На правах рукописи

**Егоров Кирилл Викторович**

# Генерация управляющих автоматов на основе генетического программирования и верификации

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

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Санкт-Петербург – 2013 г.

Работа выполнена на кафедре «Компьютерные технологии» Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики (НИУ ИТМО).

| | |
|---|---|
| Научный руководитель | доктор технических наук,<br>профессор<br>Шалыто Анатолий Абрамович |
| Официальные оппоненты | Соколов Валерий Анатольевич, доктор физико-математических наук, профессор, заведующий кафедрой «Теоретическая информатика» Ярославского государственного университета им. П.Г. Демидова |
| | Новиков Федор Александрович, доктор технических наук, профессор кафедры «Прикладная математика» Санкт-Петербургского государственного политехнического университета |
| Ведущая организация | Санкт-Петербургский институт информатики и автоматизации Российской академии наук |

Защита диссертации состоится 25 декабря 2013 года в 15 часов 30 минут на заседании диссертационного совета Д 212.227.06 при НИУ ИТМО по адресу: 197101, Санкт-Петербург, Кронверкский пр., д. 49, конференц-зал Центра Интернет-образования.

С диссертацией можно ознакомиться в библиотеке НИУ ИТМО.

Автореферат разослан «21» ноября 2013 года.

Ученый секретарь диссертационного совета И.С. Лобанов

## Общая характеристика работы

**Актуальность проблемы**. Существуют программы, в которых ошибки могут обойтись слишком дорого. Например, в программах управления транспортом, медицинским оборудованием, военной техникой и другими объектами ошибки могут привести к гибели людей или большим финансовым потерям. Для обеспечения корректности и надежности работы таких объектов большое значение имеют методы, позволяющие выявлять ошибки на разных этапах разработки и сопровождения программного обеспечения для того, чтобы устранять их.

Одним из основных методов проверки программного обеспечения является тестирование. Оно применяется на практике в большинстве случаев. Однако тестирование позволяет находить ошибки, но не доказывает их отсутствие.

Указанную проблему решает верификация – метод доказательства того, что программа соответствует спецификации. На практике часто используется метод, который называется верификацией на моделях (*Model Checking*). При этом спецификация задается в виде темпоральных формул, число которых в общем случае не известно. Поэтому соответствие программы спецификации может быть обеспечено, но это не будет доказательством правильности программы. Для программ произвольного вида применение этого метода связано с построением по программе модели. При автоматном программировании указанная проблема снимается, так как проектирование этого класса программ начинается с построения модели – управляющих конечных автоматов, которые далее будем называть автоматами. Это существенно упрощает верификацию таких программ на основе метода *Model Checking*. Поэтому в дальнейшем будем рассматривать только автоматные программы.

Еще один подход, обеспечивающий высокое качество программ, состоит в повышении уровня автоматизации их построения. Для рассматриваемого класса программ эта задача состоит из двух частей: генерация автоматов по спецификации и генерация кода программы по автоматам. Можно считать, что вторая проблема решена, а первая проблема решена лишь частично. Например, с помощью генетического программирования можно генерировать автоматы. Если удается построить автомат, то при таком подходе он не всегда соответствует спецификации. Для устранения этого недостатка предлагается в ходе выполнения алгоритма генетического программирования проводить верификацию генерируемых автоматов.

Поэтому исследование, объединяющее генетическое программирование и верификацию при построении автоматов, является актуальным.

**Цель диссертационной работы** – генерации автоматов на основе генетического программирования и верификации.

При этом **задачи, решаемые в диссертации**, состоят в следующем:

1. Предложить функцию приспособленности, учитывающую верификацию, проводимую в процессе выполнения алгоритма генетического программирования.
2. Разработать операции скрещивания и мутации, учитывающие верификацию.

3. Разработать алгоритм построения автоматов на основе генетического программирования с учетом контрактов, которые являются разновидностью темпоральных формул.
4. Разработать алгоритм построения автоматов на основе генетического программирования по сценариям работы и верификации.
5. Разработать верификатор, приспособленный для поддержки генерации автоматов на основе генетического программирования.
6. Разработать технологию и инструментальное средство для генерации автоматов с помощью генетического программирования и верификации.

**Научная новизна**. На защиту выносятся следующие новые научные результаты:

1. Предложена функция приспособленности, учитывающая выполнение формулы на части автомата. В известных работах учитывалось только выполнение или невыполнение каждой темпоральной формулы.
2. Операции скрещивания и мутации отличаются от известных тем, что в ходе их выполнения используется верификация.
3. Алгоритм генерации автоматов с учетом контрактов, который позволяет упростить запись некоторых темпоральных формул и ускорить построение автоматов по сравнению с применением темпоральных формул общего вида.
4. Алгоритм генерации автоматов по темпоральным формулам и сценариям работы, который позволяет ускорить построение автоматов по сравнению с применением тестов.

**Методы исследования**. В работе используются методы теории автоматов, теории графов, дискретной математики и генетического программирования.

**Достоверность** разработанных алгоритмов, методов и технологий, полученных в диссертации, подтверждается точной постановкой задач, корректным формулированием параметров и результатов работы генетического алгоритма, численными оценками скорости работы различных подходов, путем экспериментальных исследований.

**Практическое значение работы** состоит в том, что предложенные подходы позволяют генерировать автоматы, которые соответствуют спецификации и не требуют тестирования и верификации как самостоятельных этапов разработки программ.

**Внедрение результатов работы.** Результаты диссертации были получены при выполнении научно-исследовательских работ на кафедрах «Компьютерные технологии» и «Технологии программирования» НИУ ИТМО по следующим государственным контрактам: «*Разработка методов машинного обучения на основе генетических алгоритмов для построения управляющих конечных автоматов*» (государственный контракт № П2174 от 09.11.2009 г. по Федеральной целевой программе «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы), «*Применение методов искусственного интеллекта в разработке управляющих программных систем*» (государственный контракт № П2236 от 11.11.2009 г. по Федеральной целевой программе «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы). Результаты, полученные в

диссертации, были внедрены при построении модуля *Top Traffic Monitor* для определения узлов сети с максимальным трафиком в программном продукте, выпускаемом компанией ООО *ЭВЕЛОПЕРС* (Санкт-Петербург). Результаты работы используются в учебном процессе на кафедре «Компьютерные технологии» НИУ ИТМО в курсе «Автоматное программирование».

**Апробация результатов работы**. Основные результаты диссертации докладывались на следующих конференциях: 32-я конференция молодых ученых и специалистов Института проблем передачи информации им. А.А. Харкевича РАН «Информационные технологии и системы» (2009), VII и VIII межвузовская конференция молодых ученых (СПбГУ ИТМО, 2010, 2011), 14-th Annual Graduate Students Workshop («Genetic and Evolutionary Computation Conference» GECCO-2011, Dublin, ACM, 2011), вторая и третья межвузовская научная конференция по проблемам информатики СПИСОК-2011, СПИСОК-2012 (СПбГУ, 2011, 2012), I Всероссийский конгресс молодых ученых (СПбГУ ИТМО, 2012).

**Личный вклад автора.** Решение задач диссертации, разработанные алгоритмы и их реализация принадлежат лично автору.

**Публикации**. По теме диссертации опубликовано 11 печатных работ, три из которых в журналах из перечня ВАК.

**Структура и объем работы**. Диссертация состоит из введения, четырех глав, заключения и четырех приложений. Список литературы содержит 75 наименований. Объем работы – 151 страница, 35 рисунков и 13 таблиц.

## Содержание работы

**Первая глава** описывает основные идеи автоматного программирования, метод верификации *Model Checking*, который часто применяется на практике для проверки соответствия программы спецификации. Приведены основные принципы верификации автоматных программ с помощью этого метода.

Автоматное программирование – это парадигма программирования, при использовании которой программа или ее фрагмент рассматривается как система автоматизированных объектов управления. Объекты бывают трех типов: система управления, поставщики событий и объекты управления.

Система управления представляет собой управляющий конечный автомат или систему таких автоматов. В диссертации предполагается, что поведение программ задается одним автоматом, а при описании поведения системой автоматов, они могут быть представлены одним автоматом за счет их «произведения». Автомат состоит из множества состояний и переходов между ними. Каждый переход описывается начальным и конечным состояниями, событием, условием, являющимся булевой формулой, и выходными воздействиями, которые могут быть произвольными. События генерируются их поставщиками, а для проверки условия перехода запрашиваются значения входных переменных у объектов управления. Система управления совершает переход по событию и при выполнении условия перехода, вызывая выходные воздействия.

Формально автомат определяется следующим набором $<S, T, E, X, Z, s_0, L>$, где

- $s_0$ – начальное состояние;
- $S$ – конечное множество состояний;

- $E$ – множество событий;
- $X$ – множество булевых входных переменных;
- $Z$ – множество выходных воздействий произвольного вида;
- $T \subseteq S \times E \times 2^X \times S$ – множество переходов;
- $L : T \to 2^Z$ – функция выходных воздействий.

В настоящей работе спецификация описывается темпоральными формулами. Они задаются с помощью языка логики линейного времени (*Linear Time Logic, LTL*), так как он достаточно прост для понимания и записи утверждений об автоматной программе, которая является классической реагирующей (реактивной) системой. Особенности алгоритма верификации с применением *LTL*-формул позволили использовать его при вычислении функции приспособленности, операциях скрещивания и мутации алгоритма генетического программирования. Синтаксис языка *LTL* включает в себя темпоральные операторы, множество пропозициональных переменных *Prop* и булевы связки (*&, |, ¬*). Для составления утверждений (формул) о событиях в будущем применяются следующие темпоральные операторы:

- ***X** (ne**X**t)*, где *Xp* – предикат *p* выполняется в следующий момент времени;
- ***F** (in the **F**uture)*, где *Fp* – предикат *p* выполняется в какой-то момент в будущем;
- ***G** (**G**lobally in the future)*, где *Gp* – предикат *p* выполняется в любой момент времени (всегда);
- ***U** (**U**ntil)*, где *pUq* – предикат *q* выполняется в некотором состоянии автомата, а во всех предыдущих выполняется предикат *p*;
- ***R** (**R**elease)*, где *pRq* – либо предикат *p* выполняется в некотором состоянии, а во всех предыдущих выполняется предикат *q*, либо предикат *q* выполняется во всех состояниях.

Множество *LTL*-формул можно задать следующим рекурсивным образом:

- *True, False*;
- *Prop* – множество пропозициональных переменных;
- Если $\varphi$ и $\psi$ – *LTL*-формулы, то:
  - $\varphi\&\psi$, $\varphi|\psi$ и $\neg\varphi$ – формулы;
  - $F\varphi$, $X\varphi$, $G\varphi$, $\varphi R\psi$ и $\varphi U\psi$ – формулы

Управляющий автомат и *LTL*-формула представимы в виде автомата *Бюхи*, который формально определяется пятеркой <*S, E, T, $s_0$, F*>:

- $s_0$ – начальное состояние;
- $S$ – конечное множество состояний;
- $E$ – множество меток переходов;
- $T \subseteq S \times E \times S$ – множество переходов;
- $F \subseteq S$ – множество допускающих состояний.

Путь в графе переходов этого автомата является *допускающим*, если существует состояние из множества *F*, принадлежащее данному пути и встречающееся бесконечное число раз.

Известно, что для доказательства выполнимости некоторой *LTL*-формулы необходимо проверить, что автомат пересечения автоматов Бюхи управляющего

автомата и отрицания *LTL*-формулы не допускает ни одно слово. Следовательно, для опровержения этой формулы достаточно найти цикл в автомате пересечения, проходящий через допускающее состояние. Обычно для этого применяется двойной обход в глубину (*Depth-first search, DFS*), так как он позволяет строить автомат пересечения «на лету»: первый из них определят допускающее состояние, а второй – цикл, проходящий через него. Путь из начального состояния в допускающее является контрпримером, опровергающим формулу. На данном пути выполняется отрицание *LTL*-формулы.

Алгоритм построения автомата Бюхи по *LTL*-формуле известен, а алгоритм преобразования автомата управления в автомат Бюхи был предложен на кафедре КТ НИУ ИТМО (http://is.ifmo.ru/verification/_2007_02_report-verification.pdf). В качестве пропозициональных переменных автомата Бюхи используются предикаты. Предикатом может являться любое утверждение о текущем переходе, которое всегда выполняется независимо от предыдущих переходов. В работе используются следующие предикаты:

- *wasEvent(e)* – переход совершен по событию *e*;
- *isInState(s)* – переход совершен в состояние *s*;
- *wasInState(s)* – переход совершен из состояния *s*;
- *cameToFinalState()* – при переходе автомат перешел в завершающее состояние;
- *wasAction(z)* – во время перехода было выполнено выходное воздействие *z*;
- *wasFirstAction(z)* – во время перехода первым выполненным выходным воздействием было *z*.

Поставщики событий и объекты управления не запоминают последовательность переходов и выходных воздействий рассматриваемого автомата. Поэтому в каждый момент времени из текущего состояния автомата может быть совершен любой переход. Преобразование управляющего автомата в автомат Бюхи производится не явно, а в процессе верификации «на лету».

Изложенное реализовано в разработанном автором верификаторе *AutomataVerificator*, который доступен по адресу: http://code.google.com/p/automataverificator/.

Во **второй главе** излагается алгоритм генерации автоматов по *LTL-формулам, тестовым примерам*, *контрактам* и *сценариям работы*.

Диссертация является продолжением *совместных* работ автора и Ф.Н. Царева по генерации автоматов по тестовым примерам и *LTL*-формулам. Результаты этих работ используются в настоящей диссертации. Они состоят в том, что функция приспособленности и операции скрещивания и мутации не зависят от рассматриваемой задачи. В этих работах формулы учитывались только в функции приспособленности, и вклад каждой из них был равен нулю или единице. В настоящей работе верификация учитывается при скрещивании и мутации, а при вычислении функции приспособленности вклад каждой формулы принадлежит отрезку [0; 1]. Это приводит к повышению скорости генерации автоматов с учетом верификации.

При генерации автоматов по тестам и *LTL*-формулам, исходными данными являются:

- список событий $\{e_1, e_2, \ldots, e_v\}$;
- список входных переменных $\{x_1, x_2, \ldots, x_w\}$;
- список выходных воздействий $\{z_1, z_2, \ldots, z_t\}$;
- максимальное ожидаемое число состояний $k$;
- набор тестов $\{\{Input[1], Answer[1]\}, \ldots \{Input[n], Answer[n]\}\}$;
- набор *LTL*-формул.

Результатом работы как известного, так и предлагаемого алгоритмов генетического программирования, является автомат с числом состояний не больше $k$, который проходит все тесты и удовлетворяет всем *LTL*-формулам.

В этих алгоритмах каждый тест содержит *последовательность событий Input[i]*, поступающих на вход автомата, и соответствующую ей эталонную *последовательность выходных воздействий Answer[i]*. Причем заранее не известно отношение между элементами из *Input[i]* и элементами из *Answer[i]*. Тест с номером $i$ считается пройденным, если, подав на вход последовательность *Input[i]*, автомат вызовет в точности последовательность из *Answer[i]*.

*Особь при генетическом программировании* представлена автоматом, который содержит список переходов для каждого из состояний и номер начального состояния. Переходы задаются событиями и числом выходных воздействий, которые необходимо вызвать у объекта управления. Тем самым, в особи не кодируется конкретные выходные воздействия, а только «скелет» автомата (рисунок 1), а конкретные выходные воздействия, вырабатываемые на переходах, определяются с помощью *алгоритма расстановки пометок*.

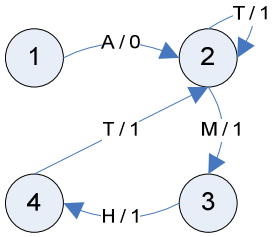



Рисунок 1 – Пример «скелета» автомата

Идея алгоритма расстановки выходных воздействий состоит в следующем: каждый из тестовых примеров запускается на «скелете» автомата, для каждого перехода запоминаются выходные воздействия, после прохождения всех тестов выбираются те выходные воздействия для перехода, которые чаще всего встречаются.

Алгоритм генетического программирования содержит три основных компоненты: функция приспособленности и операции мутации и скрещивания. Ниже укажем, в чем состоит новизна каждой из них.

Введем новую *функцию приспособленности,* задаваемую формулой:

$$FF = \varphi \cdot \frac{\sum_{i=1}^{n}\left(1 - \frac{ED(Output[i], Answer[i])}{\max(|Output[i]|, |Answer[i]|)}\right)}{n} + \psi \cdot \frac{\sum_{i=1}^{m} \frac{t_i}{T'}}{m} + \frac{C - T}{10 \cdot C}.$$

Первая ее часть показывает вклад тестов в функцию приспособленности ($FF_{test}$), вторая – вклад темпоральных свойств ($FF_{LTL}$), а третья – учитывает число переходов автомата. В формуле используются следующие обозначения:

- $\varphi$ – функция, принимающая значение $N$, если проходят все тесты, и $0.5 \times N$ – если проходит только часть из них. Здесь $N$ – «важность» тестов;
- $\psi$ – функция, принимающая значение $M$, если верны все *LTL*-формулы, и $0.5 \times M$ – если выполняется только часть из них, где $M$ – «важность» формул;
- $n$ – число тестов;
- *Output[i]* – последовательность выходных воздействий, которую сгенерировал автомат на входе *Input[i]*;
- *ED(A, B)* – редакционное расстояние между строками $A$ и $B$;
- $m$ – число *LTL*-формул;
- $T'$ – число достижимых переходов в автомате;
- $t_i$ – число переходов, для которых $i$-ая *LTL*-формула выполняется;
- $T$ – число переходов в автомате;
- $C$ – число заведомо большее числа переходов.

Отметим, что первая и третья части формулы были известны, а вторая – новой. Рассмотрим ее более подробно. В диссертации предлагается учитывать *вклад* каждой *LTL*-формулы как $\frac{t_i}{T'}$, что позволяет численно оценивать выполнение формулы значением на отрезке [0; 1], а не нулем, если формула не выполняется, или единицей при ее выполнении. Значение $t_i$ вычисляется в процессе верификации как число переходов, на которых *LTL*-формула выполнятся. Далее будем называть такие переходы «*проверенными*». В основе алгоритма верификации лежит двойной обход в глубину, поэтому «проверенными» переходами можно считать все исходящие переходы для состояний, которые «покинул» первый обход в глубину. Эти переходы и состояния не нарушают *LTL*-формулу. Тогда отношение числа «проверенных» переходов к числу достижимых переходов будет определять, на какой части автомата формула выполняется.

Пусть, например, верифицируется конечный автомат из шести состояний (рисунок 2). Цифрой «1» выделена часть автомата, на которой не удалось обнаружить контрпример («проверенные» переходы), а цифрой «2» – его часть, на которой формула опровергается. Тогда вклад формулы будет 3/7, где три – число «проверенных» переходов, а семь – число переходов в автомате.

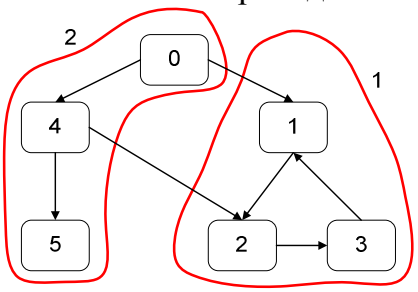



Рисунок 2 – Пример вычисления вклада *LTL*-формулы в функцию приспособленности

*Мутация* применяется к переходам. Она бывает трех видов: заменить конечное состояние перехода, заменит входное событие и изменить число выходных воздействий. В настоящей работе замены с большей вероятностью применяются к

переходам, которые образуют контрпримеры, а ранее они выполнялись для случайных переходов. Благодаря этому повышается вероятность исчезновения контрпримеров, и повышаются шансы новой особи соответствовать бо́льшему числу *LTL*-формул.

В настоящей работе, так же как и в известной, применялись понятия *большая* и *малая* мутации. Они позволяют алгоритму генетического программирования покинуть локальный максимум и не накладывать ограничение на число поколений.

*Скрещивание* происходит путем обмена переходами между двумя особями. Известно, что оно может быть случайным или учитывать прохождение тестов, а в настоящей работе предлагается учитывать «проверенные» переходы. Скрещивание между двумя особями осуществляется следующим образом. Выбираются случайные 10 % *LTL*-формул, для которых выполняется пересечение «проверенных» переходов автомата. Из этого множества переходов исключаются те, которые принадлежат контрпримерам. Такие переходы включаются в новую особь без изменений (рисунок 3). Остальные переходы случайным образом распределяются между особями.

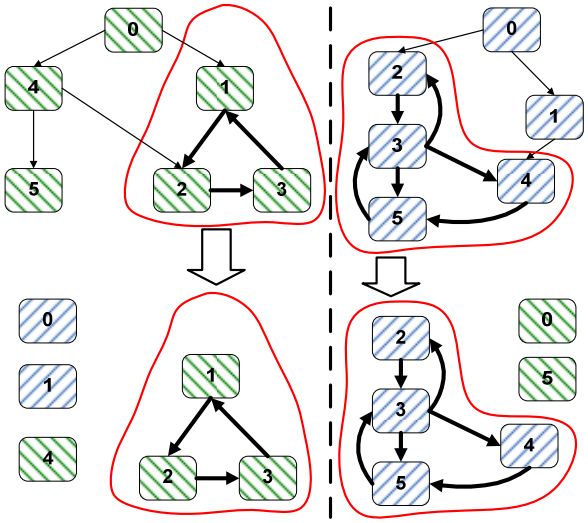



Рисунок 3 – Скрещивание по *LTL*-формулам. Переходы с «жирными» стрелками соответствует «проверенным» переходам

В результате выполнения скрещивания часть конечного автомата, на которой выполняются некоторые формулы, сохраняется. Это позволяет увеличить функцию приспособленности новой особи.

Перейдем к изложению *алгоритма генерации автоматов по LTL-формулам и контрактам*. В общем случае контракт состоит из предусловия, постусловия и инварианта, но может содержать только часть из них. При автоматном подходе контракты могут накладываться как на состояния, так и на переходы и группы состояний.

Контракты для автоматов можно определить через *LTL*-формулы особого вида. Определим *предусловие* как формулу $G(Xp_1 \rightarrow p_2)$ – если на следующем шаге выполнено $p_1$, то выполнено и $p_2$; *постусловие* как формулу $G(p_1 \rightarrow Xp_2)$ – если выполнено $p_1$, то на следующем шаге выполнено $p_2$; *инвариант* как формулу $G(p_1 \rightarrow p_2)$ – если выполнено $p_1$, то выполнено и $p_2$, где $p_1$ и $p_2$ предикаты. Так как

контракты являются *LTL*-формулами, то их выполнение проверяется указанным выше верификатором.

В этом верификаторе постусловие и предусловие оказываются эквивалентными с точностью до предикатов на переходах, так как их отрицание представляются «похожими» автоматами Бюхи. Контрактом можно назвать любую *LTL*-формулу, по отрицанию которой строится автомат Бюхи, эквивалентный автомату контракта с точностью до пометок на переходах. На рисунке 4 представлены автоматы для предусловия (а), постусловия (б) и инварианта (в) для перехода по событию *e*.

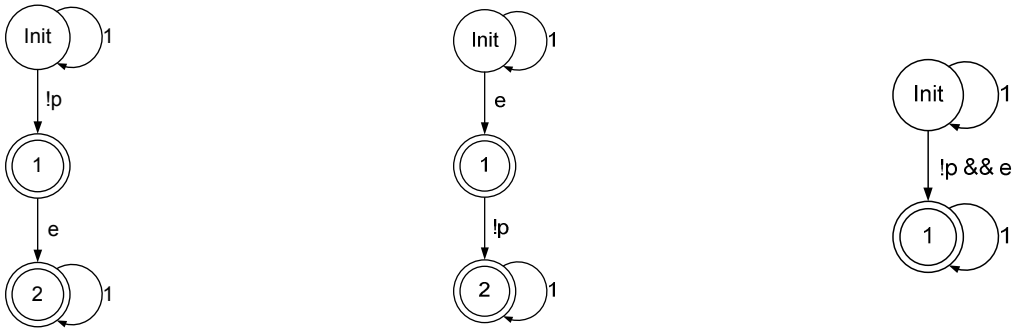



Рисунок 4 – Автоматы Бюхи, построенные для отрицания *LTL*-формул: $G(Xe \rightarrow p)$ (а), $G(e \rightarrow Xp)$ (б), $G(e \rightarrow p)$ (в)

При верификации произвольных темпоральных свойств заранее не известна их семантика. Это приводит к тому, что, обнаружив контрпример в автомате, невозможно определить, какой именно переход нарушает свойство. Когда автомат строится на основе контрактов, точно известно, какие переходы в контрпримере нарушают его. Для инварианта – последний переход, а для предусловия и постусловия – последний и предпоследний переходы одновременно. Поэтому при скрещивании и мутации предлагается использовать не весь контрпример, а только переходы, нарушающие *LTL*-формулу, что ускоряет построение автомата.

Перейдем к изложению *алгоритма построения автоматов на основе генетического программирования по сценариям работы и верификации*. В отличие от диссертации Ф.Н. Царева, в которой применялись тестовые примеры, в настоящей работе используются сценарии работы, которые могут быть либо позитивными, либо негативными.

*Позитивный сценарий* – это последовательность пар (входное воздействие, соответствующий ему список выходных воздействий). Использование таких сценариев не вносит изменений в функцию приспособленности, приведенную выше, однако требуется небольшая модификация представления особи и алгоритма расстановки пометок. Выше было отмечено, что в случае тестовых примеров каждый переход содержит только число выходных воздействий, а не сами воздействия. При использовании сценариев число выходных воздействий перехода не фиксировано. Алгоритм расстановки пометок выбирает то множество выходных воздействий для перехода автомата, которое чаще всего встречается.

Различие между тестами и сценариями показаны на рисунке 5.

| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ |
|---|---|---|---|---|
| $a_1, a_2, a_3, a_4, a_1, a_5, a_3$ | | | | |

а

| $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ |
|---|---|---|---|---|
| $a_1$ | $a_2$ | $a_3, a_4$ | | $a_1, a_5, a_3$ |

б

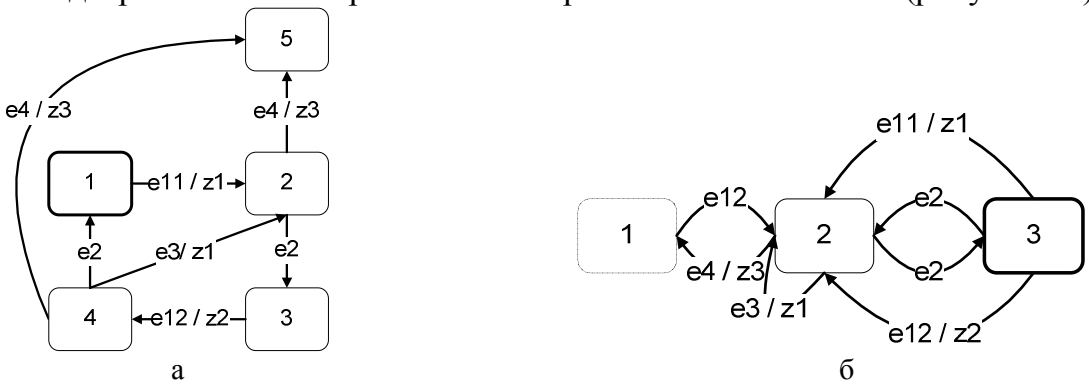
Рисунок 5 – Тестовые примеры (а) и сценарии работы (б)

*Негативный сценарий* представляет собой последовательность входных воздействий, которая не должна выполняться в автомате. При этом все префиксы негативного сценария выполняются, а только последний переход не совершается.

Вклад каждого негативного сценария в функцию приспособленности ноль, если сценарий не выполняется, -1 – в противном случае. Вклад всех негативных сценариев вычисляется как отношение суммы вкладов каждого сценария к общему числу сценариев. Негативные сценарии используются при скрещивании и мутации аналогично контрактам, когда известно, какой переход автомата нарушает выполнение контракта. С большей вероятностью мутирует последний переход негативного сценария. Он не может перейти в новую особь без изменений при скрещивании по позитивным сценариям или *LTL*-формулам.

*Экспериментальные исследования* предложенных подходов проводились для генерации автомата управления часами с будильником и автомата управления дверьми лифта (рисунок 6а). С помощью каждого подхода автоматы строились по 1000 раз без ограничений на число поколений. Для оценки их эффективности измерялось число вычислений функции приспособленности, что пропорционально времени генерации автомата. Приводимые ниже результаты относятся только к автомату управления дверьми лифта.

При этом использовалась спецификация, состоящая из девяти тестовых примеров, девяти позитивных сценариев, 30 негативных сценариев и 11 *LTL*-формул, из которых девять являются контрактами (пять инвариантов и четыре постусловия). При использовании только тестовых примеров построенный автомат в девяти случаях из 1000 соответствовал спецификации. Например, один из сгенерированных автоматов, мог после поломки лифта снова начать закрывать двери, что не соответствует спецификации. Также данный автомат мог повторно начать закрывать или открывать двери после их закрытия или открытия соответственно (рисунок 6б).




а б

Рисунок 6 – Две особи, построенные в ходе выполнения алгоритма генетического программирования

Результаты экспериментов приведены таблице. В ней не указаны результаты экспериментов только с тестовыми примерами, так как правильный автомат получался

только в 1 % случаев. Использование верификации позволяет строить автоматы, соответствующие спецификации всегда.

Таблица – Число вычислений функции приспособленности при генерации автомата управления дверьми лифта

| | Тесты и *LTL*-формулы | Тесты, *LTL*-формулы и контракты | Сценарии, *LTL*-формулы и контракты |
|---|---|---|---|
| Среднее значение | $8.372 \times 10^5$ | $7.109 \times 10^5$ | $1.616 \times 10^5$ |
| Среднеквадратичное отклонение | $7.57 \times 10^5$ | $6.32 \times 10^5$ | $1.102 \times 10^5$ |
| Минимальное значение | $6.331 \times 10^4$ | $6.153 \times 10^4$ | $3.808 \times 10^4$ |
| Максимальное значение | $5.912 \times 10^6$ | $4.589 \times 10^6$ | $8.162 \times 10^5$ |

Полученные результаты позволяют утверждать, что применение сценариев позволило в 4-5 раз сократить время построения автомата управления дверьми лифта.

В **третьей главе** приводится сравнение трех подходов построения автоматных программ с использованием верификации. Первый – построение автомата «вручную» и последующая верификация; второй – генетическое программирование по тестам; третий – генетическое программирование по сценариям работы и контрактам с учетом верификации в функции приспособленности, операциях скрещивания и мутации.

Первые два подхода обладают недостатком – необходима дополнительная верификация, и при обнаружении ошибки необходимо вносить изменения в автомат. Третий подход лишен этого недостатка – он позволяет на выходе алгоритма генетического программирования получать автомат, соответствующий спецификации за счет того, что каждая особь, генерируемая алгоритмом, подвергается верификации. На базе этого подхода создана соответствующая технология (рисунок 7).


Рисунок 7 – Схема, соответствующая технологии генерации автоматов на основе генетического программирования и верификации

Предложенная технология реализована в инструментальном средстве *GABP* (*Genetic Automata-Based Programming*), которое доступно по адресу:

http://code.google.com/p/gabp/. Это средство содержит верификатор *AutomataVerificator*, упомянутый в главе 1. Инструментальное средство *GABP* принимает на вход *XML*-файл, содержащий параметры эксперимента и спецификацию. На выходе строится *XML*-описание в формате инструментального средства *UniMod* (http://unimod.sourceforge.net/), который позволяет генерировать код на языке *Java* или непосредственно исполнять сгенерированный автомат.

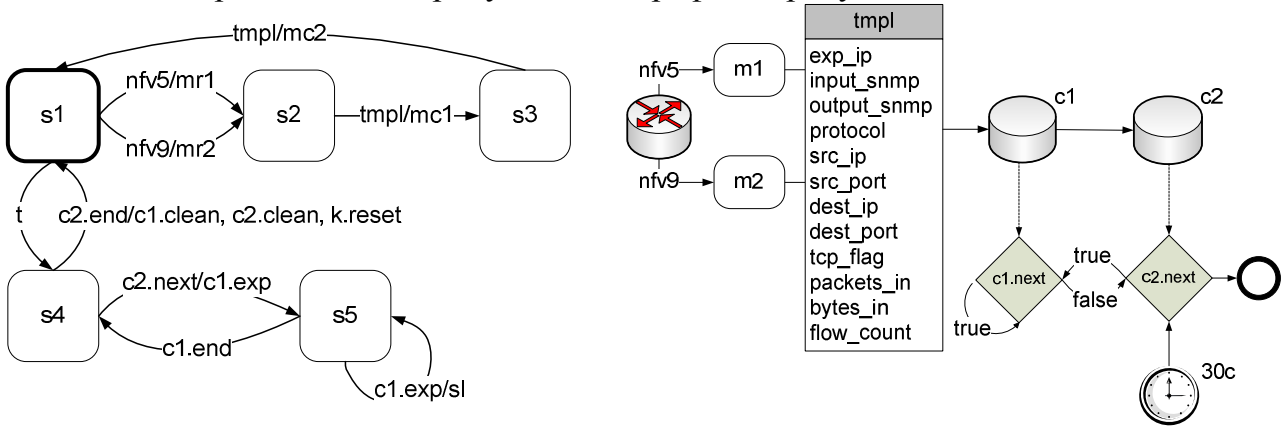
В **четвертой главе** приведены результаты внедрения разработанной технологии и предложенного инструментального средства для генерации соответствующего спецификации автомата для модуля *Top Traffic Monitor*, осуществляющего поиск узлов сети с максимальным трафиком, который входит в программный продукт, предназначенный для мониторинга сети и поиска потенциальных сетевых атак или угроз.

Каждый модуль программного продукта описывается графом потока управления и данных (*Control and Data Flow Graph, CDFG*). Вершины этого графа выполняют операции над *NetFlow*-сообщениями, а ребра бывают двух типов: *Control Flow* (последовательность выполнения операций) и *Data Flow* (связь между источниками и приемниками данных). Граф может быть преобразован в автомат и наоборот. При этом для построения графа предлагается сгенерировать автомат с помощью разработанного средства и преобразовать его в граф.

При построении модуля использовалась спецификация из 10 позитивных сценариев, 27 негативных сценариев и девяти *LTL*-формул, из которых шесть – контракты. Автомат строился 1000 раз с помощью генетического программирования без ограничения на число поколений при следующих исходных данных: размер начальной популяции – 2000, число состояний автомата начального поколения – 7, ожидаемое число переходов – 9, доля особей, переходящих в следующее поколение – 10 %, вероятность мутации – 2 %.

Для оценки скорости работы алгоритма генетического программирования измерялось число вычислений функции приспособленности. Среднее значение – $3.012 \times 10^5$. Среднеквадратичное отклонение – $1.904 \times 10^5$. Минимальное число вычислений – $8.318 \times 10^4$. Максимальное число – $2.096 \times 10^6$.

Автомат представлен на рисунке 8а, а граф – на рисунке 8б.




а б

Рисунок 8 – Модуль *Top Traffic Monitor*. Автомат (а) и граф (б)

Испытания программного продукта показали, что построенный модуль работает правильно.

## Заключение

В ходе диссертационной работы получены следующие результаты, обладающие научной новизной:

1. Предложена функция приспособленности, учитывающая верификацию. В отличие от известных, в ней вклад каждой *LTL*-формулы не ноль или единица, а значение в интервале [0; 1], что более точно оценивает выполнение формулы.
2. Разработаны операции мутации и скрещивания, учитывающие верификацию, что позволяет строить конечный автомат быстрее, чем при учете верификации только в функции приспособленности.
3. Разработан алгоритм генерации автоматов с учетом контрактов. Ранее при генерации автоматов контракты не использовались. Так как контракты – это *LTL*-формулы определенного вида, то их применение позволяет ускорить генерацию автоматов с учетом верификации.
4. Разработан алгоритм генерации автоматов по сценариям работы и верификации. Применение сценариев позволяет ускорить генерацию автоматов по сравнению с генерацией автоматов по тестам.
5. Разработан верификатор *AutomataVerificator*, приспособленный для поддержки генерации автоматов на основе генетического программирования.
6. Разработана технология генерации автоматов с помощью генетического программирования и верификации, которая не зависит от решаемой задачи.
7. На основе разработанных верификатора и технологии создано инструментальное средство *GABP* для генерации автоматов с помощью генетического программирования и верификации.
8. Это инструментальное средство было использовано для генерации автомата управления модулем *Top Traffic Monitor* для поиска узлов сети с максимальным трафиком, который входит в программный продукт, выпускаемый *ООО ЭВЕЛОПЕРС* (Санкт-Петербург). Испытания этого продукта показали, что построенный автомат соответствует спецификации, а модуль работает правильно без дополнительной отладки.

## Статьи в журналах из перечня ВАК

1. *Егоров К. В., Шалыто А. А.* Методика верификации автоматных программ // Информационно-управляющие системы. 2008. № 5, с. 15 – 21.
2. *Егоров К. В., Шалыто А. А.* Разработка верификатора автоматных программ // Научно-технический вестник СПбГУ ИТМО. 2008. Вып. 53. Автоматное программирование, с. 177 – 188.
3. *Егоров К. В., Царев Ф. Н., Шалыто А. А.* Применение генетического программирования для построения автоматов управления системами со сложным поведением на основе обучающих примеров и спецификации // Научно-технический вестник СПбГУ ИТМО. 2010. № 5 (69), с. 81 – 86.

## Другие публикации


4. *Егоров К. В., Царев Ф. Н.* Совместное применение генетического программирования и верификации моделей для построения автоматов управления системами со сложным поведением / Сборник трудов 32-й конференция молодых ученых и специалистов «Информационные технологии и системы» (ИТИС-2009). М.: Институт проблем передачи информации им. А. А. Харкевича РАН. 2009, с. 77 – 82.
5. *Егоров К. В., Царев Ф. Н., Парфенов В. Г.* Совместное применение генетического программирования и верификации моделей для построения автоматов управления системами со сложным поведением / Труды XVII Всероссийской научно-методической конференции «Телематика-2010». Т. 2. СПбГУ ИТМО. 2010, с. 344, 345.
6. *Егоров К. В., Царев Ф. Н., Шалыто А. А.* Совместное применение генетического программирования и верификации для построения автоматов управления системами со сложным поведением // Труды СПИИРАН. 2010. Вып. 15, с. 123 – 135.
7. *Егоров К. В., Царев Ф. Н.* Применение генетического программирования для построения автоматов управления системами со сложным поведением на основе верификации моделей и обучающих примеров / Материалы второй межвузовской научной конференции по проблемам информатики «СПИСОК-2011». СПб.: ВВМ. 2011, с. 343 – 350.
8. *Егоров К. В., Шалыто А. А.* Применение генетического программирования для построения автоматов управления системами со сложным поведением на основе контрактов и тестовых примеров / Материалы второй межвузовской научной конференции по проблемам информатики «СПИСОК-2011». СПб.: ВВМ. 2011, с. 351 – 355.
9. *Егоров К. В., Шалыто А. А.* Применение генетического программирования для построения автоматов управления системами со сложным поведением на основе контрактов и тестовых примеров / Сборник научных трудов VI-ой Международной научно-практической конференции «Интегрированные модели и мягкие вычисления в искусственном интеллекте. М.: Физмалит. 2011, с. 610 – 615.
10. *Egorov K., Tsarev F.* Finite State Machine Induction using Genetic Programming Based on Testing and Model Checking / Proceedings of the 2011 GECCO Conference Companion on Genetic and Evolutionary Computation. NY: ACM. 2011, pp. 759 – 762.
11. *Егоров К. В., Царев Ф. Н., Шалыто А. А.* Построение автоматов управления системами со сложным поведением на основе верификации и сценариев работы / Материалы третьей всероссийской научной конференция по проблемам информатики «СПИСОК-2012». СПб.: ВВМ. СПбГУ. 2012, с. 411 – 414.