ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ И ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

УДК 004.4’242

СРАВНЕНИЕ ТИПОВ ПРИЗНАКОВ В ЗАДАЧЕ ПОИСКА МУЗЫКИ ПО НАПЕВАНИЮ

И.А. Балтийский
Научный руководитель – к.ф.-м.н., м.н.с. С.И. Николенко
(Санкт-Петербургское отделение Математического института им. В.А. Стеклова РАН)

Для современных информационных систем характерен настолько большой объем данных, что перед пользователем встает проблема поиска интересующей информации. Одно из важнейших требований к поиску в мультимедийных системах – удобство задания критериев запроса. Для аудиоинформации одним из типичных желаемых запросов является поиск по содержимому. Например, пользователь знает отрывок из песни и хочет найти эту песню. Поиск в этом случае может выглядеть так: пользователь напевает или насвистывает этот отрывок, записывает при помощи бытового звукозаписывающего устройства и передает в качестве запроса системе. В ответ выдается список возможных соответствий.

Вопросами поиска музыкальной информации по содержимому занимается специальный междисциплинарный раздел науки – информационный поиск музыки (music information retrieval, MIR). Приведенная задача относится именно к этой области и называется поиском по напеванию (Query-by-Humming, QBH).

Один из способов решения задачи – перевод запроса пользователя и аудиоинформации, содержащейся в системе, в специальное представление, по которому в дальнейшем производится поиск. Такое представление будем называть признаками (англ. features). От выбора типа представления может зависеть качество поиска. Различными исследователями было предложено несколько вариантов признаков. Математически все они являются наборами вещественных векторов с некоторыми свойствами. Таким образом, процедура поиска алгоритмически может быть абстрагирована от выбора признаков.

Предпринята попытка сравнить существующие способы представления по качеству поиска. Для этого реализована информационная система, осуществляющая поиск по произвольному типу признаков. Реализовано 4 типа признаков: мел-частотные кепстральные коэффициенты (MFCC), нормализованное распределение энергии по хромам (CENS), а также их разновидности – MFCC с поэлементным добавлением первых и вторых разностей и CENS-признаки с модифицированным процессом поиска, использующим гармонический смысл компонентов CENS-векторов.

На основе построенной поисковой системы проведен эксперимент, в котором 24 участникам предлагалось напеть 5 популярных композиций. Для всех запросов при помощи каждого из 4 исследуемых типов признаков были получены отсортированные по релевантности списки ответов. Оценивалось среднее по пользователям положение искомой композиции в списке.

Результаты эксперимента показывают, что для всех 5 композиций оба типа CENS-признаков имеют превосходство перед MFCC-типами. Данный результат можно объяснить тем, что CENS является представлением более высокого уровня, позволяя производить поиск при возможных ошибках пользователя при напевании.

УДК 004.832.23

ГЕНЕРАЦИЯ КОНЕЧНЫХ АВТОМАТОВ С ПОМОЩЬЮ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ НАВИГАЦИИ

М.В. Буздалов
Научный руководитель – ассистент Ф.Н. Царев

Постановка проблемы. В последнее время все чаще появляются работы, направленные на решение определенных задач искусственного интеллекта методом генерации конечного автомата с помощью эволюционных алгоритмов. Специфика поиска решения эволюционными алгоритмами, в том числе генетическими, заключается в том, что поиск решения происходит в течение достаточно большого времени, а гарантии того, что решение будет найдено, алгоритм не дает. По этой причине, перед тем, как применять такие алгоритмы для решения задач в некоторой области искусственного интеллекта, сначала необходимо убедиться в том, что они вообще способны найти решение сколь-либо сложных задач в этой области.

В современных областях искусственного интеллекта, таких как робототехника, большую роль играют задачи навигации. Рассмотрим, например, простую задачу, имеющую, однако, огромную практическую ценность: движение агента в неизвестной ему области с препятствиями. Выбор задачи обусловлен тем, что она явно или неявно включается практически в любую задачу навигации с препятствиями.

Упростим задачу:

- область представляет собой бесконечное клеточное поле;
- агент занимает ровно одну клетку;
- существуют клетки-препятствия, которые агент не может занять;
- за один ход агент может переместиться из текущей клетки в одну из смежных с ней при условии, что она не является препятствием;
- в каждый момент времени агенту доступна информация о наличии/отсутствии препятствий лишь в ограниченном препятствиями и дальностью видимости множестве клеток;
- также дана клетка-цель, в которую агент должен придти из начальной клетки (или понять, что пути не существует).

Отметим, что агенту нужна дополнительная память, чаще всего – одна или более ячеек для сохранения позиций. Известно, что без дополнительной памяти невозможно построить алгоритм, успешно работающий на произвольном поле. Данная задача имеет множество эвристических решений, известных как алгоритмы семейства BUG.

Очевидно, что прежде чем решать какие-либо задачи навигации с помощью генетических алгоритмов, необходимо убедиться в целесообразности такого подхода на сравнительно простой задаче, примером которой является рассматриваемая. В случае успешного решения такой задачи, можно усложнить ее, перейдя в непрерывное пространство, добавив зону видимости и больше памяти. В случае же неудачи можно будет сделать определенные выводы о применимости вышеупомянутой методики в решении задач навигации. Во всяком случае, автор придерживается мнения, что решение данной задачи является первым и необходимым этапом в попытке применить генерацию конечных автоматов для задач навигации с помощью генетических алгоритмов.

Цель работы

1. Построить с помощью генетического алгоритма конечный автомат, решающий рассматриваемую задачу.
2. Оценить целесообразность применения генерации конечных автоматов с помощью генетических алгоритмов для решения более сложных задач навигации.

Базовые положения исследования. В данной работе для решения описанной задачи навигации генерируется конечный автомат Мили, в качестве входных воздействий использующий предопределенный набор предикатов и имеющий предопределенный набор выходных воздействий. Функция приспособленности автомата измеряется путем его тестирования на наборе тестов, который в свою очередь также эволюционирует с помощью генетического алгоритма или иных методик.

Промежуточные результаты. Получен автомат, реализующий эвристику distant bug. Однако, автомат работоспособен только в том случае, когда существует путь из начальной клетки в конечную, в противном случае автомат «зациклится».

УДК 519.683.8

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ АВТОМАТНОГО ПОДХОДА

А.В. Бульёнов
Научный руководитель – д.т.н., профессор А.А. Шалыто

Существующие подходы к разработке web-приложений имеют ряд важных недостатков: они не учитывают высокую иерархическую сложность приложений, не обладают достаточной понятностью и наглядностью. Кроме того, существует мало методик, позволяющих выполнять разработку, отладку и тестирование в комплексе, без применения сторонних средств. Развитие и активное внедрение web-фреймворков приводит к тому, что разработчик зачастую не понимает, как работает то или иное приложение. Данная работа описывает процесс автоматизации разработки web-приложений с использованием автоматного подхода, который был лишен следующих недостатков:

1. отсутствия иерархичности разработки («MVC-приложения, основанные на автоматах»[1]);
2. отсутствия единой методики разработки всего приложения (Microsoft WPF).

Цель работы – создать эффективную методику разработки web-приложений, отвечающую следующим требованиям:

1. Удобство и единообразие разработки на различных этапах создания web-приложения.
2. Высокая гибкость и масштабируемость.
3. Наличие общего визуального представления для разработки приложения и возможность генерации кода из представления.
4. Низкий порог вхождения.
5. Документируемость.
6. Большие возможности для отладки и тестирования ПО.

В качестве основного подхода в разработке методики был выбран автоматный подход. Конечные автоматы обладают высокой гибкостью и масштабируемостью, имеют свое

[1] David J. Anderson, A State Machine For Web MVC. http://bit.ly/bXxtvV

графическое представление, могут использоваться разработчиками различных уровней подготовки. Кроме того, автоматы активно используются в моделировании взаимодействий страниц и скриптов, а также в тестировании web-приложений.

Общий алгоритм работы большинства web-приложений таков:

1. Сервер получает адрес запрашиваемого ресурса (URL) и передает его некоему обработчику.
2. Переданный адрес проверяется обработчиком на соответствие заранее заданному списку шаблонов URL. Каждому шаблону URL поставлены в соответствие один или несколько модулей для генерации страницы.
3. Обработчик выполняет связанный с текущим адресом модуль или последовательность из нескольких модулей.
4. По окончании выполнения осуществляется вывод переменных в шаблоне.
5. Заполненный шаблон отдается браузеру клиента в виде web-страницы.

В соответствии с алгоритмом работы разработка поделена на три уровня: разработка URL-структуры приложения, разработка взаимодействия модулей в каждом URL, разработка каждого модуля в отдельности.

Разработка URL-структуры предусматривает представление обработчика URL в виде конечного автомата. Каждый переход в таком случае – правило обработки адреса. Каждое состояние – модуль или группа модулей, которые применяются после перехода в это состояние. Обработчик получает на вход URL и, в зависимости от формата адреса, инициирует работу тех или иных модулей. Каждое состояние предусматривает свой шаблон отображения, однако применяется только последний шаблон.

Взаимодействие модулей, применяемых для конкретного шаблона URL, также представлено в виде конечного автомата: каждое состояние – единичный модуль. На практике единичный модуль представляет собой класс с унифицированными вызовами. Переходы инициируются внутри модулей и могут использоваться разработчиком для:

1. последовательного перехода между действиями;
2. вызовов модулей-обработчиков ошибок и модулей, осуществляющих перенаправление на другой адрес;
3. ветвления и перехода на другой модуль.

Каждый модуль представлен в виде конечного автомата. Состояния модуля – функции внутри класса-модуля. Переходы между состояниями – вызовы функций. Существует заранее определенный список готовых функций, который может быть расширен разработчиком.

По окончании проектирования выполняется генерация кода, которая формирует набор готовых классов-модулей с пустыми функциями и файл с описанием последовательности обработок URL и взаимодействием модулей.

Подобный подход позволит облегчить отладку, так как средства для нее могут быть встроены в генерируемый код и отладка может выполняться с использованием графического представления, реализованного в процессе проектирования.

В ходе работы были реализованы инструментальные средства для реализации описанного подхода. Готовый web-фреймворк был расширен: заменен обработчик URL и изменена схема работы модулей. В данный момент реализованное программное обеспечение находится в стадии активного тестирования.

УДК 681.322.067

МОДЕЛЬ ОЦЕНИВАНИЯ ТРАФИКА ОТКРЫТОЙ СЕТИ В ЗАДАЧАХ ТЕХНИЧЕСКОЙ РАЗВЕДКИ ОБЪЕКТОВ ИТ ИНФРАСТРУКТУРЫ

Е.Ю. Васильева
Научный руководитель – к.т.н. М.В. Тарасюк

Классификация и модель трафика позволяет демаскировать сетевые программы, функционирующие в защищенной сети на основе сопоставления данной конкретной программы с характеристическими параметрами трафика, наблюдаемыми в открытой части сети. Демаскирование легко достигается при условии принятия достаточно «контрастных» моделей трафика. Соответственно, в ряде случаев использования специализированных приложений, определяющих критичные с точки зрения информационной технической разведки (ИТР) функции объектов, необходимо принятие мер по защите трафика от демаскирования.

Для разных наблюдаемых параметров такая защита может строиться различными способами. Например, для сетевых адресов или номеров виртуальных каналов VPN могут использоваться механизмы трансляции адресов источника (NAT/PAT).

Для длин пакетов защита может обеспечиваться выравниванием пакетов на заданную границу длины или вставка в пакеты ложных байтов данных.

Для измеряемых параметров, таких, как, например, длительность межпакетных интервалов, одним из возможных решений является передача в каналы связи маскировочных (ложных) пакетов, аналогично тому, как это делается в алгоритмах защиты от скрытых каналов.

Для проверки эффективности противодействия описанному методу разведки, подаваемый на вход детектора ИТР трафик, предварительно смешивается с маскировочным трафиком, который представляет собой, например, передачу пакетов случайной длины в случайные моменты времени. Оценка параметров алгоритма маскировки должна базироваться исходя из задания допустимых ограничений на время наблюдения и вероятность обнаружения (или навязывания) демаскирующей последовательности пакетов.

Алгоритм заключается в том, что по пробному трафику (обучающему) мы ищем сначала одинаковые тройки, затем строим из них автомат. И уже по построенному автомату ищем алгоритмом Ахо-Корасик совпадения в исследуемом трафике. Получив количество повторений можно оценить вероятность обнаружения предварительного вмешательства в исследуемый трафик.

Результаты, полученные в работе, показывают, что для рассмотренного простейшего случая маскировочный трафик сравнительно небольшой интенсивности, значительно искажает демаскирующие признаки. Фактически демаскирование оказывается возможным, если характерные последовательности пакетов наблюдаются на интервале времени, много меньшем, чем средний интервал передачи маскировочных пакетов. Однако при этом естественно ожидать, что более сложные модели трафика (например, длинные последовательности пакетов с высоким уровнем внутренней корреляции между наблюдаемыми параметрами) будут более устойчивы и потребуют маскировочного трафика значительно большей интенсивности.

В целом, можно говорить о том, что для особо критичных объектов, в качестве единственного (надежного) решения по скрытию охраняемых функций объекта может являться лишь исключение выдачи (в открытом виде) служебной информации в канал связи и полная (дополняющая) маскировка длин пакетов и межпакетных интервалов. Данное решение, с точки зрения эффективности использования ресурсов связи, эквивалентно применению в сети линейных шифраторов.

Оценивая полученные нами результаты можно сделать вывод о достаточно высоких возможностях системы распознавания и классификации объектов разведки, базирующейся на анализе демаскирующих признаков сетей шифрованной пакетной связи.

УДК 519.178

ОПТИМАЛЬНЫЕ УКЛАДКИ ГРАФОВ В ПРОСТРАНСТВЕ И ИХ ПРИЛОЖЕНИЯ К ЗАДАЧЕ ВЫПОЛНИМОСТИ

С.Э. Вельдер

Научный руководитель – д.т.н., профессор А.А. Шалыто

В докладе рассматривается задача оптимальной укладки графа в многомерном пространстве. Критерием оптимальности является объём области, в которую вложен граф. При вложении запрещаются пересечения рёбер; вершины графа реализуются шариками фиксированного радиуса, а рёбра – криволинейными проволоками фиксированной толщины.

Данная задача актуальна в рамках следующих приложений: визуализация связей; реализация логических схем на кристалле; построение кодов, исправляющих ошибки. Эти приложения требуют укладки графов на плоскости (в двумерном пространстве) или в трёхмерном пространстве, оптимальной по каким-либо критериям.

Различными авторами ранее были получены следующие асимптотические оценки на объём вложения регулярного графа из n вершин: $\Omega\left(n^2\right)$ для вложения в плоскость и $\Omega\left(n\sqrt{n}\right)$ для вложения в трёхмерное евклидово пространство.

В рамках нашего исследования было обнаружено ещё одно приложение оптимальной укладки графов – эффективные алгоритмы для задачи выполнимости булевых формул (Satisfiability problem, SAT). Все существующие алгоритмы решения SAT работают за время $O(a^n)$, где n – длина входа, $a > 1$ – некоторая константа. Для решения этой задачи могут быть полезны вложения графов в пространство размерности больше трёх. В данной работе доказано, что если бы регулярный граф из n вершин удалось вложить в выпуклую k-мерную область объёмом $V(n) = \underset{\sim}{o}\left(n^{\frac{k}{k-1}}\right)$, то SAT можно было бы решить за время $o(a^n)$ для всех $a > 1$, в идеале – за $\widetilde{O}\left(a^{\sqrt{n}}\right)$. Таким образом, сведение задачи выполнимости к задаче оптимальной укладки графов является дополнительной эвристикой для SAT.

Следующее утверждение, доказанное в работе, является отрицательным результатом о возможности укладки. Оно формулируется так: если регулярный граф с достаточно большим числом вершин n реализован в k-мерном пространстве, то для объёма вложения в худшем случае верна асимптотическая нижняя оценка $\Omega\left(n^{\frac{k}{k-1}}\right)$. Данная оценка является обобщением известных ранее оценок для плоскости и трёхмерного пространства на случай произвольной размерности.

Чтобы выяснить, насколько часто реализуется этот худший случай, в работе было доказано достаточное условие, которому должен удовлетворять граф, чтобы выполнялась эта нижняя оценка. Графы, удовлетворяющие этому специфическому условию расширения, являются разновидностью экспандеров.

В работе была доказана точная формула, дающая в явном виде число таких экспандеров. Исходя из этой формулы, было доказано, что такими экспандерами являются почти все

регулярные графы. Более точно, если через $E(n)$ обозначить число n-вершинных экспандеров данного типа, а через $U(n)$ – число всех n-вершинных регулярных графов, то $\lim_{n\to\infty}\frac{E(n)}{U(n)}=1$.

УДК 004.4’242

ВЕРИФИКАЦИЯ АВТОМАТНОЙ МОДЕЛИ МОБИЛЬНОГО БАНКОВСКОГО ПРИЛОЖЕНИЯ

С.И. Гиндин
Научный руководитель – д.т.н., профессор А.А. Шалыто

Краткое вступление, постановка проблемы. Основа успешного банковского бизнеса в области платежных карт – наличие достаточного количества удобных интерфейсов для клиентов. Примеры таких интерфейсов – это интернет-банк (web banking), мобильный банк (mobile banking), банкоматы (ATM, Automated Teller Machine), терминалы самообслуживания (self service), торговые терминалы (POS, Point Of Sale), интерактивная система самообслуживания по телефону (IVR, Interactive Voice Response), автоматизированное место оператора (APW, Automated Working Place). Мобильный банк – интерфейс, получающий все большую популярность и постепенно оттесняющий назад более привычные каналы обслуживания, благодаря невероятно широкому распространению мобильной телефонии. Здесь число услуг ограничено возможностями мобильного телефона как устройства, но присутствуют наиболее популярные. С развитием и стандартизацией возможностей, предоставляемых современными мобильными телефонами, появилась заинтересованность в приложениях для этой платформы аналогичных по удобству и функциональности другим каналам обслуживания. Рынок приложений для мобильных телефонов очень быстро вырос и по праву стал одним из самых развивающихся в информационном пространстве, что повысило требования к качеству и надежности программ. Возникла необходимость в верификации – математически строгом методе проверки того, что программа соответствует заданной спецификации.

Цель работы. Необходимо разработать общую методику построения автоматной модели и ее последующей верификации для программ на языке MBML (Mobile Banking Menu Language), используемом при написании мобильных банковских программ.

Базовые положения исследования. Язык MBML (Mobile Banking Menu Language) основан на принципах соответствия своей предметной области и системной архитектуре, гибкости и расширяемости. **Меню (Menu)** – общее название программ на MBML, после предварительного преобразования средствами серверного приложения они загружаются в интерпретатор, установленный в мобильном телефоне. **Главное назначение MBML –** решение задач предоставления пользователем мобильного интерфейса к банковским услугам. С помощью MBML созданы типовые решения программы-клиента для доступа к сервисам мобильного банка, а также большое число специализированных решений, ориентированных на конкретных заказчиков

Промежуточные результаты. Показано, что язык MBML позволяет рассматривать вопрос об автоматическом построении автоматных моделей для программ на нем с помощью метода Model Checking.

Предложена модель для программ на языке MBML, в которой поведение строится на основе работы конечных автоматов. Приведено обоснование того, что конечный автомат естественным образом моделирует работу программы, доказана корректность построения модели исходной программы относительно основных свойств, предъявляемых к программам на MBML.

Проведен информационно-аналитический обзор существующих средств верификации на основе Model Checking с учетом требований к размеру пространства состояний и основной области использования и выделено приоритетное решение.

Предложен способ автоматической реализации транслирующего отображения программ на языке MBML с помощью построенной автоматной модели для верификации выбранным средством, реализован транслятор, выполняющий данное отображение.

Основной результат. Создана возможность верификации программ на языке MBML, позволяющая выявлять ошибки в мобильных банковских приложениях. Показано практическое применение автоматных подходов к построению моделей и их верификации, позволяющее добиться повышения надежности кода и следующего из этого экономического эффекта.

УДК 004.4’242

РАЗРАБОТКА МЕТОДОВ АВТОМАТИЗИРОВАННОЙ НАСТРОЙКИ И ВЫБОРА ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ С ИСПОЛЬЗОВАНИЕМ МУРАВЬИНЫХ АЛГОРИТМОВ

В.К. Горбунов
Научный руководитель – ассистент Ф.Н. Царев

Постановка задачи. В настоящее время для решения оптимизационных задач часто применяются генетические алгоритмы (ГА). При их применении для решения оптимизационной задачи, как правило, выполняются следующие шаги:

1. Выбор типа используемого генетического алгоритма.
2. Подбор параметров генетического алгоритма для обеспечения как можно более высокой скорости поиска решения.
3. Реализация генетического алгоритма на языке программирования.
4. Проведение вычислительных экспериментов (с возможной корректировкой параметров алгоритма).

Первые три шага, как правило, выполняются вручную на основе некоторых эвристических соображений. **Целью настоящей работы** является разработка методов, которые позволят автоматизировать первые два шага – выбор генетического алгоритма и подбор его параметров.

Описание существующих методик. Для подбора параметров генетических алгоритмов существуют подходы, основанные на методах математической статистики. Одним из недостатков этих подходов является то, что при их использовании алгоритмы для анализа и настройки необходимо перезапускать заново, что увеличивает затраты времени.

Также известен подход, называемый «Метагенетическое программирование». При его использовании в процессе работы настраиваются применяемые операторы скрещивания и мутации.

Описание предлагаемого метода. Предлагаемый метод состоит из нескольких этапов:

1. Выбор набора генетических алгоритмов, которые будут использоваться для решения

оптимизационной задачи.

2. Разработка модулей метагенетического программирования, которые будут осуществлять настройку выбранных генетических алгоритмов. Функция приспособленности в рамках метагенетического программирования вычисляется на основе значений функций приспособленности особей, построенных в рамках алгоритма, и на основе затрат времени на работу алгоритма.
3. Для распределения процессорного времени между генетическими алгоритмами используется муравьиный алгоритм.

Экспериментальное исследование метода. Экспериментальное исследование разработанного метода проводилось на двух задачах:

- Задача об «Умном муравье»

Игровое поле: Тор – 32×32.

На поле расположено 89 клеток с едой.

Ограничение на количество ходов: 200.

Расположение еды и начальная позиция муравья фиксированы.

Муравей умеет определять находится ли непосредственно перед ним еда.

За один игровой ход он может совершить одно из четырех действий:

1. сделать шаг вперед, съедая еду, если она там находится;
2. повернуть налево;
3. повернуть направо;
4. ничего не делать.

Способ описания муравья – конечный автомат.

Цель – создать муравья, который съест всю еду.

- Задача «Построение лабиринта»

Игровое поле: Прямоугольник – 25×15.

Жук добирается до выхода по фиксированному алгоритму.

Начальная позиция жука и расположение выхода фиксированы.

Жук помнит сколько раз он был в каждой клетке поля.

На каждом ходу жук ориентирован в направлении своего последнего движения, изначально он смотрит вниз.

За один игровой ход жук перемещается в соседнюю клетку поля по следующему алгоритму:

1. Из соседних с ним клеток он выбирает свободные от стенок.
2. Из выбранных – выбирает с минимальным числом посещений.
3. Из оставшихся – выбирает одну, полагаясь на следующие приоритеты:
   - клетка, в направлении жука;
   - клетка снизу;
   - клетка справа;
   - клетка слева;
   - клетка сверху.

Двигается в выбранную клетку.

Цель – расставить стенки в клетки поля таким образом, чтобы жук добирался до выхода максимальное число шагов. При этом, выход должен оставаться достижимым из начальной клетки с жуком.

Модуль метагенетического программирования показал интересное поведение – адаптацию алгоритма к текущему множеству решений, которое эволюционирует в алгоритме.

Модуль распределения процессорного времени выделил наиболее эффективный из алгоритмов и предоставил ему почти все процессорное время.

Выводы. Предложен метод автоматизированной настройки и выбора генетических алгоритмов с использованием муравьиных алгоритмов. Проведено экспериментальное исследование разработанного метода на задаче об «Умном муравье» и на задаче «Построение лабиринта».

УДК 004.4`242

ПРЕДСТАВЛЕНИЕ ФУНКЦИИ ПЕРЕХОДОВ ЛИНЕЙНЫМИ БИНАРНЫМИ ГРАФАМИ ПРИ ГЕНЕРАЦИИ АВТОМАТОВ УПРАВЛЕНИЯ С ПОМОЩЬЮ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

В.Р. Данилов
Научный руководитель – д.т.н., профессор А.А. Шалыто

Генетическое программирование – метод автоматической генерации программ на основе эволюционных алгоритмов, использующий представление программ на высоком уровне абстракции. Методы, основанные на представлении функции переходов полными таблицами, оказываются неприменимыми при генерации автоматов со сложной функцией переходов.

В настоящей работе предлагается метод представления функции переходов автоматов управления, основанный на линейных бинарных графах. Наиболее близким к излагаемому в этой статье является представление, основанное на деревьях решений.

Разрешающая диаграмма является удобным способом задания булевой функции, зависящей от конечного числа булевых переменных. Важным частным случаем разрешающих диаграмм являются линейные разрешающие диаграммы или линейные бинарные графы. Узлы линейного бинарного графа могут быть пронумерованы таким образом, что из каждого нетерминального узла одно из ребер ведет в узел со следующим номером.

Автором предлагается метод определения функции переходов линейными бинарными графами. Для этого применяется следующее преобразование автомата: вместо функций переходов и действия для автомата в целом определяются функции переходов в отдельном состоянии. Каждая из таких функций может быть выражена с помощью некоторого линейного бинарного графа.

Автором разработаны генетические операции над автоматом, представленным линейными бинарными графами. При этом считается, что число состояний в автомате фиксировано. Генетические операции над автоматами выполняются через операции над линейными бинарными графами, например скрещивание автоматов – линейными бинарные графы в соответствующих состояниях скрещиваются.

Для проведения операций над линейными бинарными графами в работе разработано представление линейного бинарного графа в виде хромосомы. Указанная хромосома может быть представлена в виде строки, отдельно выделяя части, кодирующие терминальные и нетерминальные узлы.

Для скрещивания линейных разрешающих диаграмм предлагается применять кроссовер на строках. При этом часть, соответствующая терминальным узлам, должна быть скопирована

полностью из одной родительской хромосомы. Операция мутации определяется через мутацию над строкой, соответствующей хромосоме.

Разработанный подход был протестирован на задаче «Умный муравей-2». Анализ результатов показывает, что предложенный метод на указанной задаче является более эффективным по сравнению с представлением функции переходов полными таблицами.

УДК 004.4’242

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ К ГЕНЕРАЦИИ ТЕСТОВ ДЛЯ АВТОМАТНЫХ ПРОГРАММ

А.Ю. Законов
Научный руководитель – к.т.н. О.Г. Степанов

Тестирование является важным и трудоемким этапом разработки любого программного обеспечения. Автоматное программирование предполагает описание логики программы как модель какого-либо формального автомата, часто используется расширенный конечный автомат. Автомат, можно верифицировать или применять model checking для поиска ошибок. При решении любой прикладной задачи автомат взаимодействует с объектами управления, которые нельзя подвергнуть верификации. Тестирование позволяет искать ошибки в системе в целом.

Составление большого числа тестов и их прогон вручную требует больших временных затрат. В данной работе предлагается подход для автоматизации тестирования автоматных программ.

Для заданного сценария необходимо сгенерировать код теста, при запуске которого будет реализована требуемая последовательность событий и переходов, выполнена проверка требований спецификации, включенных в модель. Для этого необходимо:

- подобрать значения параметров, которые будут удовлетворять всем условиям на переходах;
- сгенерировать из спецификации тестовые оракулы, которые позволят автоматически проверить, соответствует ли реализация программы для этого сценария ее спецификации во время выполнения теста;
- попробовать подобрать такие значения параметров, которые приведут к нарушениям тестовых оракулов, т.е. выявят ошибки в системе – несоответствия реализации и спецификации.

Автоматная модель является наглядным и удобным способом описания поведения системы. Расширенная переменными и условиями на переходах автоматная модель позволяет внести большую часть спецификации в модель системы. Таким образом, объекты управления становятся максимально простыми, они не содержат логику работы системы, а отвечают только за свои непосредственные задачи. При данном подходе для тестирования системы достаточно предположения, что объекты управления соответствуют своей спецификации.

Предложенный подход позволяет включать спецификацию объектов управления в модель в виде контрактов, что позволяет убрать зависимость системы от конкретной реализации объектов управления. Это дает возможность автоматически тестировать не только автоматную модель системы, но всю систему в целом – приближено к тому, как она будет работать в реальном мире.

Модель, которая содержит в себе спецификацию, выраженную в виде охранных условий на переходах автомата, пред- и постусловий на переходах и инвариантах для состояний, позволяет

автоматизировать процесс создания тестов. Тестовый сценарий задается как последовательность переходов в автомате. Для выполнения заданной последовательности переходов в сгенерированном тесте требуется подобрать значения внешних переменных, то есть тех переменных, значения которых на практике будут приходить из объектов управления. В работе используется генетический алгоритм для поиска подходящего набора значений внешних переменных. Фитнес-функция принимает на вход массив чисел, которые являются значениями переменных, и оценивает, сколько переходов из заданной последовательности было успешно выполнено. Для всех невыполненных охранных условий и предусловий на переходах производится оценка того, насколько сильно это условие было нарушено и добавляется к значению фитнес-функции, с учетом положения этого условия в последовательности переходов. Генетический алгоритм применяется для поиска решения с наименьшим значением фитнес-функции. Значения переменных, результат фитнес-функции для которых будет ноль, подходят для создания кода теста, так как удовлетворяют всем условиям модели. При запуске сгенерированного теста динамически проверяется выполнение требований спецификации, записанных в модели в виде контрактов на языке JML.

В рамках данной работы были достигнуты следующие результаты:

1. Предложен подход к тестированию автоматных программ.
2. Предложен способ описания спецификации взаимодействия автомата и объектов управления.
3. Разработан алгоритм поиска параметров для выполнения заданного сценария.
4. Предложен способ автоматической генерации теста по описанному сценарию, в котором тестовые оракулы учитывают требования спецификации, записанные для объектов управления и автомата.
5. Предложенный подход не только позволяет искать несоответствия между системой и ее спецификацией, но также генерировать корректные тесты для регрессионного и нагрузочного тестирования.

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ НА ОСНОВЕ ОБУЧАЮЩИХ ПРИМЕРОВ ДЛЯ ПОСТРОЕНИЯ КОНЕЧНЫХ АВТОМАТОВ ДЛЯ УПРАВЛЕНИЯ МОДЕЛЬЮ БЕСПИЛОТНОГО САМОЛЕТА

А.В. Александров, С.В. Казаков, А.А. Сергушичев
Научный руководитель – ассистент Ф.Н. Царев

В настоящее время одной из актуальных задач является беспилотное управление транспортными средствами. Одним из подходов к решению этой задачи является построение управляющих конечных автоматов с помощью генетических алгоритмов. В статье Н.И. Поликарповой, В.Н. Точилина, А.А. Шалыто «Применение генетического программирования для генерации автоматов с большим числом входных переменных» (Научно-технический вестник СПбГУ ИТМО. Выпуск 53. Автоматное программирование, с. 24–42) для этого применяется генетический алгоритм с использованием метода сокращенных таблиц для представления конечных автоматов. При этом вычисление функции приспособленности основано на моделировании, поэтому оно занимает достаточно большое время.

Целью настоящей работы является разработка метода, лишенного указанного недостатка. Разрабатываемый метод основан на генетическом алгоритме, при вычислении функции приспособленности в котором используются обучающие примеры.

Для моделирования беспилотного самолета применяется свободный авиасимулятор FlightGear (http://www.flightgear.org), который позволяет осуществлять программное управление самолетом, а также сохранение параметров полета (скорость, направление полета и т.д.) и состояния самолета (положение руля, элеронов, состояние стартера и т.п.).

Обучающий пример состоит из двух последовательностей: последовательности входных данных (описывают состояние самолета и окружающей среды) и соответствующей ей эталонной последовательности выходных воздействий (описывает изменение параметров самолета). Для создания обучающего примера записывается полет под управлением человека. В настоящее время построение автомата осуществляется на основе одного обучающего примера, а в дальнейшем планируется расширить метод на большее их число.

В настоящей работе управляющий конечный автомат представляется в виде таблицы переходов. Для каждого состояния и события хранится переход в новое состояние (возможно, то же самое), а также действие (возможно, пустое), совершаемое при указанном переходе.

В зависимости от входных данных генерируется набор событий (возможные типы событий задаются вручную до начала работы генетического алгоритма), подаваемых на вход автомату, который, в свою очередь, генерирует выходные воздействия. Для вычисления функции приспособленности сгенерированная последовательность выходных воздействий сравнивается с последовательностью выходных воздействий из обучающего примера.

В процессе сравнения набор выходных воздействий в каждый момент времени рассматривается как вектор, что дает возможность определить расстояние между соответствующими наборами в последовательностях. После этого сами последовательности ввиду их одинаковой длины можно также рассматривать как векторы (элементами которых, в свою очередь, являются векторы), а потому таким же образом можно вычислить расстояние и между ними. Далее вычисляется частное от деления полученного расстояния на расстояние от эталонной последовательности выходных воздействий до последовательности, состоящей из нулей. От полученного числа вычисляется обратная экспонента. Результирующее значение функции приспособленности (при расширении метода на несколько обучающих примеров) получается в результате усреднения этой величины по всем обучающим примерам.

На данном этапе авторами получен автомат управления взлетом самолета. В дальнейшем планируется разбить полет самолета на этапы – разгон, взлет, посадка и т.д. Это позволит учитывать специфику каждого из этих этапов и разрабатывать управляющие автоматы для каждого из них отдельно. Эти автоматы, в дальнейшем, могут быть скомбинированы в один, который будет полностью управлять моделью беспилотного самолета.

УДК 519.681.3

«БЕСПЛАТНЫЕ ТЕОРЕМЫ» ДЛЯ ТЕРМОВ

Е.Р. Кирпичев
Научный руководитель – к.т.н., доцент Г.А. Корнеев

В 1989 г. Филип Вадлер в работе «Theorems for free!» заметил, что Теорема о параметричности, доказанная Джоном Рейнольдсом, позволяет, зная лишь тип полиморфной функции, сформулировать свойство, которому она удовлетворяет. Такие свойства и метод их получения были названы «бесплатными теоремами». Указанный результат может быть применен для доказательства корректности программ и их эквивалентных преобразований. В последующих работах эти результаты были обобщены на ряд расширений полиморфного лямбда-исчисления. Одновременно с этим был создан инструмент для автоматического порождения «бесплатных теорем».

Однако полезность «бесплатных теорем» не высока, так как они позволяют доказывать лишь узкий класс свойств функций. Это объясняется «скромностью» исходных данных: при построении теоремы используется информация только о типе функции.

До настоящего времени развитие аппарата построения «бесплатных теорем» осуществлялось в направлении обобщения на более богатые системы типов. Например, системы типов с классами типов, и системы, допускающие отсутствие нормализации.

В данной работе рассматривается возможность формулирования «бесплатных теорем» для функций с частично заданной реализацией. Это должно позволить получать более сильные «бесплатные теоремы», чем при классическом подходе.

При построении «бесплатной теоремы» для функции с реализацией, частично заданной в виде терма со свободными переменными, можно выделить два случая. Если терм аппликативен (т.е., не использует лямбда-абстракций), то необходимо сформулировать «бесплатную теорему» для каждой входящей в терм свободной переменной. Эти теоремы и правила типизации позволяют записать систему уравнений, решением которой является «бесплатная теорема» для терма в целом. В случае неаппликативного терма, он преобразуется в аппликативную форму при помощи комбинаторов, после чего к нему применяется процесс, описанный в первом случае. При этом используется полнота базиса комбинаторов.

Таким образом, применение информации о реализации функции в виде термов позволяет формулировать более сильные «бесплатные теоремы», чем в случае использования только информации о типе. В работе планируется исследовать границы применимости предлагаемого подхода и полезность его результатов. Также планируется разработать и реализовать алгоритмы построения таких «бесплатных теорем» в тех случаях, когда это возможно.

УДК 004.8

АВТОМАТИЧЕСКИЙ СИНТЕЗ СИСТЕМЫ УПРАВЛЕНИЯ МОБИЛЬНЫМ РОБОТОМ ДЛЯ РЕШЕНИЯ ЗАДАЧИ «КЕГЕЛЬРИНГ»

С.А. Алексеев, А.И. Калиниченко, В.О. Клебан
Научный руководитель – д.т.н., профессор А.А. Шалыто

Для построения систем управления мобильными роботами целесообразно использовать технологию автоматного программирования, в которой, в частности, предлагается строить программу как систему автоматов, взаимодействующих между собой за счет вложенности и вызываемости. Использование автоматного подхода при создании подобных систем управления обладает рядом достоинств, таких как: документируемость, возможность построения компактных моделей Крипке в целях верификации, упрощение внесения изменений и т.д.

Встречаются задачи, для которых известно, что они могут быть решены при помощи конечных автоматов, но эвристически построить для них автомат чрезвычайно сложно. Попытки решения подобных задач осуществляются, в частности, при помощи различных методов автоматического синтеза программ. Одним из таких методов является генетическое программирование.

Эффективность применения генетического программирования для синтеза автоматов показана в работах, но, к сожалению, ни в одной из этих работ не проверялась работа систем управления на реальных объектах.

В данной работе приводится пример автоматического синтеза системы управления роботом для задачи «Кегельринг» и ее проверка на реальном мобильном роботе.

Целью системы управления роботом в задаче «Кегельринг» является вытолкнуть из ринга расположенные в нем кегли, при этом, не выходя за пределы круга, ограничивающего ринг. Отметим, что сделать это необходимо за наиболее короткое время.

Для решения задачи «Кегельринг» разработан программно-аппаратный комплекс, который включает в себя среду эмуляции и приложение, реализующее генетический алгоритм.

Виртуальная модель робота была оснащена двумя моторами, дальномером и датчиком линии эквивалентными тем, которые установлены на реальном роботе.

В качестве элемента обеспечения работы виртуальной модели был разработан специальный интерпретатор, представляющий из себя программу на языке Java. С помощью этого интерпретатора исполняется автомат управления роботом.

Кроме того, был создан контроллер-супервизор, который отслеживает перемещения модели робота и кеглей, и на основе этих данных вычисляет значение функции приспособленности.

Алгоритм осуществляет селекцию, скрещивание и мутацию, в соответствии с полученными значениями функции приспособленности.

УДК 004.8

РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ МАЛОРАЗМЕРНЫМ ВЕРТОЛЕТОМ

В.О. Клебан, И.В. Широков
Научный руководитель – д.т.н., профессор А.А. Шалыто

В настоящее время ряд исследовательских групп изучает вопросы построения систем управления беспилотными летательными аппаратами, действующими внутри помещений.

При полете внутри помещения возникает необходимость преодоления коридоров, дверных проемов и других препятствий, что накладывает весьма жесткие требования на управляемость роботом. Система управления робота должна содержать приборы, позволяющие ему оценивать собственное состояние и исследовать окружающую среду.

В данной работе описывается система управления соосным микровертолетом обеспечивающая полет внутри помещения.

Управление вертолетом осуществляется по четырем каналам. Вертикальное движение осуществляется при помощи оборотов двигателя, управление курсом происходит при помощи рыскания, горизонтальный полет регулируется при помощи крена и тангажа.

Стандартная модель микровертолета подверглась существенной модернизации силовой установки и имеющейся системы радиосвязи. Также на вертолет были установлены бортовая ЭВМ и твердотельная курсовертикаль собственной разработки.

Бортовая ЭВМ представляет собой контроллер на базе архитектуры ARM7, снабженный тремя датчиками угловых скоростей, акселерометром, магнитометром, а также входами для подключения датчиков высотомера и дальномеров.

На базе данной ЭВМ была разработана курсовертикаль, показывающая ориентацию вертолета в пространстве вычисляя углы Эйлера в режиме реального времени.

Основными проблемами при разработке курсовертикали оказались:

- температурный дрейф и «уход нулей» показаний датчиков угловых скоростей, что приводило к линейно возрастающей угловой ошибке;
- ошибки калибровки, приводящие к росту угловых ошибок в зависимости от количества движений вертикали;

– собственный шум датчика и шум датчика вследствие вибраций корпуса вертолета, приводящий к росту угловой ошибки.

В целях обеспечения достоверности показаний прибора производилась его тарировка путем многократных поворотов устройства вдоль различных осей и при различных температурах на специальном стенде. Это позволило добиться высокой стабильности и достоверности показаний курсовертикали.

Для решения задачи фильтрации показаний приборов была разработана схема фильтрации сигналов, позволяющая в значительной степени снизить негативные эффекты от попадания помехи в систему. Отметим, что борьба с помехой производилась не только на уровне электронных схем, но и при компоновке вертолета.

К сожалению, избавиться от эффекта «ухода нуля» в предлагаемой схеме фильтрации не удалось.

В дальнейшем был разработан алгоритм, позволяющий производить оценку нулей датчиков угловых скоростей, непосредственно в полете.

Результат работы алгоритма автоматической оценки нуля показал, что алгоритм справляется с поставленной задачей и показывает в два раза меньшую ошибку определения угла.

В силу того, что построить полную модель полета вертолета не удалось, решено было проводить исследования непосредственно на реальном объекте. Практически с самого начала экспериментов стало ясно, что объект обладает сложным поведением, как с точки зрения логики согласования всех систем, так и с точки зрения непосредственно управления полетом. В связи с этим для реализации системы управления решено было применить автоматное программирование, что позволило эффективно реализовать систему автоматического управления вертолетом.

УДК 004.4’242

О ПРИМЕНИМОСТИ ШАБЛОНОВ ТРЕБОВАНИЙ К ФОРМАЛЬНОЙ СПЕЦИФИКАЦИИ И ВЕРИФИКАЦИИ АВТОМАТНЫХ ПРОГРАММ

А.А. Клебанов
Научный руководитель – к.т.н. О.Г. Степанов

Автоматное программирование – это метод разработки программного обеспечения (ПО), основанный на расширенной модели конечного автомата. В рамках данного подхода программы представляются системой автоматизированных объектов управления, логика поведения которых задается системой взаимодействующих управляющих автоматов.

В ряде работ показано, что к автоматным программам хорошо применима верификация на модели (Model Checking). Суть такой верификации состоит в проверке соответствия модели с конечным числом состояний (структуры Крипке) формальной спецификации, заданной в виде набора формул темпоральной логики. При верификации преимуществом автоматного подхода перед традиционными подходами к разработке ПО является высокая степень автоматизации, так как в автоматных программах модель поведения задается априори. Разработаны методы, позволяющие автоматически преобразовывать как управляющие автоматы в модель, пригодную для верификации, так и построенный верификатором контрпример в автоматную модель. Однако как при верификации автоматных программ, так и при верификации программ общего вида, существует следующая проблема – необходимость записи формальных требований в виде

формул темпоральных логик, работа с которыми достаточно трудоемка и требует значительной математический подготовки.

В одной из работ эта проблема частично решается использованием контрактов, которые являются более простым формализмом, однако значительно уступают темпоральным логикам в выразительных возможностях. Контракты к записи требований применимы только в том случае, когда необходимо специфицировать свойства инвариантности, предусловия или постусловия.

В настоящей работе описывается подход к записи требований, скрывающий сложность темпоральных логик. Предлагается записывать требования на подмножестве естественного языка (controlled natural language), заданного формальной грамматикой. Грамматика основывается на наборе шаблонов требований – обобщенном описании (формальном и на естественном языке) часто встречающихся ограничений на допустимые последовательности состояний в модели системы с конечным числом состояний. Таким образом, для каждого полученного требования существует эквивалентная формальная запись, позволяющая осуществить верификацию.

В качестве результатов работы можно выделить два основных положения.

Шаблоны требований применимы для спецификации автоматных программ. В ходе исследования было рассмотрено 77 требований из 15 источников. При этом 87% требований покрывается шаблонами (доказана формальная эквивалентность). Отсюда следует, что их последующая интеграция с инструментальным средством для разработки автоматных программ оправданна. Анализ существующих требований позволил сделать ряд дополнительных выводов:

1. В большинстве случаев перенос требований из спецификации в утверждения об автоматной модели не составляет труда, потому что в модели отражены необходимые сущности.
2. Отсутствуют конструкции, находящиеся на промежуточном уровне между шаблонами требований и элементарными утверждениями об автоматной модели.
3. Разработанная грамматика позволяет записывать верифицируемые требования на естественном языке, что, во-первых, значительно упрощает процесс формального специфицирования, а во-вторых, минимизирует число потенциальных ошибок в самой спецификации.

УДК 004+37:004

АВТОМАТИЧЕСКАЯ ГЕНЕРАЦИЯ ЗАДАНИЙ ДЛЯ ПРОВЕРОЧНЫХ РАБОТ

П.Ю. Маврин, М.В. Григорьева, А.С. Станкевич
Научный руководитель – к.т.н., доцент Г.А. Корнеев

Введение. При обучении важно своевременно проверять знания учащихся. Основная проблема, с которой сталкиваются преподаватели при проведении проверочных работ, – ученики списывают ответы друг у друга. Самый распространенный способ борьбы со списыванием – составление нескольких вариантов заданий. При этом ученики, получившие разные варианты заданий, не могут списать решения друг у друга. Данная система применяется повсеместно и хорошо себя зарекомендовала.

Однако у такой системы есть большой недостаток. Объем работы, который необходимо проделать преподавателю увеличивается пропорционально количеству вариантов. Поэтому, как правило, составляется не более пяти вариантов задания. Такого количества вариантов обычно бывает достаточно, если проверочная работа проводится один раз, одновременно для всех

учеников и под присмотром преподавателя. Если хотя бы одно из перечисленных условий не выполняется, то ученики могут достаточно быстро составить список решений всех вариантов и распространить его между собой.

В данной работе предлагается решить эту проблему при помощи автоматической генерации вариантов задания. При таком подходе можно построить неограниченное (строго говоря, ограниченное, но достаточно большое) число вариантов, что позволит избежать проблем, описанных выше.

Полученные результаты. Изложим кратко суть предлагаемого метода. Вместо составления нескольких вариантов задания строится программа-генератор, составляющая варианты заданий. Чтобы сделать генератор детерминированным (это удобно по ряду причин), будем строить его таким образом, чтобы он получал на вход натуральное число, так называемое «зерно», и выдавал на выход вариант, соответствующий этому зерну.

Передавая на вход генератору разные значения зерна, можно получить различные варианты задания.

Техническая реализация. Процесс генерации вариантов не сильно отличается от генерации других текстовых данных, например, веб-страниц, поэтому в данной работе используется язык описания веб-шаблонов JSP.

Пусть требуется составить задания по переводу чисел из одной системы счисления в другую. Тогда можно, например, составить такой генератор:

```
<%@ page import="static util.Random" %>
Переведите число <%=randomInt(0x100, 0x1000)%> в шестнадцатеричную систему счисления.
```

В данном коде используется метод int randomInt(int min, int max), возвращающий случайное целое число из интервала от min (включительно) до max (не включительно). Данный метод описан в классе util.Random (код не приводим), который импортируется в первой строке генератора.

Запустив приведенный генератор и передав ему в качестве параметра, значение зерна, получим на выходе конкретный вариант задания. Например:

```
Переведите число 1465 в шестнадцатеричную систему счисления.
```

Границы подобраны таким образом, чтобы ответы в получающихся вариантах всегда были трехзначными, при этом сложность всех сгенерированных вариантов будет примерно одинаковой.

Контроль качества составляемых вариантов. К вариантам заданий, предлагаемых на проверочную работу, предъявляется ряд требований. Перечислим некоторые из них:

1. Вариант должен проверять знания ученика по теме проверочной работы.
2. Вариант должен быть не слишком простым.
3. Вариант должен быть не слишком сложным.
4. У ученика не должно быть возможности случайно угадать правильный ответ.

Как составлять варианты удовлетворяющие требованиям? Как в известном анекдоте есть два метода: «подумать» и «потрясти».

Метод 1 «Подумать». Перед тем, как писать генератор, проектируется общий вид качественного (удовлетворяющего требованиям) варианта. Далее составляется генератор таким образом, чтобы получались только такие варианты.

Метод 2 «Потрясти». Составляется простой генератор вариантов, не учитывающий качество генерируемых вариантов, и программа-анализатор, составляющая оценку варианта с точки зрения предъявляемых требований. Далее для того, чтобы сгенерировать качественный

вариант, простой генератор запускается несколько раз до тех пор, пока анализатор не признает сгенерированный вариант качественным.

Метод «Потрясти» отличается большей универсальностью, так как проверить удовлетворение свойств, как правило, проще, чем построить объект, им удовлетворяющий. Однако применение этого метода «в лоб» часто оказывается невозможным (например, если вероятность, случайно, получить качественный вариант крайне низка) или нецелесообразным (когда проще применить метод «Подумать»). Иногда также бывает полезно использовать комбинацию этих методов (запускать «умный» генератор несколько раз, анализируя качество получаемых вариантов).

Генерация ответов. Для удобства проверки заданий преподавателю часто требуется правильный ответ (или набор правильных ответов). Для того, чтобы сгенерировать ответ к заданию можно построить программу-решатель, которая будет получать на вход текст задания и выдавать текст ответа. Такая схема удобна, например, для заданий вида «Что выведет данная программа?». В этом случае для получения правильного ответа программу надо просто скомпилировать и запустить. В других случаях построить такой решатель может быть затруднительно и удобнее использовать схему двухэтапной генерации. На первом этапе по зерну генерируются набор параметров варианта. На втором этапе из этих данных получаются два файла – задание и ответ на него. В этом случае благодаря предварительному этапу решаются две проблемы. Во-первых, решателю не требуется анализировать текст задания, а во-вторых, при надобности в параметры варианта можно включить дополнительные «подсказки» для его решения.

Заключение. Предложенный метод – действенный способ борьбы со списыванием на проверочных работах. Он был апробирован при проведении контрольных и лабораторных работ для студентов на кафедрах «Информационные системы» и «Компьютерные технологии» СПбГУ ИТМО и показал хорошие результаты. Метод применялся при составлении задач по дискретной математике, информатике, программированию, однако он может быть применен и в других областях, например в физике.

Основное достоинство метода – автоматизация создания большого количества вариантов заданий и правильных ответов на них.

Один из главных недостатков метода – обязательное участие программиста в процессе составления генератора заданий. Как показала практика, использование языка JSP сильно упрощает программистскую составляющую процесса построения генератора, однако для дальнейшего развития метода необходимо создание специализированного языка, упрощающего разработку генераторов, что позволит создавать генераторы людям, не являющихся профессиональными программистами (например, учителям математики и физики).

УДК 004.4'242

ВАЛИДАЦИЯ АВТОМАТОВ С ПЕРЕМЕННЫМИ НА ФУНКЦИОНАЛЬНЫХ ЯЗЫКАХ ПРОГРАММИРОВАНИЯ

Я.М. Малаховски
Научный руководитель – к.т.н., доцент Г.А. Корнеев

В работе [1] был предложен подход к реализации событийных конечных автоматов [2] на функциональных языках программирования. При этом для валидации функций переходов использовались свойства алгебраических типов данных.

В настоящей работе рассматривается обобщение метода, предложенного в работе [1], на структурные автоматы. При этом в качестве пометок на переходах автомата указываются не одиночные события, а комбинации входных переменных. Возможность описания структурных автоматов, а также их автоматическая валидация являются ключевыми требованиями при разработке ответственных систем. Однако у большинства распространенных языков функционального программирования системы типов недостаточно выразительны для осуществления валидации функций переходов структурных автоматов. Это связано с тем, что большинство систем типов таких языков программирования сохраняют возможность реконструкции типовых аннотаций в программе и не являются Тьюринг-полными.

При создании способа описания структурного автомата в функциональном языке программирования были разработаны способы описания:

- входных переменных и примитивов;
- логических выражений;
- переходов.

Так как провести валидацию автомата в процессе компиляции не представляется возможным, то она производится при создании состояний автомата во время исполнения программы.

Аналогично методу, используемому в [1], для представления входных переменных применяются алгебраические типы данных. Однако алгебраические типы сами по себе не могут предоставить необходимого синтаксиса и гибкости внутреннего представления для описания булевых выражений. Поэтому в настоящей работе для описания булевых выражений над произвольным множеством входных переменных разработан специальный класс типов (type class).

Валидация функции переходов автомата производится в процессе его конструирования. Это позволяет применять в процессе валидации всю вычислительную мощь базового языка программирования, при этом не допуская использование функций переходов, не прошедших валидацию. Валидатор реализован на базе исчисления секвенций [3].

Для апробации и практического применения предложенного подхода был разработан встроенный в Haskell предметно-ориентированный язык программирования (eDSL), представляющий собой «синтаксический сахар» для реализации функций переходов и автоматов.

Результатами вычислений выражений разработанного eDSL являются части автомата. При этом гарантируется, что их использование возможно только после предварительной валидации.

Литература

1. Малаховски Я.М., Шалыто А.А. Реализация конечных автоматов на функциональных языках программирования //Информационно-управляющие системы. – 2009. – №6. – С. 30–33.

2. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб.: Питер. – 2009. – 176 с.
3. Верещагин Н.К., Шень А. Лекции по математической логике и теории алгоритмов. Часть 2. Языки и исчисления. 2-е изд., стереотипн. – М.: МЦНМО. – 2002. – 288 с.

УДК 004.62

ПОСТРОЕНИЕ СИСТЕМЫ УПРАВЛЕНИЯ НАУЧНЫМИ ДАННЫМИ ДЛЯ ЗАДАЧ МОДЕЛИРОВАНИЯ НАНОРАЗМЕРНЫХ АТОМНО-МОЛЕКУЛЯРНЫХ СТРУКТУР

Ф.В. Подтелкин, С.В. Ковальчук
Научный руководитель – д.т.н., профессор А.В. Бухановский

В настоящее время существует большое количество программных пакетов для квантово-химических вычислений. Преимущества и недостатки каждого из пакетов обусловлены различиями в реализации алгоритмов, точности и наборах выходных данных, быстродействии, использовании системных ресурсов, распараллеливании. Квантово-химические пакеты созданы на основе разных подходов и программно реализованы с помощью различных технологий, обладают различными форматами и способом задания входных и выходных данных. Для выходных данных, полученных в результате расчетов разными пакетами, нет единого средства визуализации. Описанные особенности, а также плохая документированность и сложность настройки вынуждают многих ученых в процессе своих исследований отказываться от объективного выбора пакетов для расчетов в пользу более известных и привычных средств.

Цель данной работы – разработка системы управления научными данными для задач моделирования наноразмерных атомно-молекулярных структур, обеспечивающая: единообразный доступ к данным; унифицированное формирование входных данных; преобразование результатов для визуализации; хранение и каталогизация данных; расчет характеристических значений, используемых для прогнозирования времени расчетов.

В разработанной системе были использованы основные принципы методологии REST [1], что позволило организовать унифицированный доступ к научным данным в разных форматах. Для динамического формирования входных данных для различных пакетов алгоритмов были использованы шаблоны, описывающие структуру и формат входных данных, что обеспечило легкую расширяемость за счет возможности добавления новых форматов. Для преобразования выходных данных из разных пакетов в единый формат был применен метод двухфазного конвертирования, для которого был разработан оригинальный алгоритм преобразования данных. В результате анализа существующих СУБД был сделан вывод, что они мало подходят для решения поставленной задачи: из-за трудности структурирования данных и ограниченности средств хранения неструктурированных данных. Наиболее распространенные СУБД для работы с неструктурированными данными имеют только операции полного чтения и перезаписи и работают медленнее, чем прямое обращение к файловой системе. Поэтому в системе был использован гибридный метод, обеспечивающий структурированное и универсальное хранение данных: метаданные и основные параметры хранятся в базе данных, а сами данные – в виде бинарных файлов.

Разработанная система реализована на языке Java в виде отдельного web-сервиса, что обеспечивает переносимость и возможность интеграции с существующими программными комплексами. Знания о форматах входных и выходных данных реализованы в форме совокупности параметрических моделей. Параметризация моделей и выявление их структуры

производилась с использованием совокупности экспертных знаний в рассматриваемой предметной области и результатов экспериментов. Разработанная система была успешно внедрена и используется в составе программного комплекса [2].

Литература

1. Architectural Styles and the Design of Network-based Software Architectures. R.T. Fielding. 2000. Doctoral thesis, University of California, Irvine. – С. 162.
2. Высокопроизводительный программный комплекс моделирования наноразмерных атомно-молекулярных систем / В.Н. Васильев [и др.] // Научно-технический вестник СПбГУ ИТМО. Выпуск 54. Технологии высокопроизводительных вычислений и компьютерного моделирования. – СПб: СПбГУ ИТМО, 2008. – С. 3–12.

УДК 004.4'242

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ ДЛЯ ГЕНЕРАЦИИ КОНЕЧНЫХ АВТОМАТОВ ПО СПЕЦИФИКАЦИИ

С.О. Попов

Научный руководитель – к.т.н., доцент Г.А. Корнеев

Предыдущие исследования процесса генетического программирования, например в работе [1], показали, что используемые средства недостаточны из-за чего возникают неудобства, например при написании объекта управления. Генетический алгоритм может давать решение, которое учитывает особенности объекта управления и пользуется ими. Поэтому поведение решения при смене объекта управления или изменении его параметров может быть не правильным.

Основная **цель данной работы** это ускорение генерации конечных автоматов и расширение круга решаемых генетическим программированием задач.

Основные задачи данной работы:

- расширение функциональности фреймворка [2] с сохранением работоспособности всех задач;
- минимизация времени, требуемого для получения решения;
- исследование применимости вероятностных особей;
- исследование применимости CTL-формул;
- исследование генерации конечных автоматов по спецификации.

В данной работе используется спецификация для генерации конечных автоматов. Что, в простых случаях, позволяет отказаться от подсчета фитнесс функции и долгого тестирования особи. В других случаях спецификация позволяет сузить область поиска решения и упростить объект управления.

Работа основывается на фреймворке для генетического программирования, описанного в работе [2]. Все методы данной работы доступны и могут быть использованы для практического применения или продолжения исследований. Применение результатов позволяет расширить круг задач решаемых генетическим программированием и определить направление дальнейших исследований.

В процессе исследований был разработан метод построения детерминированных конечных автоматов на основе генетического программирования с особями, являющимися вероятностными автоматами.

Были реализованы и испытаны инструменты для проведения исследований. Определены направления дальнейшей работы.

Результаты испытаний показали возможности создания верифицированных автоматов и ускорение роботы генетического алгоритма.

Литература

1. Применение генетического программирования для реализации систем со сложным поведением [Электронный ресурс] / Н.И. Поликарпова, В.Н. Точилин – Режим доступа: http://vestnik.ifmo.ru/ntv/39/ntv_39.3.3.pdf, свободный.
2. Development of Software System for State Machine Generation Using Genetic Algorithms [Электронный ресурс] / Evgeny Andreevich Mandrikov, Vladimir Anatolievich Kulev, Proceedings of the Second Spring Young Researchers Colloquium on Software Engineering. SPb.: SPbSU. 2008. V. 1. – Pp. 59–60. – Режим доступа: http://is.ifmo.ru/genalg/_mandrikov-kulev_syrcose.pdf, свободный.

УДК 004.021

ВЫЧИСЛЕНИЕ МИНИМАЛЬНОГО РАССТОЯНИЯ И ДЕКОДИРОВАНИЕ НИЗКОПЛОТНОСТНЫХ КОДОВ

Р.В. Сатюков

Научный руководитель – д.т.н., профессор Б.Д. Кудряшов

Краткое вступление, постановка проблемы. В последнее время низкоплотностные (НП) коды (LDPC-коды), предложенные Р. Галлагером [1], привлекают все больше внимания как альтернатива турбо-кодам [2]. Известные НП коды можно разделить на две группы: нерегулярные (случайные или псевдослучайные) коды [3] и регулярные коды [4].

Регулярный (J,L)-НП код определяется двоичной проверочной матрицей, в которой каждая строка содержит ровно L единиц, а каждый столбец ровно J единиц. Этот класс НП кодов содержит как коды, основанные на регулярных графах, так и коды, в основе которых лежат алгебраические структуры.

Важным параметром НП кода является обхват g_T соответствующего графа Таннера. От величины g_T зависит эффективность процедур итеративного декодирования НП кодов.

Для кодов с J=2, проверочная матрица может рассматриваться как матрица инцидентности некоторого графа, поэтому такие коды называют также графовыми кодами. Хотя известно (см., например, [1]) что среди таких кодов нет асимптотически оптимальных, интерес к ним обусловлен малой сложностью их декодирования и существованием в этом классе достаточно эффективных коротких кодов. Кроме того, они могут быть использованы в каскадных конструкциях в сочетании с другими кодами (см., например, [5]).

Заметим, что минимальное расстояние графовых кодов совпадает с обхватом g соответствующего графа, а кодовые слова соответствуют подграфам данного графа, для которых степени всех вершин четны.

В случае, когда J>2, коду соответствует не обычный граф, а гиперграф. В гиперграфе ребра «соединяют» не две вершины, а несколько. Регулярному (J,L)-НП коду соответствует гиперграф, в котором каждое ребро соединяет J вершин и степень каждой вершины равна L. В этом случае неизвестно связи между обхватом и минимальным расстоянием.

Для построения графовых кодов можно использовать их связь со сверточными кодами [4, 5]. Похожий подход можно использовать и для построения кодов с J>2. Однако в этом случае мы получаем код с данным обхватом, минимальное расстояние которого неизвестно.

Цель работы. Для более точной оценки качества кодов требуется найти их минимальное расстояние, а также алгоритм декодирования. В настоящей работе рассматривается подход, который позволяет вычислить минимальное расстояние для НП кодов с J=3, которые получаются с помощью циклического усечения из небольших проверочных сверточных матриц.

Описание подхода. Подход заключается в выделении очередной единицы в текущем синдроме, выборе столбца проверочной матрицы, в котором на соответствующей позиции есть единица и рекурсивном переборе двух вариантов – варианта, когда данный столбце входит в кодовое слово, и варианте когда нет. Кроме того, используя регулярную структуру проверочной матрицы, можно существенно сократить этот перебор.

Похожий метод можно использовать и для декодирования НП-кодов. При этом удается исправлять до $d_{\min}/2$ ошибок. Отличие заключается в том, что для поиска минимального расстояние ищется слово минимального веса с нулевым синдромом, а для декодирования ищется слово минимального веса с данным синдромом.

Основные результаты. С помощью представленного подхода удалось вычислить минимальное расстояние для кода Таннера [6], для минимального расстояние которого была известна только оценка снизу, а также построить алгоритм декодирования НП-кодов для исправления не более $d_{\min}/2$ ошибок.

Литература

1. R.G. Gallager. Low-density Parity-Check Codes. MIT Press, Cambridge MA., 1963.
2. C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. In Proc. IEEE Intern. Conf. on Commun., ICC93. – V. 2. – PP. 1064–1070, Geneva, Switzerland, May 1993.
3. T.J. Richardson, M.A. Shokrollahi, and R. L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. IEEE Trans. Inform. Theory, 47: 619–637. – 2001.
4. E. Bocharova, B.D. Kudryashov, R.V. Satyukov, and S. Stiglmayr, «Short quasi-cyclic ldpc codes from convolutional codes», in Proc. IEEE International Symposium on Information Theory (ISIT'09), Seoul, South-Korea, 2009. – PP. 551–555.
5. Бочарова И.Е., Кудряшов Б.Д., Сатюков Р.В. «Сверточные и блоковые коды с малой плотностью проверок на основе графов», Пробл. передачи информ., 45:4 (2009). – С. 69–90.
6. R. Tanner, D. Sridhara, and T. Fuja «A class of group-structured LDPC codes», in Proc. ISTA, Ambleside, England. – 2001.

УДК 004.4'242

ОПТИМИЗАЦИЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ

Е.В. Селифонов
Научный руководитель – д.т.н., профессор А.А. Шалыто

Вступление. Задачи, возлагаемые на вычислительные машины, постоянно усложняются, что ведет к увеличению скорости работы компьютеров. К сожалению, невозможно достичь необходимой производительности, лишь увеличивая тактовую частоту центрального процессора. Поэтому в последнее время стали популярны многоядерные процессоры и параллельные вычисления.

Также значительно развивается сфера трехмерной графики. Связанные с ней вычисления требуют больших вычислительных мощностей. Центральный процессор уже не способен справиться с данной задачей, поэтому созданы графические процессоры со специальной архитектурой, направленной на высокопроизводительные вычисления в данной области.

В последнее время появилось направление исследований, связанное с использованием графических процессоров для вычислений, не связанных с трехмерной графикой. За счет большого числа вычислительных конвейеров возможна параллельная обработка массивов данных с практически линейным приростом производительности. В настоящее время число областей, в которых применяются графические процессоры, постоянно растет.

В силу специфики графических процессоров работа с ними значительно сложнее написания традиционной программы для центрального процессора. Были созданы специальные библиотеки для упрощения взаимодействия с графическими процессорами, но, к сожалению, они имеют значительные ограничения. Одним из них является «заточенность» библиотеки под процессоры производства ее автора (например, CUDA работает только на видеокартах nVidia).

Цель работы. Существует традиционный подход к работе с графическими процессорами, при котором вручную пишутся программы для графического процессора, и налаживается механизм взаимодействия между центральным и графическим процессором. В силу специфики графических процессоров накладываются определенные ограничения.

Основным направлением работы является исследование способов оптимизации программ по скорости, количеству подпрограмм и занимаемой памяти в рамках соблюдения данных ограничений. В качестве примера для демонстрации данных оптимизаций взята программа, имеющая прикладное значение. Данная программа осуществляет поиск движения в кадре на основе информации, получаемой с неподвижной веб-камеры.

Результаты. Реализован алгоритм работы программы на центральном и графическом процессорах. На современных видеокартах второй вариант дал значительный прирост производительности.

Разработаны и обоснованы методы оптимизации программ для графических процессоров, состоящих из нескольких подпрограмм. Написана программа, позволяющая автоматизировать процесс выявления способов оптимизации и создания оптимизированных подпрограмм.

Данная программа применена к задаче, рассматриваемой в качестве примера. В результате удалось сократить число подпрограмм, решающих задачу, с четырех до одной. При этом уменьшена занимаемая память и увеличена производительность по сравнению с изначальной реализацией алгоритма на графическом процессоре.

УДК 004.4'242

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ ДЛЯ ЛОКАЛЬНОЙ ОПТИМИЗАЦИИ КОДА

Е.В. Смирнов
Научный руководитель – д.т.н., профессор А.А. Шалыто

Вступление. Скорость работы всегда оставалась одним из основных критериев оценки качества программного обеспечения. По этой причине стадия оптимизации в той или иной мере всегда присутствует в процессе разработки. И если затратная тонкая ручная оптимизация применяется только к тем программам, в которых критична каждая лишняя миллисекунда работы, то использование автоматических средств ускорения кода дешево и подходит всем, позволяя при этом добиться заметного повышения скорости работы.

Одним из видов этапов работы, который в той или иной мере включают в себя все современные оптимизирующие средства, является так называемая «локальная оптимизация». Эта оптимизация представляет собой поиск определенных небольших последовательностей низкоуровневых инструкций (на уровне ассемблера) и замена их на более эффективный аналог. Локальный оптимизатор использует таблицу правил преобразования фрагментов кода, в которой содержатся неоптимальные наборы инструкций и то, чем их следует заменить. Обычно такая таблица составляется вручную, и может содержать лишь ограниченный набор достаточно простых правил.

Ранее был предложен метод автоматического построения таблиц с помощью полного перебора возможных наборов инструкций. Однако уже для последовательностей длиной в четыре-пять инструкций объем перебора становится слишком большим для его выполнения в разумное время.

Цель работы. Переход от ручного построения локальных оптимизаторов к автоматическому позволит добиться значительного повышения их качества работы, так как станет возможной генерация намного более подробных правил замены фрагментов кода.

Целью данной работы является проверка возможности применения генетических алгоритмов как замены полного перебора при поиске лучшего набора команд.

Результаты. В качестве целевой платформы был выбран язык виртуальной машины Java. Была разработана реализация эволюционного процесса, оперирующего наборами инструкций и при этом поддерживающего возможность их запуска. Были созданы генетические операторы, обеспечивающие лучшие условия для кандидатов, эквивалентных начальному набору инструкций, но имеющих меньшее время работы.

В качестве эксперимента был проведен поиск лучшего варианта для фрагмента кода длиной в 11 инструкций, сгенерированного компилятором для вычисления значения полинома третьей степени.

Эксперимент показал высокую эффективность генетических алгоритмов в решении задачи оптимизации программного кода. В процессе эволюции было получено несколько вариантов последовательностей в девять инструкций, реализующих ту же математическую формулу и работающих в среднем на 11 процентов быстрее. Также была получена одна нетривиальная оптимизация, для нахождения которой с использованием стандартных алгоритмов потребовался бы глубокий анализ свойств вычисляемых выражений.

УДК 004.832.28

ПРИМЕНЕНИЕ ДВУХЭТАПНОГО ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ НА ПРИМЕРЕ ПОСТРОЕНИЯ МОДЕЛИ ТАНКА В ИГРЕ ROBOCODE

Д.О. Соколов
Научный руководитель – д.т.н., профессор А.А. Шалыто

Постановка проблемы. Проблема построения управляющей системы для беспилотной техники стоит перед человечеством уже длительное время. Чаще всего данная проблема решается при помощи ручного труда. Однако данный метод не эффективен ввиду больших затрат ресурсов, а также качества построенных систем. В некоторых случаях построение управляющей системы при помощи ручного труда невозможно, ввиду сложности системы. Естественная идея – заставить компьютер строить управляющие системы.

Эволюционные вычисления являются одним из наиболее используемых направлений искусственного интеллекта и успешно применяются для автоматического создания программ. Эффективность методов напрямую зависит от способа представления программы. Для ряда задач поведение системы удобно представлять в виде конечных автоматов.

В данной работе предлагается использовать генетические алгоритмы для генерации конечных автоматов. Основными проблемами всех эволюционных вычислений, в частности генетических алгоритмов, являются: время работы программ, большое количество вариантов неоптимальных решений. Метод, представленный в работе, позволяет бороться с описанными проблемами за счет уменьшения размера пространства решений генетического алгоритма.

Цель работы. Целью данной работы является демонстрация метода, позволяющего решать основные проблемы эволюционных вычислений, на примере построения управляющего автомата для модели танка в игре Robocode.

Базовые положения исследования. Эволюционные вычисления делятся на множество типов. В данной работе рассмотрены генетические алгоритмы. Идеи основаны на скрещивании метода генетических алгоритмов и метода динамического программирования, а именно на том, что можно получать сначала отдельные состояния управляющего автомата, а только потом диаграмму переходов.

Промежуточные результаты. Были разработаны:
- метод представления автоматов при помощи деревьев разбора;
- генетические операции для данного метода;
- добавлены дополнительные генетические операторы в островной генетический алгоритм, для уменьшения вероятности нахождения локальных оптимумов;
- разработан метод разделения построения автомата на два этапа, что позволяет уменьшить размер пространства решений;
- проведены эксперименты по построению управляющих автоматов.

Основной результат. В результате апробации разработанных методов была доказана их эффективность. Построенный управляющий автомат демонстрирует лучшие результаты в сравнении с автоматами, построенными с помощью других автоматических методов. Однако победить соперника, построенного при помощи ручного труда, все еще не удается.

УДК 51-74

КИНЕМАТИКА ПЛАТФОРМЫ ГЬЮ-СТЮАРТА

И.В. Сорокин
Научный руководитель – к.ф.-м.н., доцент С.Ю. Жуков
(Санкт-Петербургский государственный политехнический университет)

Вступление. Системы подвижности широко применяются в современных тренажерах, развлекательных системах и комплексах для создания эффекта погружения. Платформа Гью-Стюарта является широко распространенной основой систем подвижности.

Платформа Гью-Стюарта представляет собой параллельный манипулятор, имеющий шесть ног изменяемой длины. Преимуществами платформы являются шесть степеней свободы манипулятора (три пространственные координаты: x, y, z; три угла: рыскание, тангаж, крен), низкая масса и высокая структурная прочность.

Недостатком платформы является то, что для изменения длины ног необходимо использовать гидравлический привод, что сказывается на стоимости платформы. Гидравлический привод можно заменить электродвигателем, тем самым уменьшив её стоимость. Данная работа посвящена решению задачи прямой и обратной кинематики для такой модификации платформы Гью-Стюарта.

Цель работы. Задача прямой кинематики состоит в определении положения платформы, по указанному положению двигателей. Задача обратной кинематики состоит в определении требуемого положения двигателей для данного положения платформы.

Обзор существующих точек зрения на проблему. Платформа Гью-Стюарта с гидравлическим приводом исследуется в работах различных авторов. В работе [1–5] построены решения для задач прямой и обратной кинематики. В работе [1] также рассматривается способ проверки отсутствия особых точек. Работа [6] исследует область допустимых положений платформы.

Результаты. В результате проведенных исследований было получено аналитическое решение задачи обратной кинематики для данной платформы. Для задачи прямой кинематики аналитического решения построено не было, однако данная задача была решена итерационными методами.

Выводы. Результат, полученный в данной работе, позволяет создать электромеханический аналог гидравлической платформы Гью-Стюарта, тем самым уменьшив стоимость ее создания и обслуживания.

Литература

1. Tiago Charters de Azevedo, Ricardo Enguica, Pedro Freitas Report on the Stewart platform problem.
2. Domagoj Jakobovic, Leonardo Jelenkovic The Forward and Inverse Kinematics Problems for Stewart Parallel Mechanisms.
3. Domagoj Jakobovic, Leonardo Jelenkovic Kinematic evaluation and forward kinematic problem for Stewart platform based manipulators.
4. Domagoj Jakobovic, Leo Budin Forward Kinematics of a Stewart Platform Mechanism.
5. Alexander V. Korobeynikov, Vadim E. Turlapov Modeling and Evaluating of the Stewart Platform.

6. Vivek Saxena, Dongming Liu, Cecil M. Daniel, John W. Sutherland A Simulation Study of the Workspace and Dexterity of a Stewart Platform Based Machine Tool.

УДК 681.5

МЕТОДЫ ОПТИМИЗАЦИИ ВЫЧИСЛЕНИЙ ПРИ РАБОТЕ КЛЕТОЧНЫХ АВТОМАТОВ

Д.И. Суясов
Научный руководитель – д.т.н., профессор А.А. Шалыто

Клеточные автоматы являются универсальной моделью параллельных вычислений, которые позволяют на основе простых локальных правил решать сложные задачи. Они применяются в распределенных приложениях, в системах распознавания текстов и образов, при моделировании физико-химических процессов в малоразмерных системах, моделировании других процессов. Основными преимуществами клеточных автоматов являются совместимость с алгоритмическими методами решения задач, простота и локальность правил, возможность параллельной обработки вычислительного поля.

Кроме преимуществ клеточные автоматы обладают и недостатками: для обработки достаточно больших объемов данных количество вычислений стремительно растет и не позволяет полноценно пользоваться данным инструментом. Для эффективной работы с клеточными автоматами необходимо иметь инструменты по оптимизации вычислений.

В данной работе проводится исследование методов оптимизации вычислений при работе с клеточными автоматами: анализируются механизмы объединения локальных правил, зональная оптимизация вычислений с оценкой применимости и методы фильтрации клеток поля.

Механизмы объединения локальных правил предполагают сокращение количества проверок условий в правилах на клетках поля клеточного автомата за счет объединения проверок правил. Объединение позволяет добиться значительного увеличения производительности при итерационных вычислениях.

Зональная оптимизация заключается в том, что на очередной итерации работы клеточного автомата не обязательно проверять все клетки поля на выполнение условий правил переходов. В случае, когда на предыдущей итерации не было изменения состояния клетки поля и ее соседей из радиуса окрестности, то в текущей итерации данная клетка не нуждается в проверке на выполнение правил клеточного автомата, так как они заведомо не будут выполнены. Зональная оптимизация проанализирована применительно при работе с отдельным клеточным автоматом и с набором клеточных автоматов. В обоих случаях значительный выигрыш в производительности достигается только при малых изменениях поля в среднем по всем итерациям.

Методы фильтрации клеток поля позволяют значительно сократить время работы клеточного автомата за счет предварительной фильтрации клеток по принципу применимости правил клеточного автомата.

В итоге, внедрение методов оптимизации вычислений при работе с клеточными автоматами увеличивает быстродействие. В задачах по выделению признаков символов при распознавании текста данное внедрение сокращает время работы на порядок.

УДК 004.4'242

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО ПОДХОДА ДЛЯ ГЕНЕРАЦИИ КЛЕТОЧНЫХ АВТОМАТОВ

А.В. Тихомиров
Научный руководитель – д.т.н., профессор А.А. Шалыто

Краткое вступление, постановка проблемы. В настоящее время широко распространено использование клеточных автоматов для симуляции физических процессов, например: диффузия энергии, разнообразные химические реакции, рост кристаллов и т.д. Однако использование клеточных автоматов существенно затруднено, когда известно состояние некоторой системы в различные промежутки времени, но не известен автомат, описывающий ее поведение, или его построение эвристическими методами затруднительно.

Цель работы. Целью настоящей работы является разработка алгоритма автоматической генерации клеточных автоматов с заранее неизвестными числом переходов и условий на них при помощи генетического программирования на основе данных о состоянии системы в различных промежутках времени.

Базовые положения исследования. Классический генетический алгоритм представляет собой эвристический метод поиска и оптимизации решений для задач моделирования, используя операции генерации случайного решения, скрещивания (комбинирование нескольких решений) и мутации (случайное изменение части известного решения). Общий принцип работы классического генетического алгоритма напоминает биологическую эволюцию.

Промежуточные результаты. В работе предложен улучшенный вариант классического генетического алгоритма для достижения поставленной цели.

В работе освещены следующие аспекты алгоритма:

- структура хромосомы клеточного автомата;
- генерация начального поколения;
- генетические операторы:
  - операторы скрещивания;
  - операторы мутации;
  - дополнительные операторы;
- проблема вырождения популяции особей и методы ее решения;
- вычисление функции приспособленности и влияние на нее размерности клеточного автомата;
- алгоритм отбора нового поколения и исследование его влияния на время работы алгоритма.

Проведено сравнение скорости работы алгоритма со временем полного перебора всех возможных вариантов автоматов.

Основной результат. Результатом настоящей работы является программная библиотека, разработанная на языке программирования Java. В качестве тестового примера была выбрана задача по построению клеточного автомата, используя только состояния клеток на каждом конкретном шаге расчетов. Разработанной системой был построен автомат, который является эквивалентным искомому.

УДК 004.932.1

ПРИМЕНЕНИЕ РЕШЕТЧАТОГО КВАНТОВАНИЯ ДЛЯ СЖАТИЯ ИЗОБРАЖЕНИЯ

Н.С. Токалов
Научный руководитель – д.т.н., профессор Б.Д. Кудряшов

Вследствие развития информационных технологий в области цифровой техники, требования, предъявляемые к сжатию изображений, неуклонно растут. Для многих изображений коэффициент сжатия традиционными методами был мал, хотя они обладали явной избыточностью. Что, в свою очередь, привело к использованию алгоритмов — сжимающих изображения с потерями.

Наиболее популярным алгоритмом сжатия с потерями является JPEG – название алгоритма образовано от комитета Joint Photographic Expert Group (Объединенная группа экспертов по фотографии). Алгоритм JPEG основывается на использовании дискретного косинусоидного преобразования (Discrete-Cosine Transform, DCT), что позволяет обеспечить довольно высокую степень компрессии при относительно небольшой потере данных. При этом сжатие растровых беспалитровых изображений выполняется следующим образом:

- преобразование цветового пространства;
- сегментация;
- линейное преобразование (DCT);
- квантование;
- энтропийное кодирование.

Декодирование осуществляется, в рамках этого подхода, в обратном порядке.

DCT преобразование выполняется для перераспределения энергии между отсчетами, декорреляции отсчетов и придания коэффициентам удобной для квантования структуры.

В качестве энтропийного кодера обычно используется арифметический кодер или кодер Хаффмана.

Целью настоящего исследования было повысить эффективность сжатия изображений, применив модифицированное решетчатое квантование.

Для достижения указанной цели были поставлены следующие задачи:

1. Реализовать модель кодера использующего квантование.
2. Реализовать скалярный квантователь.
3. Реализовать решетчатый квантователь (квантование с помощью решеток на основе сверточных кодов) и исследовать его влияние на эффективность сжатия.

В рамках настоящей работы для моделирования кодера был использован MATLAB 7.6.

Изображение конвертировалось из рисунка формата BMP (цветовое пространство RGB) в формат YUV, далее сжатие выполнялось уже конвертированного изображения. Отличительной особенностью цветового пространства YUV является явное разделение информации о яркости и цвете.

В цветовом пространстве YUV цвет представляется 3 компонентами: первая отвечает за яркость (Y) и две цветоразностных (U и V). Человеческий глаз не столь чувствителен к ошибкам цветности. В связи с этим, компоненты цветности кодируются с меньшей точностью, по

сравнению с компонентой Y. Для этого был реализован модуль, выполняющий понижение частоты дискретизации компонент цветности в 2 раза по каждой размерности. С помощью MATLAB была реализована модель кодера, использующая скалярное квантование. А позже, в ходе исследования, был реализован и модуль решетчатого квантования.

Также были реализованы модули оценки качества сжатия изображения и проведен анализ результатов сжатия изображений с использованием скалярного и решетчатого квантователей. Для описания качества сжатия изображений приводятся графики зависимости отношение пикового уровня сигнала к шуму (Peak Signal-to-Noise Ratio, PSNR) к коэффициенту сжатия изображения. В работе приводится описание полученных данных для разного уровня искажения изображения.

УДК 004.052

РЕФАКТОРИНГ АВТОМАТНЫХ ПРОГРАММ

П.В. Федотов
Научный руководитель – к.т.н., доцент О.Г. Степанов

Постановка проблемы. Для разработки программных систем со сложным поведением в последнее время все чаще используется автоматное программирование, при этом важную роль играет контроль качества разрабатываемых программ. Любая программа, используемая длительное время, подвергается модификации. Поэтому большое значение имеет проблема поддержания надежности автоматных программ при внесении в них структурных изменений.

Цель работы. При разработке объектно-ориентированных программ широко применяется рефакторинг. Автоматные программы имеют свою специфику, и применение стандартных рефакторингов в них затруднительно. Однако идеи, заложенные в основе рефакторинга, можно перенести и на автоматное программирование.

В основе рефакторинга лежит последовательность небольших эквивалентных (т.е. сохраняющих поведение) преобразований. В работе рассматриваются рефакторинги автоматных программ, т.е. такие структурные изменения автоматов, которые не меняют их внешнего поведения. Также приводятся обоснования того, что такие рефакторинги действительно сохраняют поведение автоматных программ.

Промежуточные результаты

1. Введено понятие рефакторинга автоматных программ.
2. Создан каталог рефакторингов.
3. Для каждого рефакторинга описана техника его выполнения.
4. Для каждого рефакторинга обоснована корректность его выполнения.

Основной результат. Изменения, совершаемые исключительно в целях изменения структуры автомата без изменения его поведения могут быть произведены совершенно безопасно, если они реализованы путем комбинирования нескольких рефакторингов.

Если же сложное изменение автомата касается, в том числе его поведения, то такое изменение можно разделить на две фазы:

1. Рефакторинг автомата.
2. Набор модификаций, приводящих к изменению поведения автомата.

Такой подход позволяет минимизировать и упростить небезопасные модификации автоматных программ.

УДК 004.4’242

СОВМЕСТНОЕ ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ И ВЕРИФИКАЦИИ МОДЕЛЕЙ ДЛЯ ПОСТРОЕНИЯ АВТОМАТОВ УПРАВЛЕНИЯ СИСТЕМАМИ СО СЛОЖНЫМ ПОВЕДЕНИЕМ

К.В. Егоров, Ф.Н. Царев
Научный руководитель – д.т.н., профессор А.А. Шалыто

Введение. Автоматное программирование – это парадигма программирования, в рамках которой программы предлагается проектировать в виде совокупности взаимодействующих автоматизированных объектов управления [1]. В автоматных программах выделяют три типа объектов: поставщики событий, система управления и объекты управления. Система управления представляет собой конечный автомат или систему взаимодействующих конечных автоматов. Поставщики событий генерируют события, а система управления по каждому событию может совершать переход, считывая значения входных переменных у объектов управления для проверки условия перехода.

Для многих задач автоматы удается строить эвристически, однако существуют задачи, для которых такое построение затруднительно [2–4]. Одним из авторов настоящей работы был предложен метод построения автоматов с помощью генетического программирования на основе тестов [5]. Однако, как известно, тесты не могут полностью описывать поведение программы, а их выполнимость не может служить критерием ее корректности.

Целью настоящей работы является расширение возможностей метода построения автоматных программ на основе генетического программирования за счет использования верификации на этапе вычисления функции приспособленности.

Генерация автоматных программ на основе тестов. При использовании метода построения управляющих конечных автоматов на основе тестов каждый тест представляет собой последовательность входных и соответствующую ей последовательность выходных воздействий. Под входными воздействиями понимаются события от поставщиков событий и условия переходов, а под выходными – вызываемые действия объектов управления.

Построение автоматов на основе тестов осуществляется на основе генетического программирования. Функция приспособленности вычисляется исходя из успешности выполнения тестов для автомата и количества переходов. Ее вычисление основано на редакционном расстоянии (расстоянии Левенштейна) [6]. Чем больше число пройденных тестов, тем больше функция приспособленности.

При прохождении тестов помечаются задействованные переходы, и скрещивание может происходить как случайным образом, так и с сохранением помеченных переходов.

Верификация автоматных программ. Для описания требований к автоматным программам будем применять язык логики линейного времени (Linear Temporal Logic, LTL). По любой LTL-формуле можно построить автомат Бюхи. Алгоритм верификации основан на проверке пустоты языка пересечения допускаемого конечным автоматом модели и отрицанием LTL-формулы [7, 8].

Верификатор получает на вход модель автоматной программы и LTL-формулу [9]. После проверки модели, верификатор либо сообщает, что формула выполняется, либо приводит контрпример – путь в модели, опровергающий утверждение [10].

Совместное применение генетического программирования и верификации. Предлагается при вычислении функции приспособленности учитывать как поведение автомата при обработке тестов, так и число верных для автомата LTL-формул. При этом, чем больше число верных формул и успешно пройденных тестов, тем больше значение функции приспособленности.

Верификатор помечает переходы, удовлетворяющие LTL-формулам. В результате, даже при невыполнимости LTL-формулы, у автомата получается некоторое подмножество вершин и переходов, удовлетворяющих формулам. Такие переходы могут переходить в новую особь без изменений.

Мутация учитывает путь, предложенный верификатором в качестве контрпримера, переходы на таком пути мутируют с большей вероятностью. Под мутацией подразумевается удаление перехода, изменение входных, выходных воздействий. Все эти методы позволяют не только говорить о приспособленности конкретной особи, но и позволяют увеличивать функцию приспособленности в следующем поколении.

Предложенным методом были построены два автомата: автомат управления электронными часами с будильником и управлением дверьми лифта. Однако в первом случае применение верификации замедлило процесс построения автомата. Во втором случае позволило построить правильный автомат, в отличие от построения только на основе тестов, где модель оказалась неверной. В обоих экспериментах применение верификации гарантировало построение автомата с заранее заданным поведением не только на конечном числе тестов.

Заключение. Разработанное средство показало возможность применения верификации и генетических алгоритмов для генерации модели программы. Оно позволяет автоматически строить модель программы по набору тестов и LTL-формул и позволяет в определенной степени гарантировать правильность поведения построенного автомата.

Литература

1. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб: Питер. – 2009.
2. Angeline P.J., Pollack J. Evolutionary Module Acquisition // Proceedings of the Second Annual Conference on Evolutionary Programming. 1993. http://www.demo.cs.brandeis.edu/papers/ep93.pdf
3. Jefferson D., Collins R., Cooper C., Dyer M., Flowers M., Korf R., Taylor C., Wang A. The Genesys System. 1992. http://www.cs.ucla.edu/~dyer/Papers/AlifeTracker/Alife91Jefferson.html
4. Chambers L. Practical Handbook of Genetic Algorithms. Complex Coding Systems. – Volume III. CRC Press. – 1999.
5. Царев Ф.Н. Метод построения автоматов управления системами со сложным поведением на основе тестов с помощью генетического программирования / Материалы Международной научной конференции «Компьютерные науки и информационные технологии». Саратов: СГУ. – 2009. – С. 216–219.
6. Левенштейн В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов. Доклады Академии Наук СССР 163.4. – С. 845–848.
7. Кларк Э., Грамберг О., Пелед Д. Верификация моделей программ: Model Checking. М.: МЦНМО. – 2002.
8. Gerth R., Peled D., Vardi M. Y., Wolper P. Simple On-the-fly Automatic Verification of Linear Temporal Logic / Proc. of the 15th Workshop on Protocol Specification, Testing, and Verification, Warsaw. – 1995. – Pp. 3–18.
9. Разработка технологии верификации управляющих программ со сложным поведением, построенных на основе автоматного подхода. Второй этап. СПбГУ ИТМО. – 2007. http://is.ifmo.ru/verification/_2007_02_report-verification.pdf

10. Егоров К.В., Шалыто А.А. Методика верификации автоматных программ // Информационно-управляющие системы. СПб: Политехника. – 2008. – № 5. – С. 15–21.

УДК 004.85

ПРИМЕНЕНИЕ МАШИННОГО ОБУЧЕНИЯ И КОЭВОЛЮЦИИ ДЛЯ СОЗДАНИЯ УПРАВЛЯЮЩИХ АВТОМАТОВ НА ПРИМЕРЕ КОМАНДНОЙ ВЕРСИИ ИГРЫ ROBOCODE

И.И. Чернявский
Научный руководитель – д.т.н., профессор А.А. Шалыто

В данной работе рассматривается задача построения автоматов, управляющих роботами-агентами в мультиагентной среде.

Задача рассматривается на примере командной версии игры Robocode [1], в которой программы, написанные на языке Java, «сражаются» друг с другом, управляя роботами-танками. Танки состоят из тела, способного перемещаться по прямоугольному полю, радара, с помощью которого танки находят своих противников, и пушки. Бой происходит между двумя командами танков. Танки могут обмениваться сообщениями с членами своей команды. Необходимо создать команду танков, побеждающую другие команды.

Известно множество программ [2], написанных для Robocode. Некоторые из них совсем простые, в то время как другие могут содержать тысячи строк кода и реализовывать сложные стратегии поведения. В работах [3] и [4] был впервые применен автоматный подход к созданию роботов-танков.

Большинство программ для Robocode написано вручную. Интерес представляют методы автоматического построения управляющих автоматов для роботов-танков. В [5] предложен метод такого построения, а в [6] – развитие этого метода (метод двухэтапного генетического программирования), с помощью которого успешно решена задача построения танка, побеждающего заранее заданного противника в бою «один на один». При этом с помощью генетического программирования [8] на первом этапе строятся состояния автомата, а на втором – сам автомат.

Цель данной работы – разработка метода автоматического создания команды роботов-танков, проведение сравнительного анализа различных способов ее построения.

Подход, используемый в настоящей работе, продолжает подход, предложенный в работе [6], и основан на совместном применении машинного обучения (обучение с подкреплением [7]) и коэволюции [8].

В данной работе машинное обучение используется для построения состояний автоматов. Рассмотрены несколько способов такого построения (с использованием таблиц, аппроксимации на основе нейронных сетей). Проведено сравнение метода с методом генетического программирования.

Для построения управляющих автоматов в настоящей работе применяется коэволюция. Рассмотрены вопросы взаимодействия автоматов в игре между собой.

В настоящей работе представлен подход, успешно решающий задачу построения команды роботов-танков, а также проведен сравнительный анализ различных способов такого построения.

Литература

1. Официальный сайт игры Robocode (http://robocode.sourceforge.net).
2. Сайт репозитория танков для Robocode (http://robocoderepository.com).
3. Туккель Н.И., Шалыто А.А. Система управления танком для игры Robocode. Объектно-ориентированное программирование с явным выделением состояний. Программная документация. (http://is.ifmo.ru/projects/tanks/).
4. Кузнецов Д.В., Шалыто А.А. Система управления танком для игры Robocode. Вариант 2. Проектная документация. (http://is.ifmo.ru/projects/robocode2/).
5. Бедный Ю.Д. Применение генетических алгоритмов для генерации автоматов при построении модели максимального правдоподобия и в задачах управления. Магистерская диссертация. (http://is.ifmo.ru/papers/bednij/).
6. Соколов Д.О. Применение двухэтапного генетического программирования для построения автомата, управляющего моделью танка в игре Robocode. Дипломная работа, 2009.
7. Richard S. Sutton, Andrew G. Barto Reinforcement Learning: An Introduction, The MIT Press, 1998.
8. Mitchell M. An Introduction to Genetic Algorithms. The MIT Press, 1998.

УДК 004.4’242

ПРОЕКТИРОВАНИЕ АВТОМАТИЗИРОВАННЫХ КЛАССОВ

И.С. Гунич, Д.Ю. Кочелаев
Научный руководитель – д.т.н., профессор А.А. Шалыто

В настоящее время существует ряд средств визуального проектирования программных систем. Однако в этих средствах проектирование системы необходимо производить с нуля, а также они накладывают ограничения на архитектуру создаваемой системы. Данные ограничения делают невозможным использование этих инструментальных средств при работе над уже существующими проектами, а также не предоставляют простого способа для перехода от существующей архитектуры к архитектуре, совместимой с этими инструментальными средствами.

В 2006 году Надежда Поликарпова, магистр СПбГУ ИТМО кафедры компьютерных технологий, предложила объектно-ориентированный подход к моделированию и спецификации сущностей со сложным поведением. В основе этого подхода лежит автоматное программирование – логика объектов задается с помощью автоматов. Такие объекты было предложено называть автоматизированными абстрактными типами данных (ААТД). Основной целью этой работы было построение математической модели сущности, поведение которой зависит от состояния. Таким образом, работа являлась теоретической и не предлагала какой-либо реализации для ААТД.

Целью данной работы является расширение методики, которая была предложена Н. Поликарповой, а также практическая реализация данной методики.

Основные направления работы:

- Выявление и решение открытых вопросов, связанных с практической реализацией методики.
- Практическая реализация методики в виде библиотеки на языке Eiffel.
- Разработка алгоритма перехода от реализации логики ААТД на языке Eiffel к визуальному представлению в виде диаграммы автоматов и обратно.

В работе была проведена классификация существующих подходов. Был проведен обзор генераторов декларативной части проектируемых систем. Выявлен основной недостаток этих генераторов – отсутствие возможности визуального проектирования логики системы.

Также проведен обзор генераторов логики проектируемых систем. Выявлены основные недостатки этих генераторов:

1. Сложность, а зачастую и отсутствие возможности, встраивания в существующие программные системы.
2. Невозможность использования библиотек, которые реализуют автоматное представление логики, без использования визуального редактора.

Далее рассмотрен подход с использованием автоматизированных абстрактных типов данных (ААТД). В случае использования данного подхода не возникает проблема при встраивании в существующие объектно-ориентированные программные системы. Для данного подхода отсутствует реализация и, соответственно, есть открытые вопросы связанные с ней.

Для реализации методики ААТД предлагается использовать язык программирования Eiffel, разработанный Бертраном Мейером. Eiffel является объектно-ориентированный языком программирования, который был спроектирован с целью предоставить программистам возможность рационально разрабатывать расширяемые, переиспользуемые, надежные программные системы. В работе проведен обзор языка программирование Eiffel.

Введено понятие автоматизированного класса, как реализации ААТД. Описание сущностей со сложным поведением может быть разделено на отдельное описание логики и семантики. Семантическое описание включает в себя набор вычислимых состояний (атрибутов) и действия, которые может выполнять сущность. С точки зрения объектно-ориентированного подхода семантическая часть описывается классом. Вычислимые состояния при этом задаются атрибутами этого класса, а действия командами класса.

Логическая часть сущности со сложным поведением называется контроллером, а семантическая – объектом управления. Объект управления, интегрированный вместе с контроллером в единую сущность, называется автоматизированным объектом управления. Реализацию автоматизированного объекта управления в терминах объектно-ориентированного программирования будем называть автоматизированным классом.

Рассмотрены основные этапы проектирования автоматизированных классов.

1. Декомпозиция решаемой задачи.
2. Выделяются сущности со сложным поведением, они будут реализованы с помощью автоматизированных классов.
3. Определяется набор управляющих состояний и переходов между ними, а также условия и действия на переходах.
4. Реализуется объект управления, который предоставляет запросы и команды.

Далее в работе описана архитектура библиотеки EiffelState, которая реализует работу с автоматизированными классами на языке Eiffel. В основе библиотеки лежат четыре класса, доступные клиентам библиотеки, которые позволяют построить на своей основе автоматизированный класс, реализующий логику конкретной задачи.

- AUTOMATED является абстрактным базовым классом для всех автоматизированных классов.
- STATE является вспомогательным классом, который описывает состояние системы.
- Два класса, которые используются при описании поведения, STATE_DEPENDENT_FUNCTION и STATE_DEPENDENT_PROCEDURE имеют схожее назначение. Эти классы используются для описания действий и условий на переходах (STATE_DEPENDENT_PROCEDURE), а также для описания функций,

результат которых зависит от текущего состояния автоматизированного класса (STATE_DEPENDENT_FUNCTION).

Перечислим теперь основные этапы создания автоматизированного класса на основе библиотеки EiffelState:

- Создание класса, который является наследником класса AUTOMATED.
- Создаются экземпляры класса STATE, которые соответствуют управляющим состояниям, выбранным на этапе проектирования класса.

Для каждого действия, выделенного на этапе проектирования, создаются экземпляры STATE_DEPENDENT_PROCEDURE, переходы и условия на них задаются с помощью метода add_behavior (позволяет определить поведение функции или процедуры, в зависимости от текущего состояния).

- Для каждой функции, результат которой зависит от состояния, создаются экземпляры STATE_DEPENDENT_FUNCTION.
- На заключительном этапе создаются методы класса, которые будут составлять его интерфейс и инкапсулировать его автоматную реализацию.

Во время создания библиотеки EiffelState был решен ряд вопросов, в частности была решена задача выбора перехода при наследовании автоматизированных классов путем расстановки приоритетов.

В рамках библиотеки EiffelState были разработаны алгоритм перехода от диаграммы автоматов к коду, который реализует автоматизированный класс с соответствующей логикой, и алгоритм построения диаграммы автоматов по исходному коду автоматизированного класса, написанного с помощью библиотеки EiffelState.

В заключении приведено описание пробной задачи для демонстрации легкой интеграции автоматизированных классов в существующие программные системы.

Выполнена реализация пробной задачи в терминах объектно-ориентированного программирования.

Также выполнена реализация пробной задачи в терминах автоматизированных классов. При этом интерфейсы автоматизированных классов совпадают с интерфейсами соответствующих классов «традиционной» реализации. Этот факт доказывает возможность легкой интеграции в существующие программные системы.

В качестве дальнейшего расширения работы предлагается создать инструментальное средство для визуальной разработки автоматизированных классов на базе EiffelStudio.