На правах рукописи

Корнеев Георгий Александрович

Автоматизация построения визуализаторов алгоритмов дискретной математики на основе автоматного подхода

Специальность 05.13.12 — Системы автоматизации проектирования (приборостроение)

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Санкт-Петербург
2006

Работа выполнена в Санкт-Петербургском государственном университете информационных технологий, механики и оптики

Научный руководитель	доктор технических наук, профессор Шалыто Анатолий Абрамович
Официальные оппоненты	доктор физико-математических наук, профессор Романовский Иосиф Владимирович
	доктор технических наук, профессор Тропченко Александр Ювенальевич
Ведущая организация	Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

Защита диссертации состоится 24 октября 2006 года в 15 часов 50 минут на заседании диссертационного совета Д 212.227.05 в Санкт-Петербургском государственном университете информационных технологий, механики и оптики по адресу: 197101, Санкт-Петербург, Кронверкский пр. 49.

С диссертацией можно ознакомиться в библиотеке СПбГУ ИТМО.

Автореферат разослан 22 сентября 2006 г.

Ученый секретарь совета Д 212.227.05, кандидат технических наук, доцент — Поляков Владимир Иванович

Общая характеристика работы

Актуальность проблемы. В настоящее время приборы и устройства становятся все сложнее. Для автоматизации их проектирования требуется знание алгоритмов на графах, операций над матрицами, вычислительной геометрии, линейного и динамического программирования и т.д. Зачастую, эти алгоритмы являются весьма сложными и поэтому трудны для изучения. Это требует новых подходов к разработке научных основ обучения автоматизированному проектированию, и, в частности, обучению указанным алгоритмам. Таким новым подходом является применение визуализаторов алгоритмов. Так как указанные алгоритмы сложны, то создание их визуализаторов требует разработки методов построения визуализаторов с целью формализации процесса их проектирования и реализации.

Визуализатор — это программа, в процессе работы которой на экране компьютера динамически демонстрируется применение алгоритма к выбранному набору данных. Визуализаторы позволяют изучать работу алгоритмов в пошаговом режиме, аналогичном режиму трассировки программ.

Для некоторых алгоритмов динамический вариант демонстрации их работы является более естественным, чем набор статических иллюстраций. Для родственных алгоритмов (например, алгоритмов сортировки) визуализация позволяет наглядно продемонстрировать как общий подход, так и различие в механизмах их действия.

При изучении большинства алгоритмов наряду с режимом «шаг вперед» весьма полезен также и режим «шаг назад», позволяющий более быстро и полно понять алгоритм. Например, в алгоритмах перебора с возвратом бывает необходимо сделать несколько шагов назад, для того чтобы понять, почему та или иная ветвь отброшена.

Многолетний опыт построения и применения визуализаторов на кафедре «Компьютерные технологии» СПбГУ ИТМО показал, что они могут быть использованы как основной инструмент преподавания алгоритмов дискретной математики.

Написание визуализатора «с нуля» является трудоемкой задачей. Для ее решения используются *системы визуализации*, предоставляющие как средства создания визуализаторов, так и поддержку времени выполнения.

Обычно системы визуализации предоставляют графический инструментарий, позволяющий разработчику строить визуализаторы. Некоторые системы визуализации дополнительно содержат средства, которые помогают выделять *интересные состояния* визуализируемого алгоритма либо визуализировать изменения в структурах данных.

Несмотря на то, что визуализаторы разрабатываются с 80-х годов XX века, можно утверждать, что к настоящему времени основные достижения в проектировании визуализаторов относятся к сфере их применения в учебном процессе, а успехи в сфере технологии создания визуализаторов практически отсутствуют. В частности, отсутствует метод, позволяющий по алгоритму формально и единообразно создавать логику работы визуализатора.

Поэтому исследования, направленные на решение проблем построения визуализаторов сложных алгоритмов, встречающихся в системах автоматизации проектирования (САПР) приборов и устройств, являются актуальными.

Целью диссертационной работы является разработка и реализация методов построения визуализаторов алгоритмов дискретной математики, которые позволят формализовать процесс построения визуализаторов и обеспечить корректность их построения.

Основные задачи диссертационной работы состоят в следующем.

1. Развитие методов преобразования императивных программ в автоматные.
2. Разработка языка описания визуализаторов алгоритмов дискретной математики.
3. Разработка технологии построения визуализаторов алгоритмов.
4. Внедрение результатов работы в практику программирования визуализаторов и учебный процесс в СПбГУ ИТМО.

Научная новизна. В работе получены следующие научные результаты, которые выносятся на защиту.

1. Предложен метод, позволяющий формально выполнять преобразование императивных (в том числе рекурсивных) программ в систему взаимодействующих конечных автоматов.
2. На основе указанного выше метода разработан метод, позволяющий формально преобразовывать программу в систему взаимодействующих конечных автоматов, обеспечивающую трассировку исходной программы в прямом и обратном направлениях.
3. Разработан язык описания визуализаторов алгоритмов дискретной математики.
4. На основе предложенных методов преобразования программ к автоматному виду разработана технология, автоматизирующая построение визуализаторов рассматриваемого класса.

Методы исследования. В работе использованы методы теории автоматов, теории графов, теории алгоритмов, теории компиляторов и языков программирования.

Достоверность научных положений, выводов и практических рекомендаций, полученных в диссертации, подтверждается доказательствами, корректным обоснованием постановок задач, точной формулировкой критериев, компьютерным моделированием, а также результатами использования методов, предложенных в диссертации, на практике.

Практическое значение работы состоит в том, что все полученные результаты могут быть использованы и в настоящее время уже используются на практике в учебном процессе в СПбГУ ИТМО и других учебных учреждений.

Предложенные методы позволили упростить процесс создания визуализаторов алгоритмов и внесение изменений в них. При этом за счет предложенных методов и их автоматизации уменьшается количество ошибок, допускаемых при разработке визуализаторов алгоритмов. Эти методы реализованы в системе визуализации *Vizi*, существенно упрощающей и ускоряющей процесс создания визуализаторов алгоритмов за счет автоматизации некоторых операций.

Результаты работы могут быть использованы двояко: во-первых, учащиеся могут пользоваться визуализаторами, созданными по предложенной технологии, а, во-вторых, учащиеся могут сами создавать визуализаторы.

Реализация результатов работы. Результаты, полученные в диссертации, используются на практике следующим образом:

1. В учебном процессе на кафедре «Компьютерные технологии» СПбГУ ИТМО при чтении лекций по курсу «Дискретная математика» и выполнении курсовых работ по этому курсу.
2. В учебном процессе в Лицее «Физико-техническая школа» (Санкт-Петербург).
3. В учебном процессе в Специализированном учебно-научном центре МГУ (Москва).

Научно-исследовательские работы. Результаты диссертации были получены в ходе выполнения научно-исследовательских работ по гранту конкурса персональных грантов для студентов и аспирантов вузов и академических институтов Санкт-Петербурга; по теме «Разработка технологии программного обеспечения систем управления на основе автоматного подхода», выполняемой по заказу Министерства образования РФ в 2001-2006 гг.; по теме «Разработка технологии автоматного программирования», выполненной по гранту РФФИ № 02-07-90114 в 2002-2003 гг.; по теме «Разработка технологии объектно-ориентированного программирования с явным выделением состояний», выполняемой по гранту РФФИ № 05-07-90011; по государственному контракту «Технология автоматного программирования: применение и инструментальные средства», выполняемому в рамках ФЦНТП «Исследования и разработки по приоритетным направлениям развития науки и техники на 2002 – 2006 годы».

Апробация результатов работы. Основные положения диссертационной работы докладывались на Второй Всероссийской научной конференции «Методы и средства обработки информации» (Москва, МГУ, 2005 г.); II и III конференции молодых ученых СПбГУ ИТМО (Санкт-Петербург, 2005, 2006 гг.); Политехническом симпозиуме «Молодые ученые — промышленности Северо-Западного региона» (Санкт-Петербургский государственный политехнический университет, 2005 г.); Software Engineering Conference in Russia — SECR-2005 (Москва, 2005 г.); научно-методических конференциях «Телематика-2001», «Телематика-2003» (Санкт-Петербург); XXXV научной и учебно-методической конференция СПбГУ ИТМО «Достижения ученых, аспирантов и студентов СПбГУ ИТМО в науке и образовании» (Санкт-Петербург, 2006 г.); на семинаре «Автоматное программирование» в рамках международной конференции «International Computer Science Symposium in Russia CSR 2006» (Санкт-Петербург, 2006 г.).

Публикации. По теме диссертации опубликовано 11 печатных работ, в том числе в Научно-техническом вестнике СПбГУ ИТМО (входит в «Перечень ведущих рецензируемых научных журналов и изданий, выпускаемых в Российской Федерации»); сборниках трудов конференций «Методы и средства обработки информации», «Телематика», «Межвузовская конференция молодых учёных», Политехнического симпозиума «Молодые ученые — промышленности Северо-Западного региона»; журналах «Телекоммуникации и информатизация образования», «Компьютерные инструменты в образовании».

Структура диссертации. Диссертация изложена на 181 странице и состоит из списка терминов, введения, шести глав, заключения и двух приложений (на 11 страницах). Список литературы содержит 103 наименования. Работа иллюстрирована 50 рисунками и содержит 12 таблиц.

Содержание работы

Во введении описывается предмет исследования, ставятся цель и задачи исследования, обосновывается актуальность темы диссертационной работы. Дается оценка новизны полученных результатов, формулируются положения, выносимые на защиту.

Первая глава содержит описание текущего состояния в области создания и применения визуализаторов и систем визуализации. В частности, рассмотрено применение визуализаторов в учебном процессе и предъявляемые к ним требования.

В этой главе также приводится анализ существующих визуализаторов алгоритмов и систем визуализации с точки зрения выдвинутых требований и показывается, что они им не удовлетворяют. Таким образом, обосновывается необходимость разработки новой системы визуализации.

На основе анализа литературы и многолетнего опыта применения визуализаторов на кафедре «Компьютерных технологий» в СПбГУ ИТМО выделены основные варианты применения визуализаторов алгоритмов в учебном процессе.

Для использования в учебном процессе визуализаторы алгоритмов должны удовлетворять следующим требованиям.

1. *Интерактивность* — учащиеся должны иметь возможность задавать наборы входных данных и рассматривать работу алгоритма на них.
2. *Двунаправленность* — при работе с визуализатором должна быть возможность совершать шаги алгоритма как вперед, так и назад.
3. *Автоматический режим работы* — визуализаторы должны предоставлять возможность работы без вмешательства пользователя.
4. *История* — визуализатор должен обеспечивать возможность пошагового возврата назад, вплоть до начального состояния.
5. *Отображение хода выполнения алгоритма* — визуализатор должен иметь возможность отображать не только изменения в данных, но и другие действия, например, сравнения при сортировке.
6. *Комментарии* — на каждом шаге алгоритма должны отображаться комментарии, поясняющие производимое действие.
7. *Простота использования* — интерфейс визуализатора должен быть интуитивно понятен.
8. *Свободный доступ* — у учащихся должна быть возможность доступа к визуализаторам не только в аудиториях.
9. *Платформонезависимость* — визуализатор должен работать на распространенных аппаратных конфигурациях и операционных системах.
10. *Автономность* — при работе визуализатор не должен требовать подключения к вычислительным сетям.

Системы визуализации должны позволять строить визуализаторы, удовлетворяющие указанным требованиям. Кроме того, к системам визуализации предъявляются дополнительные требования.

11. *Удобство создания визуализаторов* — простота использования системы визуализации при разработке визуализаторов.
12. *Скорость разработки визуализаторов* — возможность разработки визуализаторов на основе уже существующих компонент.

В работе произведен сравнительный анализ визуализаторов алгоритмов сортировок на основе предложенных требований. Результаты сравнения приведены в таблице 1. При этом используются следующие обозначения:

- «+» — свойство выполняется;
- «–» — свойство не выполняется;
- «±» — свойство выполняется частично.

Таблица 1. Сравнительные характеристики визуализаторов алгоритмов сортировок

Визуализаторы	1	2	3	4	5	6	7	8	9	10
Brown University	–	+	±	+	±	±	–	–	–	–
SUNY Brockport	–	±	+	±	–	+	–	+	+	+
Princeton University	±	–	+	–	–	–	±	+	+	+
Hope College	–	–	+	–	+	–	+	+	+	+
Jeliot	+	–	–	–	+	–	+	+	±	+
СПбГУ ИТМО (Казаков М.А.)	–	–	+	–	–	+	+	+	+	+

Также проанализированы девять систем визуализации алгоритмов (таблица 2).

Таблица 2. Сравнительные характеристики систем визуализации

Система визуализации	1	2	3	4	5	6	7	8	9	10
Animal	–	±	+	±	±	+	+	+	+	+
JAWAA	–	±	–	±	±	+	+	+	–	–
BALSA	–	+	±	+	±	±	–	–	–	–
Tango	–	+	+	–	–	–	–	–	–	+
XTango	–	+	+	–	–	–	–	–	–	+
Jeliot	+	–	–	–	+	–	+	+	–	–
Polka	+	–	–	±	±	–	+	–	+	+
Leonardo	+	+	+	+	–	–	–	–	–	+

Таким образом, ни одна из рассмотренных программ не удовлетворяет всем выдвинутым требованиям. Из изложенного следует, что задача разработки системы визуализации, удовлетворяющий всем выдвинутым требованиям, является актуальной.

Вторая глава содержит описание процесса построения визуализаторов алгоритмов и путей его автоматизации. Вначале выделяется структура визуализатора: основные части и связь между ними. Далее анализируются подходы к построению основных частей визуализатора, и обосновывается необходимость автоматизации их построения. В частности, рассматривается автоматный подход к построению логики визуализаторов алгоритмов. В конце главы формулируются задачи, решаемые в диссертационной работе.

Диаграмма вариантов использования визуализатора, построенная в результате анализа материала, изложенного в первой главе, приведена на рис. 1.

На основании имеющегося опыта и вариантов использования из визуализатора можно выделить следующие основные части, диаграмма компонентов которых приведена на рис. 2:

- *логика визуализатора*, обеспечивающая возможность трассировки алгоритма;
- *модель данных*, хранящая вычислительное состояние алгоритма;
- *визуальное представление* — часть визуализатора, определяющая, что и как будет отображаться пользователю в интересных состояниях;
- *набор комментариев*, определяющий какие комментарии будут отображаться пользователю в каждом интересном состоянии;
- *элементы управления*, позволяющие пользователю управлять визуализатором;
- *интерфейс визуализатора*, определяющий каким образом части визуализатора отображаются на экране;
- *проектная документация*, описывающая разработанный визуализатор.

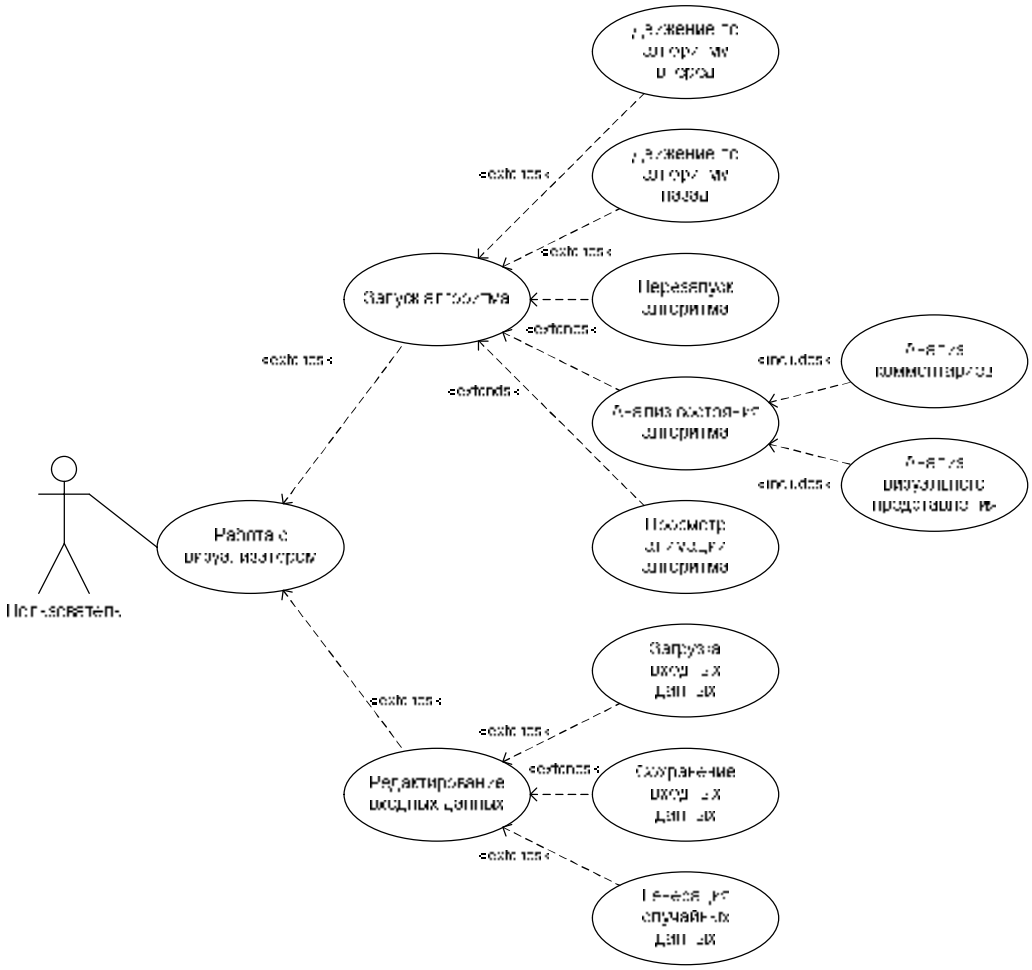


Рис. 1. Диаграмма вариантов использования визуализатора

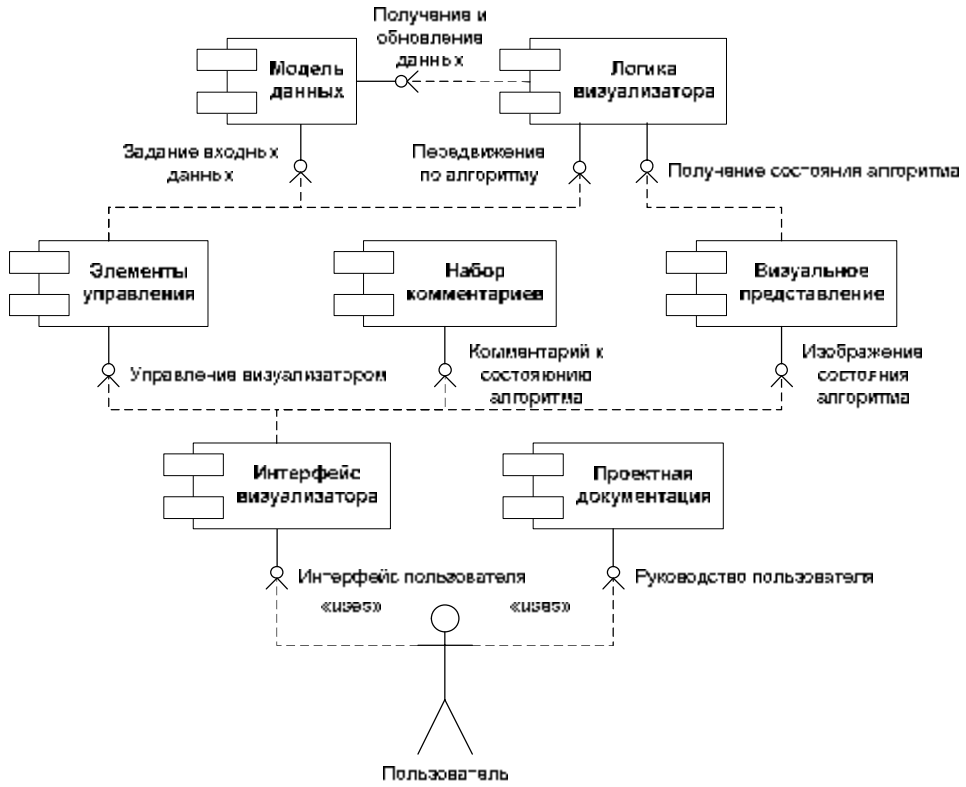


Рис. 2. Диаграмма компонентов основных частей визуализатора

В целях выделения шагов, поддающихся автоматизации, предложен порядок «ручной» разработки визуализаторов, состоящий из 12 шагов. В результате анализа установлено, что четыре шага из них могут быть автоматизированы полностью, а еще четыре — частично. При этом для автоматизации требуется разработать:

- метод автоматизации построения модели данных по программе;
- метод построения программ, обеспечивающий трассировку исходной программы в прямом и обратном направлениях;
- язык описания визуализаторов.

К логике визуализатора и модели данных выдвигаются отдельные требования. Выполнение этих требований является непростой задачей, которая может быть решена при помощи различных подходов.

В работе для построения логики визуализатора предлагается применять автоматный подход, а для модели данных — комбинация статического и стекового выделения памяти.

Таким образом, во второй главе предложена структура визуализатора и подходы к автоматизации построения визуализаторов. Сформулированы теоретические и практические задачи, решаемые в диссертационной работе.

В третьей главе рассматриваются преобразования, позволяющие упростить построение системы взаимодействующих конечных автоматов по программе. Вначале предлагается метод выделения модели данных из программы, что позволяет разделить вычислительные и управляющие состояния. Затем рассматривается преобразование программы к приведенной форме (использующей ограниченное число типов операторов), что позволяет рассматривать только такие программы.

Построение модели данных по программе можно разбить на два этапа:

1. Создание модели данных по программе.
2. Модификация программы к виду, использующему модель данных.

В работе предлагаются методы, позволяющие автоматизировать оба этапа построения модели данных. При применении этих методов по имени переменной модели можно легко узнать исходное имя переменной, и в какой процедуре она была объявлена.

Для упрощения построения системы взаимодействующих конечных автоматов по программе, ее следует преобразовать к приведенной форме. Рассмотрим типы операторов, определенных в императивных языках (в скобках приведена запись в нотации типичной для языков *Java*, *C* и *C++*):

1. Выражение (`expression`).
2. Составной оператор (`{ ... }`).
3. Укороченный оператор ветвления (`if-then`).
4. Полный оператор ветвления (`if-then-else`).
5. Цикл с предусловием (`while`).
6. Оператор вызова процедуры.
7. Цикл с постусловием (`do`).
8. Цикл со счетчиком (`for`).
9. Оператор продолжения цикла (`continue`).
10. Оператор выхода из цикла (`break`).
11. Оператор возврата из процедуры (`return`).
12. Оператор выбора (`switch`).

Программа в приведенной форме должна использовать только операторы первых шести типов. Преобразование к приведенной форме выполняется последовательным удалением «запрещенных» операторов.

Таким образом, в результате применения предложенных методов, строится программа, использующая только ограниченный набор операторов. Кроме того, в этой программе разделены вычислительные и управляющие состояния.

В четвертой главе предлагаются методы преобразования программ в систему взаимодействующих автоматов. При этом рассматриваются как неформальные, так и формальные методы преобразования. Разрабатывается метод построения системы взаимодействующих конечных автоматов, позволяющих исполнять программу в прямом направлении. Далее предлагается метод построения по программе системы автоматов, поддерживающих обратимое исполнение. В заключение для предложенных методов доказана их корректность и другие свойства.

Отметим, что процедуры могут быть рассмотрены как дерево, листами которого являются операторы присваивания и вызова процедуры, внутренними узлами — управляющие операторы, а корнем — процедура в целом.

Построение автомата по процедуре будем выполнять постепенно — от листьев дерева к корню. Для такого построения требуется ввести новые термины.

Фрагмент автомата — набор состояний и переходов. При этом у некоторых переходов может быть не определено начальное или конечное состояние. Такие переходы называются *входами* и *выходами* фрагмента соответственно. Отметим, что переход, у которого не определены ни начальное, ни конечное состояния, одновременно является входом и выходом фрагмента.

Входы и выходы фрагмента позволяют соединять его с другими фрагментами посредством операции *замыкания* — вход одного фрагмента и выход другого объединяются в один переход, у которого определены и начальное (начальное состояние выхода второго фрагмента), и конечное состояния (конечное состояние входа первого фрагмента).

Отметим, что для преобразования фрагмента автомата в автомат требуется замкнуть все его входы и выходы и указать начальное состояние. В дальнейшем будем рассматривать фрагменты автоматов с одним входом и одним выходом.

Для преобразования процедуры в конечный автомат для каждого оператора строится фрагмент автомата. Из фрагментов автомата, соответствующих отдельным операторам, строятся фрагменты, соответствующие последовательностям операторов и т.д.

В начале предлагается метод, позволяющий преобразовывать программу в систему взаимодействующих конечных автоматов, которая поддерживает обратимое исполнение исходной программы.

Для обеспечения обратимого исполнения будем строить *пары автоматов*, состоящие из прямого и обратного автоматов. *Прямой автомат* будет обеспечивать трассировку программы в прямом направлении, а *обратный автомат* — трассировку в обратном направлении.

Прямой и обратный автоматы имеют общее множество состояний, отличаясь только переходами. При этом для трассировки вперед применяется граф переходов прямого автомата, а для трассировки назад — граф переходов обратного автомата. Таким

образом, пара автоматов может быть рассмотрена как один автомат с двумя функциями переходов.

На основе пар автоматов разрабатываются методы построения систем конечных автоматов, обеспечивающих обратимое исполнение. Предлагаемые методы обобщены в таблице 3.

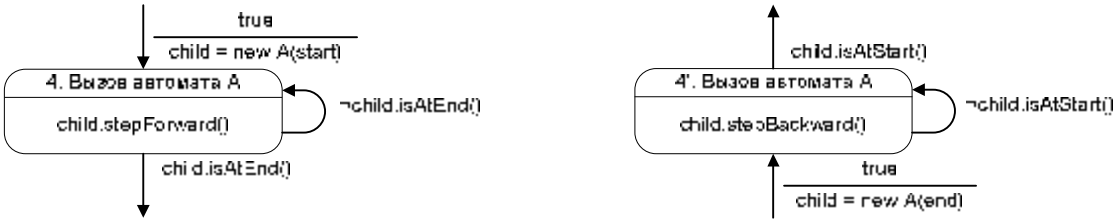
Таблица 3. Варианты построения прямого и обратного автоматов

Особенности обращения	Непосредственное обращение	Косвенное обращение		
		Оператор присваивания	Ветвления и циклы	Последовательности операторов
1	2	3	4	5
Формализм	неформальный	формальный	формальный	формальный
Тип автоматов	Мура	Мура	смешанный	Мура
Стек	не требуется	один		не требуется
Модификация прямого автомата	не требуется	требуется		

Отметим, что прямой и обратный автоматы, построенные при помощи предложенных методов, имеют одинаковую структуру переходов. При этом прямой и обратный автоматы имеют одни и те же переходы, отличающиеся только направлениями, условиями и действиями.

В рекурсивной программе одна и та же процедура может встречаться в цепочке вызовов несколько раз. Назовем каждое вхождение процедуры в цепочку вызовов *экземпляром процедуры*. Для отражения этого факта введем понятие экземпляр автомата. *Экземпляр автомата* — объект, хранящий состояние такого автомата. При этом все экземпляры автомата имеют общую логику, определяемую графом переходов. Таким образом, экземпляру автомата соответствует одно число — номер его состояния.

При прямом проходе для каждого вызова процедуры создается новый экземпляр автомата, находящийся в начальном состоянии. Соответственно, при обратном проходе для каждого вызова процедуры создается новый экземпляр автомата, находящийся в конечном состоянии. Фрагменты прямого и обратного автоматов, выполняющие вызов автомата *A*, приведены на рис. 3.



а б

Рис. 3. Фрагменты прямого (а) и обратного (б) автоматов, вызывающие автомат A

Предложенные методы проиллюстрированы на программе, осуществляющей поиск максимума в массиве натуральных чисел:

```
int max = 0;
for (int i = 0; i < a.length; i++) {
  if (max < a[i]) max = a[i];
}
```

Пара автоматов, построенная при использовании методов, соответствующих столбцам 3 и 4 таблицы 3, приведена на рис. 4.

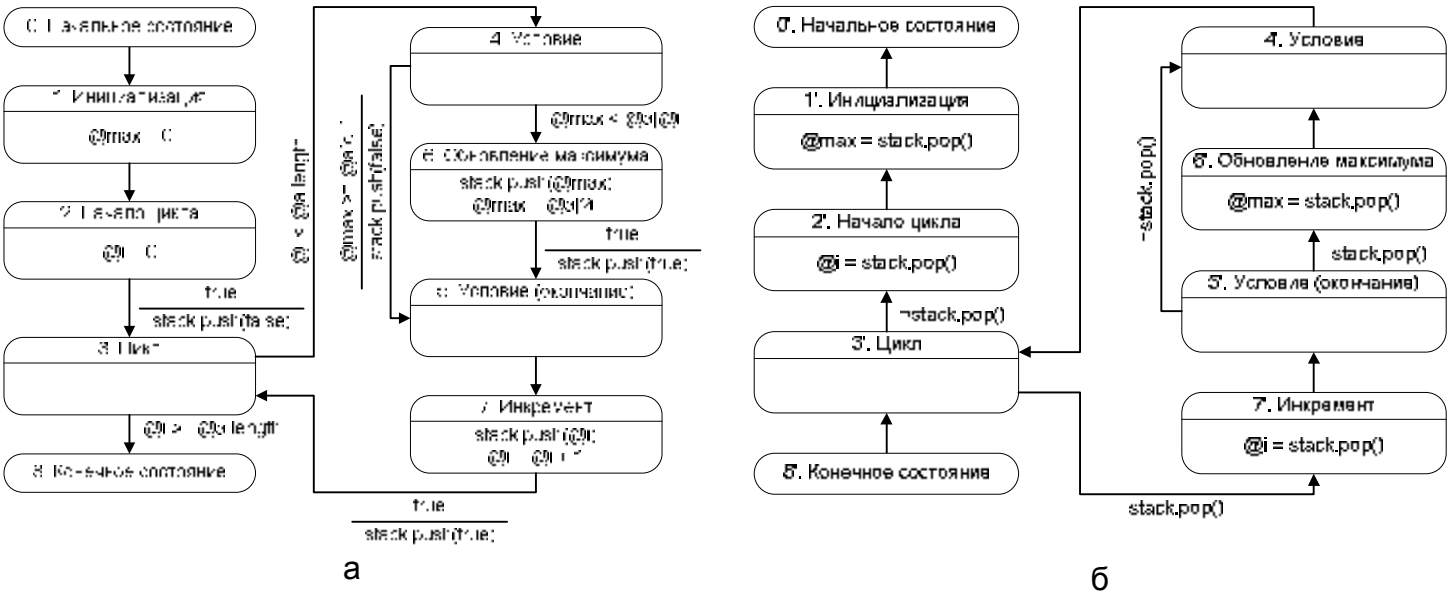


Рис. 4. Прямой (а) и обратный (б) автоматы для процедуры поиска максимума

Для предложенных методов при помощи структурной индукции доказываются следующие основные свойства:

1. *Адекватность* — система автоматов выполняет те же действия, что и исходная программа.
2. *Полнота* — для каждого состояния, кроме конечного, при любых значениях переменных условие на одном из переходов должно быть истинно (дизъюнкция всех условий на переходах из состояния является тавтологией).
3. *Непротиворечивость* — условия на переходах из одного состояния не могут быть истинными одновременно (все попарные конъюнкции условий на переходах должны быть невыполнимы).
4. *Отсутствие недостижимых состояний* — любое состояние может быть достигнуто по переходам из начального состояния.
5. *Обратимость* — при выполнении действий обратного автомата будут восстановлены исходные значения всех переменных, при условии, что в точке входа они были такими же, как в точке выхода при прямом проходе.

Также показывается, что построенная система автоматов линейна по длине исходной программы.

Таким образом, разработаны формальные методы, позволяющие преобразовывать программы в системы взаимодействующих конечных автоматов, поддерживающих обратимое исполнение исходных программ. Для разработанных методов доказаны их корректность и другие свойства.

В пятой главе предлагается язык описания визуализаторов, основанный на *XML*. При этом отдельно рассматриваются структуры описаний визуализируемого алгоритма и конфигурации визуализатора.

Описание визуализатора алгоритма можно разбить на две основные части:

1. Описание визуализируемого алгоритма, позволяющее автоматизировать реализацию: модели данных, логики визуализатора, набора комментариев, связи с визуальным представлением.
2. Описание конфигурации визуализатора, в частности, сообщений, выдаваемых пользователю, цветовой схемы визуализатора и т.д.

Первая часть описания создается один раз в процессе разработки визуализатора и после этого не изменяется. Вторая — позволяет настраивать визуализатор под конкретное окружение, например, языковое.

Процедура рассматривается как последовательность операторов, допустимых в приведенной программе. Каждому типу оператора, кроме блочного, соответствует XML-элемент. Блочные операторы записываются неявно. Отметим, что операторы вызова процедуры позволяют задавать алгоритмы, как с явной, так и с косвенной рекурсией. Оба этих случая обрабатываются корректно.

Операторы могут использовать как глобальные переменные, определенные на уровне алгоритма, так и локальные переменные, определенные в рамках процедуры.

Для оператора может быть указан его уровень, шаблон комментария и связь с визуальным представлением. Шаблон комментария определяет вид и параметры комментария, отображаемого на данном шаге. Связь с визуальным представлением позволяет обновлять визуальное представление при отображении шага.

Корневым элементом конфигурации является элемент `configuration`. Конфигурация может содержать следующие элементы:

- Описания групп (элемент `group`), свойств (`property`) и сообщений (`message`).
- Описания таблиц стилей (`style` и `style-set`), шрифтов (`font`), цветов (`color`).
- Описание элементов управления: панелей (`panel`), кнопок (`button`), панелей выбора (`adjustablePanel`).

Данные элементы позволяют гибко задавать внешний вид визуализатора, не модифицируя его код.

Таким образом, предлагаемый язык позволяет гибко описывать как логику работы визуализатора, так и его конфигурацию.

В шестой главе изложены результаты внедрения разработанных методов. Вначале описывается система визуализации *Vizi*, построенная на основе методов и подходов, разработанных в предыдущих главах. Далее приводится пример построения визуализатора на основе системы *Vizi*. Затем производится сравнение полученных результатов с существующими подходами и приводится информация о практическом внедрении системы *Vizi* и визуализаторов, построенных на ее основе.

Система визуализации *Vizi* состоит из двух основных частей: динамической, функционирующей во время исполнения визуализатора, и статической, работающей на этапе создания визуализатора.

Динамическая часть состоит из библиотеки времени исполнения и кода, сгенерированного по описанию визуализатора. При этом динамическая часть фиксирует структуру визуализатора.

Статическая часть состоит из XSLT-скриптов, генерирующих код и конфигурационные файлы по описанию визуализатора. Кроме того, статическая часть позволяет отлаживать описание алгоритма визуализатора.

Для системы визуализации *Vizi* разработаны уточненная диаграмма компонентов визуализатора и процесс построения визуализаторов, учитывающий возможности, предоставляемые системой.

Данные о затратах времени, требующегося на построение визуализатора, обычно не публикуются, поэтому для его оценки приходится пользоваться косвенными данными.

При этом считается, что время, затраченное на построение программы, примерно пропорционально размеру ее кода.

Для объективного сравнения размеров кода различных визуализаторов требуется исключить как можно больше внешних факторов, таких как используемый язык программирования, библиотеки и т.п. Поэтому в качестве базы для сравнения было решено использовать визуализаторы, построенные на основе библиотеки *BaseApplet*, также разработанной автором, и являющейся предшественником *Vizi*, но не содержащей модулей автоматизации построения визуализаторов.

Отметим сходства и различия *Vizi* и *BaseApplet*, влияющие на сравнение:

1. При построении визуализаторов используется единый язык и стиль программирования.
2. Визуализаторы выполнялись людьми примерно одинаковой квалификации.
3. К проектам визуализаторов предъявлялись схожие требования.
4. *Vizi* содержит средства автоматизации построения визуализаторов, разработанные на основе предложенных методов.

Для сравнения выбирались визуализаторы алгоритмов, отображающие одинаковый объем информации, что позволило уменьшить влияния выбранного способа отображения на результат сравнения.

При сравнении учитывался размер исходного кода и конфигурации визуализатора после удаления комментариев и пробельных символов.

Из результатов сравнения, приведенных в таблице 4, следует, что применение системы визуализации *Vizi* позволяет сократить размер кода визуализатора. Отметим, что с усложнением визуализируемого алгоритма уменьшение размера кода в процентном отношении имеет тенденцию к увеличению.

Список визуализаторов, построенных на основе системы визуализации *Vizi* в 2003-2004 учебном году, представлен в таблице 5. Здесь в столбце «A» обозначено количество пар автоматов, а в столбце «C» — суммарное количество состояний в них. В общей сложности с 2003 по 2006 год на основе системы *Vizi* было построено более 80 визуализаторов алгоритмов.

Таблица 4. Сравнение размеров исходного кода визуализаторов

Алгоритм	*BaseApplet*		*Vizi*		Уменьшение размера
	Автор	Общий размер (КБ)	Автор	Общий размер (КБ)	
Алгоритм Бойера-Мура	Пак С.	18.69	Наумов Р.	12.06	35.5%
Поиск двусвязных компонент графа	Кузнецов А.	26.54	Коломейцева О.	22.87	13.8%
Обход графа в ширину и глубину	Прощенко Ю.	32.34	Гунич И.	25.20	22.1%
Алгоритм Форда-Фалкерсона	Штучкин А.	45.80	Кулев В.	26.57	42.0%
Алгоритм Форда-Беллмана	Гаврилов М.	51.39	Кулагин Д.	26.08	49.3%
Дерево отрезков	Дронь В.	61.01	Лагунов И.	27.63	54.7%
2-3 деревья	Суясов Д.	82.28	Постников Д.	33.20	59.6%
Красно-черные деревья	Краюхин Д.	95.20	Акишев И.	62.50	34.4%

Таблица 5. Визуализаторы, выполненные на основе *Vizi*

Алгоритм	Автор	A	C
Построение кратчайшего дерева в ориентированном графе	Пименов С.	2	26
Битонический алгоритм для задачи коммивояжера	Красильников Н.	2	25
2-3 Деревья	Красильников Н.	14	195
Циклы и разрезы в графах	Ахметов И.	16	97
Алгоритм Укконена	Ахметов И.	2	56
Алгоритм Прима	Ярцев Б.	2	21
Генерация всех простых строк и построение цикла де Брюина	Лоторейчик В.	2	21
Работа со стеком	Поликарпова Н.	14	54
Венгерский алгоритм	Кудинов М.	8	46
Нахождение максимального потока в сети методом Малхотры-Кумара-Махешвари	Бедный Ю.	18	89
Нахождение максимального потока в сети методом Диница	Бедный Ю.	8	68
Алгоритм Штрассена	Котов А.	2	16
Алгоритм Флойда	Колыхматов И.	2	47
Сортировка слиянием	Паращенко Д.	6	31
Быстрая сортировка	Кочелаев Д.	4	40
Дерево отрезков	Вокин А.	2	15
Сортировка кучей	Пименов И.	8	38
Алгоритм Краскала	Данилов В.	6	32

Система визуализации *Vizi* (как и ее предшественник *BaseApplet*) была использована в учебном процессе на кафедре «Компьютерные технологии» СПбГУ ИТМО, лицее «Физико-техническая школа» (Санкт-Петербург), специализированном учебно-научном центре МГУ (Москва).

Таким образом, методы, изложенные в настоящей работе, имеют практическую ценность при обучении алгоритмам дискретной математики, широко используемым при автоматизированном проектировании в приборостроении.

Заключение

В диссертации получены следующие результаты:

1. Предложен метод, позволяющий формально выполнять преобразование императивных (в том числе, рекурсивных) программ в систему взаимодействующих конечных автоматов.
2. На основе указанного выше метода разработан метод, позволяющий формально преобразовывать программу в систему взаимодействующих конечных автоматов, обеспечивающую трассировку исходной программы в прямом и обратном направлениях.
3. Разработан язык описания визуализаторов алгоритмов дискретной математики.
4. На основе предложенных методов преобразования программ к автоматному виду разработана технология, автоматизирующая построение визуализаторов рассматриваемого класса.
5. Результаты работы были внедрены в учебном процессе различных учебных заведений.

Список публикаций

1. *Корнеев Г.А., Шалыто А.А.* Преобразование программ в систему взаимодействующих конечных автоматов / Труды Второй Всероссийской Научной конференции «Методы и средства обработки информации». М.: МГУ. 2005, с. 385-387.
2. *Корнеев Г. А.* Метод преобразования программ в систему взаимодействующих автоматов / Труды II межвузовской конференции молодых учёных. СПб.: СПбГУ ИТМО. 2005, с. 65-72.
3. *Корнеев Г. А.* Технология разработки визуализаторов алгоритмов / Труды II межвузовской конференции молодых учёных. СПб.: СПбГУ ИТМО, 2005, с. 18-23.
4. *Казаков М.А., Корнеев Г.А., Шалыто А.А.* Разработка логики визуализаторов алгоритмов на основе конечных автоматов // Телекоммуникации и информатизация образования. 2003. № 6, с. 27-58.
5. *Корнеев Г.А., Васильев В.Н., Парфенов В.Г., Столяр С.Е.* Визуализаторы алгоритмов как основной инструмент технологии преподавания дискретной математики и программирования / Труды международной научно-методической конференции «Телематика-2001». СПб.: СПбГИТМО (ТУ). 2001, с. 119, 120.
6. *Корнеев Г.А., Шалыто А.А.* Реализация конечных автоматов с использованием объектно-ориентированного программирования / Труды X международной научно-методической конференции «Телематика-2003». СПб.: СПбГИТМО (ТУ). 2003, с. 377, 378.
7. *Корнеев Г.А., Казаков М.А., Шалыто А.А.* Построение логики работы визуализаторов алгоритмов на основе автоматного подхода / Труды X международной научно-методической конференции «Телематика-2003». СПб.: СПбГИТМО (ТУ). 2003, с. 378, 379.
8. *Корнеев Г.А., Шамгунов Н.Н., Шалыто А.А.* Обход деревьев на основе автоматного подхода // Компьютерные инструменты в образовании. 2004. № 3, с. 33-37.
9. *Корнеев Г.А.* Преобразование программы в систему взаимодействующих автоматов, допускающих двустороннюю трассировку / Материалы политехнического симпозиума 2005 «Молодые ученые — промышленности Северо-Западного региона». СПб.: Политехнический университет. 2005, с. 33, 34.
10. *Корнеев Г.А., Шалыто А.А.* Построение визуализаторов алгоритмов дискретной математики // Научно-технический вестник СПбГУ ИТМО. Выпуск 23. Высокие технологии в оптических и информационных системах. СПб.: СПбГУ ИТМО. 2005, с. 118-129.
11. *Корнеев Г.А., Шалыто А.А.* VIZI — язык описания логики визуализаторов алгоритмов // Научно-технический вестник СПбГУ ИТМО. Выпуск 23. Высокие технологии в оптических и информационных системах. СПб.: СПбГУ ИТМО. 2005, с. 130-138.

Тиражирование и брошюровка выполнены в
центре «Университетские телекоммуникации»
Санкт-Петербург, Саблинская ул. 14; тел: (812)233-46-69
Тираж 100 экз.