ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ И ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

УДК 004.85

МЕТОД СБОРКИ ГЕНОМА С ПОМОЩЬЮ ВОССТАНОВЛЕНИЯ ЕГО ФРАГМЕНТОВ ПО ПАРНЫМ ЧТЕНИЯМ

А.В. Александров, В.В. Исенбаев, С.В. Казаков, А.А. Сергушичев, С.В. Мельников, Ф.Н. Царев
Научный руководитель – д.т.н., профессор А.А. Шалыто

Одной из важных задач, возникающих в биологии и медицине, является задача получения генома живого организма. Современные технологии не позволяют прочитать весь геном за один раз, поэтому применяемые в настоящее время методы основываются на считывании фрагментов нуклеотидной последовательности (этот процесс называется прямым секвенированием) и последующем восстановлении генома по этим данным (сборка). Это восстановление возможно ввиду того, что считанные фрагменты в совокупности покрывают весь геном десятки раз. Ранние методы секвенирования основывались на прочтении достаточно длинных фрагментов генома (300–1000 нуклеотидов). Применение данной технологии для получения достаточного покрытия сопряжено с большими денежными и временными затратами. В настоящее время разработаны методы, позволяющие получить хорошее покрытие дешевле и быстрее. Эти методы основаны на выделении фрагментов длиной примерно по 500 нуклеотидов и последующем чтении с их концов последовательностей длиной примерно по 100 нуклеотидов каждая – эти две последовательности называются парными чтениями. Однако последующая сборка генома при использовании данного метода становится вычислительно более сложной задачей.

Целью настоящей работы является разработка метода, решающего эту задачу. Предлагаемый метод основывается на сведении ее к задаче сборки генома из относительно длинных последовательностей (длинной примерно по 500 нуклеотидов), решение которой проще. Это достигается путем восстановления частично прочитанных фрагментов.

Метод состоит из трех этапов.

1. Исправление ошибок, присутствующих в чтениях из-за особенностей прямого секвенирования. Это позволяет увеличить число различных последовательностей фиксированной длины k (они называются k-мерами), встречающиеся в чтениях достаточное число раз, чтобы их можно было с большой уверенностью считать входящими в геном, – такие k-меры будем называть «хорошими». Исправление ошибок заключается в рассмотрении редко встречающихся k-меров и последующей попыткой их небольшого исправления на «хорошие».
2. Восстановление частично прочитанных фрагментов. Этот этап осуществляется при помощи построения подграфа графа де Брюина размерности k, в котором есть только те ребра, которым соответствуют «хорошие» (k+1)-меры. Тогда задачу восстановления можно решить с помощью поиска пути заданной длины между двумя вершинами графа, которые соответствуют k-мерам из начала и конца частично прочитанного фрагмента.
3. Сборка генома. Для полной сборки генома из относительно длинных последовательностей применяются сторонние программные решения.

Исследование данного метода проводилось на примере искусственно созданного диплоидного генома, похожего на геном позвоночных, длиной $1{,}8\times10^9$ нуклеотидов. Исследования проводились для $k = 30$.

До первого этапа было $3{,}27\times10^9$ различных k-меров («хороших» – $1{,}54\times10^9$), после стало $1{,}95\times10^9$ («хороших» – $1{,}69\times10^9$). Этот этап выполнялся одни сутки на 24-ядерном компьютере с 24 ГБ оперативной памяти.

В ходе второго этапа было восстановлено 66 процентов фрагментов. Этот этап тоже выполнялся в течение суток. Работа проводилась на 24-ядерном компьютере с 64 ГБ оперативной памяти.

На третьем этапе был запущен сборщик Newbler. В ходе этого этапа были получены большие фрагменты генома с суммарной длиной $1{,}48\times10^9$ нуклеотидов и длиной максимального фрагмента $6{,}28\times10^6$. Средняя длина фрагмента 3×10^3 нуклеотидов. Работа заняла 36 часов на 24-ядерном компьютере с 64 ГБ оперативной памяти.

В дальнейшем планируется повысить качество получаемых результатов и отказаться от использования сторонних средств.

УДК 004.415.5

КОНТРОЛЬ КАЧЕСТВА ПРОГРАММ СО СЛОЖНЫМ ПОВЕДЕНИЕМ: ИНКРЕМЕНТАЛЬНАЯ ВЕРИФИКАЦИЯ

А.А. Борисенко
(Санкт-Петербургский государственный университет информационных технологий, механики и оптики)
Научный руководитель – к.т.н., доцент Е.В. Пасынков
(ООО «ИнтеллиДжей Лабс»)

В последнее время для контроля качества программ со сложным поведением все чаще используют метод верификации на модели. Он отлично подходит для автоматных программ, при этом формулировка всех функциональных требований происходит в терминах разрабатываемой модели. Спецификация, записанная с помощью языка темпоральной логики (LTL, CTL и т.п.) затем подается на вход внешнему верификатору. Такой подход хорош тем, что в случае нарушения хотя бы одного из предъявленных к программе требований позволяет отследить соответствующий контрпример – сценарий выполнения программы.

Верификацию на модели целесообразно использовать в больших проектах, в рамках которых разрабатываются программы, реализующие сложное поведение. Цена ошибки в них может быть слишком высокой (например, космическая отрасль, военно-стратегические комплексы, медицинское ПО, операции с банковскими картами). В автоматных моделях, соответствующим таким программам, количество состояний может достигать нескольких сотен и даже тысяч, что делает процесс верификации крайне ресурсоемким и медленным.

Важно отметить, что после незначительных изменений в автоматной модели с большим количеством состояний неэффективно запускать процесс проверки спецификаций заново. В связи с этим, встает задача оптимизации вычислений с использованием результатов предыдущих запусков.

Целью данной работы является изучение возможностей инкрементальной верификации для оптимизации проверки соответствия автоматной модели программы предъявляемым ей требованиям. В частности, в работе исследуется проблема локализации подмножества состояний, влияющих на выполнимость данной темпоральной формулы и существования разбиения исходной модели на несколько независимых компонентов, допускающих безопасные модификации. Основным результатом является выделение эвристик, позволяющих эффективно проводить верификацию модели на основе данных предыдущих вычислений. В работе предлагается несколько подходов к инкрементальной верификации с подробным описанием каждого из них, включая рассуждения о применимости на практике и оценки эффективности.

УДК 004.415.53+004.832.23

ГЕНЕРАЦИЯ ТЕСТОВ ДЛЯ ОЛИМПИАДНЫХ ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ С ИСПОЛЬЗОВАНИЕМ ЭВОЛЮЦИОННЫХ СТРАТЕГИЙ

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

Введение. В мире проводится большое число олимпиад по программированию. Они способствуют выявлению талантливых программистов среди школьников и студентов. Среди них можно отметить международную студенческую олимпиаду по программированию International Collegiate Programming Contest [1], проводимую Association for Computing Machinery, с развитой сетью отборочных соревнований, международную олимпиаду школьников по информатике [2], соревнования, проводимые компанией TopCoder [3], интернет-олимпиады по информатике и программированию [4] и многие другие.

На олимпиадах по программированию предлагается решить одну или несколько задач. Решением задачи является программа, написанная на одном из разрешенных языков программирования.

Программа тестируется на наборе тестов, неизвестных участникам. На работу программы накладываются определенные ограничения, такие как максимальное время выполнения и максимальный объем используемой памяти.

Решение считается прошедшим определенный тест, если оно при работе с ним не нарушило ограничений, завершилось без ошибок, и его ответ признан правильным. О конкретных видах задач и ограничениях можно прочитать, например, на сайте олимпиады [5].

Цель работы. Создание тестов для олимпиадных задач является сложным творческим процессом. В большинстве случаев, тесты создаются вручную или с помощью программ, генерирующих тесты по некоторому шаблону. Для некоторых задач такой способ генерации тестов приводит к тому, что набор тестов оказывается слабым, и в результате засчитывается множество неэффективных решений.

Цель работы – генерация тестов, на которых неэффективные решения работают как можно дольше. Качество таких тестов может выражаться количественно, что позволяет использовать для поиска качественных тестов методы оптимизации. В качестве метода оптимизации в работе используется (1+1)-эволюционная стратегия [6].

Метод генерации тестов рассматривается на примере задачи «Work for Robots» [7], размещенной на сервере Timus Online Judge [8].

Описание предлагаемого подхода. Для генерации теста выбирается решение, против которого нужно сгенерировать этот тест. При успешной генерации теста, им может быть «покрыт» целый класс решений, таким образом, при разумном выборе решений, можно сгенерировать достаточно полный набор тестов.

Исходный код решения модифицируется, чтобы, помимо ответа на решаемую задачу, решение вычисляло функцию приспособленности теста. Функция приспособленности может быть как числом, так и более сложным объектом, например, вектором чисел, который может сравниваться с другим таким вектором лексикографически. Такой подход обладает большой гибкостью, так как позволяет описывать достаточно разнообразные функции приспособленности, которые, в свою очередь, могут ускорить процесс нахождения оптимума.

Для генерации теста используется (1+1)-эволюционная стратегия, особью которой является тест, а функция приспособленности этого теста вычисляется при запуске на нем

модифицированного решения. Для задачи «Work for Robots», используемой для демонстрации подхода, тест задается симметричной матрицей булевых значений размером 50×50 – матрицей смежности графа, который содержится в тесте.

Используется два оператора мутации. Первый из них изменяет содержимое случайно выбранной ячейки матрицы (и симметричной ей ячейки) на противоположное, второй изменяет 10, 100 или 1000 таких ячеек. Операторы мутации применяются по очереди.

Результаты. До применения описанного подхода, для задачи «Work for Robots» было засчитано 86 решений. Тесты генерировались против восьми из этих решений. По итогам генерации было выбрано пять тестов, которые оказались трудными для семи из восьми выбранных решений. Эти тесты были добавлены в набор тестов на сервер Timus Online Judge. 45 из имевшихся решений не прошли новые тесты. Этот результат показывает высокую эффективность описанного метода.

Литература

1. ACM International Collegiate Programming Contest. http://en.wikipedia.org/wiki/ACM_ICPC
2. International Olympiad in Informatics. http://www.ioinformatics.org
3. TopCoder. http://www.topcoder.com/tc
4. Интернет-олимпиады по информатике. http://neerc.ifmo.ru/school/io/
5. Правила проведения полуфинала NEERC. http://neerc.ifmo.ru/information/contest-rules.html
6. *Bäck T., Hoffmeister F., Schwefel H.-P.* A Survey of Evolution Strategies. In Proceedings of the Fourth International Conference on Genetic Algorithms. 1991. pp. 2–9.
7. Задача «Work for Robots». http://acm.timus.ru/problem.aspx?num=1695
8. Timus Online Judge. Архив задач с проверяющей системой. http://acm.timus.ru

УДК 004.85

ПРИМЕНЕНИЕ ЦЕНТРАЛЬНЫХ ПАТТЕРН-ГЕНЕРАТОРОВ И ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ ДЛЯ ГЕНЕРАЦИИ ДВИЖЕНИЙ ЧЕЛОВЕКОПОДОБНЫХ РОБОТОВ

Ю.К. Чеботарева

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

В настоящее время робототехника очень плотно вошла в нашу жизнь. Роботы используются на автомобильных сборочных конвейерах, в хирургии, в армии и дома. Роботы-пылесосы чистят пол, роботы-игрушки развлекают людей, танцуют, исполняют роль домашних животных. Весьма вероятно, что в ближайшем будущем роботы станут неотъемлемой частью повседневной жизни. По мнению Билла Гейтса, возможно, скоро наступит новая эра, «где персональные компьютеры покинут свои столы и дадут нам возможность видеть, слышать и осязать предметы, а также управлять ими там, где мы не присутствуем физически».

Современные мобильные роботы, как правило, перемещаются при помощи колес, которыми просто управлять и которые относительно легко сконструировать. Недостатком колесных роботов является их способность перемещаться только по ровной и гладкой поверхности. В лабораториях, где такие роботы создаются и проходят испытания, это не является проблемой. Но в реальном мире лестницы и пороги становятся для них непреодолимым препятствием.

В естественных условиях ходьба (т.е. перемещение в результате сложной координированной деятельности конечностей) является наиболее удобным способом

перемещения. Наблюдения за животными показывают, что организмы, обладающие двумя, четырьмя, шестью и более конечностями способны быстро ходить, бегать, лазать по различным поверхностям.

Изучение механизма локомоции животных показало, что в выполнении ритмических движений, таких как дыхание, сокращение сердечной мышцы, ходьба и т.п., важную роль играют центральные паттерн-генераторы (центральные генераторы ритма, cental pattern generator). ЦПГ представляют собой специализированные нервные клетки или особым образом связанную их совокупность в пределах нервной системы, предназначенные для контроля моторной активности.

В последнее время подход, основанный на моделировании ЦПГ, получил широкое распространение в управлении движениями человекоподобных роботов. Несмотря на значительный прогресс в изучении механизма локомоции животных и человека, вопрос о том, как этот механизм позволяет генерировать стабильное движение для различных условий окружающей среды, остается открытым. ЦПГ обладают рядом интересных свойств, которые делают их весьма перспективными для решения задач стабилизации движений:

- модели CPG обычно имеют несколько параметров, позволяющих легко менять, например, скорость и направление движения или даже тип походки;
- ЦПГ идеально подходят для интеграции в модель движения сигналов, поступающих с датчиков робота, что позволяет сделать движение более стабильным и в некоторых ситуациях предотвратить падение;
- модели ЦПГ хорошо подходят для использования совместно с алгоритмами обучения и оптимизации.

В настоящей работе рассмотрен вопрос совместного применения ЦПГ для управления движением человекоподобных роботов и генетических алгоритмов для оптимизации параметров ЦПГ.

Автор ставит своей целью разработку метода генерации движения вперед для человекоподобных роботов и апробацию полученного метода на нескольких моделях роботов в среде симуляции Webots. В качестве тестовых моделей выбраны модели роботов Nao и HOAP-2.

Одна из проблем исследований в области движения человекоподобных роботов – отсутствие общего метода создания контроллера для управления роботом (в том числе и управления движением). Такой метод не должен зависеть от того какой моделью робота будет управлять контроллер, и благодаря этому построение контроллера по этому методу может быть в значительной части автоматизировано. В данной работе дополнительно ставится цель минимизировать влияние устройства робота на метод генерации движений. Таким образом, результаты полученные, например, для робота Nao, должны адаптироваться для робота HOAP-2 с минимальными временными затратами. Это становится возможным благодаря тому, что устройство современных человекоподобных роботов во многом похоже (в большинстве своем они имеют около 20 сервомоторов, примерно одинаково расположенных)

Другой проблемой в области движения человекоподобных роботов является отсутствие методики сравнения получаемых движений. По этой причине нельзя достоверно сказать, является ли новый метод генерации движений лучше или хуже существующих, и насколько он лучше или хуже. В данной работе полученные движения сравниваются с примерами движений для указанных моделей роботов, поставляемых вместе со средой Webots.

Результатом работы является сгенерированное описанным способом движение, превосходящее по скорости движение-эталон, а также программный модуль, который может быть использован для генерации движений такого же типа для подобных роботов.

УДК 004.4’242

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ ДЛЯ ПОСТРОЕНИЯ АВТОМАТОВ УПРАВЛЕНИЯ СИСТЕМАМИ СО СЛОЖНЫМ ПОВЕДЕНИЕМ НА ОСНОВЕ КОНТРАКТОВ И ТЕСТОВЫХ ПРИМЕРОВ

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

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.

Существуют различные способы построения автоматов управления со сложным поведением. Чаще всего такие системы строятся эвристически, но они могут содержать ошибки и требуют дополнительных проверок. В работе [2] был предложен способ построения автоматов с помощью генетического программирования на основе тестовых примеров. Однако такой вариант построения конечных автоматов требовал дополнительной валидации и верификации, а в случае обнаружения ошибки, необходимо было изменять тестовые примеры и заново строить систему. В любом случае, при таком подходе нельзя гарантировать поведение построенной системы на входных данных отличных от тестовых. В работе [3] было предложено использовать верификацию при вычислении функции приспособленности, однако вклад результата верификации был дискретным – 0 или 1. В работе [4] было предложено строить систему совместно на основе обучающих примеров и темпоральных формул. Формулы записываются на языке логики линейного времени (*Linear Temporal Logic, LTL*) и позволяют утверждать, что построенная система соответствует заявленной спецификации. Результат верификации учитывается при мутации, скрещивании и при вычислении функции приспособленности, причем вклад каждой темпоральной формулы – значение на отрезке [0, 1]. При этом 0 – формула нарушается сразу же в стартовом состоянии, а значение 1 – формула выполнятся.

Однако построение конечных автоматов на основе LTL-формул приводило к замедлению процесса построения. Кроме того, запись утверждений на языке LTL оказалась сложной для неподготовленных пользователей. Допустив ошибку в формуле и не заметив ее, процесс построения автомата может никогда не завершиться, и сложно выявлять такие ошибки.

В настоящей работе предлагается вместо или совместно с LTL-формулами использовать контракты [5, 6]. Обычно выделяют три вида контрактов: предусловия, постусловия и инварианты. В классическом понимании «программирования по контрактам» в объектно-ориентированном программировании: *предусловие* – ожидание метода объекта на входные параметры и состояние класса при его вызове, *постусловие* – обязательства метода при его завершении, *инвариант* – условие выполняющиеся на протяжении всего времени жизни экземпляра класса. При автоматном подходе контракты могут накладываться как на состояния, так и на переходы и группы состояний.

При автоматическом построении автомата управления заранее не известно о состояниях конечного автомата, поэтому не представляется возможным использование контрактов для состояний или групп состояний. Определим контракты через LTL-формулы. Язык LTL состоит из пропозициональных переменных и стандартных булевых и специальных

темпоральных операторов [7]. Для настоящей работы важны два из них:

– **X** (ne**X**t) – «*Xp*» – в следующий момент выполнено *p*;

– **G** (**G**lobally in the future) – «*Gp*» – всегда в будущем выполняется *p*.

Определим предусловие как $G(Xp_1 \rightarrow p_2)$ – если на следующем шаге выполнено p_1, то выполнено p_2; постусловие как $G(p_1 \rightarrow Xp_2)$ – если выполнено p_1, то на следующем шаге выполнено p_2; инвариант как $G(p_1 \rightarrow p_2)$ – если выполнено p_1, то выполнено p_2. Например, предусловие на переход можно определить как $G((!e \;\&\&\; Xe) \rightarrow p)$ или как $G(Xe \rightarrow p)$, постусловие как $G((e \;\&\&\; X!e) \rightarrow Xp)$ или как $G(e \rightarrow Xp)$, инвариант как $G(e \rightarrow p)$, где e – «переход по событию e», p – предикат.

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]. Можно показать, что для верификатора постусловие и предусловие оказываются одинаковыми, так как их отрицание представляются «похожими» автоматами Бюхи. В принципе, контрактом можно назвать любую LTL-формулу, отрицание которой приведет к заранее заданной структуре автомата Бюхи. Под «заранее заданной структурой» будем понимать недетерминированный конечный автомат, который эквивалентен автомату контракта с точностью до пометок на переходах. На рисунке представлены автоматы для предусловия (а), постусловия (б) и инварианта (в) на переходы.

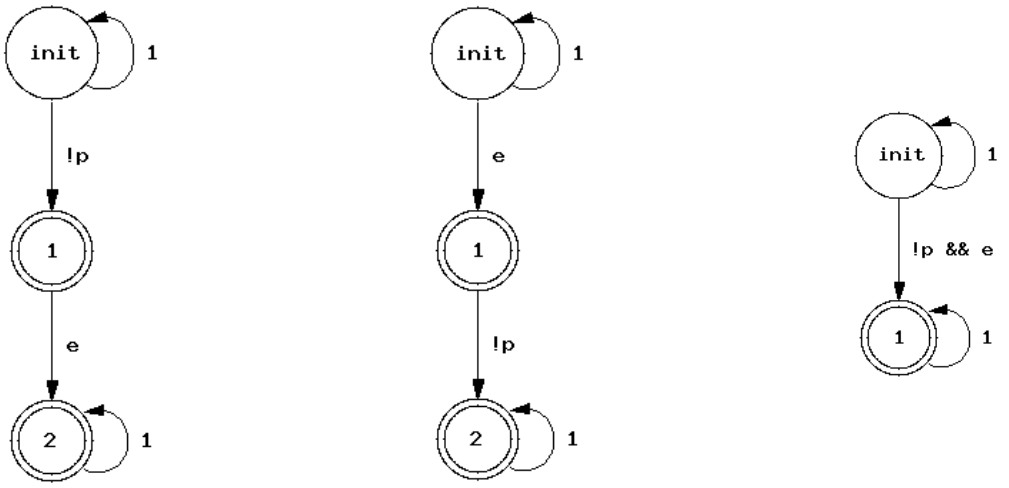

Рисунок. Автоматы Бюхи, построенные для отрицания LTL-формул G(Xe → p) (а); G(e → Xp) (б); G(e → p) (в)

При верификации произвольных темпоральных свойств, заранее не известна их семантика. Это приводит к тому, что, обнаружив контрпример в автомате, невозможно определить, какой переход нарушает формулу. Когда конечный автомат строится на основе контрактов, точно известно, какие переходы в контрпримере нарушают его. Например, для инварианта последний переход нарушает его, а для предусловия и постусловия – последний и предпоследний. В результате такой априорной информации упрощается процесс модификации автомата с целью соблюдения контракта.

В настоящей работе предлагается генетический алгоритм построения конечных автоматов на основе тестовых примеров и контрактов. Особь в каждом поколении представляет собой конечный автомат, с событием на переходах и с указанием числа выходных воздействий [2]. Тестовые примеры позволяют расставлять выходные воздействия. Контракты позволяют проверять, что конечный автомат соответствует заявленной спецификации, и исключать из популяции особи, заранее ей не соответствующие, и модифицировать автомат так, чтобы контракт соблюдался. Для этого выполнимость контракта оценивается как значение на отрезке [0, 1], и полученный результат вносит вклад в

функцию приспособленности. Переходы, нарушающие контракт, с большей вероятностью подвергаются мутации и/или не участвуют в скрещивании.

Были проведены экспериментальные исследования на примере автомата управления дверьми лифта из работы [4]. Было проведено 1000 построений для каждого из перечисленных выше методов. При использовании только тестовых примеров построенный автомат менее чем в 1% случаев соответствовал спецификации, при использовании совместно верификации и тестовых примеров среднее число вычислений функции приспособленности оказалось равным – 827857. При совместном использовании тестов и контрактов – 710882.

В работе исследована возможность применения контрактов для автоматического построения автоматов управления систем со сложным поведением, предложен метод мутации и скрещивания на основе выполнимости или не выполнимости контракта, проведено экспериментальное исследование метода при построении автомата управления дверьми лифта.

Литература

1. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. СПб: Питер, 2010.
2. 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.
3. Johnson C. Genetic Programming with Fitness based on Model Checking. Lecture Notes in Computer Science. Springer Berlin/Heidelberg. – 2007. – V. 4445. – Pp. 114–124.
4. Егоров К.В., Царев Ф.Н., Шалыто А.А. Применение генетического программирования для построения автоматов управления системами со сложным поведением на основе обучающих примеров и спецификации // Научно-технический вестник СПбГУ ИТМО. – 2010. – № 69. – С. 81–85.
5. Мейер Б. Объектно-ориентированное конструирование программных систем. – М.: Интернет-университет информационных технологий, 2005.
6. Борисенко А., Федотов П., Степанов О., Шалыто А. Разработка надежного программного обеспечения со сложным поведением / Сборник трудов конференции 5th Central and Eastern European Software Engineering Conference in Russia. – М. – 2009. – С. 125–128.
7. Кларк Э., Грамберг О., Пелед Д. Верификация моделей программ: Model Checking. – М.: МЦНМО, 2002.

УДК 004.052: 004.4'23

РАЗРАБОТКА МЕТОДОВ БЕЗОПАСНОГО ВНЕСЕНИЯ ИЗМЕНЕНИЙ В АВТОМАТНЫЕ ПРОГРАММЫ

П.В. Федотов

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

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 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. Создан каталог рефакторингов автоматов.
2. Для каждого рефакторинга описана техника его выполнения.
3. Для каждого рефакторинга обоснована корректность его выполнения.
4. Реализованы средства, позволяющие автоматизировать процесс рефакторинга в инструментальном средстве Unimod.

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 004.4'232

АЛГОРИТМЫ СРАВНЕНИЯ И СЛИЯНИЯ АБСТРАКТНЫХ СИНТАКСИЧЕСКИХ ДЕРЕВЬЕВ В СРЕДЕ ЯЗЫКОВО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

Е.В. Геращенко
Научный руководитель – к.т.н., доцент В.С. Гуров

Постановка проблемы. В настоящее время получает широкое распространение парадигма языково-ориентированного программирования. В системах, разработанных для инструментальной поддержки данной парадигмы, таких как JetBrains MPS, код программ представляется в виде абстрактных синтаксических деревьев (моделей).

Для совместной работы над проектом разработчики пользуются системами контроля версий, которые позволяют хранить и просматривать историю изменения программ, вносить изменения, создавать различные ветки разработки для независимой работы над разными частями проекта, производить их слияние, разрешая конфликты.

Популярные среды разработки, предназначенные для текстовых языков программирования (IntelliJ IDEA, Eclipse, NetBeans, PyCharm, и т. д.), как правило, имеют достаточно тесную интеграцию с системами контроля версий, позволяя выполнять стандартные процедуры прямо из среды, например, просмотр истории и интерактивное слияние изменений. Также очень полезной является подсветка текущих изменений относительно последней версии файла прямо в текстовом редакторе.

При работе с моделями необходимы иные алгоритмы для просмотра изменений, слияния и отображения текущей разницы, потому как существующие алгоритмы для текстового кода программ плохо подходят, даже если применяются к сериализованным

версиям моделей (например, в виде XML-файлов). Во-первых, они не учитывают древовидной структуры файла, и значит, в результате слияния может получиться некорректный XML-код). Во-вторых, они заставляют пользователя вникать в XML-представление файла, которое для него непривычно и сложно для восприятия. В-третьих, отображение отличий от базовой версии вообще бесполезно, потому что редактирование моделей происходит на более высоком уровне, с помощью тексто-подобных (проекционных) редакторов.

Цель работы. Разработать алгоритм построения разницы двух синтаксических деревьев (моделей) с возможностью отката отдельных изменений.

Разработать алгоритм автоматизированного слияния изменений моделей с возможностью ручного применения или отмены отдельных изменений.

Модифицировать алгоритм построения разницы, сделав его инкрементальным, т.е. быстро реагирующим на изменение модели и перестраивающим актуальный набор изменений в реальном времени.

Внедрить разработанные алгоритмы в среду разработки JetBrains MPS и создать для них удобный графический пользовательский интерфейс.

Базовые положения исследования. *Модель* – древовидная структура программы, состоящая из узлов, свойств и ссылок на другие узлы. *Разница моделей* – набор единичных структурных изменений между двумя моделями, который можно отображать визуально в графическом пользовательском интерфейсе. *Слияние изменений моделей* – объединение двух разных наборов изменений (двух версий модели) относительно одной общей исходной версии модели.

Промежуточные результаты. Формализовано понятие разницы моделей и единичного изменения.

Разработаны способы использования указанных процедур; сформулированы требования к пользовательскому интерфейсу.

Изучены возможные проблемы, связанные с внедрением алгоритмов в среду разработки JetBrains MPS.

Основной результат. Разработаны алгоритмы построения разницы двух версий модели, объединения изменений модели и инкрементальный алгоритм построения разницы двух версий модели. Разработанные алгоритмы внедрены в систему разработки JetBrains MPS, создан графический пользовательский интерфейс, визуализирующий разницу, позволяющий откатывать отдельные изменения (для разницы), разрешать конфликты в пользу применения тех или иных изменений.

УДК 004.021

УЧЕТ СЛОЖНОСТИ ВАРИАНТА ЗАДАНИЯ ПРИ ОЦЕНКЕ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ

С.И. Гиндин
Научный руководитель – к.т.н., доцент В.С. Гуров

Краткое вступление, постановка проблемы. В условиях стремительно развивающегося информационного общества и информатизации всех сфер деятельности человека требования к современному специалисту могут быть удовлетворены только путем постоянного повышения его квалификации и уровня образования. Важнейшими качествами

специалиста становятся профессиональная гибкость, готовность учиться на протяжении всей жизни и показатель приобретенной компетенции. В любой образовательной системе особое место занимает контроль – отслеживание усвоения знаний и мониторинг качества обучения. Внедрение новых образовательных и информационных технологий в учебный процесс усиливает потребность в автоматизированных системах, позволяющих объективно, быстро и качественно оценивать знания учащихся. На сегодняшний день одной из наиболее технологичных и объективных форм контроля является тестирование, повсеместно используемое во многих странах мира, в том числе, и в России, в области подготовки и сертификации специалистов, мониторинга и оценки качества образования. Развитие педагогики позволило существенно модернизировать тестовые технологии контроля знаний и поднять их на качественно иной уровень. Использование информационных технологий позволило автоматизировать обработку результатов и сделало возможным массовое тестирование, а также привело к созданию компьютерных систем тестирования знаний. С развитием на базе средств вычислительной техники систем централизованных контрольно-оценочных процедур возникла потребность в поиске новых методов и технологий для повышения эффективности универсальных способов оценки знаний в условиях большого числа параллельных вариантов тестовых заданий.

Цель работы. Необходимо разработать методику учета сложности варианта задания при оценке результатов тестирования для объективного определения уровня подготовки тестируемых.

Базовые положения исследования. Образовательное тестирование – измерение некоторых интеллектуальных показателей или достижений. В соответствии с современными представлениями, образовательное тестирование возможно только тогда, когда образовательная информация по дисциплине или предметной области может рассматриваться в виде образовательной статистической совокупности с обязательным выделением единицы данной совокупности. **Главное назначение тестирования** – контроль уровня знаний обучаемых как качественного показателя. **Методика оценки результатов тестирования** – способ перевода данного качественного показателя в количественный вид. К нему предъявляются требования точности, доступности, объективности, не зависящей от установок проверяющего, оперативности получения результатов и сопоставимости полученных результатов, насколько это возможно, без учета особенностей испытуемых и проверяющих.

Промежуточные результаты. Показана необходимость учета связи между результатами выполнения теста и относительными отличиями заданий в конкретных предъявленных из общего банка вариантах.

Проведен информационно-аналитический обзор существующих подходов к традиционной и адаптивной оценке результатов тестирования и выделены их ограничения.

Предложена модель оценки результатов тестирования, включающая как уровень обученности (как измеряемый параметр), так и трудность тестовых заданий (как параметр задания, устанавливаемый при его аттестации), позволяющая преодолеть указанные ограничения. Приведено обоснование адекватности построенной модели.

Предложен способ программной реализации предложенной модели.

Основной результат. Создана инновационная методика адаптивной оценки тестирования, учитывающая сложность предъявленного варианта, что позволяет повысить объективность и точность результатов каждого испытуемого в группе как независимо, так и относительно всей группы.

УДК 004.852

ПРИМЕНЕНИЕ СКРЫТЫХ МАРКОВСКИХ МОДЕЛЕЙ ДЛЯ КЛАССИФИКАЦИИ И СЕГМЕНТАЦИЯ СЛОИСТЫХ СЛУЧАЙНО-НЕОДНОРОДНЫХ СРЕД В ОКТ

А.Х. Киракозов
Научный руководитель – д.т.н., профессор И.П. Гуров

Автоматическая обработка получаемых в оптической когерентной томографии (ОКТ) [1] изображений имеет широкое применение в медицине и материаловедении. В данной работе рассматриваются следующие две задачи: классификация и сегментация случайно-неоднородных сред с ярко выраженной слоистой структурой.

Классификация позволяет по получаемой в ОКТ томограмме определить тип рассматриваемой среды. Сегментация представляет собой выделение на изображении областей, соответствующих слоям с одинаковыми характеристиками. Сегментация изображений в биомедицине позволяет получить ценную информацию о морфологических изменениях структуры биотканей.

Для решения задач классификации существует множество различных методик: нейронные сети, байесовские сети доверия, скрытые марковские модели (СММ) [2]. В данной работе были использованы СММ, поскольку они имеют относительно простую структуру и в явном виде учитывают стохастические свойства системы. Кроме того, скрытые состояния удобно использовать для описания каждого отдельного слоя среды, что позволяет решить не только задачу классификации, но и задачу сегментации.

СММ представляют собой математическую модель для описания статистических сигналов. Данная модель описывается следующими элементами: алфавитом наблюдаемых символов, множеством скрытых состояний, матрицей вероятности переходов из одного скрытого состояния в другое, распределением вероятностей появления символа в каждом из состояний и распределением вероятности начального состояния. В каждый момент времени система находится в одном из состояний, генерирует некоторый наблюдаемый символ и переходит в следующее состояние (или остается в том же). Переход в следующее состояние определяется только матрицей вероятностей переходов. Аналогичным образом генерируемый в каждый момент времени символ зависит только от текущего состояния системы. Распределение начального состояния системы позволяет сгенерировать состояние системы в начальный момент.

В данной работе рассматриваются томограммы слоистых сред. Основная идея состоит в том, чтобы для каждого типа среды построить свою СММ, которая впоследствии позволит классифицировать и сегментировать данную среду. Каждый А-скан такой томограммы имеет схожую структуру, поэтому последовательность значений интенсивности А-скана можно представить в виде последовательности наблюдаемых символов в СММ. Каждому слою системы сопоставляется отдельное скрытое состояние системы, которое должно описывать статистические свойства данного слоя.

Основная сложность возникает в определении матрицы вероятностей переходов и распределении вероятностей появления символов. Для решения этой задачи используется механизм обучения СММ, который, используя некоторые начальные параметры СММ и наблюдаемую последовательность сигналов, пытается изменить параметры СММ таким образом, чтобы максимизировать вероятность появления данной наблюдаемой последовательности в этой СММ. К сожалению, существующие алгоритмы, применяемые в данной ситуации, позволяют найти лишь локальный максимум, при этом важно точное определение начальных параметров СММ.

Зачастую в качестве начальных параметров СММ используются равновероятные распределения. В данной работе в ряде случаев были использованы специальные эвристики,

которые позволили определить начальные параметры системы точнее, чем обычное равномерное распределение.

Опишем подробнее методику классификации сред. Для каждого типа среды выбирается набор А-сканов, на котором обучается СММ, определяющая статистические свойства данной среды. Затем для классифицируемого А-скана, не входящего в обучающую коллекцию, вычисляется вероятность его появления в каждой СММ. А-скан относится к той среде, для которой СММ дала максимальную вероятность появления А-скана в ней. Методика классификации сред с использованием СММ была опробована на сгенерированных случайно-неоднородных средах. Более 90% сред были классифицированы верно.

Опишем подробнее методику сегментации среды. Сначала для среды аналогичным с классификацией методом строится СММ. Затем по наблюдаемому А-скану необходимо определить, какие его части какому слою принадлежат. Для этого необходимо выбрать последовательность состояний системы, которая лучше всего соответствует сегментируемому А-скану. Отметим, что данная задача является стандартной для СММ и для нее существуют готовые решения. После того, как наилучшая последовательность состояний системы найдена, остается лишь сопоставить каждому состоянию слой системы. Методика была опробована для сегментации слоев томограммы сухожилий свиной кожи [3] и позволила точно определить границы слоев для данной томограммы.

Полученные в ходе работы результаты показывают целесообразность применения СММ для решения задач классификации и сегментации случайно-неоднородных слоистых сред.

Литература

1. Гуров И.П. Оптическая когерентная томография: принципы, проблемы и перспективы. В кн.: Проблемы когерентной и нелинейной оптики // Под ред. И.П. Гурова и С.А. Козлова. СПб: СПбГУ ИТМО, 2004. – С. 6–30.
2. Lawrence R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition // Proc. IEEE. – 1989. – V. 77. – No. 3. – P. 257–286.
3. Boer J., Srinivas S., Malekafzali A., Chen Z., Nelson J. Imaging thermally damaged tissue by polarization sensitive optical coherence tomography // Optics express. – 1998. – V. 3. – No 6. – P. 212–218.

УДК 004.4'23

ПРИМЕНЕНИЕ ЗАВИСИМЫХ СИСТЕМ ТИПОВ ДЛЯ ВАЛИДАЦИИ И ВЕРИФИКАЦИИ АВТОМАТНЫХ ПРОГРАММ

Я.М. Малаховски

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

Валидация и верификация являются ключевыми требованиями при разработке ответственных систем. Автоматное программирование [1] позволяет значительно упростить процесс верификации, поскольку переход от автоматной программы к модели Крипке может быть произведен автоматически и изоморфно [2]. В работе [3] был предложен подход к реализации событийных конечных автоматов [1] на функциональных языках программирования и показаны его преимущества перед традиционными реализациями. При этом для валидации функций переходов использовались свойства алгебраических типов данных. В работе [4] было предложено обобщение данного подхода на структурные автоматы с переменными. Однако валидация производилась не в процессе компиляции, а в процессе конструирования автомата во время работы программы.

Валидация отдельного автомата обычно не требует много времени, поскольку, на практике, размер формул на дугах графа переходов весьма мал. При этом время валидации

системы автоматов есть сумма времен валидации каждого автомата в отдельности. Таким образом, добавление нового автомата не сильно увеличивает задержку при запуске программы, а потому результаты работы [4] являются пригодными для практического использования. Однако, для верификации данный подход не применим, поскольку время верификации системы автоматов есть функция от произведения мощностей множеств состояний каждого автомата. Таким образом, верификация программы должна быть частью процесса ее сборки и не может производится в процессе исполнения программы. Недостатком существующих средств верификации [2] является то, что спецификация программы, выраженная на языке, понятном верификатору, не является частью самой программы.

В настоящей работе предлагается метод верификации автоматных программ с использованием темпоральной логики CTL, производимый во время компиляции компилятором языка на котором выражены как автоматная программа, так и ее спецификация.

Разработанный подход основан на использовании зависимых систем типов в комбинации со встроенными предметно-ориентированными языками программирования (eDSL). При этом, при помощи встроенных предметно-ориентированных языков выражаются автоматные программы и спецификации на языке CTL, а проверка того, что автоматная программа удовлетворяет спецификации производится при помощи доказательства утверждений в зависимой системе типов функционального языка программирования.

Для апробации и практического применения предложенного подхода был выбран язык программирования Agda (версии 2). eDSL для описания автоматов был, насколько это возможно, перенесен из работы [4].

Литература

1. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб: Питер, 2010. – 176 с.
2. Вельдер С.Э., Лукин М.А. Шалыто А.А., Яминов Б.Р. Верификация автоматных программ. – СПб: СПбГУ ИТМО, 2011.
3. Малаховски Я.М., Шалыто А.А. Реализация конечных автоматов на функциональных языках программирования // Информационно-управляющие системы. – 2009. – №6. – С. 30–33.
4. Малаховски Я.М., Корнеев Г.А. Валидация автоматов с переменными на функциональных языках программирования // Научно-технический вестник СПбГУ ИТМО. – СПб: СПбГУ ИТМО, 2010. – № 6. – С. 73–77.

УДК 004.4'23

ВЕРИФИКАЦИЯ АСПЕКТОВ УПРАВЛЕНИЯ РЕСУРСАМИ АВТОМАТНЫХ ПРОГРАММ

Я.М. Малаховски

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

Под валидацией автоматной программы [1] обычно подразумевают проверку полноты и непротиворечивости условий, надписанных над дугами диаграмм переходов. При этом процесс валидации полностью игнорирует язык, принимаемый автоматом, и язык выходных воздействий автомата. Однако, существуют объекты управления, требующие наложения строгих ограничений на эти языки. Проверка того, что автоматная программа удовлетворяет такого рода ограничениям обычно производится при помощи верификации с применением языков темпоральных логик LTL, CTL и CTL* [2].

Однако существуют классы ограничений, не выразимых на обозначенных языках, поскольку они являются языками, описывающими регулярные свойства, в то время как, например, свойство «автомат принимает только правильные скобочные последовательности» – контекстно-свободным. При этом, для проверки таких свойств не может существовать алгоритма верификации, поскольку, в общем случае, задача о проверке того, что некоторый регулярный язык является подмножеством контекстно-свободного языка, неразрешима [3]. Тем не менее, существуют полезные свойства автоматных программ, которые являются контекстно-свободными, но не регулярными.

В настоящей работе предлагаются методы проверки некоторых из таких свойств:

– корректная работа с типизированным стеком;
– корректное управление ресурсами.

В задаче о корректной работе автомата с типизированным стеком объектом управления является стек, у которого каждому элементу соответствует элемент некоторого множества – тип. Автомату доступны две операции: «добавить элемент на стек» и «снять элемент со стека». При этом, операция добавления элемента на стек не зависит от типа добавляемого элемента, а операция снятия элемента со стека может снять только элемент «ожидаемого» типа (в случае, если на вершине стека находится элемент не «ожидаемого» типа или стек пуст, результатом операции является специальный элемент «ошибка»). Говорят, что автомат корректно работает с типизированным стеком тогда и только тогда, когда все его вызовы операций снятия элемента со стека не возвращают «ошибок».

В задаче о корректном управлении ресурсами объектом управления является бинарный вектор длины n («ресурсы»). Автомату доступны две операции: «установить ячейку номер i в еденицу» («занять ресурс») и «установить ячейку номер i в ноль» («освободить ресурс»), где i не больше n. При этом, попытка дважды установить ячейку в одно и тоже значение (занять уже занятый ресурс или освободить уже освобожденный) считается ошибкой. Требуется проверить, что автомат никогда не производит ошибочных действий.

Данные свойства предлагается проверять при помощи их сведения к задаче о правильной скобочной последовательности с последующим ее решением непосредственным анализом структуры автомата.

Разработанные методы были реализованы в качестве дополнения к системе, разработанной в работе [4].

Литература

1. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб: Питер, 2010. – 176 с.
2. Вельдер С.Э., Лукин М.А. Шалыто А.А., Яминов Б.Р. Верификация автоматных программ. – СПб: СПбГУ ИТМО, 2011.
3. Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений. – М: Вильямс, 2008.
4. Малаховски Я.М., Корнеев Г.А. Валидация автоматов с переменными на функциональных языках программирования // Научно-технический вестник СПбГУ ИТМО. – СПб: СПбГУ ИТМО, 2010. – № 6. – С. 73–77.

УДК 004.021

СУФФИКСНЫЕ АВТОМАТЫ С СОХРАНЕНИЕМ ПРОМЕЖУТОЧНЫХ ВЕРСИЙ И ИХ ПРИЛОЖЕНИЯ

Д.А. Паращенко, А.С. Станкевич
Научный руководитель – д.т.н., профессор А.А. Шалыто

Структуры данных, которые хранят свои промежуточные версии, имеют более широкий круг приложений, чем их варианты без такой возможности. Классическим примером является задача нахождения области, содержащей заданную точку. Она может быть элегантно решена с помощью применения дерева поиска с сохранением промежуточных версий.

Не существует общего механизма, позволяющего после совершения какие-либо модификаций над обычной структурой данных вернуться к ее предыдущей версии. Структуры данных, хранящие свою хронологию, называются персистентными.

Существует несколько видов персистентных структур данных. Частичная персистентность позволяет модифицировать последнюю версию структуры данных и делать запросы к любой из промежуточных версий. Полная персистентность позволяет совершать запросы и модификации с любой версией структуры данных. При таком типе персистентности версии образуют не линейную, а древовидную структуру. В обоих случаях доступ может осуществляться по номеру требуемой версии.

В настоящее время для решения множества строковых задач используются суфиксные деревья, суффиксные массивы и суффиксные автоматы. Также известны методы, позволяющие делать суффиксные деревья персистентными. Возникает вопрос о возможности эффективной реализации персистентных суффиксных автоматов.

Рассмотрены существующие методы модификаций структур данных с целью добавления в них возможности сохранения промежуточных версий приминительно к суффиксному автомату. Предложен метод, позволяющий с небольшими дополнительными затратами сделать суффиксный автомат персистентным.

Свойство персистентности позволяет суффиксному автомату поддерживать помимо операции добавления символа операцию удаления последнего символа. При этом последняя операция не вызывает никаких дополнительных временных затрат.

Кроме того, персистентность позволяет использовать суффиксные автоматы в многопоточных приложениях, в которых предъявляются особые требования к быстродействию.Применение персистентной структуры данных позволяет выполнять операции по модификации суффиксного автомата без блокировки операций чтения. В случае суффиксных автоматов это является достоинством, так как модификация суффиксного автомата может занимать до $O(n)$ времени.

В работе выполнен анализ существующих методов построения персистентных структур данных, а также разработан эффективный метод построения персистентного суффиксного автомата. При использовании предлагаемого метода доступ к прежним версиям суффиксного автомата в большинстве случаев не требует значительных дополнительных временных затрат, однако, для некоторых видов строк верхняя оценка времени выполнения запросов может увеличиться в $O(\log n)$ раз.

Предложен метод, позволяющий поддерживать множество допускающих состояний суффиксного автомата в процессе его построения. При этом требуется $O(\log n)$ дополнительного времени и $O(1)$ дополнительной памяти на каждый символ базовой строки суффиксного автомата. В связи с тем, что описанный метод основывается на использовании древовидной структуры данных, можно без дополнительных затрат хранить промежуточные версии множества допускающих состояний суффиксного автомата, тем самым, сделав упомянутое множество персистентным.

УДК 004.021

ОБРАБОТКА СТРОК НА ОСНОВЕ СУФФИКСНЫХ АВТОМАТОВ

Д.А. Паращенко, А.С. Станкевич
Научный руководитель – д.т.н., профессор А.А. Шалыто

В настоящее время для решения большого числа строковых задач применяются суффиксные деревья. При этом все известные алгоритмы построения суффиксного дерева за линейное время достаточно сложны для понимания и реализации.

В настоящей работе разработан достаточно простой алгоритм построения суффиксного дерева за линейное время, содержащий в качестве одного из своих этапов построение суффиксного автомата. Таким образом, помимо алгоритмов Вайнера, Мак-Крейта и Укконена предложен еще один алгоритм построения суффиксного дерева за линейное время. Кроме этого в работе проведено сравнение времени их работы, а также сложности их реализации.

Поясним, как появилась идея использовать суффиксный автомат для решения рассматриваемой задачи. Ввиду того, что суффиксные деревья и суффиксные автоматы являются родственными структурами данных (суффиксный автомат является минимизированным суффиксным бором, а суффиксное дерево – сжатым суффиксным бором), то автор посчитал целесообразным для решения строковых задач вместо суффиксных деревьев использовать суффиксные автоматы. Одним из преимуществ этого подхода является тот факт, что суффиксный автомат является более простой структурой данных, а алгоритм его построения значительно проще в реализации, чем алгоритм построения суффиксного дерева.

Разработанный автором алгоритм построения суффиксного дерева за линейное время немного уступает другим алгоритмам построения суффиксного дерева (Укконена и Мак-Крейта) как по количеству используемой дополнительной памяти, так и по времени работы. Однако, он реализуется намного проще и быстрее упомянутых выше алгоритмов. Это позволяет применять его в случаях, когда требуется быстро написать правильно работающий код, затратив на это минимум усилий, например, на соревнованиях ACM ICPC.

В работе также предложен метод, позволяющий избавится от использования в алгоритмах суффиксных деревьев. Суть этого метода состоит в изменении алгоритма таким образом, чтобы вместо операций над суффиксным деревом в нем использовались операции над суффиксным автоматом. В связи с тем, что суффиксный автомат является намного более простой в использовании структурой данных по сравнению с суффиксным деревом, применение данного метода позволяет в большинстве случаях ускорить работу алгоритма.

Для демонстрации эффективности предложенного метода в работе рассмотрена задача о нахождении наибольшего общего префикса двух суффиксов. Основной причиной выбора этой задачи является то, что к ней можно свести большое число других задач на строках. Установлено, что применение метода позволяет значительно сократить время фазы инициализации алгоритма для решения рассмотренной задачи.

В дальнейшем планируется выяснить, что выгоднее: сразу строить сжатый суффиксный автомат, или строить, а затем сжимать суффиксный автомат. Также планируется произвести сравнение количества памяти, требуемой для хранения суффиксного дерева, и памяти, требуемой для хранения суффиксного автомата с поддержкой всех операций суффиксного дерева.

УДК 004.021

АДАПТИВНЫЙ АЛГОРИТМ ВЫЯВЛЕНИЯ ПЛАГИАТА ИСХОДНОГО КОДА ПРОГРАММ

Д.М. Пенькин
Научный руководитель – к.т.н., доцент В.С. Гуров

Краткое вступление, постановка проблемы. Одной из задач образования является необходимость осуществлять контроль знаний учащихся. Традиционными формами такого контроля являются устный и письменный опросы, проверка которых ручным способом занимает много времени. С появлением компьютеров и предметов, связанных с написанием программ, ручная проверка стала особенно неэффективной. Помимо нетривиальности проверки корректности работы написанной программы существует проблема обнаружения плагиата, проще говоря, списывания кода одним учащимся у другого. Есть множество способов с помощью всего нескольких действий модифицировать исходный код программы таким образом, что он останется работоспособным, однако с виду будет совсем не похож на оригинал. Большинство сред разработки в разы облегчают возможность запутать проверяющего и завуалировать плагиат. Уже существуют алгоритмы сравнения исходного кода двух программ, позволяющие по некоторой шкале оценить их схожесть. Если требуется сравнить несколько решений одной задачи, то необходимо сверять их попарно. Однако с развитием компьютерных систем тестирования знаний стало возможно массовое тестирование, при котором попарная сверка решений становится неэффективной по времени, а значит возникает потребность разработать алгоритм выявления плагиата, не предполагающий явного сравнения каждой пары решений.

Цель работы. Необходимо разработать алгоритм выявления плагиата исходного кода программ, более эффективный, чем сравнение каждой пары решений. Этот алгоритм должен учитывать как синтаксические, так и логические изменения кода, например, переименование переменных, выделение методов и пр.

Базовые положения исследования. *Плагиат* в рассматриваемой задаче – умышленное присвоение авторства чужого исходного кода программы, чужих идей его организации. В аспекте тестирования плагиат ведет к недостоверности оценки знаний учащихся. Выявление плагиата позволяет точнее оценить уровень знаний и предпринять меры для повышения качества образования. *Алгоритм выявления плагиата* – алгоритм, позволяющий среди множества решений найти семантически схожие и оценить степень этой схожести. *Главное назначение алгоритма выявления плагиата* – эффективно обрабатывать большое количество экземпляров исходного кода программ.

Промежуточные результаты. Проведен информационно-аналитический обзор существующих алгоритмов обнаружения сходства исходного кода программ, определены основные факторы их эффективности.

Показана возможность адаптивного применения существующих алгоритмов в зависимости от типа проводимого сравнения.

Предложен способ создания наращиваемого в процессе сравнения «следа», строящегося на основе семантической структуры исходного кода, позволяющий сравнивать программы не попарно, а с этим «следом». Приведена оценка трудоемкости разработанного алгоритма.

Предложен метод программной реализации алгоритма для конкретного языка программирования.

Основной результат. Предложен эффективный алгоритм выявления плагиата исходного кода программ, позволяющий значительно ускорить проверку результатов массовых тестирований по программированию.

УДК 004.8

МНОГОЭТАПНОЕ ОБУЧЕНИЕ АГЕНТОВ, УПРАВЛЯЕМЫХ АВТОМАТАМИ, С ПОМОЩЬЮ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

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

Краткое вступление, постановка проблемы. В работе рассмотрено многоэтапное генетическое программирование. Генетическое программирование – один из способов нахождения оптимального решения задачи, основанный на идеях биологической эволюции и естественного отбора [1–3]. Задача ускорения поиска оптимума актуальна, так как любые эксперименты с использование генетического программирования требуют использования больших вычислительных ресурсов. Многоэтапное генетическое программирование в данном случае состоит в выращивании особи по частям, изменении условий решаемой задачи от более простых к более сложным, а также в смене в процессе эксперимента операторов отбора, скрещивания и их параметров.

Цель работы. Необходимо проверить, имеет ли смысл применять многоэтапное генетическое программирование для ускорения получения результата.

Базовые положения исследования. В работе рассмотрена модифицированная задача «Умный муравей-3» [4]. В изначальной постановке особе, управляемой конечным автоматом [5], требуется за заданное число ходов на тороидальном поле съесть как можно больше «яблок». В модифицированной задаче размеры поля произвольны, и существует возможность добавить на поле «золотые» яблоки с ценой в 10 раз большей, чем у обычных яблок.

Промежуточные результаты. Приведем несколько утверждений, которые будут использоваться в дальнейшем. На основе известной теоремы схем сформулирована гипотеза.

Гипотеза 1. Генетический алгоритм стремится достичь близкого к оптимальному результата за счет комбинирования схем с приспособленностью выше среднего малого порядка и малого охвата.

В книге [3] сформулирована еще одна гипотеза.

Гипотеза 2. Использование меньшего числа точек разрыва в генетических операторах приводит к повышению быстродействия генетического поиска.

Для начала в работе было необходимо хотя бы приблизительно узнать, какой оператор скрещивания является лучшим для задачи «Умный муравей-3». В результате эксперимента к 500-му поколению было получено несколько операторов скрещивания, обеспечивающих хороший результат. Был выбран 1000-ти точечный оператор скрещивания. Он использовался в дальнейших опытах. Эксперимент довольно хорошо исключает влияние вероятности на результат, так как в каждом опыте в каждом поколении значение функции приспособленности считалось на 50 случайных полях, а каждый опыт для каждого оператора скрещивания проводился 10 раз.

Результаты явно ставят под сомнение гипотезу 2 (одно и двух точечные операторы скрещивания дали плохие результаты). Исходя из гипотезы 1, результат генетического алгоритма получается за счет комбинирования схем с приспособленностью выше среднего

малого порядка и малого охвата. Поэтому оператор скрещивания с большим числом точек скрещивания (разрыва) дает большое разнообразие хромосом в поколении и тем самым ускоряет поиск оптимума, в то время как в генетическом алгоритме с малым числом точек скрещивания у оператора скрещивания такого разнообразия особей нет.

После выбора лучшего оператора скрещивания, была сделана попытка ускорить получение результата. Автомат особи выращивался до 500-го поколения. При этом использовался 1000-точечный оператор скрещивания и оператор отбора «Рулетка». Изменялась заполненность поля обычными яблоками первые 250 поколений. При этом оставшиеся 250 поколений поле было заполнено на 0,3 обычными яблоками. Затем сравнивались результаты к 500-му поколению.

По результатам стало понятно, что если необходимо вырастить автомат на поле с определенным числом объектов, то выгоднее при выращивании сначала в полтора-два раза увеличить число этих объектов, а затем уменьшить. Это можно объяснить тем, что в случае многоэтапного выращивания автомата особь первые 250 поколений активно учится справляться с ситуациями, когда она видит довольно много яблок перед собой, затем же оставшиеся 250 поколений учится также в ситуации, когда она видит перед собой меньше яблок. В случае одноэтапного выращивания автомата, когда все 500 поколений число яблок на поле не изменяется, особь с лучшим автоматом успевает лучше научиться действовать в ситуации, когда она видит немного яблок, но не успевает научиться хорошо реагировать на большое число яблок перед собой из-за довольно редкой встречаемости такой ситуации. Лучше всего изменять число яблок на 150 – 350 поколениях из 500 – по прошествии 30%–70% процентов от запланированного числа поколений. Разброс значений в опытах 3%–4%.

Введем в эксперименты золотые яблоки. Пусть обычные яблоки, использованные в предыдущих опытах, будут иметь цену единица. Тогда золотые яблоки имеют цену 10. В опытах было выполнено сравнение, что выгоднее в условиях, когда обычных яблок намного больше, чем золотых: обучать муравья на поле, на котором уже есть и обычные и золотые яблоки, или сначала обучить особь съедать только обычные яблоки, а затем добавить золотые. И наоборот, если на поле намного больше золотых яблок, чем обычных, что выгоднее: обучать муравья в ситуации, когда на поле имеются оба типа яблок, или сначала обучить муравья на золотых яблоках, а затем добавить обычные.

По результатам эксперимента стало ясно, что в случае, когда на поле яблок одного типа намного больше, чем другого, намного выгоднее сначала вырастить автомат, реагирующий только на один тип яблок, а затем добавить другой тип яблок. Это можно объяснить тем, что в случае, когда особь видит яблоки только одного типа, ее автомат имеет $2^8 = 256$ входных воздействий, и его представление в виде байтовой строки в программе не очень длинное. В случае же, когда на поле имеются и обычные и золотые яблоки автомат особи имеет $3^8 = 6561$ входное воздействие, и его представление в виде байтовой строки более чем в 10 раз длиннее, а чем длиннее искомая строка, тем дольше получается оптимальная строка.

Основной результат. Поставлена под сомнение известная гипотеза о том, что оператор с наименьшим числом точек скрещивания является лучшим для генетических алгоритмов.

Было показано, что если при обучении на некоторое время увеличить число объектов по сравнению с их числом, определенном в условии задачи, то это ускорит получение наиболее приспособленной особи. При наличии нескольких объектов различного типа, которые могут быть видны особи, управляемой автоматом, и при превалировании по числу одних объектов над другими на поле, выгодно выращивать автомат многоэтапно, постепенно добавляя новые типы объектов.

Литература

1. Koza J. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, 1992.

2. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. М.: Горячая линия-Телеком, 2008.
3. Емельянов В.В., Курейчик В.В., Курейчик В.М. Теория и практика эволюционного моделирования. М.: Физматлит, 2003.
4. Бедный Ю.Д., Шалыто А.А. Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей». СПбГУ ИТМО, 2007. http://is.ifmo.ru/works/ant
5. Поликарпова Н. И., Шалыто А. А. Автоматное программирование. – СПб: Питер, 2010. http://is.ifmo.ru/books/_book.pdf

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

В работе [1] выращивалась разливочная линия. При этом особи приспосабливались к объему емкости и скорости работы линии. Итоговое решение было приспособлено под конкретные настройки объекта управления. Для избегания подобного, пришлось, перед тестированием новой популяции, менять объем емкости и скорость линии. Подобные особенности генетического алгоритма не всегда можно предусмотреть и сложно исправлять уже после получения результатов и их анализа.

В предыдущей работе [2] рассматривались способы устранения непредвиденного решения задачи. За основу бралась логика CTL из-за простоты проверки ее утверждений на автоматных программах. Выражения, записанные на логике CTL, выступали в качестве спецификации. В операторах можно было использовать предикаты и действия автомата, что позволяло записывать следующие утверждения: Если под емкостью находится бутылка, то открой сливной клапан. Создание CTL формул требует гораздо меньше затрат, чем редактирование кода объекта управления. Теоретически возможно добавление новой CTL формулы в процессе работы генетического алгоритма. При проверке CTL формулы находился контр пример правильной работы. Анализируя информацию после проверки можно было сразу изменять любую особь в заведомо лучшую сторону, что очень сильно способствовало получению решения и ускоряло работу.

Недостаток использования логики CTL состоял в необходимости проверки выполнимости формул на особях. Затрачиваемое время на проверку зависело прямо пропорционально от числа особей и количества формул. Для сокращения числа особей в работе использовались вероятностные особи, с помощью которых можно было сократить количество особей в популяции и в итоге получить детерминированный автомат.

В результате работы [2] удалось получить простейшие автоматы и изучить особенности использования вероятностных особей. Исследования проводились с использованием фреймворка [1].

В данной работе изучается расширение функциональности CTL формул. Состояния выращиваемого автомата помечаются маркерами. Их использование в операторах CTL будет позволять создавать аналог вложенных автоматов с помощью генетического программирования. С помощью формул CTL можно будет контролировать переходы автомата между помеченными маркерами состояниями, и записывать формулы только для состояний, помеченных одинаковым маркером.

Расширенная функциональность должна позволить упростить генетическое программирование по спецификации, разбив программу на части. Что позволит программировать более сложные задачи, получая надежные решения.

Литература

1. 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.
2. Применение генетического программирования для генерации конечных автоматов по спецификации [Электронный ресурс] / Попов С.О. Сборник тезисов докладов конференции молодых ученых, Выпуск 5. Труды молодых ученых: СПбГУ ИТМО, 2010. – С. 22–23. – Режим доступа: http://fppo.ifmo.ru/kmu/kmu7/Выпуск%205/2_Технолог_программир.pdf, свободный.

УДК 004.8

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ И АВТОМАТОВ В АСИММЕТРИЧНЫХ ГИБРИДНЫХ ИГРАХ

Ю.И. Попов

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

Краткое вступление, постановка проблемы. В природе особи доказывают свое превосходство при взаимодействии с другими. Например, даже растения захватывают с помощью потомства наиболее благоприятные территории. Совместное развитие особей, при котором они друг на друга влияют, называется коэволюцией [1]. В математике подобные отношения исследует теория игр [2]. В работе предлагается решить математическую проблему с помощью коэволюции, подсказанной природой.

В данной работе на примере многошаговой игры на поле показывается возможность искусственного построения игрока, успешно справляющегося со своей задачей. Игрок рассматривается как конечный автомат, который на входные воздействия (текущая позиция), выдает выходное воздействие (следующая позиция). Однако, например, для шахмат на хранение подобного автомата необходимо хранилище сопоставимое с размерами Луны. Поэтому в работе рассматривается класс игр, в которых игрок видит лишь часть доски и ходит оптимально исходя из локального оптимума.

Цель работы. Возникает вопрос, как получить автомат, если в урезанной игре он помещается в память, но все же велик и сложен для рассмотрения человеком. В работе показывается, что построение подобных автоматов возможно с помощью генетических алгоритмов.

Базовые положения исследования. «Задача о муравьеде и муравьях» является обобщением задачи «Умный муравей-3» [3]. Задано поле – тороидальная поверхность произвольного размера с координатной сеткой, которая делит поверхность на клетки. На поле расположены муравьед, муравьи (еда для муравьеда) и яблоки (еда для муравьев). Яблоки и муравьи располагаются случайным образом. На одной из клеток находится муравьед (рисунок).

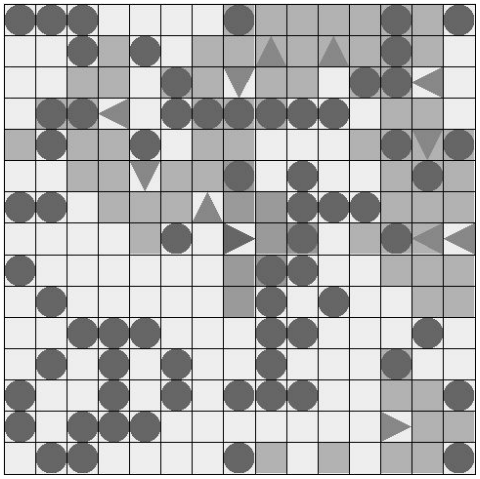

Рисунок. Тороидальная поверхность с муравьедом, муравьями и яблоками. Темный треугольник в центре изображает муравьеда, серые клетки рядом – это те клетки, которые он видит. Серые треугольники – муравьи, серые клетки рядом с ними – та часть поверхности, которую видят муравьи. Темные кружочки обозначают места расположения яблок

Цель каждого муравья – за заданное число шагов съесть как можно больше яблок и «не попасться» муравьеду. Цель муравьеда – помешать им сделать это, съев за заданное число шагов максимальное число муравьев. Муравьи не должны «наступать» на других муравьев и муравьеда, так как они при этом погибают. Правила игры изложены в работах [4, 5]. Для построения генетического алгоритма необходимо разработать способы представления автоматов, выбрать функции приспособленности, операторы скрещивания, мутации и отбора.

В игре с довольно сложной системой правил, картой обзора, простые эвристические методы не удастся найти за приемлемое время. На примере рассматриваемой игры показывается применимость генетических алгоритмов, которые существенно облегчают задачу поиска оптимальных стратегий.

Промежуточные результаты. В работе для решения гибридной асимметричной игры используются различные подходы: коэволюция, случайные особи, выращивание на случайных особях. Каждый из этих трех подходов применяется для получения автоматов, управляющих муравьедом и муравьем. В конце из всех экспериментов выбираются средние по приспособленности особи и сравниваются между собой. Заметим, что независимо от метода, муравьи выращиваются в процессе коэволюции, так как хотя, например, не влияют на случайного муравьеда, они активно взаимодействуют друг с другом.

В экспериментах по выращиванию в процессе коэволюции автомат муравьеда постепенно учится съедать все большее число муравьев. При этом если вначале муравьед передвигается случайным образом и чаще всего вращается на месте, то, как показали эксперименты, уже к 50-му поколению прослеживается его стремление догнать жертву, хотя иногда он сворачивает не туда. У муравьев также отмечается процесс роста числа съеденных яблок от поколения к поколению. На результат влияет то, что муравьи фактически соревнуются между собой за яблоки.

Основной результат. Лучший результат для автомата муравьеда получается при выращивании на муравьях со случайными автоматами. Коэволюция дает почти такой же результат, и особенно важно, что значения функций приспособленности автоматов муравьедов совпадают, когда их соперниками являются муравьи, управляемые случайными автоматами. Для автоматов муравьев ситуация похожа. Лучшим оказался автомат муравья, выращенный на случайных автоматах муравьеда.

Важно отметить, что коэволюционное выращивание автоматов муравьеда и муравья не способствовало их замыканию на успешное решение задачи только против тех, с кем они

взаимодействовали в процессе выращивания. В [1] также приведена игра, при решении которой с помощью коэволюции более половины выращенных решений сводились к известному для игры равновесию Нэша. Это подтверждает уместность использования коэволюционных методов наравне с другими при решении задач этого класса. Серия экспериментов показала возможность применения генетических алгоритмов для выращивания игроков в многошаговых асимметричных гибридных играх с неполной информацией. При этом успешному выращиванию способствовала простота представления особей – автоматы.

Литература

1. Koza J. Genetic programming. On the Programming of Computers by Means of Natural Selection. MA: The MIT Press, 1998.
2. Петросян Л.А. и др. Теория игр: Учеб. Пособие для ун-тов. – М: Высш. шк., Книжный дом «Университет», 1998.
3. Бедный Ю.Д., Шалыто А.А. Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей». СПбГУ ИТМО, 2007. http://is.ifmo.ru/works/ant
4. Попов С.И., Попов Ю.И., Шалыто А.А. Задача о муравьеде и муравьях / Сборник статей третьей Всероссийской научной конференции «Нечеткие системы и мягкие вычисления». Том II. Волгоград, 2009. – С. 57–63.
5. Попов С.И., Попов Ю.И., Шалыто А.А. Задача о муравьеде и муравьях // Информационные технологии. – М.: Новые технологии. – № 8. – 2010.

УДК 004.4'242

ОПТИМИЗАЦИЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ С ИСПОЛЬЗОВАНИЕМ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ

Е.В. Селифонов

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

Вступление. В последнее время для решения многих научных и прикладных задач требуется использование высокопроизводительных вычислений. Для поддержания возможности их проведения производители процессоров постоянно увеличивали их тактовую частоту, а также количество вычислительных ядер. Однако данных мер оказалось недостаточно для многих тяжелых задач, в которых требуется применение некоторых схожих операций для каждого элемента большого массива данных. Для таких вычислений стали использоваться изначально предназначенные для расчета трехмерной графики графические процессоры, так как в них поддерживается большое количество вычислительных конвейеров. Данная особенность позволяет одновременно обрабатывать несколько сотен элементов массива данных.

В первых компьютерах центральный процессор производил все необходимые расчеты. Далее появились дополнительные функциональные модули: сопроцессоры и отдельные процессоры. Они предназначались для выполнения определенных задач с более высокой, чем центральный процессор, скоростью и были выполнены в виде отдельных микросхем. Таким образом, графический процессор (*GPU*) был полностью отделен от центрального процессора (*CPU*). В настоящее время крупные производители процессоров решили вновь решили объединить данные модули на одной плате, что привело к появлению *APU* (*Accelerated Processing Unit*), например, *AMD Fusion*, *Intel Sandy Bridge*, *NVIDIA Project Denver*.

Данная работа является продолжением работы «Оптимизация программ для графических процессоров» и дополняет примененные в ней оптимизации программ для графических процессоров новыми, в том числе использующими преимущества *APU*.

Цель работы. Первой целью работы является доработка результатов предыдущей работы, в том числе, оптимизация генерации кода для графического процессора из нескольких шейдеров (подпрограмм) и генерация программы на языке *C++*, позволяющей провести описанные в специальной форме вычисления на графическом процессоре.

Второй целью является исследование возможностей применения связки из *CPU* и *GPU* для проведения высокопроизводительных вычислений, что позволит максимально эффективно использовать новый вид процессоров, *APU*. Предлагается унифицировать запись программ для центрального и графического процессоров с использованием особенностей их архитектур, а также найти оптимальное соотношение объема данных для одновременной обработки на этих двух различных типах процессоров.

Результаты. Реализован инструмент, позволяющий на основе набора шейдеров и описания схемы вычислений в специальном формате получить готовую к исполнению на любом современном графическом процессоре оптимизированную программу. В программе использован адаптивный алгоритм распределения нагрузки на центральный и графический процессоры.

УДК 004.891.2

РАЗРАБОТКА КОМПЬЮТЕРНЫХ МАНИПУЛЯТОРОВ ДЛЯ ФОРМИРОВАНИЯ МАТЕМАТИЧЕСКИХ УМЕНИЙ

В.А. Шишов
Научный руководитель – д.т.н., профессор С.К. Стафеев

Манипуляторы-тренажеры следует рассматривать как основное средство поддержки практического курса математики в условиях электронного обучения (e-learning). Представляя собой простейшую экспертную систему, они позволяют добиться практического усвоения изучаемого материала, имитируя деятельность наставника-преподавателя. Система обучения может иметь структуру «мастера» для пошаговых инструкций, а также быть основанной на базе знаний в случае сложного и неоднозначного процесса обучения.

Несмотря на проработанность структуры на уровне концептуализации для данных имитирующих систем, в разработке манипуляторов-тренажеров наблюдается дефицит унифицированных структур на уровне специализации и реализации. Предметом нашего исследования является разработка унифицированной модульной структуры, описывающей произвольный манипулятор-тренажер. Данная структура должна отвечать современным требованиям к ПО образовательного назначения, в том числе, стандартам проектирования программного обеспечения, в особенности, экспертных систем. Широкое и активное внедрение систем ИИ, в частности манипуляторов, в обучающий процесс делают проблему актуальной, а результаты практически значимыми.

Основным результатом работы является построение базовой модульной архитектуры манипулятора. Данная архитектура представлена в виде шаблона проектирования «Manipulator», отвечающего стандартам представления паттернов проектирования программного обеспечения. Модульная структура обеспечивает гибкость в настройке, позволяя разработчику сосредоточиться на разработке отдельных компонентов манипулятора. В докладе представлены схемы реализации элементов базовой модели, генератора состояний модели, механизма визуализации базовой модели и пользовательского интерфейса. Представлена концепция визуализации модели элементами другой визуальной модели на базе адаптированного шаблона MVC. Приведена реализация описанного паттерна на языке Java. Разработан ресурс информационной поддержки развития проекта.

Паттерн «Manipulator» разработан в соответствии с принципом модульной архитектуры: отдельные части архитектурного объекта получают возможность автономного существования, как с точки зрения архитектурной самодостаточности, так и с функциональной точки зрения. Структура состоит из пяти модулей: Tuttors (наставники или система обучения), Model (базовая модель и механизмы генерации), ModelStates (управление состояниями базовой модели), ModelViews (управление визуализацией базовой модели), ModelControllers (инкапсулирует методы модификации моделей).

Описаны методики применения шаблона на примерах реальных манипуляторов, в частности приведен пример визуализации базовой модели векторными объектами.

Литература

1. Шалыто А.А. Автоматное программирование, 2008. http://is.ifmo.ru
2. Казаков М.А., Корнеев Г.А., Шалыто А.А. Разработка логики визуализаторов алгоритмов на основе конечных автоматов // Телекоммуникации и информатизация образования. 2003.
3. Гамма Э., Хелм Р., Джонсон Р., Влиссидис Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб: Питер, 2001.

УДК-004.4'416

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ ДЛЯ ЛОКАЛЬНОЙ ОПТИМИЗАЦИИ ПРОГРАММНОГО 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.

Одним из 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 либо вручную (и может содержать лишь ограниченный набор достаточно простых правил), либо с помощью полного перебора наборов инструкций (что требует значительных затрат времени).

Целью данной работы является проверка эффективности применения генетических алгоритмов как замены полного перебора при поиске лучшего (в данном случае в смысле «работающего быстрее») набора команд.

Ход работы. Была разработана реализация эволюционного процесса, оперирующего наборами инструкций и при этом поддерживающего их корректность (возможность запуска) и эквивалентность исходному набору. Были созданы реализации, позволяющие производить оптимизацию байткода виртуальной машины *Java* и инструкций архитектуры *x86*. Полученное программное решение может быть в дальнейшем расширено для поддержания других целевых платформ.

В качестве эксперимента были проведены поиски более быстрых вариантов для нескольких наборов инструкций, взятых из различных прикладных программ.

Результат работы генетического процесса показал его высокую эффективность в деле оптимизации. Были найдены как стандартные решения (например, замена целочисленного умножения на два на сдвиг), так и нетривиальные оптимизации, для нахождения которых с помощью прочих методов потребовался бы глубокий анализ свойств вычисляемых выражений и детальное знание целевой архитектуры.

Генетические алгоритмы позволяют найти достаточно хорошее решение за намного меньшее число итераций, чем полный перебор. Например, для одной из рассмотренных последовательностей длиной в 11 инструкций, результат в большинстве случаев получается за 100–150 поколений, по 100 особей в каждом, тогда как всего возможно около 60 миллиардов наборов инструкций. Результат при этом оказывается лучше выдаваемого другими средствами на 3–15%.

В процессе работы была выявлена отличительная черта такой генетической оптимизации: сильная привязка к архитектуре машины, на которой производилось тестирование. Некоторые оптимизации, дающие выигрыш по сравнению с результатами работы других программных средств, могут быть объяснены только деталями работы процессора, на которых тестировалась их скорость работы. Таким образом, наиболее эффективным будет проверка производительности на целевой платформе (в тех случаях, когда это возможно).

УДК 004.4'242

ГЕНЕРАЦИЯ КЛЕТОЧНЫХ АВТОМАТОВ НА ОСНОВЕ ОБУЧАЮЩИХ ПРИМЕРОВ ПРИ ПОМОЩИ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

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 многих физических процессов, например: диффузия энергии, разнообразные химические реакции, рост кристаллов и т.д. Однако использование клеточных автоматов затрудняется тогда, когда физическая система известна, а описывающий ее клеточный автомат нет, или построение его обычными эвристическими методами затруднительно, так как автомат может иметь большое число состояний, переходов, условий и действий на переход.

Цель работы – устранить этот недостаток, используя генетическое программирование.

Целью настоящей работы является разработка алгоритма автоматической генерации клеточных автоматов с заранее неизвестными числом переходов и условий на них при помощи генетического программирования.

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;
- 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 состояния клеток и данные в них на каждом конкретном шаге расчетов. Разработанной системой был построен автомат, который является эквивалентным искомому.

УДК 004.4’242

ПРИМЕНЕНИЕ МЕТОДОВ РЕШЕНИЯ ЗАДАЧИ О ВЫПОЛНИМОСТИ БУЛЕВОЙ ФОРМУЛЫ ДЛЯ ПОСТРОЕНИЯ УПРАВЛЯЮЩИХ КОНЕЧНЫХ АВТОМАТОВ ПО СЦЕНАРИЯМ РАБОТЫ

В.И. Ульянцев
Научный руководитель – аспирант Ф.Н. Царев

Введение. При применении парадигмы автоматного программирования для реализации сущности со сложным поведением выделяется система управления и объект управления. На начальном этапе проектирования программы выделяются события, входные переменные и выходные воздействия. После этого проектирование программы может идти разными путями. Один из них состоит в написании сценария работы программы, по которому далее эвристически строится автомат. Пример построения автомата таким способом приведен в работе М.А. Мазина, В.Г. Парфенова, А.А. Шалыто «Разработка интерактивных приложений Macromedia Flash на базе автоматной технологии» (http://is.ifmo.ru/download/flash.pdf).

Цель работы. В указанной работе сначала был составлен неформальный сценарий в виде текста, далее он был формализован – был получен текст с указанием обозначений входный событий и выходных воздействий, соответствующих стадиям работы программы. После этого выполнялось построение автомата. Все три указанных этапа выполнялись вручную.

Первые два этапа, скорее всего, не могут быть автоматизированы и должны выполняться человеком. Целью настоящей работы является разработка метода автоматизированного построения управляющих конечных автоматов по сценариям работы.

Базовые положения исследования. Сценарием работы программы будем называть последовательность троек $<e, f, A>$, где e – входное событие; f – булева формула от входных переменных, задающая охранное условие; A – последовательность выходных воздействий.

На вход разрабатываемому алгоритму подается набор сценариев работы. Построение управляющего автомата осуществляется в пять этапов.

1. Построение дерева сценариев.
2. Построение графа совместимости вершин дерева сценариев.
3. Построение булевой КНФ-формулы, задающей требования к раскраске построенного графа и выражающей непротиворечивость системы переходов результирующего автомата.
4. Запуск сторонней программы, решающей задачу о выполнимости булевой КНФ-формулы.
5. Построение автомата по найденной выполняющей подстановке.

Промежуточные результаты. Доказана NP-полнота задачи построения управляющего конечного автомата по сценариям работы. Разработан и реализован алгоритм построения графа совместимости вершин дерева сценариев. Этот алгоритм основан на принципе динамического программирования на дереве. Реализован метод построения булевой КНФ-формулы и метод построения автомата по выполняющей подстановке.

Основной результат. Разработан метод автоматизированного построения управляющих конечных автоматов по сценариям работы. Этот метод основан на сведении указанной задачи к задаче о выполнимости булевой формулы. Работоспособность метода проверена на задаче построения автомата управления часами с будильником и на задаче построения автомата управления дверями лифта. На этих задачах соответствующие управляющие автоматы были построены корректно, а время работы алгоритма составляло меньше секунды на персональном компьютере с процессором Intel Core 2 Quad Q9400, что позволяет говорить о достаточно высокой производительности разработанного метода.

УДК 519.713

АВТОМАТИЧЕСКИЕ ДОКАЗАТЕЛЬСТВА АНАЛОГОВ ГИПОТЕЗЫ ЧЕРНИ-ПЭНА

С.Э. Вельдер
Научный руководитель – д.т.н., профессор А.А. Шалыто

В докладе рассматриваются некоторые задачи теории синхронизируемых автоматов. Проблемы, связанные с синхронизируемостью, встречаются в различных областях информационных технологий, таких как теория кодирования, робототехника, и др. Свойство синхронизируемости детерминированного конечного автомата в узком смысле означает существование *синхронизирующего слова* – последовательности символов (команд), переводящей все состояния автомата в одно и то же состояние. Т.е. синхронизирующее слово переводит множество всех состояний автомата в синглетон. Говорят, что слово *имеет дефект* k в заданном автомате из n состояний, если оно переводит множество всех его состояний (имеющее размер n) в множество размера $n – k$. Синхронизируемость автомата определяется существованием для него слова дефекта $n – 1$.

Одной из основных и наиболее сложных задач в этой теории является оценка минимальной длины (в худшем случае) синхронизирующего слова или слова заданного дефекта при условии, что такое слово существует. Длину минимального слова дефекта k в худшем случае обозначим через $L(k)$. Гипотеза Ж.-Э. Пэна (1978) гласит, что $L(k) = k^2$. Ее частный случай при $k = n – 1$ известен как гипотеза Я. Черни (1964). Функция L перечислима снизу путем перебора всех автоматов. Данный факт означает следующее. Рассмотрим двуместный предикат: через $P(k, l)$ будем обозначать высказывание «в любом

автомате, имеющем слово дефекта не менее k, найдется слово дефекта не менее k и длины не более l». Если $P(k, l)$ не выполняется для некоторых k и l, то это можно подтвердить предъявлением соответствующего автомата – он будет являться сертификатом, опровергающим $P(k, l)$. Возможность перебирать такие сертификаты – это и есть перечислимость снизу функции L. В случае же, когда $P(k, l)$ верно для некоторых k и l, этот факт является неочевидной теоремой и доказывается для каждых k и l по-своему. Известно, например, следующее:

1) $P(0, 0)$, тривиально;
2) $P(1, 1)$, тривиально;
3) $P(2, 4)$, нетрудно;
4) $P(3, 9)$, Ж.-Э. Пэн (1981);
5) $P(4, 16)$, Я. Кари (2001) – последний пример (опровергающий гипотезу Пэна) был получен полным перебором автоматов.

Результаты, излагаемые в докладе, состоят в построении алгоритмов, позволяющих автоматически генерировать доказательства утверждений такого вида (когда они верны), а также определять случаи, когда они неверны, не перебирая все автоматы. Наличие такого алгоритма, в частности, доказывает, что функция L перечислима и сверху (т. е. вычислима). В работе введены специальные комбинаторные структуры, которые имеют конечный размер и кодируют разбиения (бесконечного) множества всех автоматов на конечное число (бесконечных) классов эквивалентности. Эти структуры играют роль автоматически проверяемых сертификатов, подтверждающих истинность $P(k, l)$ для заданных k и l. Корректность алгоритма обусловлена наличием таких сертификатов.

УДК 004.823

АНАЛИЗ ПРИМЕНИМОСТИ ПРАВИЛ ВЫВОДА КОНТЕКСТНО-СВОБОДНЫХ ГРАММАТИК В ФРЕЙМОВЫХ МОДЕЛЯХ ЭКСПЕРТНЫХ СИСТЕМ

И.А. Вотинов, М.В. Григорьева
Научные руководители:
к.ф.-м.н., доцент Д.А. Зубок, к.т.н., доцент Г.А. Корнеев

Введение. Исследователи ЭС используют теории представления знаний из когнитологии. Фреймовые системы пришли в ЭС из теорий обработки информации человеком. Поскольку знание используется для достижения разумного поведения, фундаментальной целью дисциплины представления знаний является поиск таких способов представления, которые делают возможным процесс логического вывода, т.е. создание знания из знаний. При этом основным недостатком фреймовых систем является отсутствие специальных рекомендаций для формирования механизмов управления логическим выводом в условиях заданных объектом анализа.

Цель работы: проведение анализа применимости правил вывода контекстно-свободных грамматик в фреймовых моделях экспертных систем.

Базовые положения. Экспертные системы состоят из: базы знаний, механизма вывода, интерфейса взаимодействия с пользователем.

Фрейм – это модель абстрактного образа, минимально возможное описание сущности какого-либо объекта, явления, события, ситуации, процесса. Фрейм может быть представлен в виде списка свойств, таблицы, текста на натуральном языке, а также в нотации Бэкуса-Наура. Таким образом, фрейм может представлять собой как единицу базы знаний, так и

выполнять роль интерфейса взаимодействия с пользователем (если фрейм представить в виде текста на натуральном языке).

Представление фреймов в нотации Бэкуса-Наура позволяет связать фреймовую модель с теорией контекстно-свободных грамматик. Таким образом можно применять правила вывода контекстно-свободных грамматик для создания механизма управления выводом в фреймовой системе.

В результате, становится возможным построение фреймовой модели системы, которая включает в себя все три основных компонента экспертной системы.

Результаты. Математическая модель, связывающая фреймовую модель и контекстно-свободные грамматики и позволяющая применять правила вывода контекстно-свободных грамматик в фреймовых моделях экспертных систем.

УДК 004.4’242

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ К ПОИСКУ ВЫПОЛНИМЫХ ПУТЕЙ ДЛЯ ТЕСТИРОВАНИЯ АВТОМАТНЫХ ПРОГРАММ

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

Автоматная программа состоит из конечного автомата и набора объектов управления, с которыми взаимодействует модель. Наиболее распространенным способом проверки автоматных программ является Model Checking, однако проверка моделей позволяет верифицировать только автомат, но не всю систему в целом.

В данной работе предлагается использовать тестирование для проверки автоматных программ в целом. Тестирование является трудоемкой и ресурсоемкой задачей, для преодоления этих недостатков в работе предлагается подход к тестированию автоматных программ и способ автоматизировать процесс создания тестов при помощи использования генетических алгоритмов.

Генетический алгоритм – это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путем случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих биологическую эволюцию. При тестировании традиционных программ генетические алгоритмы эффективно используются для поиска тестовых значений и генерации кода тестов.

Постановка задачи. При тестировании моделей, построенных с использованием расширенных конечных автоматов, выделяется две подзадачи: выбор или генерация выполнимых путей (feasible paths) для тестирования и подбор входных значений для выполнения этих путей. Данная работа изучает проблему генерации выполнимых путей.

Пути для тестирования можно задавать вручную, выбирая наиболее интересные сценарии для проверки. Но для обнаружения ошибок в автоматной программе необходимо проверить все возможные переходы между состояниями и последовательность состояний системы. Таким образом, требуется получить набор путей, создав тесты по которым, разработчик получит полное покрытие тестами автоматной программы. Поиск набора тестов вручную для больших систем трудновыполним, поэтому необходим способ автоматизации решения поставленной задачи. Для этого предлагается использовать генетические алгоритмы.

Задачи, рассмотренные в данной работе:

– определить метрики покрытия автоматной программы тестами;

– разработать способ оценки трудности создания теста ПО для заданного пути;
– преложить метод автоматизации процесса создания набора тестов с заданным покрытием для автоматной программы.

Практические результаты. Разработан метод оценки выполнимости пути в автоматной программе – оценка того, насколько сложно будет подобрать набор внешних переменных, удовлетворяющих описанному пути. Определение набора внешних переменных необходимо для создания теста автоматной программы, проверяющего поведение программы на данном пути. Таким образом, для генерации успешного набора тестов необходимо подобрать набор выполнимых путей, покрывающих переходы и состояния автоматной программы.

Выполнимость пути оценивается путем анализа охранных условий на переходах и контрактов, записанных для переходов, и зависимостей между переходами. Охранные условия и требования контрактов разбиваются на простые элементы вида «Выражение1 Оператор Выражение2». Для каждого элемента учитываются следующие свойства:

– наличие контекстных переменных в Выражение1;
– наличие контекстных переменных в Выражение2;
– зависимость Выражение1 от внешних переменных;
– зависимость Выражение2 от внешних переменных;
– наличие параметров-констант.

Также учитывается влияние выходных действий, заданных на переходах, на состояние переменных системы:

– изменение контекстных переменных;
– влияние на глобальные переменные.

С учетом приведенных оценок предложено определение фитнесс-функции, оценивающей выполнимость выбранного пути. На основе полученной фитнесс-функции разработан метод генерации выполнимых путей для заданной автоматной программы с использованием генетического алгоритма.

УДК 004.852

ВЕРИФИКАЦИЯ АВТОМАТНЫХ ПРОГРАММ ПРИ ПОМОЩИ ВЕРИФИКАТОРА SPIN

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

Верификация программ общего вида (в части построения модели) практически не может быть автоматизирована. Для автоматных программ эта задача решаема.

Поэтому в работе ставится задача разработки метода автоматической верификации визуальных [1] автоматных программ [2]. Наиболее известным верификатором является верификатор *SPIN* [3], который является открытым и бесплатным. При верификации [4, 5] на его основе требования к программе записываются на языке линейной темпоральной логики (*LTL*) [6]. Инверсии каждой *LTL*-формулы может быть сопоставлен автомат *Бюхи* [7]. Собственно верификация состоит в том, что верификатор с целью построения контрпримера (если он имеется) должен «пересечь» модель *Крипке* [4] и автомат *Бюхи*. *М*одель *Крипке* автоматически строится верификатором по модели программы, записанной на языке *Promela*.

Для визуальных автоматных программ, во-первых, построение модели на указанном языке по графам переходов может быть автоматизировано, а, во-вторых, переход от контрпримера на модели к контрпримеру в терминах автоматов также может быть

автоматизирован.

Отличительной особенностью предлагаемого метода является возможность верифицировать параллельные программы.

Для поддержки метода было создано инструментальное средство Stater, которое позволяет разрабатывать автоматные программы, включая генерацию кода на разных языках программирования и верификацию. Для инструментального средства был разработан удобный интерфейс, подсвечивающий контрпример. Кроме того, Stater позволяет создавать автоматизированные классы [8], которые удобно встраиваются в уже существующие проекты.

Литература

1. Новиков Ф.А. Визуальное конструирование программ. http://is.ifmo.ru/works/visualcons/
2. Шалыто А.А. Технология автоматного программирования. http://is.ifmo.ru/works/tech_aut_prog/
3. SPIN home page. http://SPINroot.com
4. Кларк Э., Грамберг О., Пелед Д. Верификация моделей программ: Model Checking. М.: МЦНМО, 2002.
5. Лифшиц Ю. Верификация программ и темпоральные логики. Лекция №3 курса «Современные задачи теоретической информатики». http://download.yandex.ru/class/lifshits/lecture-note03.pdf
6. Linear temporal logic. http://en.wikipedia.org/wiki/Linear_temporal_logic
7. Büchi automaton. http://en.wikipedia.org/wiki/Büchi_automaton
8. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. СПб: Питер, 2009.

УДК 004.852

ВЕРИФИКАЦИЯ АВТОМАТНЫХ ПРОГРАММ ПРИ ПОМОЩИ ВЕРИФИКАТОРА UNIMOD.VERIFIER

Б.Р. Яминов

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

В результате постоянно увеличивающейся автоматизации жизни человека все более остро возникает необходимость заранее обнаруживать и устранять ошибки в программах, управляющих устройствами. Чем раньше ошибки находятся, тем меньше вреда они приносят, поэтому возникает интерес к задаче поиска ошибок еще на этапе моделирования управляющей программы.

Метод *Model Checking* [1] используется для верификации моделей программ: он позволяет проверять, что указанное свойство модели выполняется всегда, для любого сценария работы программы. Этим он отличается от традиционного тестирования: тесты проверяют работу программы лишь на ограниченном множестве сценариев. Однако сложность применения *Model Checking* связана с тем, что этот метод работает не с исходными кодами программы, а с моделью программы. Поэтому для верификации программы требуется вручную создать модель этой программы, распознаваемую верификатором. В результате корректность модели не гарантирует корректность исходной программы.

Автоматный подход [2] хорошо подходит для программирования управляющих систем, поскольку автоматы позволяют просто и наглядно представлять логику работы программы. Кроме того, программы, реализованные при помощи автоматного подхода, априори обладают моделью: автоматной системой. Таким образом, если научиться верифицировать автоматные системы, то для автоматных программ можно будет применять метод *Model Checking* полностью автоматически, без необходимости разрабатывать дополнительно модель

программы.

В работе предложен метод верификации автоматных систем, и этот метод реализован в верификаторе *UniMod.Verifier*. *UniMod* [3] – это инструментальное средство для разработки автоматных программ. Верификатор *UniMod.Verifier* реализован как дополнение к *UniMod*, что позволяет использовать одно инструментальное средство для разработки, верификации и запуска автоматных программ.

Отмечено преимущество разработанного метода верификации по сравнению с другими существующими методами, состоящее в том, что при верификации непосредственно используется интерпретатор автоматных систем инструментального средства *UniMod*. В результате этот метод проверяет корректность работы автоматной программы не абстрактно и отвлеченно от процесса работы автоматной системы, а конкретно применительно к исполнению автоматной системы инструментальным средством *UniMod*. Это позволяет избежать несоответствия между верифицируемой и исполняющейся автоматной системой, а следовательно, избежать ошибочной верификации.

Кроме того, для верификатора *UniMod.Verifier* был разработан удобный интерфейс, позволяющий запускать верификацию и получать ее результаты прямо из инструментального средства *UniMod*. При этом контрпример графически отображается на диаграммах автоматов.

Литература

1. Кларк Э., Грамберг О., Пелед Д. Верификация моделей программ: Model Checking. М.: МЦНМО, 2002.
2. Шалыто А.А., Туккель Н.И. SWITCH-технология – автоматный подход к созданию программного обеспечения «реактивных» систем // Программирование. – 2001. – № 5.
3. Гуров В.С., Мазин М.А., Шалыто А.А. UniMod – инструментальное средство для автоматного программирования // Научно-технический вестник СПбГУ ИТМО. Вып. 30. Фундаментальные и прикладные исследования информационных систем и технологий. СПбГУ ИТМО. – 2006. – С. 32–44.

УДК 519.688

ИСПОЛЬЗОВАНИЕ МОДИФИЦИРОВАННОЙ ПЛАТФОРМЫ СТЮАРТА ДЛЯ МОДЕЛИРОВАНИЯ ДИНАМИКИ ДВИЖУЩЕГОСЯ ОБЪЕКТА

И.В. Сорокин

Научный руководитель – ассистент В.М. Лесин

Параллельные манипуляторы на базе платформы Стюарта широко применяются в авиасимуляторах, развлекательных системах и медицине. Преимуществами данных манипуляторов является высокая точность позиционирования, высокая структурная жесткость и хорошие динамические характеристики [1]. Сложностями, возникающими при разработке таких манипуляторов, являются ограниченность рабочего пространства, а также наличие в нем особых точек [2–4], в которых эффектор получает дополнительные степени свободы.

Данная работа посвящена модификации платформы Стюарта использующей вместо ног переменной длины кривошипно-шатунный механизм. Задачи прямой и обратной кинематики для данной задачи имеют более сложное решение по сравнению с обычной платформой Стюарта. В частности, в отличие от обычной платформы, задача обратной кинематики для рассмотренной модификации имеет не единственное решение [5].

При использовании в авиасимуляторах очень важно уметь моделировать движениями платформой, имеющей ограниченную рабочую область, поведение некоторого объекта пространства, не имеющего ограничений на рабочую область.

В данной работе предлагается две модели такого поведения. Одна применима для объектов, для которых важно точно передавать их ориентацию в пространстве, другая для тех, чья ориентация в пространстве не играет существенной роли, а важно лишь её изменение.

Литература

1. Tsai L.W. Robot Analysis: the Mechanics of Serial and Parallel Manipulators. A Wiley-Interscience Publication, 1999.
2. Charters T., Enguica R., Freitas P. Detecting Singularities of Stewart Platforms // Mathematics-in-Industry Case Studies Journal. – V. 1. – 2009. – P. 66–80.
3. Peter Donelan. Kinematic Singularities of Robot Manipulators // Advances in Robot Manipulators. – 2010. – P. 401–416.
4. Qimi Jiang. Singularity-Free Workspace Analysis and Geometric Optimization of Parallel Mechanisms. Ph. D. Thesis, 2008.
5. James E. Dieudonne, Russell V. Parish, Richard E. Bardusch. An Actuator Extension Transformation for a Motion Simulator and an Inverse Transformation Applying Newton-Raphson's Method // NASA Technical Note, 1972.

УДК 004.031.42

ПОСТРОЕНИЕ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ С ИСПОЛЬЗОВАНИЕМ ВЛОЖЕННЫХ ТЕГОВ В КАЧЕСТВЕ АБСТРАКЦИИ НАД ПРОИЗВОЛЬНЫМ ВИДОМ ДАННЫХ

А.П. Чеботаев

Научный руководитель – д.т.н., профессор А.В. Бухановский

Постановка проблемы. В настоящее время объем информации, доступной человеку стал настолько велик, что даже используемые сервисы для поиска информации не всегда справляются с задачами людей. Для осуществления поиска объектов необходимо знать об их существовании. С ростом объема доступной информации знать обо всем становится затруднительно. Данную задачу решают рекомендательные системы, которые связывают между собой объекты и пользователей и ищут для пользователей объекты, неизвестные им, но вероятно их интересующие.

Среди них можно выделить два наиболее успешных класса: алгоритмы коллаборативной фильтрации и алгоритмы, основанные на анализе данных о поведении пользователя (data mining, machine learning, etc.).

Алгоритмы из первой группы обладают рядом проблем, наиболее существенными из которых являются проблема холодного старта, первого рейтинга, избирательности внимания и долгого времени работы при пересчете данных. Проблема первого рейтинга заключается в том, что невозможно или бессмысленно делать прогноз об объектах, которым ни один пользователь не поставил оценки. Избирательность внимания возникает из-за того, что плотность оценок неравномерно распределена, причем эта неравномерность усиливается со временем из-за того, что хорошо оцененные объекты чаще попадаются пользователям. Из-за малой плотности распределения данных по пользователям такие системы дают очень размытые рекомендации при своем первом запуске (холодный старт), а чем больше объектов в системе, тем больше информации требуется от каждого нового пользователя, чтобы выдать первую качественную рекомендацию. Причем, при больших размерах матрицы оценок пользователей для объектов, сравнение объектов и пересчет множеств похожих пользователей («соседей») занимает много времени.

Алгоритмы, основанные на анализе данных о поведении, всегда привязаны к специфике

области, к которой они применяются. Они редко переносимы на другие типы данных. Например, ключевые слова новостей не применимы к рекомендациям фильмов.

Для решения большинства из этих проблем, предлагается выделить из объектов единицу смысла и использовать ее в качестве нового представления данных, и новый алгоритм для рекомендаций на основе этого типа данных.

Цель работы. Разработать структуру данных и алгоритм поиска похожих объектов, который решит описанные выше проблемы.

Базовые положения исследования. Предлагается отказаться от понятия рейтингов или оценок, и заменить их семантическими связями между объектами «состоит из» (в смысле сущности объектов, а не физического устройства вещей). Каждую сущность в системе рекомендаций (пользователей, объекты и их характеристики) предлагается представлять как цельную единицу информации.

Каждую единицу информации предлагается представлять в виде уникального ключа (id), и множества взвешенных ссылок на другие единицы информации.

Например, единица информации, соответствующая книге Олдоса Хаксли «Дивный новый мир», будет содержать ссылки на «Олдос Хаксли», «Книга», «Фантастика» и «Антиутопия». В свою очередь, единица информации «Олдос Хаксли» будет ссылаться на «Писатель» и «Утопист». Сравнение объектов осуществляется путем сравнения доли весов общих ссылок от общего веса ссылок.

При взаимодействии пользователя с объектом (добавление объектов в свой профиль или их оценка) предлагается изменять веса ссылок на одинаковые единицы информации у пользователя и у объекта и добавлять новые ссылки, как пользователю, так и объекту. Другими словами, пополнять информацию о пользователе используя информацию об объекте и наоборот неявно.

Это позволяет решить проблему избыточного внимания и проблему долгих пересчетов, потому как для сравнения двух объектов не требуется обращения ко всем остальным.

Автоматический сбор единиц информации и неявный обмен взвешенными ссылками между объектами решает проблемы холодного старта и первого рейтинга.

Используя один предложенный формат данных для разных рекомендательных систем, можно объединять результаты их работы. Таким образом, профиль пользователя в одной системе позволит рекомендовать ему объекты другой рекомендательной системы. Это решает проблему специфики объектов.

Промежуточные результаты

- Предложена структура данных рекомендательной системы.
- Предложен алгоритм быстрого поиска похожих объектов.

Основной результат

1. Разработан прототип рекомендательной системы, работающей на основе единиц информации.
2. Система опробована на данных MovieLens и данных конкурса рекомендательных систем Netflix.
3. Произведено сравнение разработанного алгоритма с алгоритмом Slope One.

УДК 004.89

УСКОРЕНИЕ ЛОГИЧЕСКОГО ВЫВОДА В ПРОДУКЦИОННОЙ МОДЕЛИ ЗНАНИЙ С ПОМОЩЬЮ РЕЛЯЦИОННЫХ СУБД

Р.С. Катериненко
Научный руководитель – к.т.н., доцент И.А. Бессмертный

Вступление. В данной работе рассматриваются вопросы, связанные с построением продукционной модели знаний. На сегодняшний день существует множество средств формализации общих знаний о мире. Несмотря на это, повсеместное применение таких средств в интеллектуальных информационных системах сталкивается с существенной проблемой – быстрый рост трудоемкости логического вывода при росте размера базы знаний.

Целью работы является исследование ускорения логического вывода в подобных системах.

Базовые положения. Продукционная модель была выбрана в качестве основной модели ввиду большого потенциала для ускорения логического вывода.

Авторами разработан алгоритм логического вывода фактов в продукционной базе знаний. Ускорение достигается за счет применения операций реляционной алгебры, эффективная реализации которых имеет место в современных СУБД.

Алгоритм реализует две основные функции: загрузка фактов, продукций в базу знаний и поиск фактов по запросу пользователя с применением логического вывода (продукционного).

При загрузке фактов происходит и загрузка в таблице СУБД. При загрузке продукций происходит обновление внутренних структур данных алгоритма. К важнейшим внутренним структурам алгоритма относятся: таблица «предикат – продукция» и граф зависимостей продукций. Эти структуры служат для ограничения множества продукций, применяемых для логического вывода фактов при обработке запроса пользователя. Существует множество способов анализа множества продукций. Мы предлагаем анализировать зависимость продукций по входящим в них предикатам. Продукция $p1$ называется зависимой от продукции $p2$ по предикату $a(x)$, если $a(x)$ встречается в теле $p1$ в голове $p2$. Другими словами, с применением продукции $p2$ к множеству фактов могут быть получены новые факты, которые могут активизировать продукцию $p1$. Эта зависимость выражается в таблице «предикат – продукция». Информация в таблице обновляется при изменении, добавлении или удалении продукции.

Выше указанная таблица играет вспомогательную роль для быстрого построения графа зависимостей продукций. Узлами графа являются вершины-продукции и вершины-предикаты. Двум зависимым продукциям, согласно введенному выше свойству зависимости, в графе соответствует существование двух вершин-продукций соединенных дугами с вершиной-предикатом. Граф является ориентированным, поскольку отношение зависимости ориентировано. Граф является двудольным, поскольку всякие две продукции либо соединены через предикат, либо вообще не соединены.

Граф играет важную роль для оптимизации логического вывода. При получении запроса пользователя – найти факты, удовлетворяющие заданным предикатам, происходит поиск продукций, связанных с этим предикатом. В результате чего, из графа будет выделен подграф необходимых продукций. Этот подграф является деревом с вершиной-предикатом.

Дальнейшим этапом является логический вывод по фактам базы знаний с помощью найденного подмножества продукций. Логический вывод заключается в поиске фактов, удовлетворяющих условиям продукций. Срабатывание продукций приводит к появлению новых фактов (выведенных). Выведенные факты тоже, в свою очередь, могут, приводить к

срабатыванию продукций и получению выведенных фактов из выведенных и т.д. Для логического вывода в нашем подходе используется SQL. Каждой продукции сопоставляется SQL-выражения, выполнение которого эквивалентно срабатыванию продукции. Зависимость продукций приводит к образованию вложенных SQL-запросов. Конечное SQL-выражение, результатом которого является ответ на пользовательский запрос, генерируется по найденному выше дереву продукций. Дерево продукций транслируется в дерево SQL-операций. Для этого каждой вершине сопоставляется одна из операций – *select*, *union* или *join*. В результате получается дерево, похожее абстрактное синтаксическое дерево арифметического выражения (*AST*). При обходе дерева, операции упорядочиваются, и получается SQL-выражение, готовое для исполнения в СУБД.

Основные результаты. Данный подход позволяет использовать преимущества СУБД по автоматической оптимизации запросов, эффективному составлению планов выполнения запросов и оптимизациям в случае распределенной СУБД.

Возможность использовать комбинацию SQL-запроса и запроса с логическим выводом в одном запросе, дает возможность расширить возможность существующих приложений, работающих с СУБД функциями интеллектуальной базы знаний.

Прототипная реализация алгоритма дает ускорение по сравнению с Jess - реализацией алгоритма Rete на 20%.

УДК 004.8

СИСТЕМА ПОДДЕРЖКИ АЛГОРИТМОВ КОЛЛЕКТИВНОГО РАЗУМА

Д.О. Конончук
(Уральский государственный университет им. А.М. Горького)
Научный руководитель – к.ф.-м.н. Ю.С. Окуловский
(Уральский государственный университет им. А.М. Горького)

Алгоритмы коллективного разума (АКР) являются в настоящее время одной из самых динамично развивающихся отраслей алгоритмов искусственного интеллекта (ИИ).

Они основаны на так называемом «интеллекте толпы» – явлении, при котором система из множества слабо интеллектуальных, равноправных и децентрализованных сущностей проявляет свойства интеллектуальности.

Несмотря на большое количество исследований, связанных с АКР, для них до сих пор не существует общеизвестной системы поддержки, решающей то множество сервисных задач, которое возникает при программной эмуляции подобных алгоритмов.

В связи с этим, в каждой реализации того или иного АКР приходится заново решать проблемы распараллеливания вычислений, ввода и вывода данных из системы, ее отладки и многие другие. Это усложняет процесс разработки и делает практически невозможной интеграцию различных решений. Кроме того, из-за отсутствия единого фреймворка производительность различных алгоритмов сложнее сравнивать: время работы и потребление памяти разных решений будут в первую очередь зависеть не от принципиальных различий алгоритмов, а от особенностей их реализации.

В некоторых случаях в качестве фреймворка могут быть использованы т.н. системы агент-ориентированного моделирования. Но, к сожалению, эти системы не поддерживают многих свойств, являющихся существенными для АКР. Кроме того, среди них нет открытых решений на языке C#.

Таким образом, для облегчения реализации, интеграции и сравнения различных типов и вариаций АКР требуется создание универсальной системы поддержки алгоритмов данного типа. Эта система должна подходить для создания на ее базе эмуляций всех общеизвестных

АКР и, кроме того, быть удобной для использования, изучения и расширения (поскольку возможно применение ее также и для поддержки огромного количества систем, схожих с АКР).

В рамках моей работы был выполнен анализ и обобщение существующих на данный момент алгоритмов коллективного разума, их классификация. На основе выявленных общих свойств была построена модель произвольного АКР, с одной стороны, удобная для программной реализации, а с другой – позволяющая легко описывать АКР любого типа. Была разработана система поддержки, предоставляющая множество удобных инструментов для описания алгоритмов ИИ, удовлетворяющих ограничениям построенной модели.

Была создана последовательность прототипов и, затем, полноценная реализация системы поддержки. На каждой итерации разработки производились различные тестирования, результаты которых использовались для дальнейшего усовершенствования продукта. Система реализована на языке C# 4, под платформу .Net 4.0, совместимо с платформой Mono. В ходе разработки активно использовались технологии контрактного и параллельного программирования.

Также в работе рассмотрены как планируемые, так и уже реализованные расширения системы поддержки и лежащей в ее основе модели, позволяющие существенно расширить область ее применимости, в частности, для эмуляции игр.

УДК 519.688

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ РЕШЕНИЯ ЗАДАЧИ АНАЛИТИЧЕСКОГО ПРЕДСТАВЛЕНИЯ КОЭФФИЦИЕНТА ПРОПУСКАНИЯ АТМОСФЕРЫ

М.И. Моисеева
Научный руководитель – д.т.н., профессор А.В. Демин

С целью повышения достоверности измерений, проводимых при мониторинге объектов в оптическом диапазоне спектра, необходимо решить задачу оценки величины коэффициента пропускания атмосферы. Данный коэффициент определяется поглощением составляющих атмосферу газов и рассеянием на ее частицах, молекулах и аэрозолях. Для учета коэффициента пропускания атмосферы при дистанционном зондировании объектов обычно используют метод расчета, требующий дополнительных измерений температуры воздуха, метеорологической дальности видимости, относительной влажности. Однако представляется возможным сократить количество измеряемых величин, построив и исследовав аналитическую модель коэффициента пропускания атмосферы на основе экспериментальных данных о значении этого коэффициента при различных условиях.

Аналитическая модель для оценки коэффициента пропускания атмосферы может быть представлена в виде композиции двух компонент. Первая компонента определяет спектральное пропускание атмосферы. Вторая – необходима для учета текущих географических и климатических внешних факторов.

Целью работы является разработка программного комплекса для получения спектральной компоненты аналитической модели на основе исходных экспериментальных данных для отдельных окон прозрачности атмосферы.

Ранее в ходе исследований был разработан алгоритм аналитического представления коэффициента пропускания атмосферы для двух спектральных диапазонов в инфракрасной области оптического спектра – [3; 5,2] и [8; 14] мкм. Данный алгоритм был апробирован с использованием ряда экспериментальных данных, известных из литературных источников. Были получены отдельные аналитические зависимости для различных климатических и географических условий. Однако чтобы вывести единую формулу для оценки величины

коэффициента пропускания атмосферы при любых вероятных внешних условиях без проведения дополнительных измерений, необходимо провести дополнительные вычислительные эксперименты.

В программном комплексе должен быть реализован разработанный ранее алгоритм аналитического представления коэффициента пропускания атмосферы, а также предусмотрена возможность визуализации исходных данных и получаемых расчетных значений. Разрабатываемый комплекс должен обладать модульной структурой с возможностью его дальнейшего развития для применения в других спектральных диапазонах.

В настоящее время спроектирована структура программного комплекса. Часть модулей комплекса находится в стадии отладки и тестирования, часть – на этапе реализации.

УДК 62-231.1

ПРОГРАММА ПОСТРОЕНИЯ КИНЕМАТИЧЕСКИХ СХЕМ НА ОСНОВЕ РАЗРАБОТАННОЙ КЛАССИФИКАЦИИ

А.С. Суворов
Научный руководитель – ст. преподаватель С.С. Гвоздев

Кинематическая схема входит в комплект чертежно-конструкторской документации и служит для установления и изучения связей между кинематическими элементами устройства или прибора. Для построения кинематических схем применяются условные графические обозначения – графические обозначения кинематических элементов, которые приведены в ГОСТ 2.770 [1]. Правила построения этих схем даны в ГОСТ 2.703 [2].

В учебном процессе эти схемы используются в нескольких предметах.

В соответствии с этими ГОСТами имеющиеся обозначения кинематических элементов разнообразны и сгруппированы в некоторые пронумерованные классы, такие, например, как серии подшипников или разнообразные типы передач. Недостатком приведенных групп кинематических элементов является отсутствие их четкого упорядочивания. Кроме того обозначения некоторых кинематических элементов состоит из нескольких более простых кинематических элементов.

В чертежно-графической среде «КОМПАС», которая позволяет создавать чертежи, имеется библиотека кинематических элементов, состоящая из обозначений входящих в ГОСТ 2.770 [1]. Однако, в среде «КОМПАС» нет графических обозначений, например, соединительных частей звена, неподвижных соединений с валом и пружин: спиральной и листовой.

В приборостроении используются обозначения кинематических элементов ГОСТа 2.770, а также такие обозначения, которые не указанны в ГОСТе 2.770 [1]. Например, такими кинематическими элементами являются: датчики, шкалы, цанги, фиксаторы.

Общая структура классификации кинематических элементов может иметь следующий вид:

1. по виду соединения:
   - неподвижное;
   - подвижное;
   - с временной фиксацией взаимного положения;
2. по виду движения:
   - вращательное;
   - поступательное;
   - передача движения;
   - преобразование движения;

3. по функции представления информации:
   – шкальные устройства;
   – датчики;
1. по виду привода:
   – ручной привод;
   – автоматизированный привод.

Приведенный принцип классификации может быть полезен, как в учебном процессе для изучения кинематических элементов устройств приборов, так и при ознакомлении с принципами работы устройств. Приведенная классификация позволят быстро сориентироваться при выборе необходимого обозначения кинематического элемента.

На основе разработанной классификации была создана компьютерная база данных кинематических обозначений элементов, которая меньше по объему и более структурирована, чем структуры, приведенные в ГОСТе и чертежно-графической среде «КОМПАС».

Также была разработана программа построения кинематических схем, в которой имеется база данных классифицированных кинематических элементов, с помощью которой пользователь ЭВМ осуществляет построение кинематической схемы. Программа позволяет производить соединения отдельных элементов и строить кинематические схемы, как на плоскости, так и в аксонометрии независимо от их ориентации [3].

С использованием разработанной программы можно быстро выбирать необходимое обозначение кинематического элемента, добавить его на область рисования и путем добавления дополнительных элементов создавать кинематические схемы. При необходимости можно сохранить результат построения в файл. Сохраненный файл имеет расширение .jpg и предназначен нести графическую информацию об изображении построенной кинематической схемы.

В программе имеются отображения основных кинематических элементов в аксонометрической проекции. Такое отображение позволяет представить выбранный элемент наглядней и понятней, чем по сравнению с простым двухмерным отображением. Также в программе появилась возможность задавать положение кинематического элемента произвольным образом на области рисования в зависимости от задания углов поворота по трем пространственным осям.

Простой интерфейс программы и наглядное представление обозначений кинематических элементов делают возможным использование такой программы в учебном процессе для ознакомления и изучения кинематических обозначений и построения кинематических схем.

Литература

1. ГОСТ 2.770–68. Обозначения условные графические в схемах. Элементы кинематики [Текст]. – Введ. 01–01–71. – М.: Изд-во стандартов, 1998. – 13 с.
2. ГОСТ 2.703–68. Правила выполнения кинематических схем. – Введ. 01–01–71. – М.: Изд-во стандартов, 1987. – 12 с.
3. Сборник трудов конференции «Оптика и образование-2010». – СПб: СПбГУ ИТМО, 2010. – 114 с.

УДК 681.3.06

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ WEB-ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ АВТОМАТНОГО ПРОГРАММИРОВАНИЯ НА ПРИМЕРЕ REST- И MVC-АРХИТЕКТУР

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

При разработке сложных программных систем очень трудно создать код непосредственно на основе интуитивных представлений. Поэтому перед этапом программирования системы должен быть дополнительный – моделирование системы.

В процессе создания программного обеспечения часто возникает необходимость создания систем со сложным поведением. При реализации таких систем целесообразно применять автоматное программирование называемое также «программирование от состояний» или «программирование с явным выделением состояний». При совместном использовании объектной и автоматной парадигм программирования этот подход был назван «объектно-автоматным программированием». Данный метод разработки позволяет устранить такие сущности, которые возникают при традиционной программной реализации, как, например, избыточные переменные, называемые флагами, которым не соответствуют никаким элементам предметной области.

Объектно-автоматное программирование основано на объектной и автоматной парадигмах программирования. Оно совмещает в себе их основные преимущества, такие как гибкость, расширяемость и наличие мощного механизма описания сложного поведения, основанного на конечных автоматах. В работе показано, что наследование и вложение состояний, применяемое в объектно-автоматном программировании, позволяет значительно сократить требуемое число переходов для описания сложного автомата.

Бурное развитие сети Internet, наблюдающееся в последние десятилетия, привело к активному использованию средств и протоколов, использующихся в этой сети, в качестве базовой инфраструктуры при создании современных информационных систем. В первую очередь, речь идет об использовании WEB-интерфейсов для взаимодействия системы с пользователем, а так же построению систем на основе WEB-служб или WEB-сервисов, которые являются единицей модульности сервисно-ориентированной архитектуры приложения. Достоинствами WEB-служб являются: независимость от платформы, наличие и использование открытых стандартов и протоколов, использование HTTP протокола для передачи данных.

Сейчас WEB-приложения являются значимой частью представления и обработки информации и их, несомненно, можно отнести к классу систем со сложным поведением, так как на одно входное воздействие могут совершить несколько действий. Таким образом, для их реализации целесообразно применять объектно-автоматное программирование. Это позволяет повысить качество разработки WEB-приложений, увеличить отказоустойчивость и автоматически обеспечить документирование.

Целью работы является проектирование и разработка WEB-приложения с использованием объектно-автоматного программирования. Для примера были рассмотрены WEB-приложения на MVC (Model, View, Controller) и REST (Representational State Transfer) архитектурах.

В данной работе было показано, что использование наследования и вложения автоматных классов позволяет уменьшить необходимое количество переходов для реализации автоматных программ, а применение объектно-автоматного подхода для реализации WEB-приложений обеспечивает:

- удобное визуальное оформление на этапе разработки;
- эффективную работу со сложными иерархическими структурами;
- сохранение истории;
- сохранение контекста выполнения.

УДК 004.93

МЕТОД ПРИНЯТИЯ РЕШЕНИЯ О ГРАНИЦАХ БЫСТРОПРОТЕКАЮЩИХ ПРОЦЕССОВ ФИЗИКИ ПЛАЗМЫ НА ОСНОВЕ ИСКУССТВЕННОЙ НЕЙРОННОЙ СЕТИ

С.А. Фёдоров
(Санкт-Петербургский государственный политехнический университет)
Научный руководитель – к.т.н., доцент В.В. Амосов
(Санкт-Петербургский государственный политехнический университет)

В работе рассмотрены существующие методы определения движения и границ быстропротекающих процессов при высокоскоростной фотографии, определены особенности границ быстропротекающих процессов физики плазмы и предложен метод принятия решения о границах быстропротекающих процессов физики плазмы на основе нейронной сети.

Создание высокоскоростных устройств детектирования, хранения и передачи оптической информации, поступающей в результате различных физических явлений (взрывы, вспышки, разрушения, баллистика и т.п.), позволяет разрабатывать новые методики исследования этих процессов. Среди большого числа возможных применений высокоскоростных видеокамер наиболее сложные задачи возникают в исследованиях физики плазмы, что связано со следующими условиями и требованиями:

- работа в условиях жестких электромагнитных помех;
- большие перепады яркости во времени и в различных участках детектируемого изображения;
- необходимость передачи данных большому числу пользователей на удаленные центры мониторинга и обработки данных;
- короткое время существования плазменного разряда (от неск. мс до неск. с);
- объем отснятой за это время информации – 0,5–100 Гб;
- процент несущих полезную информацию кадров – 1–10% (рис. 1).

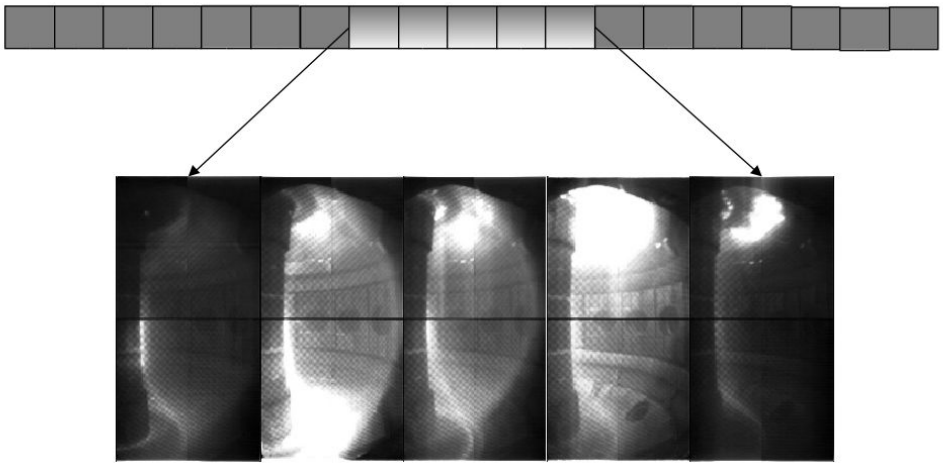

Рис. 1. Передаваемые кадры и кадры с полезной информацией

Анализ рынка выпускаемых к настоящему времени высокоскоростных видеокамер показал, что, обеспечивая в принципе необходимую скорость регистрации данных, они не удовлетворяют совокупности перечисленных выше требований.

Основной задачей, решаемой для удовлетворения этих требований является разработка метода принятия решения о границах быстропротекающих процессов физики плазмы, который позволит автоматизированной системе научных исследований (АСНИ) по

оптической диагностике обрабатывать информацию и передавать на удаленные центры только полезные кадры.

Предлагаемый метод основан на рассмотрении разности кадров – отличающихся пикселей. Для такой разности строится 4 укрупненные сетки следующего, второго уровня (элемент новой сетки содержит 4 пикселя прежней сетки). На укрупненных сетках второго уровня определяют число различающихся элементов (рис. 2).

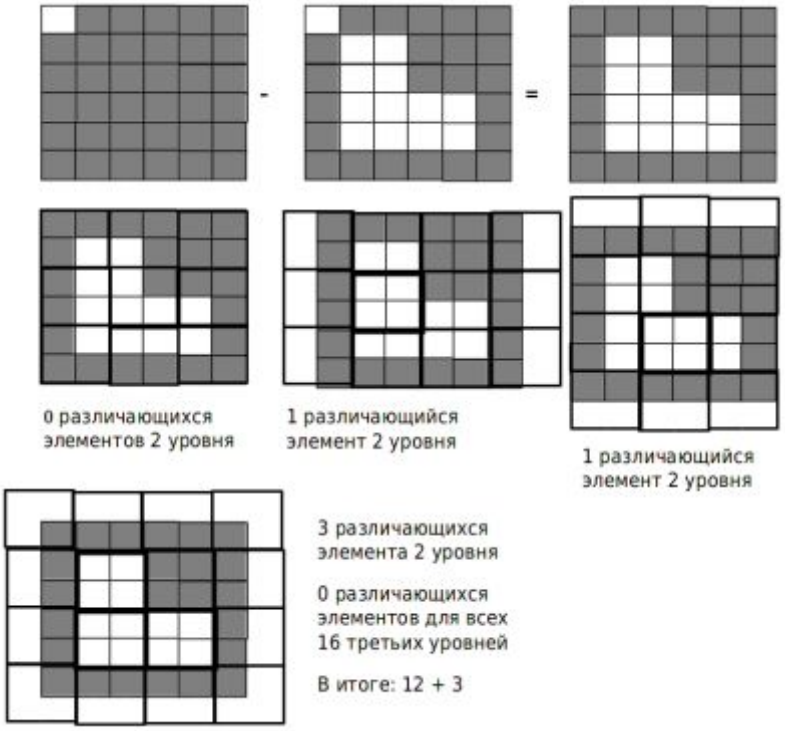

Рис. 2. Построение укрупненных сеток

Этот метод описан с помощью искусственной нейронной сети, каждый слой которой представляет собой очередной уровень сеток (рис. 3).

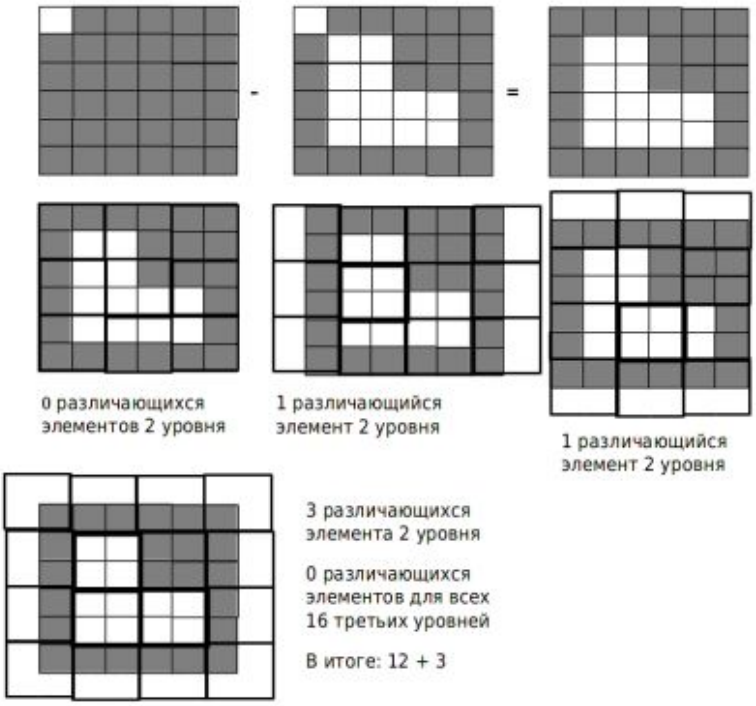

Рис. 3. Искусственная нейронная сеть для метода принятия решения

Число слоев сети меняется при обучении. Это позволяет предъявлять самые жесткие требования к отличию кадров с началом/окончанием процесса физики плазмы от кадров с помехами.

Функционалы и математические выкладки в тезисах для наглядности опущены. Проводится апробация этого метода и уже получены первые результаты. Этот метод планируется внедрить в высокоскоростные камеры ФТИ им. А.Ф. Иоффе РАН.