев И/ИЛИ, у которых отсутствуют И-узлы. Этот класс деревьев строится на основе представления мощности комбинаторного множества в виде рекуррентного выражения, состоящего только из аддитивных составляющих. Для сочетаний получен алгоритм генерации по номеру num на основе приближенного решения уравнения

$$n \approx [\sqrt[m]{num \cdot m!} + \frac{m-1}{2}]. \tag{5}$$

где n – номер бита, m – общее число единиц в сочетании.

Для разбиений натурального числа n доказана теорема, что число разбиений натурального числа n на части, каждая из которых не менее m определяется следующей формулой:

$$p_m(n) = p_m(n-m) + p_{m+1}(n). \tag{6}$$

На основании данной формулы были получены оригинальные алгоритмы: последовательной генерации для разбиений, генерации разбиений по номеру и нумерации разбиений с указанными ограничениями. Отличительной особенностью данных алгоритмов от известных является универсальность, при условии, что среднее число операций push и pull на одно разбиение равно 4. Это объясняется тем, что по формуле 6 получается дерево И/ИЛИ, которое является полным двоичным деревом и число узлов в таком дереве равно $(2n-1)$, где n – число листьев. Откуда число операций push на один вариант будет равно $\frac{2p_m(n)-1}{p_m(n)}$. Это согласуется с известным алгоритмом генерации разбиений, у которого число операций равно $4 - C/\sqrt{(}n) + O(1/n)$ (Д. Кнут "Генерация всех разбиений"). Построен треугольник разбиений, получены производящие функции для разбиений $p_m(n)$.

В третьей главе рассмотрены классические комбинаторные множества: перестановки, множества, мощности которых заданы формулами Каталана, Стирлинга и Сильвестра. Для каждого множества по рекуррентной формуле построена схема рекурсивной композиции деревьев И/ИЛИ, получен автомат последовательной генерации и соответствующие алгоритмы First и Next, разработаны алгоритм генерации по номеру и алгоритм нумерации элементов соответствующего множества. Ниже в таблице представлены схемы рекурсивной композиции.

Для перестановок имеется большое число алгоритмов комбинаторной генерации, это объясняется большим числом разнообразных форм представления. Мощность множества перестановок выражается через рекуррентные формулы, основанные на числах Стирлинга, Эйлера, Каталана. Все эти формулы имеют рекуррентную природу. Предложен метод построения алгоритмов генерации перестановок на основе формулы перестановок мультимножества: параметр множества перестановок n представляется как композиция натурального числа, $n_1 + n_2 + \ldots + n_k = n$. Тогда

$$n! = \left(\begin{array}{c} n \\ n_1, n_2, \ldots, n_k \end{array} \right) \cdot \prod_{i=1}^{k} n_i!.$$

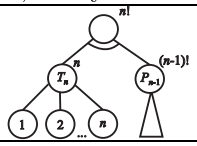
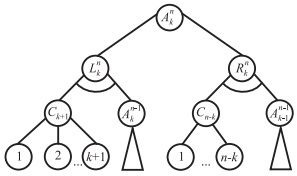
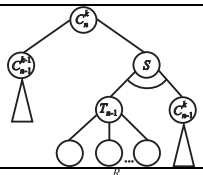
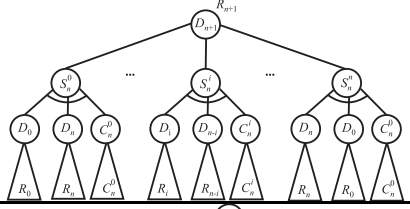
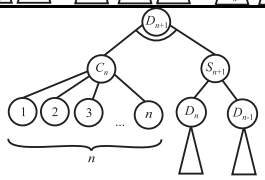
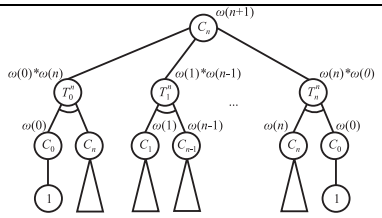
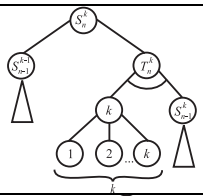
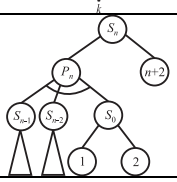
Или

$$n! = [C_n^{n_1} C_{n-n_1}^{n_2} C_{n-n_1-n_2}^{n_3} \ldots C_{n_k}^{n_k}] \cdot \prod_{i=1}^{k} n_i!.$$

Отсюда, задавая алгоритмы генерации композиции числа n, алгоритм генерации перестановки мультимножества и алгоритмы генерации перестановок для n_i, получим уникальный алгоритм генерации перестановки (см. рис. 6).

Используя данный подход можно предложить следующий алгоритм генерации перестановки. Число $n > 1$ представляется в виде $2 + 3k$, $3k$, $4 + 3k$.

Таблица 1. Схемы рекурсивных композиций, полученных в главе 3

1	Перестановки (инверсии)	
2	Перестановки (формула Эйлера)	
3	Перестановки (формула Стирлинга)	
4	Перестановки (возрастающие деревья)	
5	Беспорядки	
6	Числа Каталана	
7	Числа Стирлинга 2 рода	
8	Числа Сильвестра	

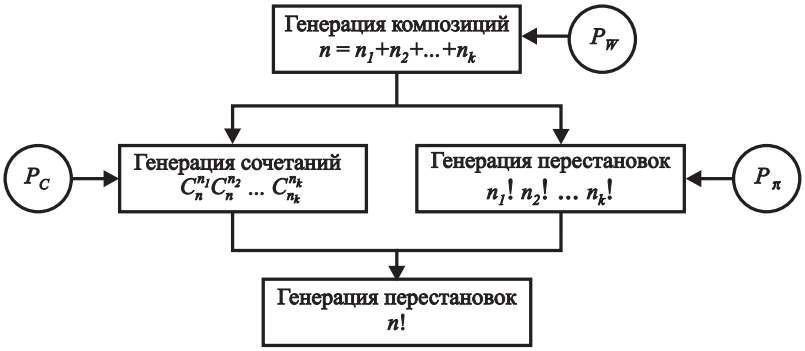


Рис. 6. Комбинация алгоритмов генерации перестановок

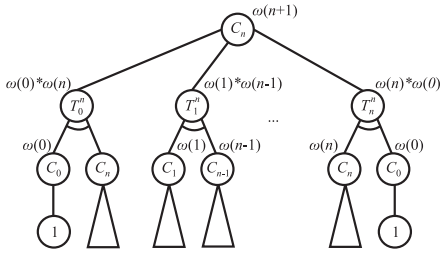


Рис. 7. Схема рекурсивной композиции для множеств, описываемых числами Каталана

Необходимо иметь все перестановки для чисел 2, 3 и 4. Тогда все перестановки мощностью 2, 3, 4 могут быть представлены массивами констант. А сочетания примут вид C_m^2,C_m^3,C_m^4, для которых можно построить эффективные алгоритмы генерации.

Рассмотренный метод применен для построения алгоритмов комбинаторной генерации для множеств, описываемых числами Каталана. Представленное рекуррентное выражение $C_{n+1} = C_0C_n+C_1C_{n-1}+\ldots+C_0C_n, C_0 = 1$, является рекурсивной функцией алгебры $\{N, +, \times, R\}$. Тогда для данной функции строится схема рекурсивной композиции дерева И/ИЛИ. Правила построения следующие:
1. Корнем будет узел $\{C_{n+1}\}$, это будет ИЛИ узел, у которого ровно $n + 1$ сыновей.
2. Каждый i-й сын ИЛИ-узла является И-узлом, имеющий поддеревья C_i и C_{n-i}. Схема рекурсивной композиции представлена на рис. 7. Вариантом для дерева И/ИЛИ, основанного на данной схеме рекурсивной композиции, будет являться двоичное дерево, т.к. все И-узлы имеют двух сыновей. Тогда автомат последовательной генерации будет иметь следующий вид:

1. Множество магазинных символов:
   $\{C_i\}_{i=0}^{n+1}$ – символы, соответствующие узлам вычисления чисел Каталана.
   $\{T_j^n\}_{j=0}^{n}$ – символы, соответствующие членам суммы.
   1 – константа, соответствующая листу.

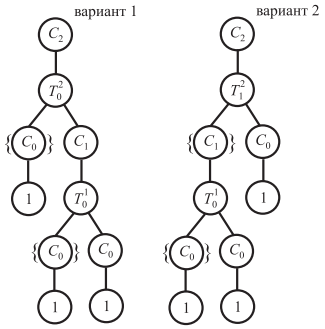


Рис. 8. Представление вариантов для множеств Каталана

Таблица 2. Значения функции числа операций для генерации элементов множеств, описываемых формулой Каталана

n	$Op(n)$	C_n	$\frac{Op(n)}{C_n}$
7	625	429	1.4568764568
8	2055	1430	1.4370629370
9	6917	4862	1.4226655697
10	23713	16796	1.4118242438
11	82499	58786	1.4033783553
12	290511	208012	1.396606926
13	1033411	742900	1.3910499394
14	3707851	2674440	1.3864027609
15	13402696	9694845	1.3824559340

2. Начальный символ C_{n+1}.

3. Правила First
   1) $\text{First}(C_i) \rightarrow \text{First}(T_0^{i-1})$
   2) $\text{First}(T_j^i) \rightarrow \text{First}(C_i)\text{First}(C_{i-j})$
   3) $\text{First}(C_0)$->1

4. Правила Next
   1) $\text{Next}(C_i)\text{Prev}(T_j^i)(j < i)$->$\text{First}(T_{j+1}^i)$
   2) $\text{Next}(C_i)\text{Prev}(T_j^i)(j = i)$->pop
   3) $\text{Next}(C_0)\text{Prev}(1)$->pop
   4) $\text{Next}(T_j^i)$->$\text{First}(C_i)\text{Next}(C_{i-j})$

Автомат, построенный на приведенных правилах, будет последовательно генерировать варианты для заданного параметра n. Определим вычислительную сложность алгоритма последовательной генерации. В соответствии с выражением для числа операции для последовательной генерации вариантов

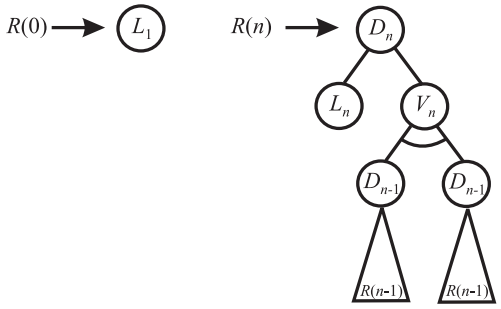


Рис. 9. Схема рекурсивной композиции для полных двоичных деревьев высотой

получим, что

$$Op(n+1) = \sum_{i=0}^{n} [Op(i) \cdot C_{n-i} + Op(n-i) + 1] + 1.$$

Для листа $Op(0) = 1$. Работу автомата можно оптимизировать, заметив, что от магазинных символов T_i можно избавиться. Тогда число операций будет выражаться формулой

$$Op(n+1) = \sum_{i=0}^{n} [Op(i) + 1] C_{n-i}.$$

Результаты моделирования для полученной рекуррентной формулы приведены в таблице 2. Как видно из приведенной таблицы 2 результаты согласуются с формулой, приведенной Д.Кнутом $4C_n/3 + O(1/n^2)$.

Четвертая глава посвящена разработке алгоритмов комбинаторной генерации корневых деревьев. Показано эффективное применение предложенной методологии для генерации практически всех классов корневых деревьев: двоичных деревьев с высотой не более n, с заданной высотой n, с заданным числом узлов n, полных двоичных деревьев, АВЛ деревьев, красно-черных деревьев, 2-3 деревьев, крашеных деревьев, троичных деревьев, t-арных деревьев, упорядоченных, неупорядоченных, деревьев Кемпа, Шредера, Риордана. Приведем пример использования методологии для получения алгоритмов комбинаторной генерации для полных двоичных деревьев высотой не более n. Первым шагом предлагаемой методологии является построение рекурсивной композиции деревьев И/ИЛИ полных двоичных деревьев высотой не более n. Для этого предположим, что имеется множество полных двоичных деревьев высотой не более $n-1$ и для этого множества имеется схема рекурсивной композиции $(n-1)$. Тогда множество полных двоичных деревьев высотой не более n состоит из дерева высотой 0 и деревьев, у которых левое и правое поддеревья высотой не более $n-1$. Отсюда, схема рекурсивной композиции будет следующая (см. рис. 9)

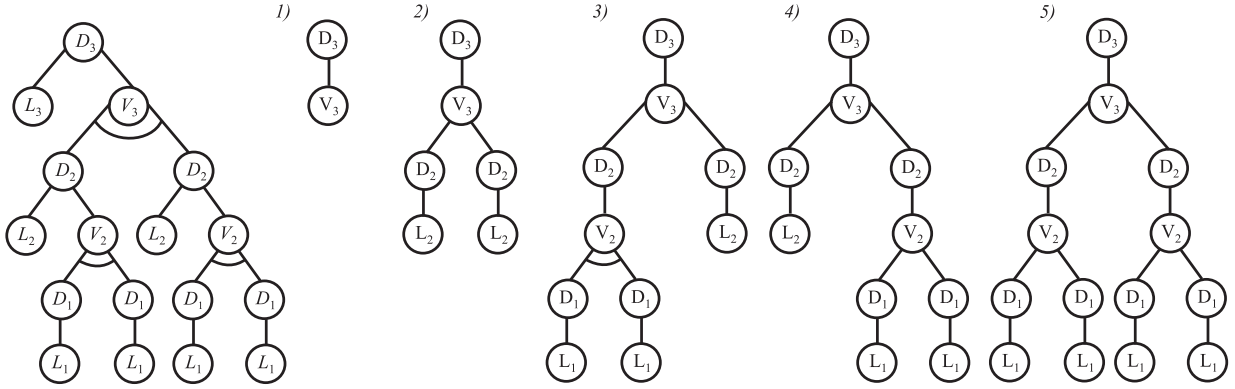


Рис. 10. Пример дерева И/ИЛИ для $n = 3$ и всех его вариантов

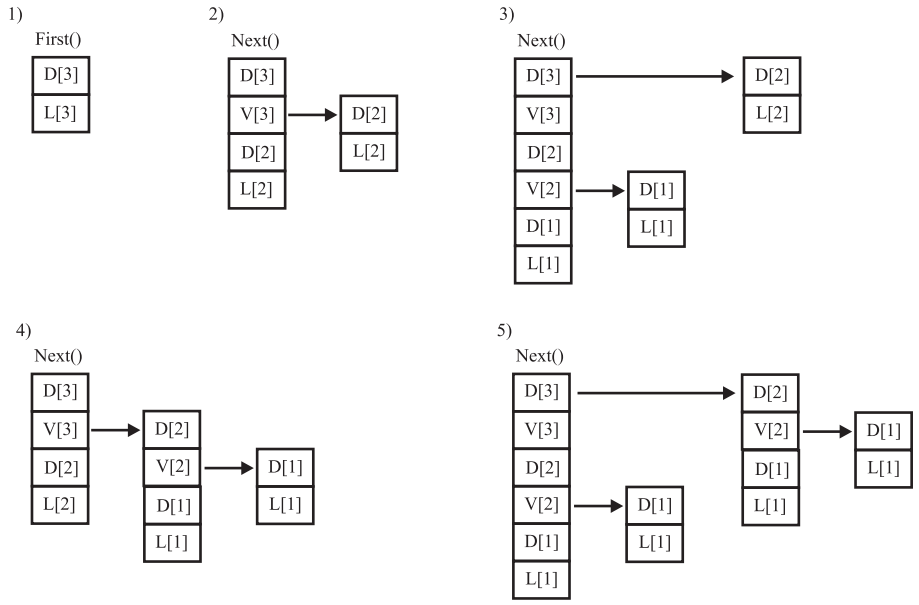


Рис. 11. Стековое представление вариантов для $C(3)$

Получим рекурсивное выражение для мощности полных двоичных деревьев высотой не более n. Для этого по схеме рекурсивной композиции получим деревья d_g и d_h (рис. 9). Тогда $g(x) = 1$, $h(x) = 1 + x \cdot x$. Отсюда, множество полных двоичных деревьев высотой не более n выражается следующим рекурсивным выражением:

$$\omega(n) = \begin{cases} 1, & n = 0, \\ 1 + \omega^2(n-1), & n > 0. \end{cases} \tag{7}$$

Данное выражение согласуется с формулой, полученной А.Ахо и Н.Слоун. При $n = 3$ получим следующее И/ИЛИ дерево (см. рис. 10) Число вариантов в дереве равно 5. По каждому варианту однозначно получается соответствующее полное двоичное дерево.

Алгоритм последовательной генерации. Запишем автомат $(B, N, P_{First}, P_{Next})$: Магазинные символы: $\{D_i\}_{i=1}^{n}$, $\{V_i\}_{i=1}^{n}$, $\{L_i\}_{i=1}^{n}$.

Таблица 3. Результаты моделирования для среднего числа операций в алгоритме последовательной генерации

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$Op(n)/\omega(n)$	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

Начальный символ: D_n.
Правила для операции $First$:
$First(D_i) \to L_i$
$First(V_i) \to First(D_{i-1})First(D_{i-1})$.
Правила для операции Next:
$Next(D_1)Prev(L_1) \to pop$
$Next(D_i)Prev(L_i) \to First(V_i)$
$Next(D_i)Prev(V_i) \to pop$
$Next(V_i) \to First(D_{i-1})Next(D_{i-1})$.
На рис. 11 перечислены все стековые конфигурации для последовательной генерации полных двоичных деревьев высотой не более 3.

Запишем число операций push для последовательной генерации вариантов:
$Op(D_n) = Op(L_n) + Op(V_n) + 1$,
$Op(L_n) = 1$,
$Op(V_n) = Op(D_{n-1})\omega(n-1) + Op(D_{n-1}) + 1$.
Тогда $Op(n) = Op(n-1)[\omega(n-1) + 1) + 3$.
Результаты моделирования приведены в таблице 3. Запишем алгоритм генерации полных двоичных деревьев высотой не более n.

```
algorithm GenFullBinaryTree(num,n) begin
    if n=0 then return Nil
    Create(node) { создается узел дерева }
    if num==0 then return Nil; {сыновей нет}
     begin {есть два сына  }
      num := num − 1
      l1:=num  mod ω(n − 1)
      l1:=num/ω(n − 1)
      node.left:=GenFullBinaryTree(l1,n − 1)
      node.right:=GenFulBinaryTree(l2,n − 1)
     end
     return node
end

    algorithm RankFullBinaryTree(Node,n) begin
     if Node=Nil or n=0 then return 0
```

```
    begin
      l1:=RankFullBinaryTree(Node.left,n − 1)
      l2:=RankFullBinaryTree(Node.right ,n − 1)
      return l1 + l2ω(n − 1)
    end
end
```

Модифицирована процедура полного разбиения, которую впервые предложил Н.Я. Виленкин, как подсчет процессов последовательных разбиений конечного множества, а Р.Стенли использовал эту процедуру для полного разбиения конечного множества S, в результате выполнения которой получается некоторое корневое помеченное дерево. В этой процедуре предложено вместо конечного множества S использовать натуральное число n. Тогда, имея некоторый алгоритм $R(n)$ представления натурального числа n в виде суммы, в общем случае, неотрицательных чисел λ_i, строится непомеченное n-узловое корневое дерево. Алгоритм построения дерева следующий:

1. Создаем корень дерева r и пару $< n, r >$ заносим в стек;

2. Если стек пуст, то завершаем процедуру.

3. Вынимаем из стека очередную пару $< k, z >$. Если $k = 1$, то данный узел становится листом и переходим на шаг 2, иначе, в соответствии с алгоритмом R, получаем представление $\pi : [\lambda_1 + \lambda_2 + \ldots + \lambda_m]$, такое, что $\sum_{i=1}^{m} \lambda_i = k - 1$, и переходим на шаг 4.

4. Создаем m узлов $\{s_i\}_{i=1}^{m}$, присоединяем их в качестве сыновей к узлу z в соответствии с порядком, заданным в представлении $\{\lambda_i\}_{i=1}^{m}$. Все пары $< \lambda_i, s_i >$ записываем в стек. Переходим на шаг 3.

Свойства предложенного алгоритма полного разбиения:

1. Если задан $R(n),\ n > 0$ и $\lambda_i \geq 1$ для всех представлений $\pi \in P_n$, то предложенный алгоритм всегда строит дерево.

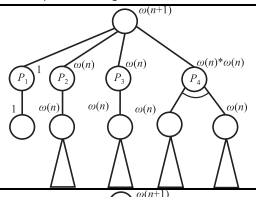
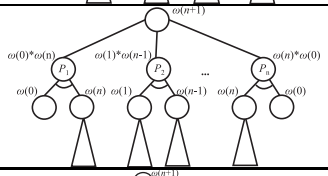
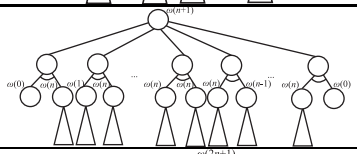
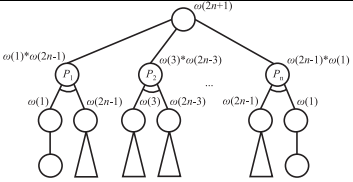
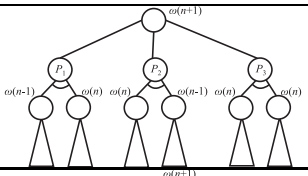
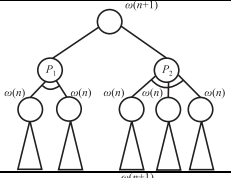
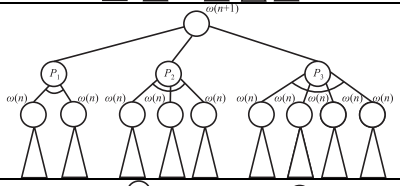
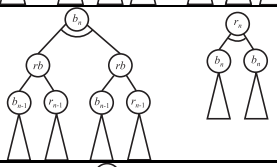
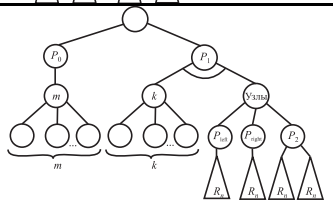
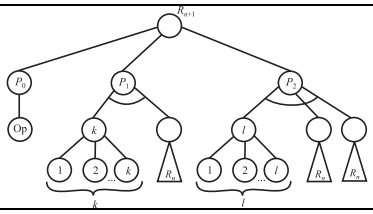
2. Число деревьев для заданного числа узлов равно:

$$K(n) = \sum_{\pi \in P_{n-1}} \prod_{i=1}^{m} K(\lambda_i), \tag{8}$$

где $\pi = \{\lambda_1 + \lambda_2 + \ldots + \lambda_m = n - 1\}$, P_{n-1} – множество представлений числа $(n - 1)$, в соответствии с алгоритмом R.

Используя алгоритмы генерации разбиений, композиций и разбиений в процедуре полного разбиения, были получены алгоритмы генерации n-узловых t-арных деревьев, корневых упорядоченных деревьев, корневых неупорядоченных деревьев. При исследовании процедуры полного разбиения основан-

Таблица 4. Схемы рекурсивных композиций, полученных в главе 4

1	двоичные деревья высотой не более n	
2	двоичные деревья с заданным числом узлов	
3	двоичные деревья высотой n	
4	полные двоичные деревья с числом узлов $2n+1$	
5	АВЛ-деревья высотой n	
6	2-3 деревья высотой n	
7	2-3-4 деревья высотой n	
8	красно-черные деревья высотой n	
9	цветные двоичные деревья высотой не более n	
10	дерево операций	

ной на композициях была получена новая функция:

$$T(i) = \begin{cases} 1, & i = 0, \\ T(i - 2^m)\frac{4k+2}{k+2}, & i > 0, \end{cases} \tag{9}$$

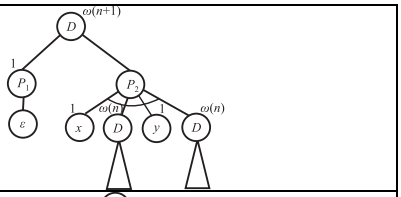
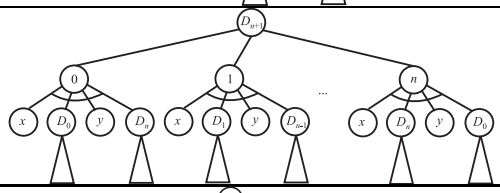
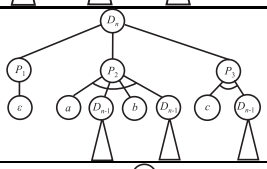
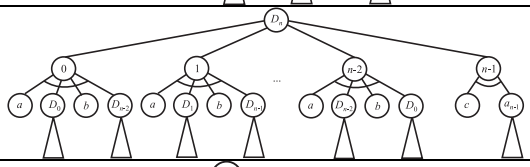
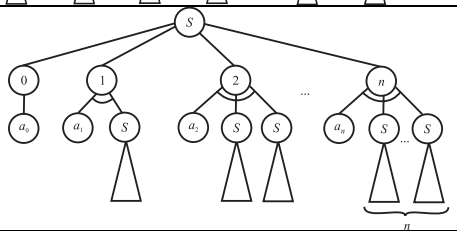
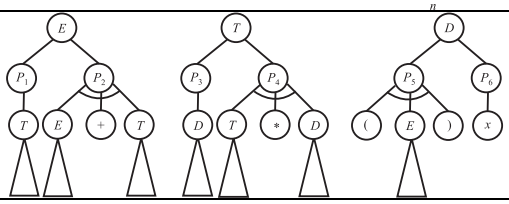
где $m = \lfloor \log(i) \rfloor$,

$$k = \begin{cases} m - \lfloor \log(2^{m+1} - i - 1) \rfloor - 1, & 0 \le i < 2^{m+1} - 1, \\ m, & i = 2^{m+1} - 1. \end{cases}$$

В пятой главе показано применение предложенной методологии для построения алгоритмов комбинаторной генерации для множеств выражений языков, описываемых однозначными контекстно-свободными грамматиками (B, T, N, P), где B – начальный символ, T – множество терминальных символов, N – множество нетерминальных символов, P – правила подстановки. Для построения схемы рекурсивной композиции деревьев И/ИЛИ предложены следующие правила: для каждого нетерминала $A \in N$ строится И/ИЛИ дерево. Корнем этого дерева будет являться узел, помеченный нетерминалом A, это будет ИЛИ-узел, у которого число сыновей равно числу правил подстановки, у которых в левой части записан этот нетерминал (они помечаются номерами $P_1, P_2 \dots P_k$). Каждый узел P_i является И-узлом, у которого каждый сын соответствует символу грамматики, присутствующий в правой части правила P_i. Для каждого дерева И/ИЛИ нетерминала грамматики вводится параметр рекурсии, при фиксации которого для рекурсивной композиции получается фиксированное дерево И/ИЛИ, описывающее подмножество выражений данного языка. Отсюда вариант дерева И/ИЛИ, построенного для грамматики будет являться деревом вывода для конкретного выражения. Поскольку между вариантом и выражением должно быть взаимно-однозначное соответствие, то контекстно-свободная грамматика должна быть грамматикой с однозначным выводом.

В шестой главе представлен программный комплекс исследования, разработки и применения алгоритмов комбинаторной генерации. Этот комплекс включает инструментальное программное обеспечение и прикладные программные системы.

Таблица 5. Схемы рекурсивных композиций, полученных в главе 5

1	выражения языка Дика	
2	выражения языка Дика длиной n	
3	выражения языка Моцкина	
4	выражения языка Моцкина длиной n	
5	выражения языка Лукасевича длиной n	
6	арифметические выражения	

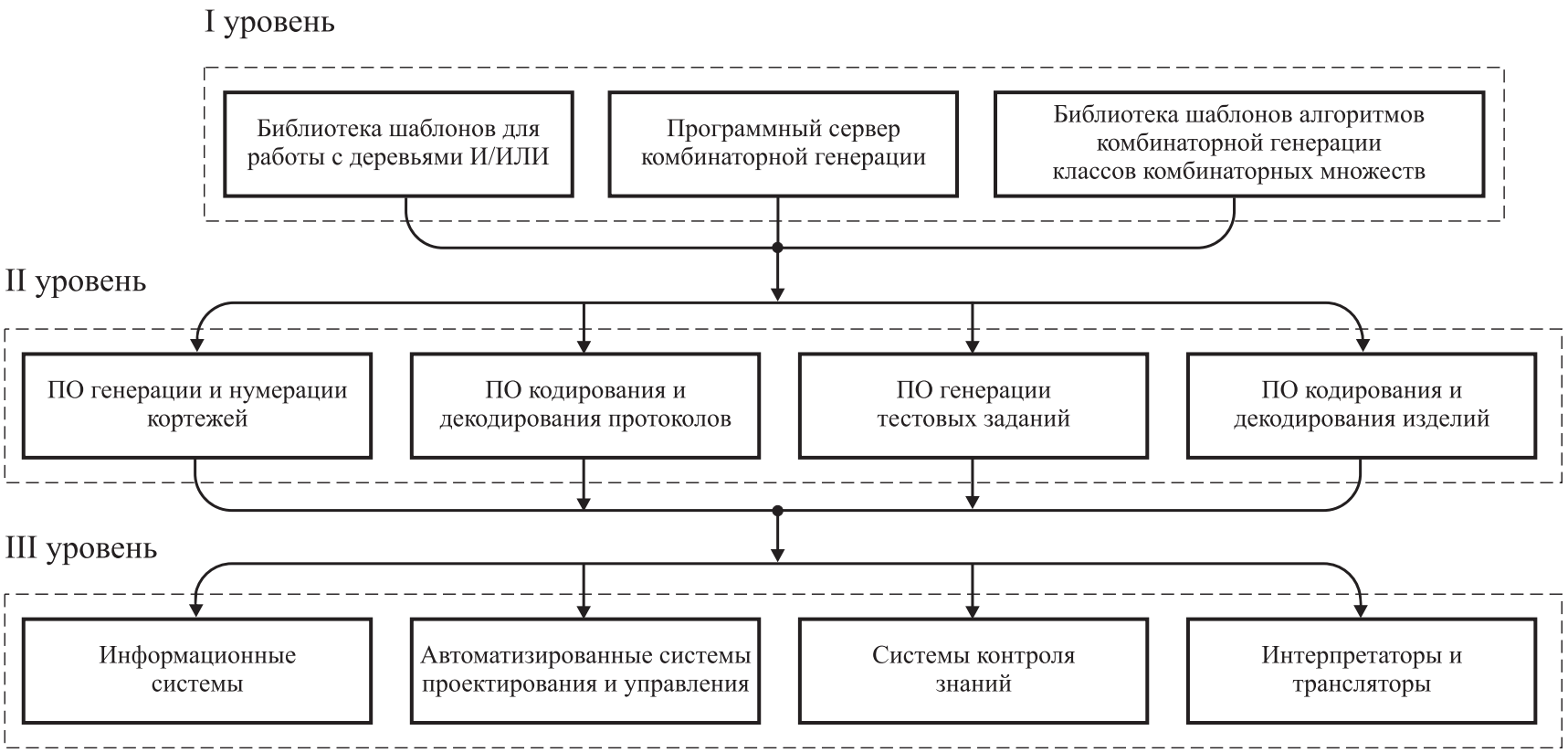


Рис. 12. Уровни программного комплекса комбинаторной генерации

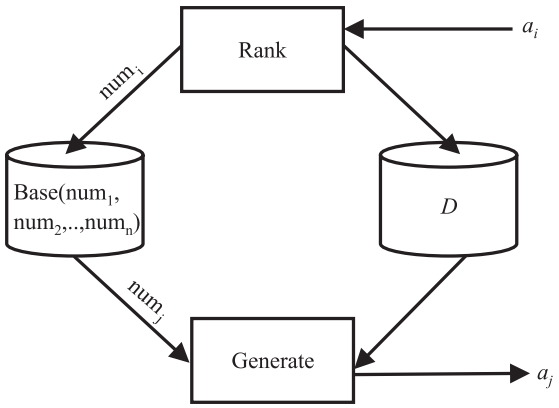


Рис. 13. Описание структуры базы данных

На рис. 12 представлены основные уровни программного комплекса комбинаторной генерации. На первом уровне представлено инструментальное программное обеспечение, включающее в себя:
1) библиотеку шаблонов для работы с деревьями И/ИЛИ;
2) программный сервер, обеспечивающий основные функции метода комбинаторной генерации;
3) библиотека шаблонов классов комбинаторных множеств, исследованных в диссертации.

Для реализации сервера были разработаны программные интерфейсы: для представления деревьев И/ИЛИ и вариантов, для представления и манипулирования системой стеков.

Второй уровень программного комплекса комбинаторной генерации обеспечивает реализацию решения задач при кодировании и декодировании реляционных баз данных и архивов, протоколов в автоматизированных системах управления, кодирования изделий в системах производства, построения генераторов тестовых заданий и тестовых примеров.

Третий уровень представляет конкретные разработки программных систем, внедренных в практику.

Для кодирования реляционной базы данных предложен метод, основанный на представлении доменов атрибутов реляционной таблицы в виде дерева И/ИЛИ (рис. 13). Тогда алгоритм $Rank(a)$ будет кодировать кортеж a, алгоритм $Generate(num_j)$ по номеру num_j генерировать этот кортеж. Так как значение $a \in A_1 \times A_2 \times \ldots \times A_n$ является комбинацией элементов из множеств $\{A_i\}_{i=1}^n$, то корень дерева будет И-узлом, имеющим n сыновей, каждый i-й сын соответствует множеству A_i, графическое изображение такого соответствия показано на рис. 14.

Общее число множества значений определяется по формуле:

$$\omega(T) = \prod_{i=1}^{n} \omega(A_i).$$

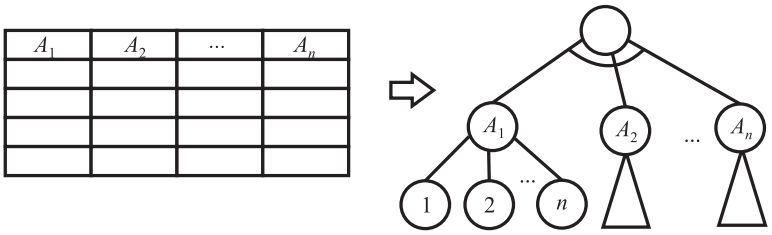


Рис. 14. Соответствие между таблицей и деревом И-ИЛИ

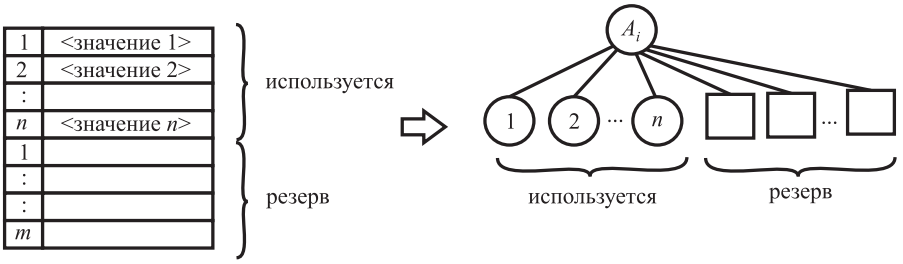


Рис. 15. Соответствие между справочником и деревом И-ИЛИ

Далее для каждого множества строится свое дерево И-ИЛИ.
В общем случае можно выделить следующие типы:
1. Множество значений представлено справочником.
2. Множество значений представлено числовым интервалом.
3. Множество значений представлено деревом И-ИЛИ.
Для представления множества уникальных объектов, которые используются в базе данных для некоторого домена, используется справочник. Справочник имеет две части, первая часть содержит пронумерованные уникальные объекты, вторая часть – резервная, предназначена для внесения новых объектов. Соответствие между справочником и деревом И-ИЛИ показано на рис. 15. Справочник представляется ИЛИ-узлом, а все сыновья являются элементами справочника. Тогда общее число вариантов дерева (или элементов множества) равно:

$$\omega(A_i) = n + m.$$

Использование данного подхода позволило создать защищенный архив удостоверяющего центра с историями сертификатов открытого ключа. Обобщенная структура архива показана на рис. 16. Основные модули и подсистемы: подсистема ввода сертификата; подсистема поиска, обеспечивающая контекстный поиск в архиве; подсистема управления доменами, обеспечивающая поиск и занесение заданных значений полей; модуль Rank, выполняющий формирование номера (кода) для данного сертификата; модуль Generate по номеру (коду) в базе данных формирует соответствующий кортеж сертификата; индексный файл для организации поиска (Idx); база данных (Base), хранящая коды кортежей; D(K) – домен, хранящий значения атрибута K(C,L,ST, O,OU,CN); S_i – кортеж сертификата; num_i – код сертификата.

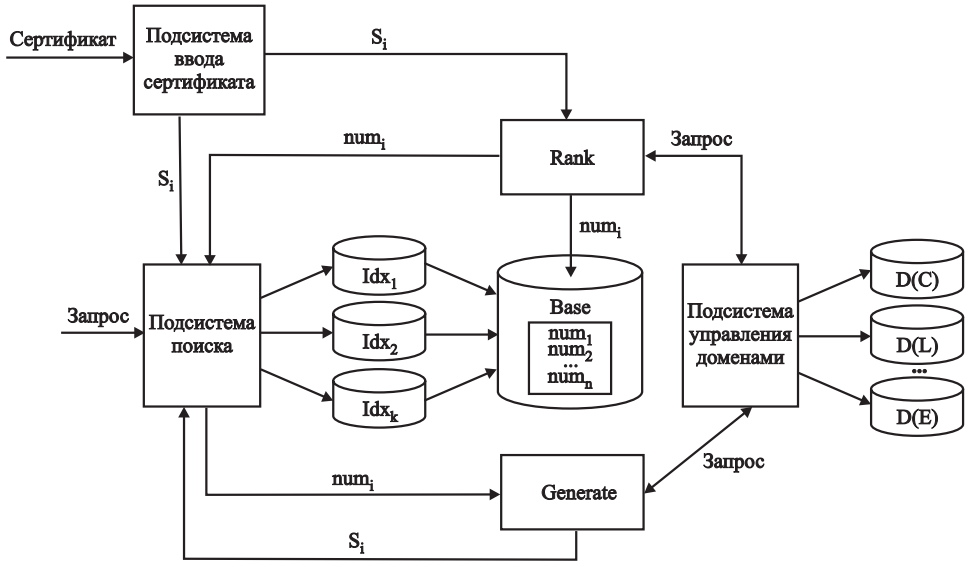


Рис. 16. Структура архива

Применение инструментального ПО при разработке и серийном производстве систем управления технологическими процессами и безналичными расчетами за нефтепродукты с применением магнитных, электронных и смарт - карт в ЗАО НПФ "Сибнефтекарт"позволило:

1) построить экономные алгоритмы представления и хранения реляционных баз данных, сокращающие на 18% объем баз данных;

2) разработать алгоритмы кодирования протоколов обмена данными между сервером приложений и автоматизированными рабочими местами системы управления технологическими процессами и безналичными расчетами за нефтепродукты. Разработанное инструментальное ПО было применено при создании системы контроля знаний в дистанционных технологиях, основанное на применении генераторов тестовых заданий. Обобщенная структура программных систем контроля знаний показана на рис. 17. Основные модели и программные системы, внедренные в реальную технологию дистанционного обучения:

1. Модели и алгоритмы генераторов тестовых заданий, основанные на применении деревьев И/ИЛИ.

2. Инструментальная система и технология, позволяющие строить компьютерные учебные программы с генераторами тестовых заданий.

3. Пакет генераторов тестовых заданий общим числом 130, число шаблонов – 6428.

4. Система проведения компьютерных экзаменов, реализованная в двух вариантах: локальном и сетевом на сайте Томского межвузовского центра дистанционного образования (www.tcde.ru).

5. Система проведения компьютерных контрольных работ.

6. База компьютерных контрольных работ и экзаменов, насчитывающая

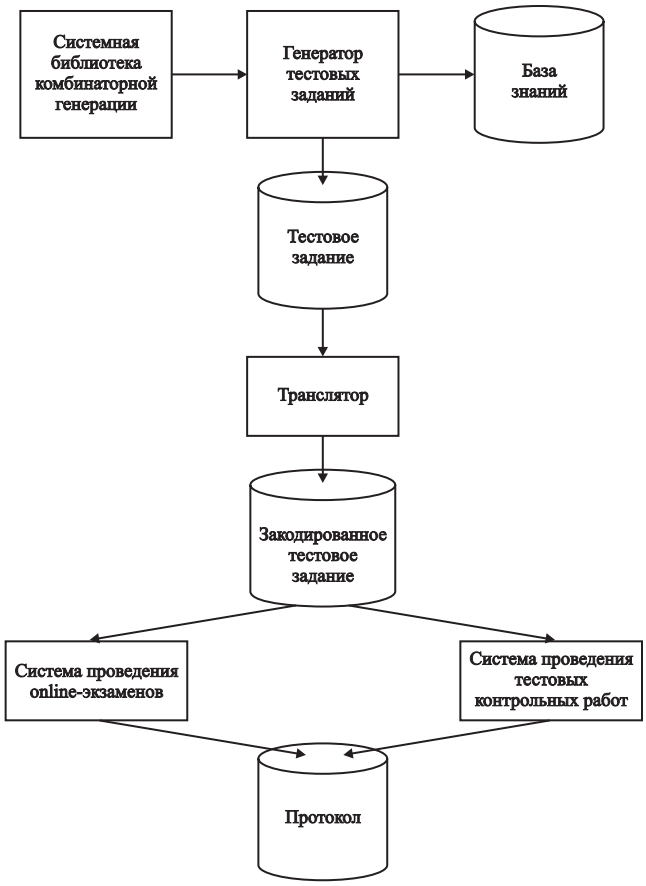


Рис. 17. Система тестирования уровня знаний на основе генерации тестовых заданий

1300 единиц, общим числом вопросов свыше 150000.

Эффект от внедрения следующий:

Внедрение перечисленных программных систем, пакета генераторов и базы компьютерных контрольных работ и экзаменов позволило решить следующие задачи:

• организовать действенный текущий и итоговый контроль в дистанционной технологии обучения;

• повысить надежность компьютерного контроля, сокращая объем негативных явлений (плагиат, шпаргалки, взлом аттестационных программ);

• разработчикам компьютерных учебных программ дало возможность оперировать не конкретным множеством вопросов, а моделями и базами знаний предметной области.

Перечисленные программные системы используются для 20 специальностей ТУСУРа, общим числом в 900 дисциплин, с числом студентов, обучающихся по дистанционной технологии, 7500. В год система проведения компьютерных контрольных работ тиражируется объемом 15 000 компакт дисков. В год обрабатывается 120000 протоколов компьютерных экзаменов и контрольных работ.

Заключение. Совокупность полученных в настоящей диссертации на-

учных результатов составляет основы решения важной проблемы построения программного обеспечения комбинаторной генерации, внедрение которого вносит значительный вклад в построение и развитие инновационной экономики страны. Получены следующие результаты:

1. Показано, что между функцией $f(n) \in A = \{N, +, \times, R\}$, где N – множество натуральных чисел, R – оператор примитивной рекурсии, и схемой рекурсивной композиции $R_n(d_g, d_h)$ деревьев И/ИЛИ существует взаимнооднозначное соответствие, которое обеспечивает следующие свойства:
- при фиксации параметра $n = n^*$ по схеме рекурсивной композиции R_{n^*} строится дерево И/ИЛИ, число вариантов которого равно значению $f(n^*)$;
- если для некоторого комбинаторного множества получена схема рекурсивной композиции R_n, то мощность этого множества выражается функцией $f(n)$;
- между комбинаторным множеством, мощность которого равна $f(n^*)$ и множеством вариантов дерева И/ИЛИ R_{n^*} существует биекция.

2. Если для некоторого комбинаторного множества E_n существует дерево И/ИЛИ R_n, то:
– Однозначно строится автомат последовательной генерации$\{B, M, P_{Fst}, P_{Nxt}\}$, где B – начальный магазинный символ, M – множество магазинных символов, P_{Fst} – правила для операции First, P_{Nxt} – правила для операции Next. Временная сложность алгоритма последовательной генерации равна

$$Op(z) = \begin{cases} \sum_{i=1}^{k} Op(s_i) + 1, & \text{для ИЛИ-узла;} \\ \sum_{j=1}^{k-1} [Op(s_j) \cdot \prod_{i=j+1}^{k} \omega(s_i)] + Op(s_k) + 1, & \text{для И-узла;} \\ 1, & \text{для листа.} \end{cases}$$

– Строится алгоритм генерации элемента комбинаторного множества по номеру.
– Строится алгоритм нумерации элементов комбинаторного множества.
– Временная сложность алгоритма генерации по номеру и нумерации равна:

$$Op_z = \begin{cases} \sum_{i=1}^{n_z} [(n_z - i)\omega_i^z + Op_{s_i}], & \text{для ИЛИ-узла;} \\ (n_z - 1 + \sum_{i=1}^{n_z} \frac{Op_{s_i}}{\omega_i}) \cdot \omega_z, & \text{для И-узла;} \\ 0, & \text{для листа.} \end{cases}$$

3. Применяя метод построения алгоритмов комбинаторной генерации и приближенные алгоритмы решения уравнений, были разработаны оригинальные алгоритмы генерации сочетаний и элементов множеств, заданных числами Фибоначчи. Получено следующее выражение для числа разбиений натурального числа n на части не менее m:

$$p_m(n) = p_m(n - m) + p_{m+1}(n).$$

Получены оригинальные алгоритмы генерации и нумерации композиций и разбиений натурального числа n.

4. Эффективное применение разработанных методов убедительно показано на классических комбинаторных множествах: классов перестановок, множеств, описываемых числами Фибоначчи, Каталана, Стирлинга, Сильвестра.

5. Разработаны схемы рекурсивных композиций деревьев И/ИЛИ для построения алгоритмов генерации и нумерации классов деревьев: двоичных деревьев высотой не более n, заданным числом узлов и высотой, полных двоичных деревьев, АВЛ-деревьев, 2-3-деревьев, 2-3-4-деревьев, красно-черных деревьев, цветных двоичных деревьев, деревьев операций.

6. Развит подход для построения алгоритмов комбинаторной генерации для классов корневых деревьев, основанный на процедуре полного разбиения, и получены:

– Алгоритмы генерации t-арных и полных t-арных деревьев с заданным числом узлов.

– Выдвинута гипотеза, что для разложений справедлива формула:

$$\prod_{i=1}^{t} K_{t+k}(\lambda_i) = \binom{(k+t)n+k}{n} \frac{k+1}{(k+t-1)n+k+1},$$

где t – число частей разложения числа n, $t > 2$, $\sum_{i=1}^{t} \lambda_i = n$, k – смещение относительно большего разложения $(t+k)$.

– Алгоритм генерации упорядоченных корневых деревьев.

– Функция, связывающая номер композиции и значение числа упорядоченных корневых деревьев относительно процедуры полного разбиения для данной композиции.

$$T(i) = \begin{cases} 1, & i = 0, \\ T(i - 2^{\lfloor \log(i) \rfloor}) \frac{4k+2}{k+2}, & i > 0. \end{cases}$$

где $k = \begin{cases} \lfloor \log(i) \rfloor - \lfloor \log(2^{\lfloor \log(i) \rfloor + 1} - i - 1) \rfloor - 1, & 0 \le i < 2^{\lfloor \log(i) \rfloor + 1}, \\ \lfloor \log(i) \rfloor, & i = 2^{\lfloor \log(i) \rfloor + 1} - 1. \end{cases}$

– Алгоритм и функция для генерации неупорядоченных корневых деревьев относительно процедуры полного разбиения, основанного на разбиениях натурального числа n.

7. Разработаны схемы рекурсивных композиций для языков Дика, Моцкина, Лукасевича и арифметических выражений. Показано, что для построения алгоритмов комбинаторной генерации для выражений языков, необходимо иметь язык, описываемый контекстно-свободной грамматикой с однозначным выводом.

8. Разработано универсальное программное обеспечение направленное на:

1) проектирование и исследование алгоритмов комбинаторной генерации;

2) создание генераторов тестовых заданий для систем контроля знаний студентов;

3) проектирование программных систем сжатия и кодирования в автоматизированных и информационных системах. Данное программное обеспечение внедрено на производственном объединении «Контур», на ЗАО НПФ «Сибнефтекарт», в дистанционные технологии обучения студентов: Томского государственного университета систем управления и радиоэлектроники по 38 специальностям, в Кемеровском технологическом институте пищевой промышленности по трем специальностям, в Югорском государственном университете.

Список публикаций

Монографии

1. Кручинин, В. В. Комбинаторика композиций и ее приложения / В. В. Кручинин. — Томск: Изд-во «В-Спектр», 2010. — 156 с.

2. Кручинин, В. В. Методы построения алгоритмов генерации и нумерации комбинаторных объектов на основе деревьев И/ИЛИ / В. В. Кручинин. — Томск: Изд-во «В-Спектр», 2007. — 200 с.

3. Кручинин, В. В. Генераторы в компьютерных учебных программах / В. В. Кручинин. — Томск: Изд-во Том. ун-та, 2003. — 200 с.

4. Кручинин, В. В. Разработка компьютерных учебных программ / В. В. Кручинин. — Томск: Изд-во Том. ун-та, 1998. — 211 с.

Статьи в журналах из списка ВАК

1. Кручинин, В. В. Подходы к созданию защищенного архива на основе разделения секрета / В. В. Кручинин, А. А. Шелупанов // Доклады ТУСУР. — 2008. — № 2 часть 1. — С. 67–72.

2. Кручинин, В. В. Представление множеств деревьями И/ИЛИ / В. В. Кручинин // Доклады ТУСУР. — 2008. — № 2(17). — С. 107–112.

3. Кручинин, В. В. Алгоритмы генерации и нумерации композиций и разбиений натурального числа n / В. В. Кручинин // Доклады ТУСУР. — 2008. — № 2(17). — С. 113–119.

4. Кручинин, В. В. Язык описания генераторов комбинаторных множеств / В. В. Кручинин, А. В. Титков // Известия Томского политехнического университета. — 2008. — Т. 312. — № 5. Управление, вычислительная техника и информатика.– С.89—93

5. Кручинин, В. В. Рекурсивные композиции деревьев и их свойства / В. В. Кручинин // Доклады ТУСУР. — 2007. — Т. 16. — С. 74–80.

6. Кручинин, В. В. Подход к созданию баз данных, основанный на алгоритмах генерации и идентификации кортежей / В. В. Кручинин, С. Л. Хо-

мич, А. В. Титков// Известия Томского политехнического университета. — 2006. — Т. 309, № 8. — С. 28–32.

7. Кручинин, В. В. Использование деревьев И/ИЛИ для генерации вопросов и задач / В. В. Кручинин // Вестник ТГУ. — 2004. — № 284, серия «Математика. Кибернетика. Информатика». — С. 185–189.

8. Кручинин, В. В. Программные генераторы (обзор) / В. В. Кручинин // Доклады ТУСУР. — 2004. — № 2(10). — С. 64–68.

10. Компьютерный учебник «ТМЦДО. Высшая математика-1» / С. И. Борисов, А. В. Долматов, В. В. Кручинин, В. А. Томиленко // Открытое образование. — 2004. — № 3. — С. 12–17.

11. Кручинин, В. В. Применение генераторов в компьютерных технологиях обучения / В. В. Кручинин, С. И. Борисов // Интеграция образования. — 2004. — № 4. — С. 116–121.

12. Кручинин, В. В. Модели и алгоритмы генерации задач в компьютерном тестировании / В. В. Кручинин, Ю. В. Морозова // Известия Томского политехнического университета. — 2004. — № 5. — С. 127–131.

13. Кручинин, В. В. Алгоритмы и перечислительные свойства деревьев И/ИЛИ / В. В. Кручинин, Вестник ТГУ, №284, серия «Математика. Кибернетика. Информатика», декабрь 2004.– С.181–184.

14. Кручинин, В. В. Система тестирования, основанная на генерации вопросов и тестовых заданий / В. В. Кручинин, М. Ф. Молочко // Открытое образование. — 2004. — № 4. — С. 30–35.

15. Кручинин, В. В. Использование деревьев И/ИЛИ для перечисления выражений контекстно-свободных языков / В. В. Кручинин // Вестник Томского государственного педагогического университета. — 2004. — № 6(43). — С. 84–88.

16. Жуков, В. К. Новые подходы к организации контроля знаний в вузе / В. К. Жуков, В. В. Кручинин // Известия МАН ВШ. — 2004. — № 2(28). — С. 113–118.

17. Кручинин, В. В. Число разбиений натурального числа n на части, каждая из которых не менее m// Матем. заметки . — 2009. — № 4(86). — С. 538–542.

18. Кручинин, В. В. Алгоритмы генерации корневых деревьев на основе процедуры полного разбиения// ПДМ. Приложение 1 . — 2009. — С. 99–101.

19. Кручинин, В. В. Метод кодирования информационных объектов на основе деревьев И/ИЛИ / В. В. Кручинин Б. А. Люкшин // Доклады ТУСУР. — 2010. — № 1(21), ч. 1. — С. 170–172.