На правах рукописи

Казаков Матвей Алексеевич

Методы построения визуализаторов алгоритмов дискретной математики на основе автоматного подхода

Специальность 05.13.06 – Автоматизация и управление технологическими процессами и производствами (образование)

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Санкт-Петербург
2010

Работа выполнена в Санкт-Петербургском государственном университете информационных технологий, механики и оптики (СПбГУ ИТМО)

Научный руководитель — доктор технических наук, профессор Парфенов Владимир Глебович

Официальные оппоненты — доктор физико-математических наук, профессор Романовский Иосиф Владимирович

кандидат технических наук, доцент Тимченко Борис Дмитриевич

Ведущая организация — Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

Защита диссертации состоится 27 декабря 2010 года в 11 часов 00 минут на заседании диссертационного совета Д 212.227.06 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 24 ноября 2010 г.

Ученый секретарь диссертационного совета,
доктор технических наук, профессор Л.С. Лисицына

Общая характеристика работы

Актуальность проблемы. С вводом Федеральных государственных образовательных стандартов (ФГОС) усиливается роль самостоятельной работы студентов (СРС), что требует разработки новых образовательных технологий для ее поддержки. Кроме того ФГОС содержат требование к обеспечению не менее 10–20% аудиторных занятий студентов в интерактивной форме. Это обстоятельство ставит новые задачи по разработке интерактивных средств обучения, призванных индивидуализировать образовательный процесс массовой подготовки выпускников вузов. При уровневой подготовке выпускников вузов информационной направленности важное место занимают методы и алгоритмы дискретной математики, изучение которых создает базис (базовые знания, умения и навыки) для формирования различных универсальных (общекультурных) и профессиональных навыков выпускников. Уже сейчас во многих вузах РФ применяются виртуальные лаборатории, тренажеры и электронные тьюторы для поддержки практических и самостоятельных занятий по изучению алгоритмов дискретной математики. Одним из ключевых инструментов при изучении таких алгоритмов являются *визуализаторы* (программы для реализации динамических иллюстраций). При построении учебных курсов по дискретной математике разработчики таких курсов сталкиваются с проблемой отсутствия эффективных и унифицированных методов построения визуализаторов, демонстрирующих работу алгоритмов на каждом шаге их выполнения. Автоматный подход, хорошо зарекомендовавший себя в области проектирования и верификации программ, позволяет явным образом выделить состояния действий алгоритма. Автором настоящей работы было предложено использовать автоматный подход в качестве методологической основы для построения визуализаторов алгоритмов дискретной математики. Применение такого подхода при автоматизации построения визуализаторов было исследовано автором данной работы в сотрудничестве с Г.А. Корнеевым. В настоящей работе проводилось исследование методов ручного построения визуализаторов алгоритмов дискретной математики, т.к. именно ручное построение позволяет глубже понять и изучить природу как самих алгоритмов, так и процессов их визуализации. Кроме того, ручные методы являются самостоятельной технологией программирования для обучения студентов младших курсов. Поэтому разработка методов ручного формализованного построения визуализаторов алгоритмов, обеспечивающих интерактивность и индивидуальность образовательных технологий массовой подготовки студентов в области дискретной математики, является актуальной.

Цель диссертационной работы – разработка методов построения визуализаторов алгоритмов дискретной математики на основе автоматного подхода. Для достижения данной цели были поставлены и решены **следующие задачи**.

1. Анализ возможностей автоматного подхода для решения проблемы построения визуализаторов алгоритмов дискретной математики.
2. Разработка формализованного метода построения визуализаторов алгоритмов на основе конечных автоматов.
3. Разработка метода построения визуализаторов алгоритмов на основе системы взаимодействующих автоматов.
4. Разработка метода простой анимации алгоритмов при построении визуализаторов.
5. Исследование разработанных методов и внедрение результатов работы в учебный процесс.

Научная новизна. На защиту выносятся результаты, обладающие научной новизной.

1. Подход к реализации визуализаторов алгоритмов дискретной математики на основе конечных автоматов.
2. Метод построения визуализаторов на основе конечных автоматов.
3. Метод построения визуализаторов на основе системы взаимодействующих автоматов.
4. Метод для обеспечения простой анимации визуализаторов на основе автоматного подхода.

Методы исследования. В работе использованы дискретная математика, методы автоматного и объектно-ориентированного программирования.

Достоверность научных положений, выводов и практических рекомендаций, полученных в диссертации, подтверждается корректным обоснованием постановок задач, точной формулировкой критериев, компьютерным моделированием, а также актами их внедрения на практике.

Практическое значение. Результаты, полученные в диссертации, используются на практике:

1. На кафедре «Компьютерные технологии» СПбГУ ИТМО (КТ) при разработке визуализаторов для образовательного портала «Интернет-школа программирования» (http://ips.ifmo.ru).
2. В учебном процессе на кафедре КТ по курсу «Теория автоматов в программировании».
3. На кафедре КТ при разработке визуализаторов в рамках учебного курса «Дискретная математика» (http://rain.ifmo.ru/cat/).
4. Данные методы являются основой для построения многих других визуализаторов алгоритмов дискретной математики.

Внедрение результатов работы. Результаты диссертации использованы при выполнении следующих научно-исследовательских работ: «Разработка технологии программного обеспечения систем управления на основе автоматного подхода» (гос. контракт № 10038 по заказу Министерства образования РФ в 2001 – 2005 гг.) и «Разработка технологии автоматного программирования», выполненной по гранту Российского фонда фундаментальных исследований № 02-07-90114 в 2002, 2003 гг.

Апробация результатов работы. Основные положения диссертационной работы докладывались на научно-методических конференциях: «Телематика-1999» (СПб.), «Дистанционное обучение. Проблемы и перспективы взаимодействия вузов Санкт-Петербурга с регионами России» (СПб., 1999), «Телематика-2000», «Телематика-2001», «Телематика-2002», «Телематика-2003» (СПб.), «Конференция молодых ученых СПбГУ ИТМО» (СПб., 2004), «Телематика-2004» (СПб.), «Телематика-2005» (СПб.), «Профессиональное образование, наука, инновации в XXI веке» (СПб., 2007), «Научное программное обеспечение в образовании и научных исследованиях» (СПб., 2008).

Публикации. По теме диссертации опубликовано 28 научных работ, из них 21 печатная работа, в том числе 11 статей, из которых пять статей опубликованы в журналах из перечня ВАК, 7 отчетов по научно-исследовательской работе.

Награды. В 2008 году автор стал лауреатом премии Правительства Российской Федерации в области образования в составе авторского коллектива научно-практической разработки «Инновационная система поиска и подготовки высококвалифицированных специалистов в области производства программного обеспечения на основе проектного и

соревновательного подходов». В рамках этого проекта автор работал над разработкой методов построения визуализаторов и созданием Интернет-школы программирования (http://ips.ifmo.ru/).

Структура диссертации. Диссертация изложена на 178 страницах и состоит из введения, пяти глав, заключения и приложения. Список литературы содержит 97 наименований. Работа иллюстрирована 96 рисунками и 7 таблицами.

Содержание работы

Во введении описывается предмет исследования, ставятся цель и задачи исследования, обосновывается актуальность темы диссертационной работы. Формулируются положения, выносимые на защиту.

В первой главе приведен общий обзор визуализаторов алгоритмов, описаны общие свойства визуализаторов, описывается роль визуализаторов в образовательном процессе и, в частности, в дистанционном обучении. Анализ обучения программированию и дискретной математике показал, что для эффективного обучения необходимо использовать визуализаторы.

Визуализатор – это программа, иллюстрирующая выполнение алгоритма при определенных входных данных. В задачи визуализатора алгоритма входит:

1. Отображение входных и выходных данных и служебных переменных в наглядной форме.
2. Отображение процесса воздействия алгоритма на входные данные и внутренние переменные, процесс принятия решений.
3. Комментарии к каждому действию алгоритма.
4. Пошаговое исполнение алгоритма.

В этой же главе проводится обзор существующих систем визуализации и подходов к построению визуализаторов, делается вывод о том, что при ручном построении визуализаторов алгоритмов существует три подхода:

- **Традиционный эвристический**. Визуализатор строится из общих соображений учащегося.
- **Автоматный эвристический**. Автоматный подход основан на описании поведения визуализатора алгоритма на основе автоматов.
- **Автоматный формализованный**. Визуализатор с использованием автоматов строится на основе алгоритма формализовано.

Автоматный формализованный метод, рассматриваемый в настоящей диссертации, превращает автоматный эвристический подход в метод формального построения визуализаторов.

Далее в первой главе после анализа традиционного эвристического построения визуализаторов и делаются следующие выводы относительно эвристического подхода:

- Отсутствует формализованный подход к переходу от алгоритма к визуализатору.
- Отсутствует метод выделения визуализируемых контрольных точек.
- Отсутствует метод перехода между контрольными точками визуализатора.

Таким образом, традиционный эвристический подход к построению визуализаторов не может быть использован, как метод в силу отсутствия повторяемости и универсальности.

В этой же главе проведено сравнение (табл. 1) технологических платформ для построения визуализаторов по следующим требованиям:

1. Наличие готовых библиотек по построению пользовательского интерфейса.
2. Возможность встраивать приложения в *Web*-страницы.

3. Кросс-платформенность.
4. Возможность работы без использования браузера.
5. Широкое распространение и поддержка.

Из табл. 1 следует, что в качестве платформы для построения визуализаторов выбрана технология *Java Applets*, как наиболее полно удовлетворяющая необходимым требованиям.

Автором диссертации предлагается использовать метод автоматного программирования для построения визуализаторов. Наличие состояний в автоматах позволяет выделять в алгоритме, реализуемым автоматом, контрольные точки, что, в свою очередь, позволяет преобразовывать алгоритм в набор статических кадров. Такой подход обеспечивает возможность однозначного и формального перехода от автомата, реализующего алгоритм, к визуализатору. При этом простая плавная анимация также являет собой последовательность быстро сменяющих друг друга состояний – кадров.

Таблица 1. Результаты сравнения платформ для разработки визуализаторов

Платформа	**1**	**2**	**3**	**4**	**5**
Flash	+	+	+	−	+
Adobe AIR	+	+	+	+	−
Silverlight	+	+	−	−	−
Delphi	+	−	−	+	−
Java Applets	+	+	+	+	+
JavaScript	±	+	+	±	±

Таким образом, методы автоматного программирования позволяют явно выделять состояния в императивных алгоритмах. Это дает возможность построить методы формального перехода от алгоритма к визуализаторам.

Первая глава завершается формулировкой задач, решаемых в диссертационной работе.

Основной вывод из первой главы: анализ существующих методов разработки и систем визуализации показал, что известные методы не позволяют формально преобразовывать логику алгоритма в логику построения визуализаторов. Таким образом, существует необходимость создания методов формализованного построения визуализаторов.

Вторая глава посвящена изложению предлагаемого формализованного метода построения визуализаторов алгоритмов. В рамках данной работы визуализатор представляет собой *динамический набор статических иллюстраций*, отображающих воздействие алгоритма на входные данные и служебные переменные. Целью построения визуализатора является формирование набора таких статических иллюстраций по ходу выполнения алгоритма. Поэтому основная сложность при построении визуализатора состоит в преобразовании традиционного императивного алгоритма в *набор контрольных точек* (состояний) системы. Наличие такого набора позволяет сформировать набор искомых иллюстраций. Другим важным аспектом при реализации визуализатора является возможность перехода от одной контрольной точки (иллюстрации) к следующей.

Программирование с явным выделением состояний предоставляет инструмент для формирования необходимых шагов построения визуализатора. Наличие *состояний* в полученной автоматной реализации алгоритма создает набор контрольных точек, в

которых появляется возможность сформировать искомый *набор статических иллюстраций*. Поскольку автоматное решение реализует визуализируемый алгоритм – воздействует на выходные и выходные переменные так же как императивный алгоритм, описанный в литературе, то для целей визуализации алгоритма поведение визуализатора, построенного на основе автоматной реализации алгоритма, не будет отличаться от поведения визуализатора, построенного традиционным путем.

Поскольку при автоматной реализации алгоритма в каждом состоянии существует однозначный переход в следующее состояние, автоматный подход к реализации визуализаторов предоставляет возможность реализации переходов между контрольными точками алгоритма.

Визуализатор «по-крупному» предлагается строить на основе паттерна проектирования *Model – View – Controller*:

- по алгоритму формализованным способом строится автомат, описывающий поведение визуализатора (контроллер – *Controller*);
- выбираются визуализируемые переменные (модель – *Model*);
- проектируется формирователь иллюстраций и комментариев, который преобразует номер состояния и соответствующие значения визуализируемых переменных в «картинку» и поясняющий текст (представление – *View*).

Перед рассмотрением предлагаемого *формализованного метода* построения визуализаторов приводятся описание автоматного эвристического подхода, лежащего в основе предлагаемого метода, и демонстрация этого подхода на примере алгоритма «решето Эратосфена». При этом формируются основные шаги построения визуализатора, разработанного автором:

1. Постановка и решение задачи в словесной форме.
2. Автоматная реализация алгоритма решения задачи.
3. Реализация построителя иллюстраций.
4. Реализация визуализатора.

Из построения визуализатора делается вывод о том, что эвристический автоматный подход может использоваться для построения некоторых визуализаторов, методом он по своей сути не является, поскольку обладает следующими недостатками.

1. Построение автоматной реализации алгоритма требует специальных навыков. В некоторых случаях решение этой задачи может потребовать существенных трудозатрат.
2. При построении автоматного решения задачи, автомат не всегда является готовым к использованию в визуализаторе.

Далее во второй главе рассматриваются две модификации базового метода построения визуализаторов. При использовании модификации на основе автоматов Мили управляющие состояния не содержат действий над выходными переменными, в то время как все действия осуществляются на переходах. При использовании модификации на основе автоматов Мура действия совершаются в управляющих состояниях. При этом отмечено, что число состояний в автоматах Мура обычно больше (если строго – не меньше), чем их число в эквивалентном автомате Мили. Приведем этапы формализованного метода построения визуализаторов алгоритмов, который назовем базовым:

1. **Постановка задачи**.
2. **Решение задачи** (в словесно-математической форме).
3. **Выбор визуализируемых переменных**.

4. **Анализ алгоритма для визуализации**.
5. **Реализация алгоритма решения задачи**.
6. **Реализация алгоритма на выбранном языке программирования**.
7. **Построение схемы алгоритма по программе**.
8. **Преобразование схемы алгоритма в граф переходов автомата** Мили либо Мура.
9. **Формирование набора невизуализируемых переходов (состояний)**.
10. **Выбор интерфейса визуализатора**.
11. **Сопоставление иллюстраций и комментариев с состояниями автомата**.
12. **Выбор архитектуры программы визуализатора**.
13. **Программная реализация визуализатора**.

В табл. 2 выполнено сравнение модификаций предлагаемого метода на примере «Задачи о рюкзаке».

Автоматы Мура имеют больше состояний, однако наличие единственной иллюстрации в каждом «интересном» состоянии упрощает построение формирователя иллюстраций, реализовываемого с помощью оператора *switch*. В случае автоматов Мили, в некоторых состояниях приходится формировать несколько иллюстраций. Это усложняет логику формирователя иллюстраций.

Таблица 2. Сравнение модификаций метода на примере «задачи о рюкзаке»

Особенность	**Автоматы Мили**	**Автоматы Мура**
Число состояний	10	18
Исключение иллюстраций	«Неинтересные» переходы	«Неинтересные» состояния
Иллюстрации	Состоянию соответствует одна или несколько иллюстраций	Каждому состоянию соответствует одна иллюстрация
Число строк кода для реализации автоматов	85	187

В завершение второй главы на примере алгоритма «Пузырьковая сортировка» выполнено сравнение *традиционного эвристического*, *автоматного эвристического* и *автоматного формализованного метода построения визуализаторов*, и сделаны следующие выводы:

- несмотря на большую распространенность традиционный эвристический подход методом не является;
- для алгоритмов дискретной математики явное выделение состояний по словесной формулировке может быть выполнено сравнительно редко. Однако когда это удается, применение автоматного метода для построения визуализаторов весьма эффективно;
- метод построения визуализаторов на основе формализованного построения графа переходов по схеме алгоритма совмещает достоинства предыдущих двух методов: по алгоритму эвристически строится программа, что характерно для традиционного программирования, а граф переходов автомата получается по этой программе формально.

Таким образом, показано, что формальное построение графа переходов по программе превращает подход к построению визуализаторов в метод.

В третьей главе на основе метода, предложенного во второй главе, предлагается метод построения визуализаторов алгоритмов на основе системы взаимодействующих автоматов Мура. Важной особенностью визуализаторов, построенных этим методом,

является возможность отображения хода алгоритма на разных уровнях визуализации за счет введения *крупных* шагов визуализатора, состоящих из набора более мелких шагов.

Для реализации визуализатора в базовый метод вносятся следующие изменения.

a) После построения программы по схеме алгоритма на шестом этапе проводится преобразование программы с целью выделения крупных шагов в процедуры.
b) При построении схемы алгоритма по программе на седьмом этапе строится не одна, а несколько схем: **одна схема строится для главного алгоритма и по одной схеме для каждой процедуры**.
c) На восьмом этапе проводятся преобразования **для каждого автомата**.
d) На девятом этапе, происходит формирование набора неинтересных **состояний системы автоматов**.
e) На одиннадцатом этапе при формировании иллюстраций, они сопоставляются не состояниям автомата, а **состояниям системы автоматов**.

Каждый вызов вложенной процедуры преобразовывается в запуск вложенного автомата. При этом используется следующее преобразование (рис. 1).

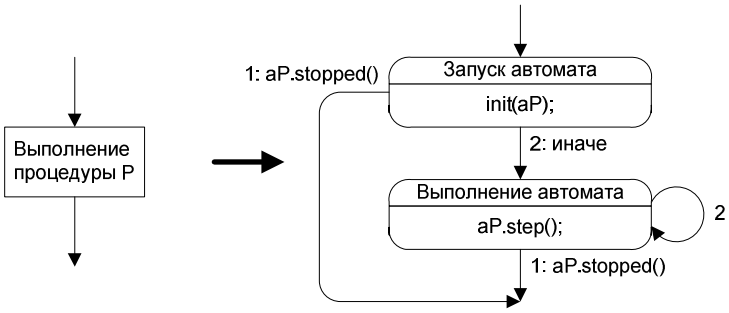

Рис. 1. Преобразование вызова вложенной процедуры

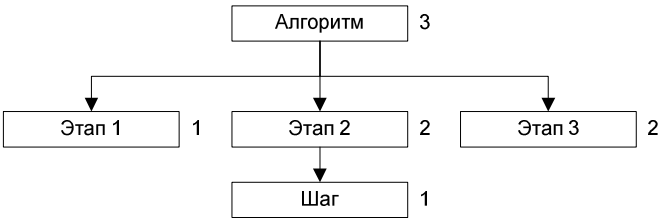

Рис. 2. Назначение уровней каждому автомату

На рис. 2 используются следующие обозначения: *P* – процедура, выполняющая крупных шаг алгоритма; *aP* – автомат, полученный преобразованием схемы алгоритма процедуры *P* в граф переходов автомата Мура; *aP.stopped*() – булево выражение, которое принимает значение *true* в случае окончания работы автомата; *aP.step*() – выполнение одного перехода вложенного автомата Мура.

В дополнение к стандартным в метод добавляется дополнительное преобразование.

f) **Каждому из полученных автоматов Мура сопоставляется его уровень**. При этом, чем выше уровень, тем более крупные шаги алгоритма реализует автомат.

Уровни автомата (рис. 2) позволяют реализовать возможность пропускать некоторые шаги алгоритма для более быстрого перехода к интересующим шагам. В визуализаторе при этом появляются кнопки различных шагов визуализации.

Формирователь иллюстраций должен реализовывать функцию $F_S(S) = F_S(s_1, s_2, \ldots, s_N)$, переводящую состояния системы автоматов в иллюстрацию. Поскольку до построения формирователя иллюстраций строится список «интересных» состояний $\{V_j\}_{j=1}^{K}=\{\{v_1,v_2,\ldots,v_N\}_j\}_{j=1}^{K}$, то при формировании операторов *switch* необходимо построить иллюстрации лишь для интересных состояний.

Итак, при использовании системы автоматов этапы базового метода с первого по шестой не изменяются, а этапы, начиная с седьмого, преобразуются следующим образом:

7. **Преобразование программы с целью выделения крупных шагов в процедуры**.

8. **Построение схем алгоритма по программе**: по одной схеме для главного алгоритма и каждой вложенной процедуры.
9. **Преобразование схем алгоритма в графы переходов системы автоматов Мура**.
10. **Формирование набора «неинтересных» состояний для каждого автомата**.
11. **Назначение уровня детализации каждому из автоматов.**
12. **Выбор интерфейса визуализатора**.
13. **Сопоставление иллюстраций и комментариев с состояниями автоматов**.
14. **Выбор архитектуры программы визуализатора**.
15. **Программная реализация визуализатора**.

Сравнение двух реализации приведено в табл. 3, из рассмотрения которой видна бо́льшая трудоемкость при построении визуализаторов на основе системы автоматов, однако выделение уровней детализации делает визуализаторы алгоритмов более удобными в восприятии и улучшает понимание алгоритма учащимся, реализующим визуализатор.

Таблица 3. Сравнение метрик автоматов для примера «Задача о рюкзаке»

Особенность	Один автомат	Система автоматов
Число автоматов	1	6
Общее число состояний автоматов	16	32
Число строк кода для реализации автоматов	187	455

В четвертой главе предлагается расширение метода построения визуализаторов с целью включения возможности простой анимации, под которой понимается обеспечение не скачкообразного, а непрерывного перехода алгоритма из одной вершины в другую. Для этого предлагается применять четыре автомата:

- *автомат, реализующий алгоритм* (формируется на восьмом этапе);
- *автомат визуализации* (формируется на девятом этапе), который последовательно отображает *статические иллюстрации* в каждом состоянии, что приводит к *статической (пошаговой) визуализации*;
- *преобразованный автомат визуализации*, кроме статических иллюстраций, отображает также и *динамические иллюстрации* в дополнительно вводимых анимационных состояниях. Это позволяет говорить о *динамической визуализации (анимации)*;
- *автомат анимации* отображает статические иллюстраций на экране через определенные промежутки времени.

В этой главе базовый метод расширяется за счет введения следующих этапов:

a) **выбор состояний автомата визуализации, в которых выполняется анимация** (такие состояния будем назвать *анимационными*);
b) **построение преобразованного автомата визуализации путем замены каждого из анимационных состояний тремя состояниями** по схеме указанной ниже;
c) **разработка анимационного автомата и обеспечение его взаимодействия с преобразованным автоматом визуализации**;
d) **разработка и реализация функции вывода каждой динамической иллюстрации, зависящей от состояния автомата визуализации и шага анимации**.

Отметим, что формирование автомата визуализации состоит во введении в каждое состояние *S* выходного воздействия *z0*, осуществляющего формирование статического изображения и комментария $F_S(S)$. При этом переход к следующему состоянию осуществляется по событию *e0* (нажатие клавиши «шаг» в визуализаторе).

На этапе *b* в автомате визуализации для получения преобразованного автомата этого типа каждое анимационное состояние формально заменяется тремя состояниями (рис. 3). При этом *S* – «интересное» состояние, в котором визуализируемым переменным v_i присваивается значение выражений $< expr_i >$. Преобразование состоит в замене выбранного состояния на три состояния — *S′, S* и S_A.

В состоянии *S′* старые значения переменных $v_1, \ldots, v_n$ предварительно сохраняются в переменных $old\_v_1, \ldots, old\_v_n$, а также проводятся все действия, выполняемые в состоянии *S* автомата визуализации. В состоянии *S′* преобразованного автомата визуализации, в отличие от состояния *S* исходного автомата этого типа, не происходит ожидания события, а выполняется переход в одно из состояний S_A или *S*.

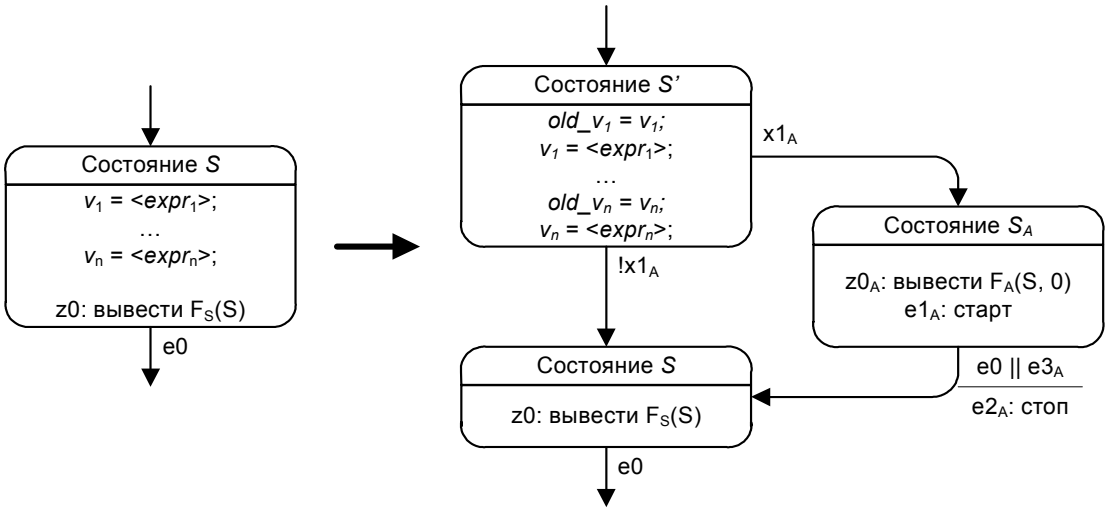

Рис. 3. Преобразование анимационного состояния

При этом $x1_A$ принимает значение истина, если анимация включена. Анимация выполняется, пока преобразованный автомат визуализации находится в состоянии S_A. При входе в это состояние выполняется запуск автомата анимации при помощи события $e1_A$: *старт* и отображается иллюстрация $F_A(S, 0)$, соответствующая началу анимации. Выход из состояния анимации и переход в состояние *S* происходит, как по событию *e0* (нажатие клавиши), так и по событию $e3_A$ (окончание анимации), и сопровождается событием $e2_A$: *стоп*. Таким образом, обеспечивается как автоматическое, так и ручное завершение анимации. Состояние *S* – завершающее, в нем выполняется отображение статической иллюстрации $F_S(S)$.

Следует отметить, что при выключении анимации ($x1_A$ принимает значение ложь) процесс работы преобразованного автомата визуализации полностью повторяет исходный автомат визуализации. Поэтому включение анимации может рассматриваться как расширение исходного визуализатора.

На этапе *c* строится автомат анимации (рис. 4), изменяющей по таймеру значения переменной *step*, которая хранит номер шага анимации.

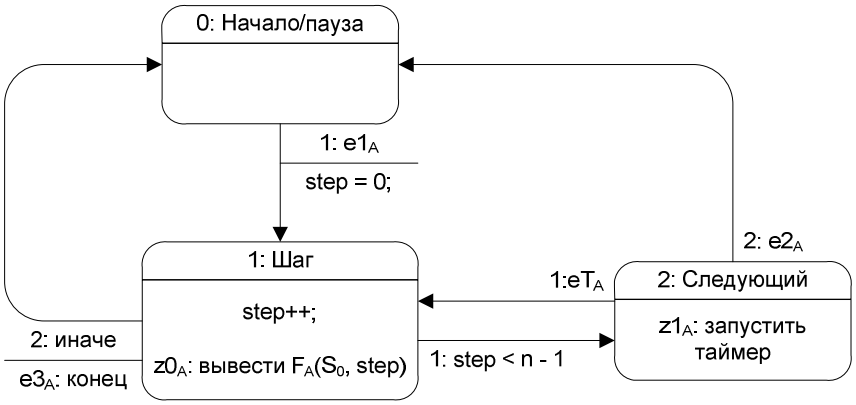

Рис. 4. Автомат анимации

Автомат анимации формирует следующие выходные воздействия и события:

- $z1_A$*: запустить таймер* – запускает таймер, который через период времени T генерирует событие eT_A;
- $z0_A$*: вывести* $F_A(S_0, step)$ – информирует «рисовальщик» о необходимости перерисовать иллюстрацию;
- $e3_A$*: конец анимации* – сигнализирует о том, что анимация закончилась.

Анимационный автомат реагирует на следующие события:

- $e1_A$: *старт* – автомат визуализации перешел в состояние, в котором должна начаться анимация;
- $e2_A$: *стоп* – автомат визуализации перешел в состояние, в котором анимация должна закончиться;
- eT_A – событие от таймера, по которому осуществляется переход к следующему шагу анимации.

На этапе *d* автомат визуализации и автомат анимации связываются с помощью событий и выходных воздействий.

На этапе *e* в режиме анимации «рисовальщик» принимает на вход не только состояние, но и значение номера шага. Если функция F_S реализует статические иллюстрации, а функция F_A – динамические (анимационные) иллюстрации, то при переходе от статики к анимации выполняется преобразование вида:

$$F(state) \to \begin{cases} F_S(state); \\ F_A(state, step) \end{cases}$$

При этом $F_S(state) = F(state)$, $F_A(state, 0) = F_S(state - 1)$, $F_A(state, N) = F_S(state)$. Другими словами преобразованный автомат визуализации формирует статические иллюстрации, совпадающие с иллюстрациями исходного автомата. Формирование анимационных иллюстраций в состоянии *state* происходит таким образом, что начальная иллюстрация совпадает со статической иллюстрацией в предыдущем состоянии автомата, а конечная – со статической иллюстрацией в состоянии *state*.

При введении новых этапов в базовый метод этапы с первого по девятый не изменяются, а последующие этапы преобразуются следующим образом:

10. **Выбор состояний, в которых будет выполняться анимация**.
11. **Замена каждого из выбранных состояний тремя состояниями**.
12. **Использование автомата анимации и обеспечение его взаимодействия с преобразованным автоматом визуализации**.
13. **Выбор интерфейса визуализатора**.
14. **Сопоставление иллюстраций и комментариев с состояниями автомата**.

15. **Обеспечение выбора в каждом анимационном состоянии статического либо динамического изображения, зависящего не только от состояния основного автомата, но и от номера шага анимац**ии.
16. **Выбор архитектуры программы визуализатора**.
17. **Программная реализация визуализатора**.

Применение метода проиллюстрировано на примерах алгоритмов пирамидальной сортировки и обхода дерева в глубину. Таким образом, формальный метод предложенный в четвертой главе является универсальным средством обеспечения простой анимации в визуализаторах алгоритмов дискретной математики.

В пятой главе изложены результаты внедрения разработанных методов в учебный процесс. В начале главы приводится анализ курсовой работы студента кафедры КТ Г. Удова (2004 г.), в которой автор диссертации являлся консультантом. В этой работе студент проводит проверку на практике того факта, что автоматный эвристический подход существенно упрощает процесс построения визуализаторов по сравнению с традиционным эвристическим подходом. В заключении работы студент делает следующие выводы: *«Автоматная реализация расширяет использование конечных автоматов в программировании. Их применение делает естественным процесс визуализации, так как, не вводя состояния, в общем случае не понятно, в какой момент необходимо выполнять шаг визуализации».*

Начиная с 2002 г. по настоящее время, студенты кафедры КТ используют формальный автоматный подход при разработке визуализаторов в рамках курсовых работ по курсу «Дискретная математика». В общей сложности на основе автоматного формализованного метода построения визуализаторов, предложенного в данной диссертации, было спроектировано и построено более 100 визуализаторов алгоритмов. Пример одной из таких работ приведен в Приложении к диссертации.

В рамках проекта «Интернет-школа программирования», выполняемого в 1999 – 2001 гг. на кафедре КТ, визуализаторы алгоритмов дискретной математики разрабатывались традиционным методом. Начиная с 2002 г. в процессе построения визуализаторов, использовался метод, предложенный в настоящей работе. При этом сначала использовался ручной метод, а в дальнейшем визуализаторы создавались на базе системы визуализации *Vizi*, которая основана на совместной статье автора настоящей диссертации с Г.А. Корнеевым.

На основании опыта преподавания на кафедре «Компьютерные технологии» СПбГУ ИТМО был собран статистический материал по трудоемкости создания визуализаторов алгоритмов, который обобщен в табл. 4.

Таблица 4. Сравнение трудозатрат на разработку визуализатора в часах

Подход	**Затраты студента**		**Затраты преподавателя**
	Простые алгоритмы	Сложные алгоритмы	
Традиционный эвристический	80–120	120–180	8–10
Автоматный эвристический	10–20	40–80	2–3
Автоматный формализованный	2–5	20–40	менее одного часа
Система *Vizi*	5–10	10–20	менее одного часа

Таким образом, использование формализованного подхода к построению визуализаторов позволило существенно снизить затраты времени как студентов, так и преподавателей. Кроме того, имеющийся опыт свидетельствует о том, что использование этого метода привело к существенному улучшению качества визуализаторов.

Заключение

Анализ опыта разработки и применения визуализаторов алгоритмов в учебном процессе показал, что их разработка велась на основе эвристических подходов, что существенно отражалось на трудозатратах, как их разработчиков, так и преподавателей.

В ходе проведенных исследований были разработаны методы формального построения визуализаторов алгоритмов дискретной математики. Их применение обеспечило индивидуализацию практических и самостоятельных занятий студентов в СПбГУ ИТМО с использованием эффективных средств автоматизации образовательного процесса в интерактивной форме.

В диссертации получены следующие результаты.

1. В ходе проведенного анализа установлено, что для построения визуализаторов алгоритмов дискретной математики эффективной основой является автоматной подход, который естественным образом позволяет выделять и визуализировать состояния действий алгоритма.
2. Разработан метод формализованного построения визуализаторов алгоритмов дискретной математики на основе конечных автоматов, который имеет не только все достоинства автоматного подхода, но и дает четкую последовательность действий для перехода от визуализируемого алгоритма к его визуализатору.
3. Разработан метод построения визуализаторов алгоритмов дискретной математики на основе системы взаимодействующих конечных автоматов, который позволил проводить визуализацию алгоритмов как пошагово, так и поэтапно.
4. Разработан метод обеспечения простой анимации в визуализаторах на основе конечных автоматов. Использование этого метода в визуализаторах позволило анимировать динамические иллюстрации.
5. При внедрении предложенного *автоматного эвристического подхода* в процесс построение визуализаторов, затраты разработчика снизились в шесть–восемь раз по сравнению с традиционным эвристическим подходом, при этом время, затраченное преподавателем снизилось в три–четыре раза.
6. При внедрении *автоматного формализованного метода* к построению визуализаторов алгоритмов трудозатраты на разработку одного визуализатора снизились до 30 раз по сравнению с традиционным подходом, а время, затраченное преподавателем, снизилось в 10 раз.
7. Предложенные методы успешно используются с 2002 г. в учебном процессе на кафедре КТ СПбГУ ИТМО. За разработку и внедрение методов построения визуализаторов алгоритмов дискретной математики и создание Интернет-школы программирования автор данной работы совместно с авторским коллективом научно-практической разработки «Инновационная система поиска и подготовки высококвалифицированных специалистов в области производства программного обеспечения на основе проектного и соревновательного подходов» стал лауреатом премии Правительства Российской Федерации в области образования за 2008 год.

Статьи в журналах из перечня ВАК

1. *Казаков М. А., Васильев В. Н., Корнеев Г. А., Парфенов В. Г., Шалыто А. А.* Три кита подготовки программистов // Открытые системы. 2009. № 3, с. 54–56.
2. *Казаков М. А., Шалыто А. А.* Методы построения логики визуализаторов алгоритмов // Открытое образование. 2005. № 4, с. 53–58.
3. *Казаков М. А., Шалыто А. А.* Реализация анимации при построении визуализаторов алгоритмов на основе автоматного подхода // Информационно-управляющие системы. 2005. № 4, с. 51–60.
4. *Казаков М. А., Шалыто А. А.* Использование автоматного программирования для реализации визуализаторов // Компьютерные инструменты в образовании. 2004. № 2, с. 19–33.
5. *Казаков М. А., Шалыто А. А.* Автоматный подход к реализации анимации в визуализаторах алгоритмов // Компьютерные инструменты в образовании. 2005, № 3, с. 52–76.

Другие публикации

6. *Казаков М. А., Осипова Т. Г., Парфенов В. Г., Столяр С. Е.* Интернет-школа программирования в СПбГИТМО (ТУ) / Тезисы докладов Всероссийской научно-методической конференции «Телематика'99». СПбГИТМО (ТУ). 1999, с. 165, 166.
7. *Казаков М. А., Васильев В. Н., Парфенов М. А., Столяр С. Е.* Проблемы подготовки профессиональных программистов и дистанционное обучение в Интернет-школе СПбГИТМО (ТУ) / Материалы II межрегиональной научно-практической конференции «Дистанционное обучение. Проблемы и перспективы взаимодействия вузов Санкт-Петербурга с регионами России». 1999, с. 45–48.
8. *Казаков М. А., Столяр С. Е.* Визуализаторы алгоритмов как элемент технологии преподавания дискретной математики и программирования / Тезисы докладов международной научно-методической конференции «Телематика'2000». СПбГУ ИТМО. 2000, с. 189–191.
9. *Казаков М. А., Шалыто А. А., Туккель Н. И.* Использование автоматного подхода для реализации вычислительных алгоритмов / Тезисы докладов международной научно-методической конференции «Телематика'2001». СПбГУ ИТМО. 2001, с. 174–176.
10. *Казаков М. А., Мельничук О. П., Парфенов В. Г.* Интернет школа программирования в СПбГИТМО (ТУ). Реализация и внедрение / Тезисы докладов Всероссийской научно-методической конференции «Телематика'2002». СПбГУ ИТМО. 2002, с. 308, 309.
11. *Казаков М. А.* Создание системы проведения Интернет-соревнований и дистанционного обучения программированию //Телекоммуникации и информатизация образования. 2002. № 6, с. 81–100.
12. *Казаков М. А., Корнеев Г. А., Шалыто А. А.* Построение логики работы визуализаторов алгоритмов на основе автоматного подхода / Тезисы докладов Всероссийской научно-методической конференции «Телематика'2003». СПбГУ ИТМО. 2003, с. 378, 379.
13. *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.
14. *Казаков М. А.* Использование автоматного подхода для построения визуализаторов / Вестник конференции молодых ученых СПбГУ ИТМО. 2004, с. 166–180.

15. *Казаков М. А.* Система автоматического тестирования программных решений и проведения соревнований в режиме on-line / Вестник конференции молодых ученых СПбГУ ИТМО. 2004, с. 181–189.
16. *Казаков М. А., Шалыто А. А.* Реализация визуализаторов на основе автоматного программирования / Тезисы докладов Всероссийской научно-методической конференции «Телематика'2004». СПбГУ ИТМО. 2004, с. 191, 192.
17. *Казаков М. А., Шалыто А. А.* Технология построения визуализаторов алгоритмов на основе автоматного подхода / Тезисы докладов Всероссийской научно-методической конференции «Телематика'2005». СПбГУ ИТМО. 2005, с. 507–509.
18. *Казаков М. А.* Реализация концепции многоуровневой системы дистанционного обучения на базе Интернет школы программирования. / Вестник конференции молодых ученых СПбГУ ИТМО. 2005, с. 176–183.
19. *Казаков М. А., Васильев В. Н., Корнеев Г. А., Парфенов В. Г., Шалыто А. А.* Инновационная система поиска и подготовки высококвалифицированных разработчиков программного обеспечения на основе проектного и соревновательного подходов / Труды Первого Санкт-Петербургского конгресса «Профессиональное образование, наука, инновации в XXI веке». СПбГУ ИТМО. 2007, с. 84–97.
20. *Казаков М. А., Васильев В. Н., Корнеев Г. А., Парфенов В. Г., Шалыто А. А.* Применение проектного подхода на основе автоматного программирования при подготовке разработчиков программного обеспечения / Труды Первого Санкт-Петербургского конгресса «Профессиональное образование, наука, инновации в XXI веке». СПбГУ ИТМО. 2007, с. 98–100.
21. *Казаков М. А., Васильев В. Н., Корнеев Г. А., Парфенов В. Г., Шалыто А. А.* Автоматное программирование и проектный подход при подготовке разработчиков программного обеспечения / Труды научно-технической конференции «Научное программное обеспечение в образовании и научных исследованиях». СПбГПУ. 2008, с. 248–250.

Личный вклад. В работах, выполненных в соавторстве, личный вклад автора состоит в применении автоматов для построения визуализаторов алгоритмов дискретной математики.

Тиражирование и брошюровка выполнены
в центре «Университетские телекоммуникации»
Санкт-Петербург, Саблинская ул. 14, тел. (812)233-46-69.
Объем 1,0 у.п.л. Тираж 100 экз.