На правах рукописи

Титков Антон Вячеславович

СИСТЕМА ПОСТРОЕНИЯ ГЕНЕРАТОРОВ КОМБИНАТОРНЫХ МНОЖЕСТВ НА ОСНОВЕ ДЕРЕВЬЕВ И/ИЛИ

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

Автореферат
Диссертации на соискание ученой степени
кандидата технических наук

Томск – 2010

Работа выполнена в Томском государственном университете систем управления и радиоэлектроники (ТУСУР)

Научный руководитель — кандидат технических наук, доцент Кручинин Владимир Викторович

Официальные оппоненты — доктор технических наук, профессор Погребной Владимир Кириллович

кандидат технических наук, доцент Осипов Александр Леонидович

Ведущая организация — Новосибирский государственный технический университет

Защита состоится «14» апреля 2010 г. в 15.00 на заседании совета по защите докторских и кандидатских диссертаций Д 212.269.06 при ГОУ ВПО «Томский политехнический университет» по адресу: 634034, г. Томск, ул. Советская, 84/3, институт «Кибернетический центр».

С диссертацией можно ознакомиться в научно-технической библиотеке ГОУ ВПО «Томский политехнический университет» по адресу: г. Томск, ул. Белинского, 55.

Автореферат разослан «10» марта 2010 г.

Ученый секретарь
совета по защите
докторских и кандидатских
диссертаций Д 212.269.06,
к.т.н., доцент — Сонькин М. А.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность работы. Программные генераторы комбинаторных множеств применяются в различных системах при решении разнообразных задач. С их помощью моделируются случайные события, тестируются знания и программные комплексы, они применяются для нумерации товаров и кодирования информации. Требования к разработке генераторов постоянно возрастают.

За время существования информатики как дисциплины комбинаторная генерация стала самостоятельным, динамично развивающимся направлением. Комбинаторной генерации посвящено множество работ. Основной вклад в её развитие внесли Д. Кнут, Э. Рейнгольд, Ф. Раски. С развитием комбинаторной генерации развивались и методы построения алгоритмов генерации. Самый распространённый, эмпирический метод, стал уступать по эффективности методам построения алгоритмов генерации, претендующим на некоторую универсальность (Д. Крехер, Е. Баргутччи, Ф. Флажоле, К. Мартинец). Такие методы позволяют строить алгоритмы генерации для целого ряда предметных областей, однако сильно к ним привязаны и обладают рядом существенных ограничений, что затрудняет автоматизацию процесса построения программных генераторов на их основе.

Сравнительно недавно появился метод построения алгоритмов генерации и нумерации комбинаторных множеств на основе деревьев И/ИЛИ, предложенный В.В. Кручининым. Метод заключается в представлении множества, элементы которого требуется генерировать, в виде дерева И/ИЛИ. В этом случае дерево И/ИЛИ является генератором элементов данного множества. Этот метод обладает широкой областью применения и не имеет таких существенных ограничений, как другие методы.

Темп развития инструментальных средств построения генераторов не соответствует теоретическим достижениям, полученным в области комбинаторной генерации. На сегодняшний день создано большое количество библиотек алгоритмов комбинаторной генерации, реализованных на различных языках программирования. Однако данные библиотеки не обеспечивают системного подхода при построении генераторов, т.к. обладают ограниченным функционалом и не используют главного преимущества универсальных методов построения алгоритмов генерации – единого подхода к построению программных генераторов. Для некоторых универсальных методов существуют программные наработки в виде специализированных библиотек для пакетов математического моделирования (Combinatorica для Mathematica, MuPAD-Combinat для MuPAD). Использование таких средств в прикладном программном обеспечении затруднено в связи с тем, что данные библиотеки разработаны в первую очередь для научных исследований. Пакеты математического моделирования, на которых основаны данные библиотеки, не предназначены для использования в разработке программного обеспечения (ПО), что является серьезным недостатком, т.к. генераторы обычно используются как часть программных комплексов и систем. Таким образом, актуальной является задача разработки программных систем построения

генераторов комбинаторных множеств на основе универсальных методов построения алгоритмов генерации и нумерации.

Целью диссертационной работы является разработка и исследование системы построения генераторов комбинаторных множеств на основе деревьев И/ИЛИ.

Задачи исследования:

1) анализ методов и инструментальных средств построения программных генераторов, выявление требований к системе построения генераторов комбинаторных множеств;
2) формализация процесса построения генераторов комбинаторных множеств на основе деревьев И/ИЛИ;
3) разработка математического, алгоритмического и программного обеспечения системы построения генераторов комбинаторных множеств;
4) разработка технологии построения генераторов на основе деревьев И/ИЛИ;
5) экспериментальные исследования системы построения генераторов комбинаторных множеств на примерах практических задач.

Методы исследования. В работе использованы понятия и методы системного анализа, теории множеств, теории алгоритмов, лямбда-исчисления, формальных грамматик, теории объектно-ориентированного и функционального программирования.

Научная новизна:

1) Впервые определены теоретико-множественные операции на деревьях И/ИЛИ, введены операции И-произведения и ИЛИ-объединения. Показано, что выявленное множество алгебраических операций на деревьях И/ИЛИ и система лямбда-исчисления позволяют формализовать процесс построения деревьев И/ИЛИ.
2) Создан оригинальный язык описания генераторов GIL, реализующий метод построения алгоритмов генерации и нумерации комбинаторных множеств на основе деревьев И/ИЛИ.
3) Разработана архитектура универсального STL-совместимого контейнера для представления древовидных структур данных, на основе которой реализован интерпретатор языка GIL.
4) Предложена технология построения программных генераторов, обеспечивающая разработку генераторов для широкого круга предметных областей.

Основные положения, выносимые на защиту:

1) Теоретико-множественные операции на деревьях И/ИЛИ позволяют формализовать процесс построения деревьев И/ИЛИ.
2) Функциональный язык программирования GIL обладает удобными выразительными средствами для описания программных генераторов на основе деревьев И/ИЛИ.

3) Архитектура универсального STL-совместимого контейнера для представления древовидных структур данных позволяет создавать классы деревьев с заданными свойствами.
4) Технология построения программных генераторов обеспечивает снижение сроков разработки генераторов и объема их программного кода.

Практическая ценность. Результаты диссертационной работы позволяют:

1) Исследовать алгоритмы генерации и нумерации комбинаторных множеств на основе деревьев И/ИЛИ в разработанной системе построения генераторов.
2) Разрабатывать генераторы комбинаторных множеств на основе библиотеки комбинаторных алгоритмов и языка построения генераторов GIL для различных предметных областей.
3) Применять разработанные модели генераторов на основе грамматик, представленных в виде дерева И/ИЛИ, для построения генераторов тестовых заданий систем контроля знаний и генераторов тестовых данных систем тестирования программного обеспечения.
4) Разработанная библиотека комбинаторных алгоритмов, включающая библиотеку деревьев и библиотеку алгоритмов комбинаторной генерации, расширяет возможности комбинаторной генерации библиотеки STL, позволяет создавать контейнеры деревьев с заданными свойствами.

Достоверность результатов работы. Достоверность полученных результатов достигается применением научных основ системного анализа, теории множеств, лямбда-исчисления, а также широким внедрением результатов диссертационной работы в практику.

Внедрение результатов. Разработанное программное обеспечение внедрено в технологии дистанционного обучения Томского политехнического университета, Томского государственного университета систем управления и радиоэлектроники, Новосибирского государственного университета экономики и управления, применяется для разработки и тестирования программного обеспечения компаниями «Эль Софт» и «Томск Софт».

Апробация результатов. Результаты исследований представлены в 12 публикациях различного уровня. Две из них опубликованы в рекомендованных ВАК изданиях. Представлено 15 докладов на различных научно-методических и научно-практических конференциях, среди которых:

1) Международная научно-методическая конференция «Современное образование: проблемы и перспективы в условиях перехода к новой концепции образования», Томск (2009);
2) Международная научная конференция «Космос, астрономия и программирование» (Лавровские чтения), Санкт-Петербург (2008);
3) Международная научно-методическая конференция «Современное образование: вызовам времени – новые подходы», Томск (2008);

4) VII межрегиональная конференция «Информационные технологии и решения для «Электронной России», Ханты-Мансийск (2008);
5) XIV Всероссийская научно-методическая конференция «Телематика'2007», Санкт-Петербург (2007);
6) XI Международная научно-практическая конференция студентов и молодых ученых «Современные техника и технологии», Томск (2005);
7) Всероссийская научно-техническая конференция студентов и молодых ученых «Научная сессия ТУСУР-2005», Томск (2005).

Работа выполнена в рамках ведомственной научной программы «Развитие научного потенциала высшей школы», подпрограммы 1 «Фундаментальные исследования», фундаментальных НИР, выполняемых в Томском государственном университете систем управления: 1.3.09 «Создание и исследование методов и алгоритмов комбинаторной генерации».

Личный вклад. Основные результаты работы, алгоритмы автоматизации процесса построения генераторов на основе деревьев И/ИЛИ, язык построения генераторов GIL и его интерпретатор, библиотека комбинаторных алгоритмов, примеры практического применения системы построения генераторов комбинаторных множеств получены лично автором.

Структура и объем диссертации. Работа состоит из введения, четырех глав, заключения, списка литературы из 67 наименований, одного приложения. Общий объем диссертации 105 страниц, в том числе 23 рисунка, 4 таблицы.

СОДЕРЖАНИЕ РАБОТЫ

Во введении кратко представлено основное направление исследований, обозначены проблемы комбинаторной генерации, показана актуальность создания системы построения генераторов на основе деревьев И/ИЛИ. Исходя из этого, сформулированы актуальность, цели и задачи исследования. Обозначены методы исследования и элементы научной новизны работы. Сформулированы практическая значимость работы, основные защищаемые положения, обоснована их достоверность. Представлены результаты внедрения, апробации работы и опубликованных материалов по теме исследования.

Первая глава посвящена обзору и анализу методов построения алгоритмов генерации и нумерации комбинаторных множеств, систем построения программных генераторов. На основе этого анализа выбирается направление исследований, уточняется постановка задачи и требования к результатам исследований.

При создании программных генераторов комбинаторных множеств ведущую роль играет алгоритм генерации. Алгоритмы генерации могут быть получены эмпирически, путем анализа множества и выявления закономерностей между его элементами, либо на основе одного из методов построения алгоритмов генерации и нумерации комбинаторных множеств, претендующих на некоторую универсальность. Используемый метод накладывает ограничения на функциональность генератора и способы его

реализации. Так, одни методы позволяют производить последовательную генерацию элементов множества и генерацию элементов по номеру. Другие методы позволяют нумеровать элементы рассматриваемого множества, однако требуют дополнительных условий, например, наличие производящей функции для генерируемого множества. Существуют методы, применяемые к широкому кругу предметных областей и не накладывающие существенных ограничений на генерируемое множество, например, метод, основанный на представлении множества в виде дерева И/ИЛИ.

В настоящее время существуют различные инструментальные средства построения программных генераторов. Любой язык программирования подходит для реализации алгоритмов генерации, полученных эмпирическим путем. Однако, реализация алгоритмов, полученных универсальными методами, на таких языках не столь эффективна, как на специализированных средствах, для этого предназначенных. Большинство систем, основанных на универсальных методах построения алгоритмов генерации, реализованы в виде библиотек для пакетов математического моделирования, что затрудняет их применение в прикладном ПО.

Анализ существующих методов построения алгоритмов генерации и инструментальных средств создания генераторов показал, что системы построения генераторов должны удовлетворять следующим требованиям. Такие системы должны:

- позволять осуществлять последовательную генерацию элементов комбинаторного множества;
- позволять осуществлять нумерацию всех элементов комбинаторного множества;
- позволять осуществлять генерацию элементов комбинаторного множества по номеру;
- позволять осуществлять случайную генерацию элементов комбинаторного множества;
- обладать минимальным порогом вхождения, т.е. пользователь должен иметь минимальный уровень знаний, чтобы начать работу с системой;
- обладать богатым функционалом, ориентированным на разработку программных генераторов;
- позволять встраивать полученные на выходе системы генераторы в различное программное обеспечение.

В представленном обзоре инструментальных средств построения генераторов ни одна из систем не удовлетворяет всем требованиям.

Вторая глава посвящена анализу метода построения алгоритмов генерации и нумерации, основанного на деревьях И/ИЛИ, с целью формализовать процесс построения деревьев И/ИЛИ, что необходимо для автоматизации процесса построения генераторов на основе этого метода.

Деревом И/ИЛИ называется дерево, состоящее из узлов 2-х типов: И-узел и ИЛИ-узел. На рисунке 1 изображено схематическое представление узлов дерева И/ИЛИ.

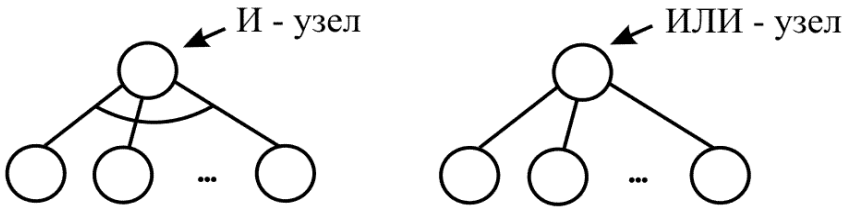

Рисунок 1. Схематическое представление узлов дерева И/ИЛИ

Вариантом дерева И/ИЛИ называется дерево, полученное из данного путём отсечения всех дуг, кроме одной, у ИЛИ-узлов. Корнем варианта будет корень исходного дерева. На рисунке 2 показано дерево И/ИЛИ и все его варианты.

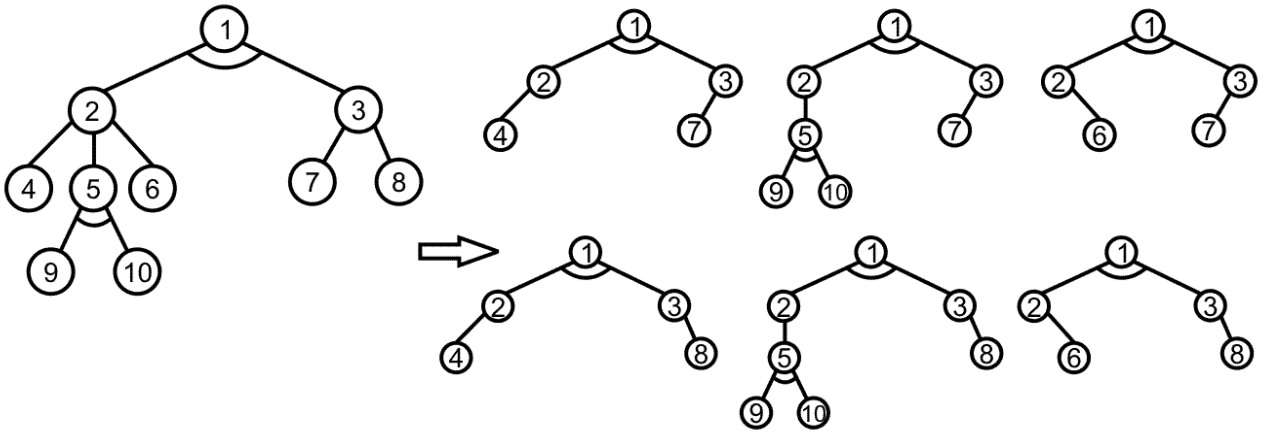

Рисунок 2. Дерево И/ИЛИ и все его варианты

Мощностью дерева И/ИЛИ называется количество вариантов, которое оно содержит.

Автор метода, Кручинин В.В., в своей работе показал, что между вариантом дерева И/ИЛИ и элементом кодируемого множества существует однозначное биективное отображение.

Анализ деревьев И/ИЛИ как алгебраической структуры показал, что на деревьях И/ИЛИ определены теоретико-множественные операции $|T|$, $\cup$, $\cap$, $\setminus$, $\times$. Действительно, между вариантом дерева И/ИЛИ и элементом множества есть биективная связь:

$A \leftrightarrow TA$, где TA – дерево И/ИЛИ для множества A;

$B \leftrightarrow TB$, где TB – дерево И/ИЛИ для множества B;

$a \leftrightarrow wTA$, где $a \in A$, $wTA \in TA$ - множество вариантов дерева TA;

$b \leftrightarrow wTB$, где $b \in B$, $wTB \in TB$ - множество вариантов дерева TB,

следовательно,

$C = A\ op\ B$, $C \leftrightarrow TC$, если $TC = TA\ op\ TB = wTA\ op\ wTB$, где op - бинарная операция.

Например, для операции объединения можно записать:

$C = A \cup B$, $TC \leftrightarrow C$, где $TC = TA \cup TB = wTA \cup wTB$.

Таким образом, результатом объединения двух деревьев И/ИЛИ является дерево И/ИЛИ, вариантами которого будет объединение вариантов двух

исходных деревьев. С помощью таких же рассуждений все остальные операции, определенные над множествами, переносятся на деревья И/ИЛИ.

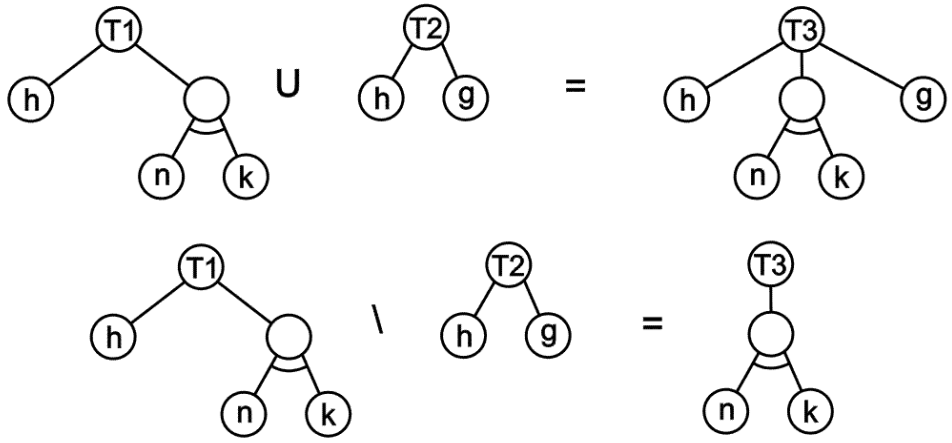

Рисунок 3. Объединение и разность двух деревьев И/ИЛИ.

Приведенные на рисунке 3 операции возможны только в том случае, если структура деревьев имеет идентичную структуру. При несовпадении структур деревьев произвести операции над ними таким способом невозможно. Анализ деревьев И/ИЛИ показал, что можно производить эквивалентные преобразования деревьев И/ИЛИ путем добавления и удаления узлов таким образом, чтобы не менялась мощность дерева. Поэтому операцию объединения предлагается осуществлять ИЛИ-узлом, а операцию декартового произведения И-узлом, сыновьями которых становятся деревья операнды (рисунок 4). Такие операции будем называть ИЛИ-объединением ($\overset{or}{\cup}$) и И-произведением ($\overset{and}{\times}$).

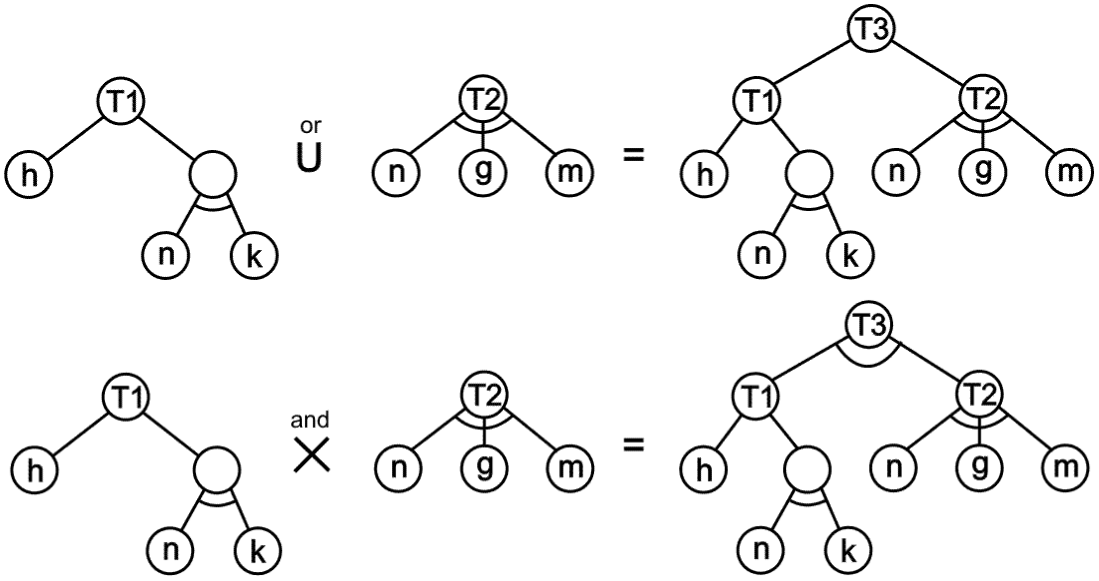

Рисунок 4. ИЛИ-объединение и И-произведение деревьев И/ИЛИ.

Для формализации процесса построения деревьев И/ИЛИ предлагается использовать формальную систему лямбда-исчисления. Использование лямбда-исчисления для формализации процесса построения деревьев И/ИЛИ дает возможность применять для автоматизации процесса построения генераторов

на деревьях И/ИЛИ мировой опыт в области реализации систем программирования, основанных на лямбда-исчислении.

В.В. Кручинин для построения деревьев И/ИЛИ, заданных выражениями, вводит операции композиции и рекурсивной композиции деревьев И/ИЛИ. Композиция S_i определяется как построение дерева D_3 путем замены листа $l_i \in D_1$ на корень дерева D_2: $D_3 = S_i(D_1, D_2)$. Применяя каррирование и β-редукцию лямбда-исчисления, операцию композиции деревьев И/ИЛИ можно записать следующим образом:

$$\lambda x.\ \lambda y.\ (\lambda z.\ x)\ y,$$

где x – выражение построения дерева И/ИЛИ, свободно содержащее z;

y – выражение построения дерева И/ИЛИ, которое требуется подставить в выражение x вместо переменной z.

Для выражения $D_3 = S_i(D_1, D_2)$ дерево D_1 является x, D_2 – y, l_i – переменной z:

$$S_i(D_1, D_2) = (\lambda x.\ \lambda y.\ (\lambda z.\ x)\ y)\ D_1\ D_2\ l_i$$

Любая алгебраическая операция на деревьях И/ИЛИ записывается в виде:

$$D_1\ op\ D_2 = (\lambda x.\ \lambda y.\ x\ op\ y)\ D_1\ D_2$$

Операция рекурсивной композиции выражается через Y-комбинатор:

$$Y = \lambda f.\ (\lambda x.\ f(x\ x))\ (\lambda x.\ f(x\ x))$$

$$S = (\lambda x.\ \lambda y.\ (\lambda z.\ x)\ y)$$

$$Sn = \lambda f\ m\ n.\ (\ iszero\ m \rightarrow 0\ )\ |\ f\ (m\text{-}1)\ n$$

$$Sr = Y\ Sn$$

В главе приводятся примеры построения деревьев И/ИЛИ на основе формальной системы лямбда-исчисления и теоретико-множественных операций на деревьях И/ИЛИ для выражений из различных алгебр и языков. Показано, что построить дерево И/ИЛИ для множеств, заданных выражениями или правилами преобразования других множеств, возможно только в том случае, если для каждого операнда в выражении можно построить дерево И/ИЛИ и все операции в выражении имеют отображения в алгебре $\{T, \overset{or}{\cup}, \overset{and}{\times}, \overset{tree}{-}, R\}$, где T – дерево И/ИЛИ, $\overset{or}{\cup}$ – операция ИЛИ-объединения, $\overset{and}{\times}$ – операция И-произведения, $\overset{tree}{-}$ – операция удаления ветви дерева И/ИЛИ, R – оператор рекурсии.

Третья глава посвящена разработке системы построения генераторов комбинаторных множеств.

Процесс построения генераторов комбинаторных множеств на основе деревьев И/ИЛИ можно разделить на 3 части:

1) реализация обобщённых алгоритмов деревьев И/ИЛИ (нумерации и генерации вариантов);
2) реализация алгоритма построения дерева И/ИЛИ;
3) получение варианта дерева И/ИЛИ по объекту в алгоритмах нумерации и получение объекта по варианту в алгоритмах генерации.

Первые две задачи процесса построения генераторов можно автоматизировать, решение третьей зависит от конкретного вида объекта и полностью ложится на плечи программиста.

Систему предлагается реализовать как совокупность библиотеки комбинаторных алгоритмов и специализированного языка построения генераторов. Такой подход обеспечит максимальную универсальность системы, позволит решать задачу преобразования варианта дерева И/ИЛИ в элемент множества и обратную задачу несколькими способами, в том числе и средствами языка построения генераторов. Структурная схема системы приведена на рисунке 5.

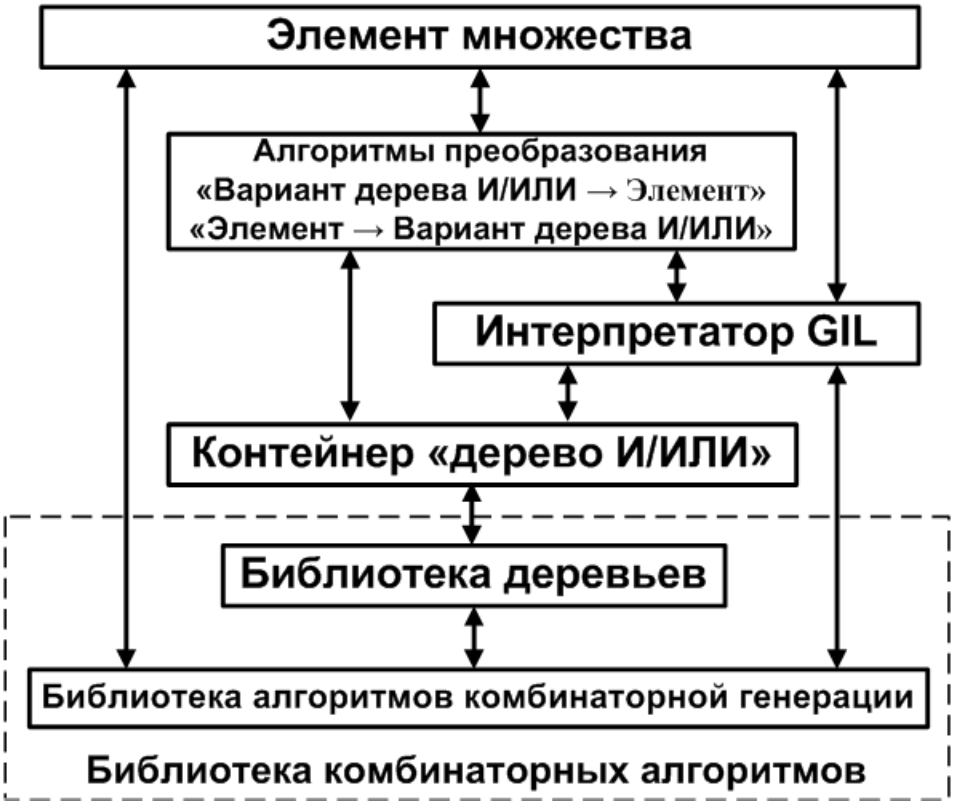

Рисунок 5. Структурная схема системы построения генераторов комбинаторных множеств

Для реализации системы выбран язык программирования C++, т.к. он является на сегодняшний день самым популярным языком программирования для различного программного обеспечения и поддерживает различные парадигмы программирования.

Библиотека комбинаторных алгоритмов является расширением библиотеки STL и реализована средствами обобщенного программирования, что позволяет применять её для различных типов данных.

Библиотека комбинаторных алгоритмов состоит из следующих модулей:

1) Библиотеки алгоритмов комбинаторной генерации, включающей функции для нахождения основных комбинаторных величин (вычисление факториала, подсчет количества сочетаний, перестановок

и т.д.) и функции-генераторы комбинаторных объектов, таких как сочетания, перестановки, разбиения и.т.д.

2) Библиотеки деревьев, включающей реализацию контейнеров «дерево», «дерево И/ИЛИ» и алгоритмы на деревьях И/ИЛИ.

Функционал комбинаторной генерации библиотеки STL сильно ограничен и включает в себя только последовательную генерацию перестановок. Библиотека алгоритмов комбинаторной генерации расширяет возможности STL алгоритмами генерации перестановок, сочетаний, разбиений, разложений по номеру, бинарных деревьев.

Библиотека STL предоставляет все возможные контейнеры, кроме деревьев. Это связано с тем, что при решении различных задач к деревьям предъявляются разнообразные требования и выработать обобщенный шаблон этой структуры данных достаточно сложно. Несмотря на это, существуют несколько реализаций контейнеров деревьев, таких как библиотека Kasper Peeters, библиотека STL+ Andy Rushton, библиотека TCL, Mitchel Haas. Был произведен анализ этих библиотек по следующим критериям:

- универсальность интерфейсов классов;
- совместимость с STL;
- возможность расширения и модификации;
- возможность написания источников данных для узлов дерева;
- возможность наследования от базового класса дерева различных реализаций деревьев.

Всем требованиям не удовлетворяет ни одна из рассмотренных библиотек.

На основе критериев и анализа существующих библиотек была разработана оригинальная библиотека деревьев, позволяющая реализовывать контейнеры деревьев с заданными свойствами. Основные идеи реализации были следующие:

1) Логика отдельных модулей контейнера дерева выделяется в отдельные сущности с заданными интерфейсами:
   а) Узел дерева. Осуществляет логику хранения данных узла и предоставляет доступ к ним остальным классам библиотеки.
   б) Итераторы. Предоставляют доступ к данным узла дерева и позволяют осуществлять обход узлов дерева.
   в) Класс-контейнер. Реализует логику работы с древовидными структурами: вставка, удаление, замена узлов; доступ к итераторам дерева.
   г) Аллокаторы. Позволяют реализовать различные схемы выделения динамической памяти для данных, хранящихся в дереве.
2) Для реализации дерева с заданными свойствами необходимо реализовать только те компоненты, которые отвечают за эти свойства.

В библиотеке деревьев реализованы контейнеры «n-арное дерево», «бинарное дерево», «дерево И/ИЛИ».

Для описания процесса построения деревьев И/ИЛИ разработан оригинальный язык GIL – Generation and Identification Language (язык генерации и нумерации). Процесс построения деревьев И/ИЛИ формализуется системой лямбда-исчисления и алгеброй на деревьях И/ИЛИ, поэтому основными чертами языка построения генераторов стали функциональность и возможность производить операции на деревьях И/ИЛИ. Принцип создания генераторов с помощью языка GIL сводится к написанию правил преобразования входных данных генератора к выходным.

Для записи деревьев И/ИЛИ в языке GIL используется скобочная нотация: круглыми скобками обозначаются И-узлы, фигурными – ИЛИ-узлы, узлы без скобок являются листами: `root(alpha{a,b}, digit{1,2,3,4,5})`.

Операция объединения и декартового произведения деревьев И/ИЛИ в языке GIL моделируются операциями ИЛИ-объединения и И-произведения. Введены операции добавления, удаления и поиска узла/ветви, причем операция поиска рассматривается как операция поиска по образцу (pattern matching).

Все выражения GIL делятся на 2 категории: простые выражения (описание дерева И/ИЛИ) и правила преобразования деревьев И/ИЛИ. Операции над деревом записываются в квадратных скобках после выражения, описывающего дерево. Если требуется произвести несколько операций, то они записываются через запятую. Образец для поиска является шаблоном дерева И/ИЛИ или его ветви, по которому ведётся поиск подходящего выражения. В образце для поиска можно задать как строгое, так и нестрогое соответствие узлов дерева:

`root(1{a, b}, 2{c, d}, 1{a, b})[- 1{c@ch}];` – выражение описывает дерево И/ИЛИ, в котором удаляется ветвь, соответствующая образцу `1{c@ch}`, где `c@ch` означает любой список сыновей.

Правила преобразования позволяют указать интерпретатору способ модификации данных определённого вида. Таким образом, интерпретатор сам строит дерево вывода для выражений. Например,

```
pow(@x, @y) = x*pow(x, y-1);
pow(@x, 1) = x;
find_pow(pow(2, 4), pow{x, 2});
```

Результатом выполнения будет дерево И/ИЛИ

```
find_pow(16, pow{x, 2});
```

Для разграничения программного кода модулей в GIL используется механизм пространства имён. Интерпретатор языка выполняет только простые выражения из глобального пространства имён в порядке их объявления.

Интерпретатор языка GIL выполнен в виде отдельного класса, что позволяет встраивать его в различное программное обеспечение.

Интерпретатор имеет возможность расширения путем подключения внешних функций, реализованных на языке программирования C++. Таким способом реализованы алгоритмы генерации и нумерации вариантов дерева И/ИЛИ (правила `var(@tree, n@num)` и `num(@tree, @var)`).

В **четвертой главе** дано описание технологии разработки генераторов в системе построения генераторов комбинаторных множеств, дается анализ применения системы на различных прикладных задачах.

Технология построения генераторов в системе комбинаторных множеств заключается в использовании следующих правил при построении генераторов:

1) Если имеется конечное множество, то оно представляется как ИЛИ-узел, листьями которого являются элементы этого множества.
2) Если имеется выражение некоторого языка, для которого есть функтор, отображающий все операнды выражения в деревья И/ИЛИ и отображающий операции выражения в алгебраические операции на деревьях И/ИЛИ, то дерево И/ИЛИ для него строится по следующим правилам:
   а) операции в выражении заменяются на эквивалентные им операции на деревьях И/ИЛИ;
   б) в полученном выражении операция объединения заменяется операцией ИЛИ-объединения;
   в) в полученном выражении операция декартового произведения заменяется операцией И-призведения;
   г) другие операции анализируются на предмет возможности реализации их через операции ИЛИ-объединения, И-произведения, добавления и удаления узлов;
   д) определяются правила построения дерева И/ИЛИ для каждого операнда выражения;
   е) если в выражении присутствует рекурсия, то задаются условия её ограничения.
3) Если требуется произвести объединение множеств, то оно производится при помощи операции ИЛИ-объединения.
4) Если требуется произвести декартово произведение множеств, то оно производится при помощи операции И-произведения.
5) Построенное дерево И/ИЛИ и его варианты анализируются для построения алгоритма получения элемента исходного множества.
6) Построенное дерево И/ИЛИ и элементы исходного множества анализируются для построения алгоритма получения варианта дерева И/ИЛИ по элементу множества.

В главе приводятся примеры использования системы для построения генераторов в различных предметных областях.

Язык GIL прекрасно подходит для исследования алгоритмов комбинаторной генерации. С его помощью можно строить дерево И/ИЛИ для функции мощности множества, анализировать варианты дерева для построения генератора нужного комбинаторного множества, проверить реализацию генератора на GIL, при необходимости перенести полученный алгоритм на какой-либо из императивных языков программирования. Например, генерация сочетаний в виде битовой последовательности на основе рекурсивной формулы

подсчета количества сочетаний $C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$ осуществляется следующей программной на языке GIL:

```
C(n@m, n@n) = {L(C(m, n-1)), R(C(m-1, n-1))};
C(0, n@m) = ;
C(n@m, n@m) = ;
comb(n@m, n@n, n@num) = flat(var(C(m, n), num)[
     L(@ch) --> (0, ch), R(@ch) --> (1, ch),
     C(@mm, @mm)->repeat(1, mm), C(0, @nn)->repeat(0, nn)
]);
```

Результатом выполнения `comb(2, 4, 5)` будет `1100`.

Показано применение языка GIL для генерации тестовых заданий по различным дисциплинам, таким как программирование, высшая математика. Рассмотрим пример генерации тестового задания по высшей математике (вычисление производной выражения), использующего возможности символьных вычислений языка GIL.

Шаблон вопроса:

Требуется генерировать математическое выражение для следующей грамматики:

```
grammar -> E (opT | opD) func
E -> T | T opT T
opT -> + | -
T -> D | D opD D
opD -> * | /
D -> variable | func | number
variable -> ‘x’
number -> -3 | -2 | -1 | 1 | 2 | 3
func -> pow | ln
pow -> ‘pow(’variable‘, ’number‘)’ | ‘pow(e, ’variable‘)’
ln -> ‘ln(’ variable ‘)’
```

Студент должен вычислить производную для сгенерированного выражения. В ответ требуется ввести значение производной в точке a, $a \in (-5;5), a \neq 0$.

Генератор выражений на GIL для заданной грамматики будет следующим:

```
grammar = (E, {opT, opD}, func);
E = {T, (T, opT, T)};
opT = {'+', '-'};
T = {D, (D, opD, D)};
opD = {'*', '/'};
D = {variable, func, number};
variable = {x};
number = range{-3, 3}[-0];
func = {pow, ln};
pow = {pow(variable, number), pow(e, variable)};
ln = ln(variable);
a = range{-5, 6}[-0];
test = (grammar, a);
```

Проверку ответа осуществляет следующий код:

```
answer((@exp, @a)) =
```

```
        calculate.calc(differential.diff(exp), ‘x’, a);
    test_check(n@num, n@answer) =
        eq(answer(var(test, num)), answer);
```

Мощность описанного генератора равна 138,532,800. Это гарантирует получение индивидуального задания для каждого студента. Ответ проверяется правилом `answer`, которому в качестве параметра передается вариант дерева `test`. Ниже приведена программа на GIL для дифференцирования математических выражений:

```
namespace differential {
    diff((c@childs)) = diff(childs);
    diff(pow(e, n@x)) = pow(e, x);
    diff(pow(s@x, n@y)) = (y, '*', pow(x, y-1));
    diff(ln(s@x)) = (1, '/', x);
    diff(@u, '+', @v) = ((diff(u), '+', diff(v)));
    diff(@u, '*', @v) = ((diff(u), '*', v), '+', (u, '*',
diff(v)));
    diff(@u, '/', @v) = (((diff(u), '*', v), '-', (u, '*',
diff(v))), '/', pow(v, 2));
    diff(ls@x) = 1;
    diff(n@x) = 0;
    diff(tg(@u)) = ((1, '/', cos2(u)), '*', diff(u));
    diff(pow(e, @u)) = (pow(e, u), '*', diff(u));
}
```

Важным этапом разработки программного обеспечения является этап тестирования. Ни один серьезный программный продукт не обходится без него. Система построения генераторов комбинаторных множеств может быть успешно применена для тестирования ПО.

Построим генератор тестовых данных для конвертора математических формул. Конвертор преобразует формулы из формата MathML в TeX и обратно. Тестирование заключается в следующем:

1) генерируется выражение языка MathML;
2) полученное выражение конвертируется в TeX;
3) конвертированное выражение преобразуется конвертором обратно в MathML;
4) если исходное выражение не равно результату конвертирования, значит, конвертирование происходит неправильно.

Код генератора может быть следующим:

```
mathml = 'mathml'{mrow, mfrac};
operation = 'mo'{'-', '+', '*', '/'};
token = {mi, mn, func, msub, msup};
identificator = {'x', 'y'};
number = {10, 1234};
mi = 'mi'(identificator);
mn = 'mn'(number);
mn1 = 'mn'(100);
msub = 'msub'(mi, mn1);
msup = 'msup'(mi, mn1);
func = 'mrow'(funcname, 'mo'('&ApplyFunction;'),
'mrow'('mo'('('), {mi, mn}, 'mo'(')')));
funcname = 'mi'{'sin', 'ln'};
```

```
mrow = {mrow1, mrow2};
mrow1 = 'mrow'(token, operation, token);
mrow2 = 'mrow'(token, operation, func);
mfrac = 'mfrac'(mrow, mrow2);
```

Данный генератор строит дерево И/ИЛИ для грамматики, описывающей выражения на языке разметки математических формул MathML. Мощность описанного генератора равна 787,968. Каждый вариант построенного дерева подается на вход конвертера для проверки корректности его работы.

Анализ результатов применения системы построения генераторов показал, что система полностью удовлетворяет всем требованиям, выработанным в первой главе.

В Томском межвузовском центре дистанционного образования (ТМЦДО) технология тестирования знаний обучающихся, основанная на использовании генераторов тестовых заданий, действует с 2003 года. За это время было создано большое количество генераторов тестовых заданий по различным дисциплинам. Генераторы создавались прямым кодированием на языке программирования C++ без использования специализированных библиотек. С целью экспериментальной проверки эффективности применения системы построения генераторов было решено реализовать несколько уже созданных на языке программирования C++ генераторов тестовых заданий с помощью системы построения генераторов.

Условия эксперимента были следующие:

1) специалист, реализовавший генератор на языке C++, реализовывал этот же генератор в системе построения генераторов, таким образом квалификация специалиста выполняющего работу оставалась неизменной;
2) генераторы выбирались по количеству шаблонов вопросов от 30 до 150.

Результаты эксперимента показали, что время разработки генераторов тестовых заданий уменьшилось в 2 раза, объем программного кода генераторов также сократился в 2 раза (рисунки 6, 7). Эксплуатация разработанных с помощью системы генераторов показала, что объем ошибок по сравнению с реализацией генераторов на языке программирования C++ уменьшился на 50%.

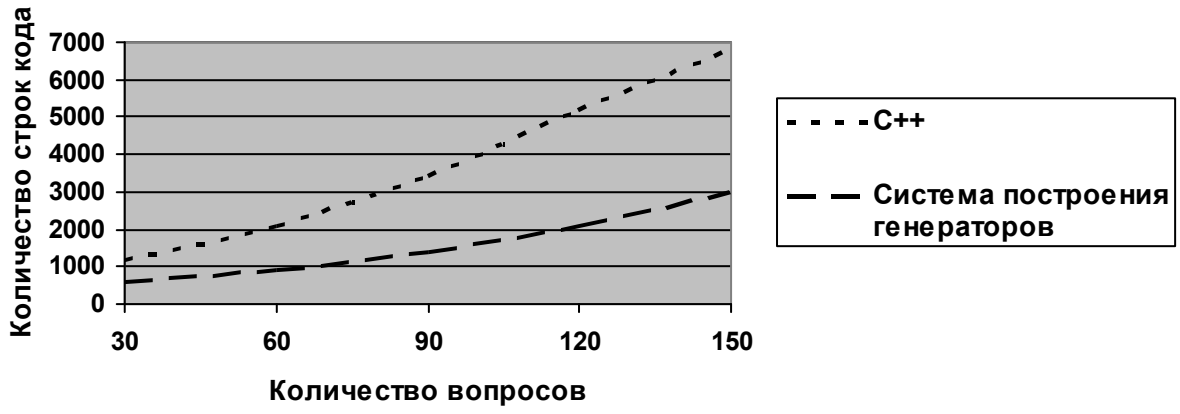

Рисунок 6. Зависимость количества строк кода генератора от количества шаблонов вопросов

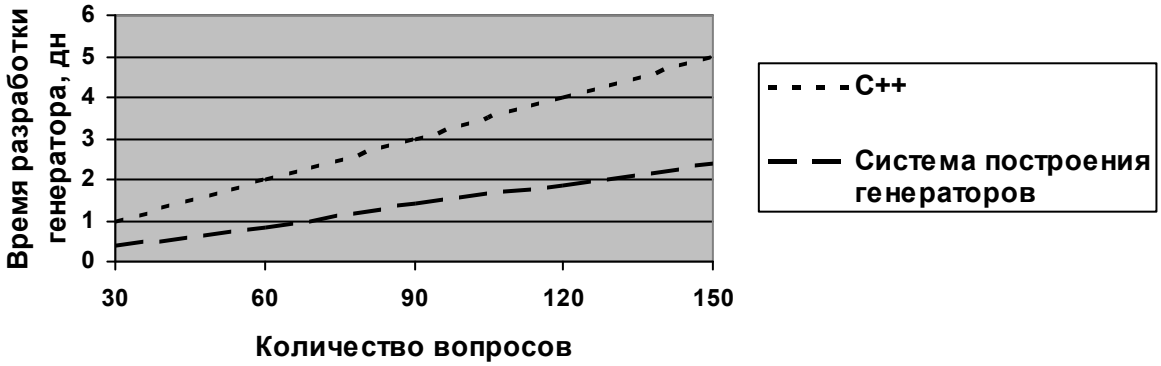

Рисунок 7. Зависимость времени разработки генератора от количества шаблонов вопросов

Внедрение системы построения генераторов позволило вывести генераторы тестовых заданий на качественно новый уровень, ускорило время их разработки, повысило их надежность. Использование символьных вычислений позволило создавать генераторы, которые невозможно или очень сложно было создать прямым кодированием без использования специализированного программного обеспечения.

В **заключении** сформулированы основные результаты диссертационной работы:

1) Произведен анализ деревьев И/ИЛИ с точки зрения теории множеств, показано, что теоретико-множественные операции также определены и на деревьях И/ИЛИ. Показано, что множество алгебраических операций на деревьях И/ИЛИ и система лямбда-исчисления формализуют процесс построения деревьев И/ИЛИ.
2) Разработана и реализована система построения генераторов комбинаторных множеств на основе деревьев И/ИЛИ. Структура разработанной системы определяется методом построения алгоритмов генерации и нумерации на основе деревьев И/ИЛИ, являющегося базисом системы, и включает в себя библиотеку комбинаторных алгоритмов и язык построения генераторов GIL.
3) Разработанная библиотека комбинаторных алгоритмов расширяет библиотеку STL алгоритмами комбинаторной генерации и библиотекой деревьев, позволяющей создавать контейнеры деревьев с заданными свойствами.
4) Разработанное программное обеспечение реализует метод построения алгоритмов генерации и нумерации комбинаторных множеств на основе деревьев И/ИЛИ и позволяет строить программные генераторы для широкого круга предметных областей.
5) Разработанная технология построения генераторов апробирована на широком круге задач и показала свою эффективность.

ОСНОВНЫЕ ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1) Титков, А.В. Язык описания генераторов комбинаторных множеств / А.В. Титков, В.В. Кручинин // Известия Томского политехнического университета. – 2008. – Т. 312. – № 5. – С. 89–93.
2) Титков, А.В. Подход к созданию баз данных, основанный на алгоритмах генерации и идентификации кортежей. / В.В. Кручинин, А.В. Титков. // Известия Томского политехнического университета. – 2006. – Т. 309. – № 8. – С. 28–32.
3) Титков, А.В. Генерация тестовых заданий на основе грамматик в системе построения генераторов на деревьях И/ИЛИ. / А.В. Титков // Материалы международной научно-методической конференции «Современное образование». – 2010. – С. 163–164.
4) Титков, А.В. Применение системы программирования GIL для генерации тестовых заданий. / А.В. Титков, В.В. Кручинин // Материалы международной научно-методической конференции «Современное образование». – 2009. – С. 178–179.
5) Титков, А.В. Реализация библиотеки шаблонов алгоритмов комбинаторной генерации средствами STL / А.В. Титков // Материалы международной научной конференции «Космос, астрономия и программирование». – 2008. – С. 309–311.
6) Титков, А.В. Разработка алгоритмов параллельной генерации комбинаторных множеств / В.В. Кручинин, А.В. Титков // Труды XV Всероссийской научно-методической конференции «Телематика-2008». – 2008. – С. 119.
7) Титков, А.В. Система курсового online-обучения ТМЦДО / А.В. Титков // Труды XIV Всероссийской научно-методической конференции «Телематика-2007». – 2007. – Т. 1 – С. 228.
8) Титков, А.В. Технология проведения экзаменационных сессий с использованием сети Интернет / А.В. Титков, В.В. Кручинин // Доклады Международной научно-практической конференции "Электронные средства и системы управления". – Томск: Издательство Института оптики атмосферы СО РАН. – 2005. – С. 241–242.
9) Титков, А.В. Компьютерные учебные программы томского межвузовского центра дистанционного образования / В.В. Кручинин, А.В. Титков, М.А. Песков // Дистанционное и виртуальное обучение. – 2007. – № 5. – С. 35–37.
10) Титков, А.В. Система online-обучения ТМЦДО / А.В. Титков // материалы отчетной конференции Томского межвузовского центра дистанционного образования. По итогам работы в 2006 г. – Томск : Томский межвузовский центр дистанционного образования. – 2007. – С. 58–63.
11) Титков, А.В. Электронный учебник «Информатика - I» / А.В. Титков, С.С. Свистунов, К.С. Балуева // Наука технологии инновации

часть 2 / Материалы всероссийской научной конференции молодых ученых. Новосибирск: Издательство НГТУ. – 2006. – С. 101–102.
12) Титков А.В. Мультимедийный компьютерный учебник «Программирование на языке С» / А.В. Титков, В.В. Кручинин, С.С. Свистунов, С.Л. Хомич, // Материалы всероссийской научной конференции молодых учёных. – Новосибирск: Издательство НГТУ. – 2006. – С. 105–106.

Тираж 100. Заказ XXX.
Томский государственный университет
систем управления и радиоэлектроники
пр. Ленина, 40