ISSN 1818–1015 (Print)
ISSN 2313–5417 (Online)

Министерство образования и науки Российской Федерации
Ярославский государственный университет им. П. Г. Демидова

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ

Том 23 № 6(66) 2016

Основан в 1999 году
Выходит 6 раз в год

Адрес редакции: ЯрГУ, ул. Советская, 14, г. Ярославль, 150003, Россия
Website: http://mais-journal.ru, e-mail: mais@uniyar.ac.ru; телефон (4852) 79-77-73

Научные статьи в журнал принимаются по электронной почте. Статьи должны содержать УДК, аннотации на русском и английском языках и сопровождаться набором текста в редакторе LaTeX. Плата с аспирантов за публикацию рукописей не взимается.

12+

СОДЕРЖАНИЕ

Моделирование и анализ информационных систем. Т. 23, №6. 2016

Свидетельство о регистрации СМИ ПИ № ФС 77 – 66186 от 20.06.2016 выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций. Учредитель – Федеральное государственное бюджетное образовательное учреждение высшего образования "Ярославский государственный университет им. П. Г. Демидова". Подписной индекс – 31907 в Объединенном каталоге "Пресса России". Редактор, корректор А.А. Аладьева. Редактор перевода Э.И. Соколова. Подписано в печать 12.12.2016. Дата выхода в свет 30.12.2016. Формат 60x84$^1/_8$. Усл. печ. л. 22,3. Уч.-изд. л. 20,0. Объем 193 с. Тираж 50 экз. Свободная цена. Заказ 082/016. Адрес типографии: ул. Советская, 14, оф. 109, г. Ярославль, 150003 Россия. Адрес издателя: Ярославский государственный университет им. П. Г. Демидова, ул. Советская, 14, г. Ярославль, 150003 Россия.

ISSN 1818–1015 (Print)
ISSN 2313–5417 (Online)

P.G. Demidov Yaroslavl State University

MODELING AND ANALYSIS OF INFORMATION SYSTEMS

Volume 23 No 6(66) 2016

Founded in 1999
6 issues per year

Editorial Office Address: P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl 150003, Russia
Website: http://mais-journal.ru, e-mail: mais@uniyar.ac.ru

Contents

Modeling and Analysis of Information Systems. Vol. 23, No 6. 2016

От редактора специального выпуска

В. А. Захаров

В этом выпуске журнала представлены статьи, подготовленные на основе научных докладов, которые были сделаны на седьмом международном семинаре «Семантика, спецификация и верификация программ: теория и приложения» (Seventh Workshop on Program Semantics, Specification and Verification: Theory and Applications, PSSV–2016). Этот семинар проводился 14–15 июня 2016 г. в Санкт-Петербурге, в Санкт-Петербургском отделении Математического института имени В.А. Стеклова в рамках 9-го Международного симпозиума по компьютерным наукам в России (9th International Computer Science Symposium in Russia).

Программа семинара PSSV–2016 включала 12 регулярных докладов и 4 лекции приглашенных докладчиков. В выступлениях участников семинара были представлены результаты исследований широкого круга задач в области математического моделирования, анализа и верификации программных систем. Основное внимание было уделено методам дедуктивной проверки правильности программ, методам верификации моделей программ, формальным подходам к тестированию, а также ряду вопросов построения и анализа формальных моделей информационных систем. Данный выпуск журнала включает 7 статей участников семинара PSSV–2016.

С.Н. Баранов и В.В. Никифоров описывают подход к проверке выполнимости многозадачных приложений реального времени в различных сочетаниях дисциплины планирования и протокола доступа к разделяемым общим информационным ресурсам при исполнении приложений на многоядерных вычислительных платформах. Предложенный подход позволяет находить и выбирать оптимальное сочетание дисциплины планирования и протокола доступа для многозадачного приложения с заданным профилем.

Статья Н.В. Визовитина, В.А. Непомнящего, А.А. Стененко продолжает цикл исследований по применению математического аппарата раскрашенных сетей Петри (РСП) для анализа и верификации моделей Use Case Maps (UCM). Приводятся описания алгоритма трансляции UCM в РСП, а также алгоритма конвертации РСП в описание системы взаимодействующих процессов на языке Promela для системы верификации моделей программ SPIN.

В статье Н.О. Гараниной и Е.А. Сидоровой описан метод верификации для семейств распределенных систем, которые порождаются контекстно-зависимой сетевой грамматикой специального вида. Свойства порождаемых систем специфицируются с помощью универсальной логики ветвящегося времени с конечными детерминированными автоматами в качестве атомарных формул. Авторы показывают, что предложенный метод верификации можно применять для проверки некоторых свойств мультиагентных систем разрешения конфликтов, в частности систем разрешения неоднозначностей при пополнении онтологий.

В статье М.Л. Громова и Н.В. Шабалдиной рассмотрена задача построения каскадной параллельной композиции временных автоматов. Авторы статьи предлагают оригинальную процедуру построения такой композиции, которая, в отличие от ранее известных методов, не требует выполнения затратной процедуры детерминизации автоматов. В качестве инструмента для построения композиции используется программно-инструментальное средство BALM-II.

Расширенные автоматы активно используются при построении тестов для программного обеспечения на основе формальных моделей. В статье А.Д. Ермакова и Н.В. Евтушенко для построения тестовых последовательностей предлагается использовать шаблонную реализацию расширенного автомата в языке Java и опираться на предложенный авторами метод построения множества тестовых последовательностей, обнаруживающих

функциональные ошибки в шаблонной реализации расширенного автомата. Показано, что исходный тест, расширенный такими различающими последовательностями, обнаруживает значительно больше функциональных ошибок в программных реализациях системы, для которой расширенный автомат используется в качестве спецификаций.

В статье В.А. Захарова и Г.Г. Темербековой конечные автоматы-преобразователи использовались для моделирования последовательных реагирующих программ. Для этой модели вычислений была исследована задача минимизации программ и установлены достаточные условия, при которых она имеет эффективное решение.

Статья В.М. Ицыксона посвящена вопросам спецификации структуры и поведения программных библиотек. Автор статьи формулирует требования к создаваемому формализму, который мог бы позволить специфицировать все свойства библиотек, необходимые для автоматизации нескольких классов задач: обнаружение дефектов в программном обеспечении, миграция приложений в новое окружение, генерация программной документации.

Моделирование и анализ информационных систем. Т. 23, №6 (2016), с. 673–687
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 673–687

DOI: 10.18255/1818-1015-2016-6-673-687

УДК 004.04

Имитационное моделирование для анализа выполнимости приложений реального времени

Баранов С. Н.[1], **Никифоров В.В.**

получена 15 марта 2016

Аннотация. Описывается развиваемый авторами подход к проверке выполнимости многозадачных приложений реального времени в различных сочетаниях дисциплины планирования и протокола доступа к разделяемым общим информационным ресурсам при исполнении данного приложения на многоядерной вычислительной платформе. Структура приложения задается в виде простого формализованного профиля, состоящего из сегментов трех видов, и описывающего доступ задач приложения к разделяемым информационным ресурсам; для каждого сегмента дается оценка необходимого ему объема вычислительного ресурса процессора. В основе данного подхода лежит введенное авторами понятие плотности программного приложения, которое характеризует потенциальную эффективность использования вычислительного ресурса приложением с определенным профилем. Значение эффективности определяется путем оценки выполнимости приложения с заданным профилем в зависимости от производительности процессора. Практическим инструментом для такой оценки служит разработанная авторами программа имитационного моделирования, обеспечивающая более точные, по сравнению с известными аналитическими методами, оценки. Приводится архитектура этого инструмента и общие сведения по его двум разнородным реализациям, а также представленные графиками результаты проведенных с их помощью экспериментов на ряде эталонных примеров, включая конфигурации Лю-Лейланда многозадачного приложения реального времени, вместе с их анализом и объяснением. Предложенный подход позволяет находить и выбирать оптимальное сочетание дисциплины планирования и протокола доступа для многозадачного приложения с заданным профилем.

Ключевые слова: имитационное моделирование, реальное время, плотность приложений, выполнимость приложений

Для цитирования: Баранов С. Н., Никифоров В.В., "Имитационное моделирование для анализа выполнимости приложений реального времени", *Моделирование и анализ информационных систем*, **23**:6 (2016), 673–687.

Об авторах:
Баранов Сергей Николаевич, orcid.org/0000-0001-6692-8432, д-р. физ.-мат. наук, проф.,
СПИИРАН, университет ИТМО,
Кронверкский пр., 49, г. Санкт-Петербург, 197101 Россия, e-mail: SNBaranov@gmail.com

Никифоров Виктор Викентьевич, orcid.org/0002-0007-1896-0876, д-р. тех. наук, проф.,
СПИИРАН, 14-я линия 39, г. Санкт-Петербург, 199178 Россия, e-mail: nik@iias.spb.su

Благодарности:
[1]Работа выполнена при государственной финансовой поддержке ведущих университетов Российской Федерации (субсидия 074-U01).

Введение

Программные приложения для систем реального времени (СРВ) обычно строятся как совокупности *задач* с их *приоритетами*, каждая из которых считается последовательной программой, замкнутой в себе по передачам управления. Во время своего исполнения такие задачи могут разделять *общие системные ресурсы*, как *исполнительные* – процессор или процессорные ядра в случае многоядерной платформы, так и *информационные* – глобальные массивы данных, интерфейсные регистры периферийных устройств, элементы человеко-машинного интерфейса и т.п. Доступ к исполнительным ресурсам управляется применяемой *дисциплиной планирования*, а доступ к информационным ресурсам – выбранным *протоколом доступа*.

Каждая задача τ_i приложения СРВ характеризуется своим *периодом* T_i, *предельным сроком* ее выполнения D_i и *абсолютным весом* W_i. Период и предельный срок задаются в абсолютных единицах времени (например, миллисекундах) и обычно рассматриваются как «внешние ограничения» данного приложения, тогда как его абсолютный вес считается «внутренним ограничением» – он характеризует «объем вычислительной работы» (и, таким образом, количество вычислительного ресурса), необходимой данной задаче для выполнения своей функции, и задается в виде числа эталонных машинных операций в данной реализации данной задачи. При заданной *производительности процессора* P в виде числа тех же эталонных операций в единицу времени абсолютный вес W_i задачи τ_i может быть преобразован в ее *относительный вес* C_i, выраженный в единицах времени:

$$C_i = \frac{W_i}{P}. \tag{1}$$

Поведение приложения состоит в параллельном исполнении ряда экземпляров $\tau_i^{(j)}$ его задач τ_i, называемых *заданиями*, которые порождаются с периодичностью T_i. Поскольку задания $\tau_i^{(j)}$ конкурируют между собой за общие системные ресурсы, то исполнение любого из них может быть приостановлено и затем возобновлено через какой-то промежуток времени. *Временем отклика* R_i задачи τ_i является максимальный из интервалов времени между моментами порождения и завершения заданий $\tau_i^{(j)}$, которые называются *интервалами существования* этих заданий.

Ключевым требованием к программному приложению СРВ является его *выполнимость*, формулируемая как $\forall i\ (D_i \geq R_i)$ для всех допустимых сценариев взаимодействия данного приложения с внешней средой. Выполнимость приложения может быть установлена путем аналитической оценки времени отклика для каждой задачи приложения или путем имитационного моделирования поведения данного приложения при заданном сценарии его взаимодействия с внешней средой с помощью соответствующего программного инструмента.

Для приложений СРВ, предназначенных к исполнению на одноядерном процессоре, точные аналитические оценки их выполнимости существуют еще с начала 1970-х годов [1–3], однако для многоядерных процессоров такие оценки до сих пор не известны, а предлагаемые грубые методы дают пессимистические оценки, по сравнению с реальным поведением таких систем [4, 5]. По этой причине возникает необходимость в программном инструменте имитационного моделирования для получения оценок выполнимости приложений СРВ, предназначенных к исполнению

на многоядерных платформах при различных сочетаниях дисциплин планирования и протоколов доступа, более точных, чем оценки, какие дают известные аналитические методы. В данной работе описывается архитектура такого инструментального средства [6] и результаты ряда экспериментов, выполненных с его помощью.

Данная работа дополняет результаты исследований [7]. В разделах 1 и 2 приводятся введенные в [7] понятия и обозначения, необходимые для дальнейшего изложения (плотность приложения, общая схема работы и архитектура программного инструментального комплекса). Отсутствующее в [7] описание понятия профиля многозадачного приложения и способа его представления введено в разделе 3, что позволило в разделе 4 дополнить результаты проведенных экспериментов точным указанием параметров исследуемых примеров приложений и выводами по особенностям полученных результатов измерений.

1. Плотность приложения

Производной характеристикой поведения задачи τ_i при заданной производительности P многоядерного процессора является его *полезная нагрузка*:

$$u_i = \frac{C_i}{T_i}, \tag{2}$$

вычисляемая как доля в периоде задачи, которая уходит на собственно вычислительную работу для данной задачи; таким образом, *общая полезная нагрузка* U многозадачного приложения из n задач может быть вычислена как

$$U = \frac{1}{k} \times \sum_{i=1}^{n} u_i, \tag{3}$$

где k – количество ядер в процессоре на данной платформе, каждое с производительностью P операций в единицу времени. Величина $1 - U$ задает долю процессорного времени, которая не используется данным приложением (процессор либо простаивает, либо занят вычислениями, не связанными с работой данной СРВ). Увеличение производительности процессора ведет к увеличению доли его простоя для данного программного приложения СРВ.

Рассмотрим вспомогательную характеристику задачи – ее *жесткость*:

$$H_i = \frac{T_i}{D_i}. \tag{4}$$

Если $H_i \geq 1$, то интервалы существования двух последовательных экземпляров одной и той же задачи τ_i – заданий $\tau_i^{(j)}$ и $\tau_i^{(j+1)}$ – не пересекаются. Противоположное условие $H_i < 1$ означает, что интервалы существования таких заданий могут пересекаться. Если все задачи приложения имеют одну и ту же жесткость H, то эта величина H называется жесткостью всего приложения. Часто более удобным оказывается рассматривать обратную к жесткости величину H^{-1}.

В работе [8] понятие *плотности приложения* определяется как максимальное значение его общей полезной нагрузки: $Dens = \max_P\{U(P)\}$ для всех значений P

производительности процессора, при которых данное приложение выполнимо. Очевидно, что для любого конкретного приложения найдется столь малое число ϵ, что данное приложение выполнимо при $P > \epsilon^{-1}$ и не выполнимо при $P < \epsilon$. Также очевидно, что если приложение выполнимо при значениях производительности P_1 и P_2, где $P_1 < P_2$, то оно выполнимо и для любого $P \in [P_1, P_2]$. Таким образом, существует значение плотности $Dens$, соответствующее минимальной производительности P_0, при которой данное приложение все еще выполнимо: для всех $P < P_0$ приложение не выполнимо, а для всех $P \geq P_0$ выполнимо.

Приведенное выше рассуждение приводит к экономичному алгоритму «поимки льва в пустыне» для вычисления этого значения $Dens$. Начиная с интервала $[a, b]$ для значений производительности, где $a = 0$ и $b = P_{max}$, причем P_{max} настолько велико, что выполнимость приложения при такой производительности обеспечивается, выполняется имитационное моделирование поведения приложения с производительностью процессора $P = (b-a)/2$. Следующим рассматриваемым интервалом будет либо $[a, P]$, если сеанс имитационного моделирования подтверждает выполнимость приложения, либо $[P, b]$ в противном случае. Цикл завершается на интервале $[P-\epsilon, P+\epsilon]$, где P – результирующее значение этой минимальной производительности с точностью до заданного значения $\epsilon > 0$. Плотность приложения, определяемая внешними факторами и структурными характеристиками приложения, а также выбором сочетания дисциплины планирования с протоколом доступа к разделяемым информационным ресурсам, может использоваться как объективный критерий экономичности данного сочетания при заданных характеристиках приложения.

Целью данного исследования было нахождение и анализ зависимостей между жесткостью приложения и его плотностью для различных сочетаний дисциплин планирования и протоколов доступа. Для этой цели в рамках единой программной архитектуры были параллельно разработаны два разнородных и независимых прототипа инструментального средства для оценки выполнимости приложений CPB: один на языке C# [9] объемом 1,4 KLOC, другой объемом 0,9 KLOC на языке Форт [10] в стандарте Forth 2012 [11], одновременно с набором эталонных приложений. Результаты имитационного моделирования поведения указанных эталонных приложений на этих двух инструментах различались менее чем на 0,1%, что является сильным аргументом в пользу их достоверности.

2. Проверка выполнимости

Общая схема работы обоих симуляторов для определения выполнимости приложений представлена на рис. 1. Работа начинается с инициализации, которая состоит в выборе желаемого сочетания дисциплины планирования и протокола доступа, установки режима наследования заданиями приоритетов, задания соответствующих констант, считывания файла с описанием структуры приложения и формирования соответствующих этому внутренних объектов – ресурсов и задач. Затем формируется начальный список *EventList* системных событий, состоящий из активизации всех задач в моменты системного времени, определяемые их фазовыми сдвигами относительно начала отсчета системного времени. Счетчики максимального време-

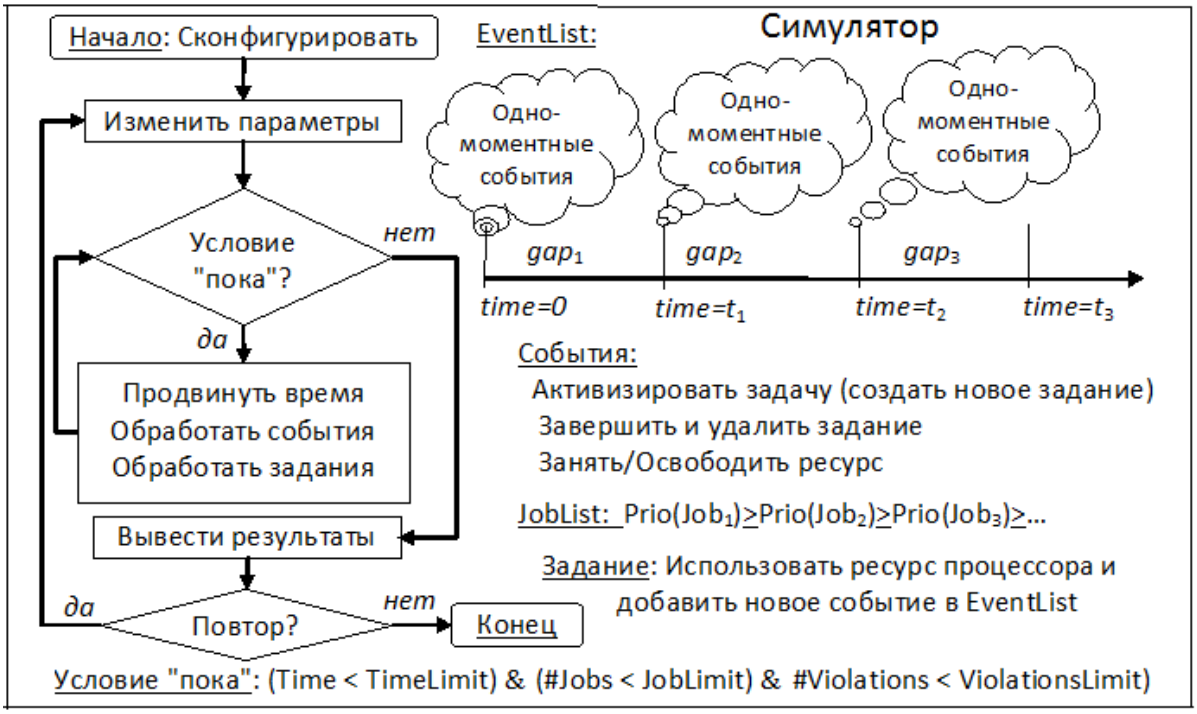


Рис. 1. Общая схема работы симулятора
Fig. 1. The overall structure of the simulator workflow

ни отклика для каждой задачи устанавливаются в нуль и все ресурсы переводятся в состояние «свободен».

В главном цикле моделирования рассматривается первая группа одномоментных событий в упорядоченном списке *EventList*, системное время симулятора устанавливается на этот момент, и по очереди обрабатываются все события из этой первой группы в соответствии с их типом.

1. Если очередное обрабатываемое событие – активизация задачи, то создается новое задание (экземпляр данной задачи), которое добавляется в список *JobList* активных заданий со своим приоритетом и запланированным моментом начала, равным текущему значению системного времени, а в список событий *EventList* добавляется новое событие – активизировать следующий экземпляр данной задачи (новое задание) в момент времени, не меньший чем текущее значение системного времени плюс период T_i данной задачи.

2. При завершении какого-либо задания обновляется время отклика соответствующей задачи как максимум из его текущего значения и полученного интервала существования данного задания. Если величина этого интервала превышает заданный предельный срок D_i данной задачи, то фиксируется нарушение ее выполнимости. Завершаемое задание удаляется из списка *JobList*.

3. В случае занятия/освобождения ресурса, если занимается уже занятый ресурс, то соответствующее задание переносится из списка *JobList* активных заданий в упорядоченный по приоритетам список заданий, ожидающих осво-

бождения данного ресурса; в противном случае данный ресурс занимается данным заданием. При освобождении ресурса, если список заданий, ожидающих его освобождения, не пуст, то первое задание из этого возвращается в список *JobList* активных заданий в соответствии со своим приоритетом и данный ресурс занимается этим заданием; в противном случае ресурс освобождается.

По завершении обработки события оно удаляется из списка *EventList*. После того, как будут обработаны все одномоментные события данной группы, рассматривается список *JobList* активных заданий (он мог измениться в результате обработки событий). Если этот список не пуст, то выбирается его первый элемент и счетчик процессорного времени, затраченного данным заданием на выполнение своей вычислительной работы, увеличивается на соответствующую величину. При этом возможно порождение нового события – завершение данного задания. После этого повторяется основной цикл симулятора. Условием завершения основного цикла (Условие «пока» на рис. 1) является либо исчерпание установленного лимита системного времени на данную сессию имитационного моделирования, либо достижение заданного числа созданных в процессе моделирования заданий, либо выявление заданного числа зарегистрированных нарушений выполнимости.

Результаты сеанса имитационного моделирования – максимальное время отклика задач, количество зафиксированных нарушений предельных сроков их выполнения, плотность приложения и другие данные – выводятся на экран. Также может быть выведен системный журнал моделирования, в котором регистрируются все возникшие системные события вместе с моментом времени их возникновения и другими сопроводительными данными. Вся эта информация может быть легко перенесена в MS Excel для удобного и наглядного представления полученных результатов и записей в системном журнале.

3. Профиль многозадачного приложения

В рассматриваемых ниже моделях программных приложений код каждой задачи представляется в виде последовательности сегментов. Каждый сегмент содержит фрагмент программного кода, замыкаемый оператором, связанным с каким-либо системным событием. Используются три типа сегментов, отличающиеся разновидностью замыкающего сегмент системного события:

- финальный сегмент – его замыкающим оператором является оператор завершения исполняемого задания;
- сегмент с запросом ресурса – его замыкающим оператором является оператор запроса доступа задания к одному из разделяемых ресурсов;
- сегмент с освобождением ресурса – его замыкающим оператором является оператор завершения одного из занятых заданием разделяемых ресурсов (оператор освобождения ресурса).

Задачи, не содержащие операторов доступа к разделяемому ресурсу, состоят из единственного (финального) сегмента.

Текстовое представление сегмента начинается спецификатором типа сегмента и завершается заданием его абсолютного веса. Спецификаторы сегментов имеют вид:

- «E_» – спецификатор финального сегмента;
- «L_» – спецификатор сегмента типа *lock* (запрос ресурса);
- «U_» – спецификатор сегмента типа *unlock* (освобождение ресурса).

Вес сегмента представляется целым числом, отражающим объем вычислений, исполняемых в рамках сегмента. Условимся задавать этот объем (т.е. оценку веса сегмента) в виде эквивалентного количества эталонных операций (эталоном такого рода может выступать, например, арифметическая операция с плавающей запятой).

Представление финального сегмента содержит два элемента: спецификатор и абсолютный вес. Например, последовательность символов «E_20» представляет финальный сегмент с абсолютным весом $w = 20$ эталонных операций.

В представлении *lock*-сегмента между спецификатором и значением абсолютного веса помещается целое число, соответствующее номеру запрашиваемого разделяемого ресурса – например, символы «L_1_10» представляют сегмент с абсолютным весом $w = 10$, завершающийся оператором запроса доступа к разделяемому ресурсу с номером 1. Аналогично символы «U_1_40» представляют *unlock*-сегмент с абсолютным весом $w = 40$, завершающийся оператором освобождения ранее занятого разделяемого ресурса с номером 1.

Для представления значений периодов задач в формальной модели приложения используется конструкция типа «T=1000», означающая, что период данной задачи равен 1000 единиц физического времени (например, миллисекунд). Значения предельных сроков D_i представляются аналогичными конструкциями типа «D=1000».

Текстовое представление модели задачи имеет вид строки, начинающейся описаниями ее параметров, за которыми следует описание последовательности сегментов ее кода. Описания отдельных параметров и сегментов разделяются пробелами. Например, для периодической задачи τ_i, представляемой строкой:

$$\text{T=1000 D=1000 L\_1\_10 U\_1\_40 E\_20}$$

ее экземпляры – задания $\tau_i^{(j)}$ $(j = 1, 2, ...)$ – порождаются каждую секунду (T=1000 миллисекунд), причем допустимая продолжительность существования каждого из них не превышает одной секунды (D=1000 миллисекунд), абсолютный вес W_i кода задачи τ_i составляет 10+40+20=70 эталонных операций. Задача содержит три сегмента. Первый сегмент с абсолютным весом 10 эталонных операций заканчивается операцией запроса информационного ресурса g_1. Второй сегмент весом 40 эталонных операций заканчивается операцией освобождения информационного ресурса g_1. Третий сегмент весом 20 эталонных операций заканчивается операцией завершения исполнения задачи.

Относительный вес C_i одного задания типа τ_i (т.е. выраженный в единицах физического времени объем ресурса процессора, требуемый для его исполнения) зависит от производительности P процессора, задаваемой числом эталонных операций за единицу физического времени (например, миллисекунду) по формуле (1). Величина u_i в формуле (2) характеризует полезную нагрузку – долю процессорного времени,

необходимого задаче τ_i. Сумма U в формуле (3) дает общую полезную нагрузку на каждое ядро процессора от данного многозадачного приложения. Таким образом, описание пяти задач $\tau_1, ..., \tau_5$ в правом столбце матрицы (5)

τ_1	T=1000 D=1000 E_150
τ_2	T=1330 D=1330 E_198
τ_3	T=1150 D=1150 E_168
τ_4	T=1520 D=1520 E_218
τ_5	T=1750 D=1750 E_260

(5)

представляет характеристики приложения из этих пяти независимых (без сегментов типа *lock/unlock*) задач. Следующий пример (6) представляет приложение с пятью задачами, разделяющими четыре ресурса $g_1, ..., g_4$:

τ_1	T=1000 D=1000 L_1_0001 U_1_0147 E_0002
τ_2	T=1150 D=1150 L_2_0001 L_3_0004 U_2_0002 U_3_0002 E_0159
τ_3	T=1330 D=1330 L_3_0001 L_4_0004 U_3_0002 U_4_0002 E_0189
τ_4	T=1520 D=1520 L_2_0001 L_4_0004 U_2_0002 U_4_0002 E_0219
τ_5	T=1750 D=1750 L_1_0001 U_1_0257 E_0002

. (6)

Данные о составе и параметрах задач, последовательности и параметрах сегментов каждой из задач, составляют *профиль* данного программного приложения. Данные о профиле приложения, дополненные указанием используемой дисциплины планирования и применяемого протокола доступа к разделяемым ресурсам, составляют *конфигурацию* исполнения приложения.

4. Результаты экспериментов

Для выполнения экспериментов по имитационному моделированию должны быть заданы конкретные конфигурации исследуемых программных приложений. Применяемые в практике промышленного программирования информативные и ценные конфигурации приложений реального времени, как правило, являются конфиденциальной информацией их владельцев и не публикуются; поэтому описываемые эксперименты проводились с использованием приложений, представляющих в большей степени методологический интерес – в частности, с приложениями, профиль которых отвечает следующим условиям:

- значения нагрузки одинаковы для всех задач: $\forall i, j\ (u_i = u_j)$ – приложения, отвечающие этим условиям, будем называть сбалансированными по нагрузкам, или просто сбалансированными;

- периоды задач распределены логарифмически (см. рис. 2, левая часть).

Приложения отвечающие этим условиям, впервые были рассмотрены в работе [1] — такие приложения будем называть приложениями с профилем Лю-Лейланда.

Профиль (7) дает пример приложения из 10 задач с профилем Лю-Лейланда:

τ_1	T=1000 D=1000 E_72		τ_6	T=1410 D=1410 E_102
τ_2	T=1070 D=1070 E_77		τ_7	T=1510 D=1510 E_109
τ_3	T=1140 D=1140 E_83		τ_8	T=1650 D=1650 E_116
τ_4	T=1230 D=1230 E_89		τ_9	T=1740 D=1740 E_124
τ_5	T=1320 D=1320 E_95		τ_{10}	T=1870 D=1870 E_132

. (7)

В левой части рис. 2 показана зависимость значений периодов T_i задач τ_i в приложении с профилем (7) от номера i задачи – в логарифмическом масштабе эта зависимость отображается прямой линией. В правой части того же рис. 2 показана зависимость плотности $Dens$ приложения от жесткости $H = T_i/D_i$ (фактически H^{-1}) ее задач для двух классических дисциплин планирования: «темпо-монотонной» (Rate Monotonic – RM) и «монотонной по срочности» (Earliest Deadline First – EDF) при исполнении на одноядерном процессоре. Значение H^{-1}, обратное величине жесткости H, изменяется на графике рис. 2 от 0,4 до 1,0 (оно предполагается одинаковым для всех 10 задач; т.е., фактически является жесткостью всего приложения).

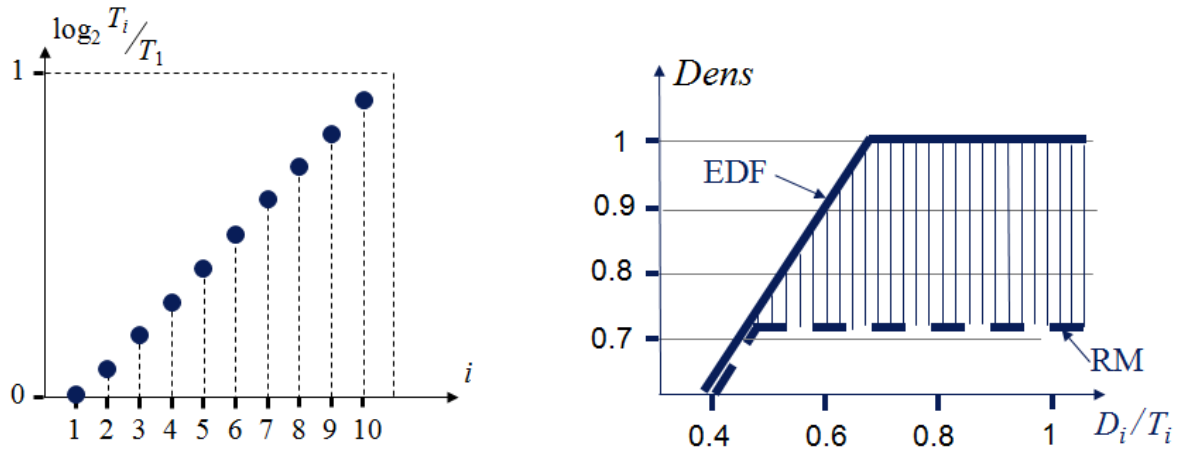


Рис. 2. Конфигурация Лю-Лейланда из 10 задач на одноядерной платформе
Fig. 2. The Liu-Layland configuration of 10 tasks on a single-core platform

Как видим, обе дисциплины планирования – EDF и RM – демонстрируют одинаковое поведение приложения, когда предельный срок существенно меньше периода (т.е. жесткость велика или, что то же самое, обратная величина к жесткости мала). Затем, с уменьшением жесткости (т.е. с увеличением ее обратной величины $H^{-1} = H_i^{-1} = D_i/T_i$) при дисциплине планирования EDF достигается максимально возможная плотность приложения, равная 1, тогда как при дисциплине RM плотность не превышает значения 0,72.

На рис. 3 представлены результаты моделирования поведения приложения из 5 независимых задач с профилем (5) с теми же двумя дисциплинами планирования при исполнении на одно- (слева) и двуядерной (справа) платформе. Профиль (5), как и профиль (7), отвечает требованиям сбалансированности нагрузок и логарифмического распределения периодов задач. Поэтому характер графиков плотности при дисциплинах планирования EDF и RM повторяет характер аналогичных графиков на рис. 2. Однако в случае использования дисциплины планирования EDF

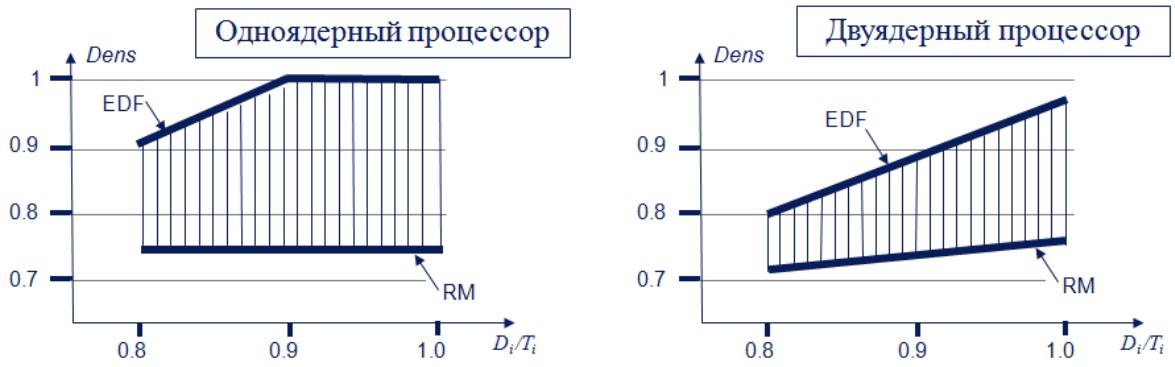


Рис. 3. Исполнение приложения из 5 независимых задач с профилем Лю-Лейланда
Fig. 3. Execution of an application of 5 independent tasks with the Liu-Layland profile

потенциальная эффективность 100% достигается лишь при значении $H^{-1} = 0,9$. Для приложений с профилем (7) такая эффективность достижима при существенно более жестких требованиях к срочности D_i (до величины $H^{-1} = 0,7$) – см. рис. 2.

Можно заметить, что при оптимальной производительности процессора P применение дисциплины планирования EDF с плотностью приложения, близкой к 1, обеспечивает 100% загрузку процессора ($Dens = 1$) на одноядерной платформе, тогда как на двуядерном процессоре не только дисциплина RM, но даже и EDF не может обеспечить такой экономичности.

На рис. 4 представлены результаты моделирования исполнения приложения из пяти задач с профилем (6) на одноядерной платформе. Здесь задачи $\tau_1, ..., \tau_5$ уже не являются независимыми – они используют 4 разделяемых информационных ресурса $g_1, ..., g_4$ с применением мьютексов для охраны соответствующих критических интервалов.

Рис. 4. Конфигурация из 5 зависимых задач на одноядерной платформе
Fig. 4. Configuration of 5 dependent tasks on a single-core platform

На графиках рис. 4 представлена зависимость плотности приложения от жесткости при дисциплинах планирования RM и EDF как с наследованием приоритетов

(что позволяет, как правило, ускорить выполнение приложения), так и без него. Стоит отметить два неожиданных факта, обнаруженных в данном случае:

- при отсутствии наследования приоритетов значение плотности для RM и EDF в точности совпадают;
- при добавлении наследования приоритетов плотности оказываются такими же, как для независимых задач для обеих дисциплин планирования.

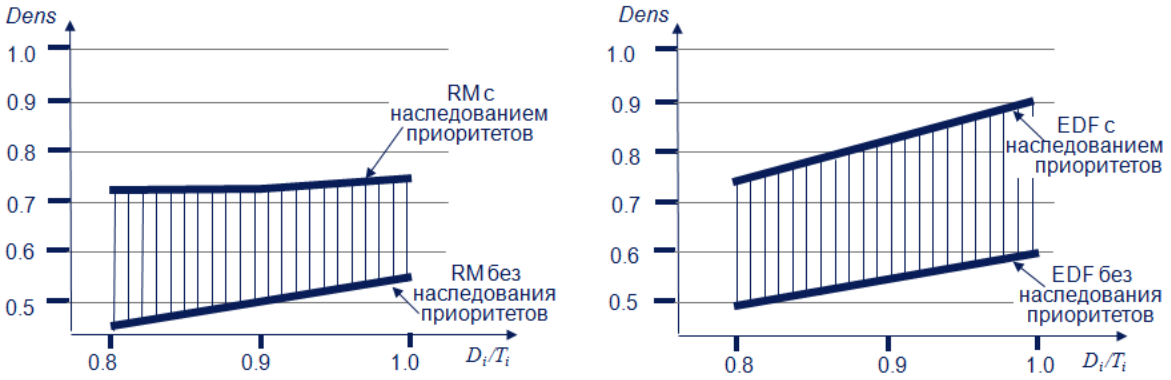


Рис. 5. Исполнение приложения из 5 зависимых задач на двуядерной платформе
Fig. 5. Execution of an application of 5 dependent tasks on a two-core platform

Когда та же конфигурация с добавленным механизмом наследования приоритетов исполняется на двуядерном процессоре, то преимущества дисциплины планирования EDF перед RM становятся менее значительными, в сравнении с тем, что наблюдалось на одноядерном процессоре (см. рис. 5).

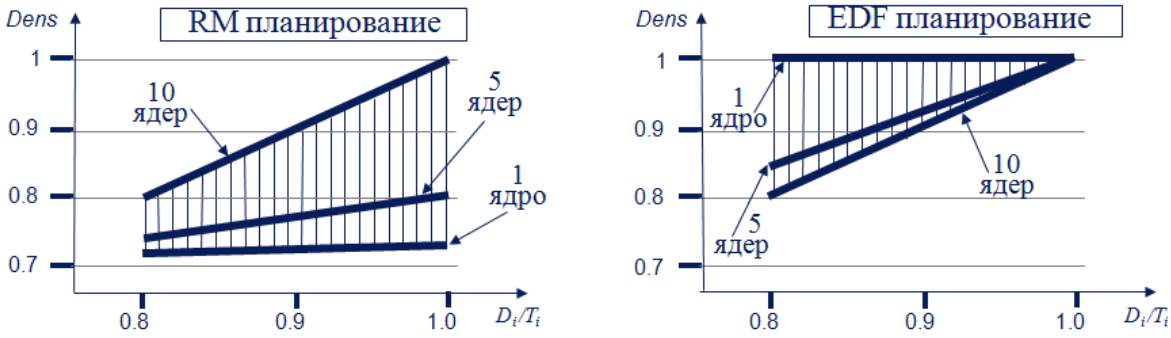


Рис. 6. Исполнение приложения из 10 независимых задач на многоядерной платформе
Fig. 6. Execution of an application of 10 independent tasks on a multi-core platform

На рис. 6 показано, как при увеличении числа ядер процессора исчезает различие между дисциплинами планирования RM и EDF для приложений с профилем (7).

На рис. 7 представлены результаты имитационного моделирования конфигурации из 4-х задач с неравномерной полезной нагрузкой для приложения из независимых задач – профиль (8) и для приложения с взаимозависимыми задачами – профиль (9).

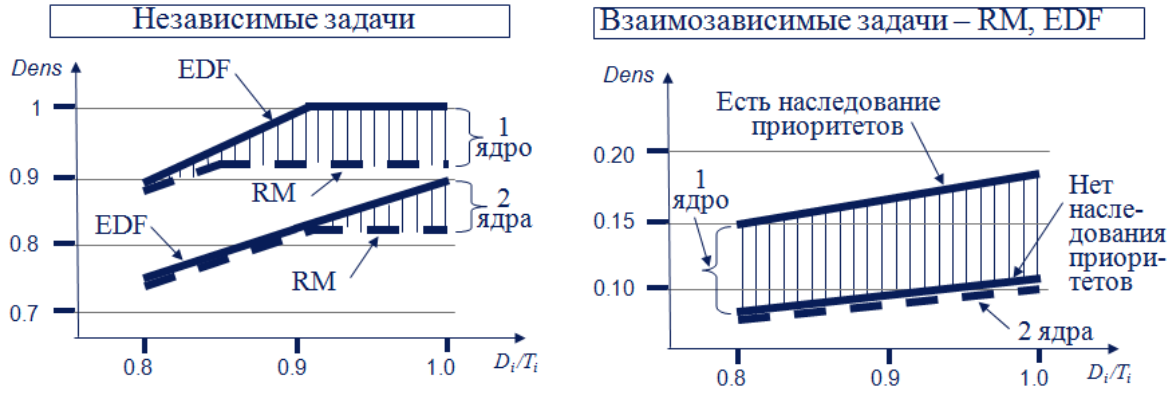


Рис. 7. Сравнение дисциплин RM и EDF для 4-х зависимых и независимых задач
Fig. 7. Scheduling modes RM vs. EDF for 4 dependent and independent tasks

Независимость задач на рис. 7 задается профилем (8):

τ_1	T=0050 D=0050 E_005
τ_2	T=0500 D=0500 E_200
τ_3	T=0550 D=0550 E_055
τ_4	T=0600 D=0600 E_240

, (8)

а их взаимозависимость в виде разделения ими 2-х информационных ресурсов (задачи τ_1 и τ_3 разделяют ресурс g_1, а задачи τ_3 и τ_4 разделяют ресурс g_2) задается профилем (9):

τ_1	T=0050 D=0050 L_1_0001 U_1_0003 E_001
τ_2	T=0500 D=0500 E_200
τ_3	T=0550 D=0550 L_1_0005 L_2_0005 U_2_0035 U_1_0005 E_005
τ_4	T=0600 D=0600 L_2_0005 U_2_0230 E_005

. (9)

Как следует из графиков на рис. 7 слева, для независимых задач с жесткостью, близкой к 1, дисциплина планирования EDF существенно более экономична, чем RM, как на одноядерной, так и на двуядерной платформе. Эта разница исчезает с уменьшением величины H^{-1}.

Анализ графиков на рис. 7 справа показывает, что:

- при наличии взаимозависимости между задачами приложения эффективность использования процессорного времени снижается в несколько раз при исполнении как на одноядерном, так и на двуядерном процессоре;
- применение дисциплины планирования EDF не приводит к повышению эффективности использования процессорного времени в сравнении с дисциплиной планирования RM;
- при исполнении на двуядерном процессоре применение механизма наследования приоритетов не приводит к повышению эффективности использования процессорного времени.

Дисциплины планирования RM и EDF оптимальны в своем классе приложений при исполнении на одноядерной платформе. Но эти дисциплины не оптимальны на многоядерных процессорах, что хорошо видно на приложениях со сдвинутой нагрузкой, т.е. при существенно неравномерном распределении общей полезной нагрузки между задачами. В приложении с профилем (10)

τ_1	T=1040 D=1040 E_840
τ_2	T=1000 D=1000 E_100
τ_3	T=1010 D=1010 E_101
τ_4	T=1020 D=1020 E_102
τ_5	T=1030 D=1030 E_103

(10)

60% от общей полезной нагрузки приходится на задачу τ_1: ее можно назвать «тяжелой», тогда как задачи $\tau_2, ..., \tau_5$ следует назвать «легкими». В таких случаях целесообразно применить дисциплину RM в следующей модификации: тяжелой задаче придается высший приоритет, а планирование легких задач следует обычной дисциплине RM (см. рис. 8).

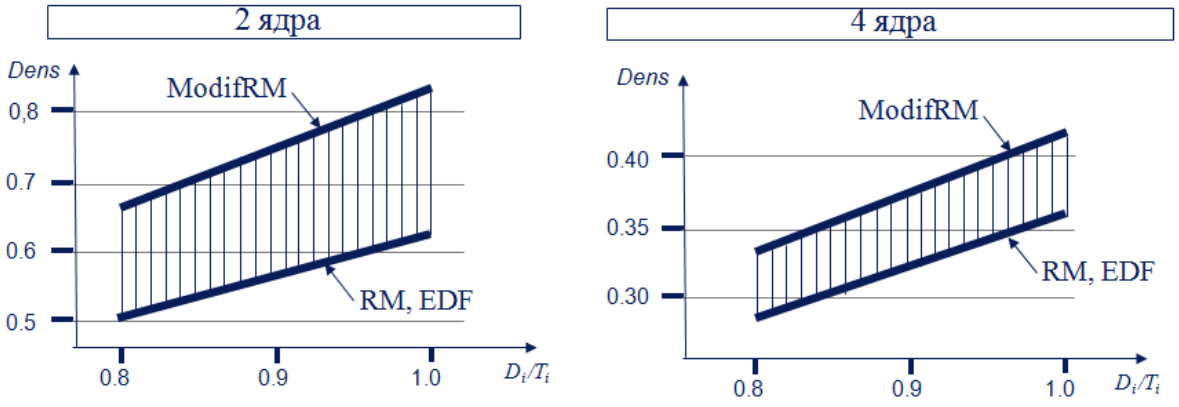


Рис. 8. Сравнение дисциплин RM и EDF для задач со сдвинутой нагрузкой
Fig. 8. Scheduling modes RM vs. EDF for tasks with shifted utility load

Графики на рис. 8 показывают преимущество этой дисциплины ModifRM (Modified Rate Monotonic) на платформе с 2 и 3 ядрами, тогда как применение дисциплин RM и EDF в этом случае одинаково неэкономично.

5. Заключение

Применение описанных методов имитационного моделирования на программном симуляторе для оценки выполнимости приложений для многоядерных платформ позволяет получить объективные данные для выбора оптимального сочетания дисциплин планирования и протоколов доступа. Точные аналитические методы для таких оценок известны только для одноядерных платформ; будучи примененными к многоядерным платформам, они дают слишком пессимистические результаты. Таким

образом, имитационное моделирование становится важным инструментом для выявления оптимальных структур приложений и платформ для многозадачных приложений реального времени. Разработанные симуляторы могут использоваться для проверки выполнимости таких приложений.

Дальнейшие исследования будут нацелены на расширение номенклатуры изучаемых дисциплин планирования, протоколов доступа и профилей приложений.

Список литературы / References

[1] Liu C., Layland J., "Scheduling Algorithms for Multiprocessing in a Hard Real-Time Environment", *Journal of the ACM*, **20**:1 (1973), 46–61.

[2] Andersson B., Baruah S., Jonsson J., "Static-Priority Scheduling on Multiprocessors", *Proc. 22nd IEEE Real-Time Systems Symposium*, 2001, 193–202.

[3] Laplante P.A., *Real-Time Systems Design and Analysis*, John Wiley & Sons, Inc., 2004.

[4] Baker T., "Multiprocessors EDF and Deadline Monotonic Schedulability Analysis", *Proc. 24th IEEE Real-Time Systems Symposium*, 2003, 120–129.

[5] Andersson B., "Global Static-Priority Preemptive Multiprocessor Scheduling with Utilization Bound 38%", *Proc. 12th International Conference on Principles of Distributed Systems*, 2008, 73–88.

[6] Baranov S.N., "Real-Time Multi-Task Simulation in Forth", *Proc. 18th Conf. FRUCT Association*, 2016, 21–26.

[7] Baranov S.N., Nikiforov V.V., "Application Density and Feasibility Checking in Real-Time Systems", *System Informatics*, 2016, № 7, 1–9.

[8] Baranov S.N., Nikiforov V.V., "Density of Multi-Task Real-Time Applications", *Proc. 17th Conf. FRUCT Association*, 2015, 9–15.

[9] Никифоров В. В., *Программа ОЭКПП для оценки эффективности конфигураций программных приложений*, Свидетельство о государственной регистрации программы для ЭВМ №2016618872 от 9 августа 2016 (RU), 2016, http://www1.fips.ru/wps/portal/Registers/.
[Nikiforov V. V., *Program OEKPP for Estimating the Efficiency of Software Application Configurations*, Certificate of computer program registration number 2016618872 of 9 August 2016 (RU), 2016, http://www1.fips.ru/wps/portal/Registers/].

[10] Баранов С. Н., *Программа RTMT для имитационного моделирования исполнения многозадачных приложений*, Свидетельство о государственной регистрации программы для ЭВМ №2016613095 от 16 марта 2016 (RU), 2016, http://www1.fips.ru/wps/portal/Registers/.
[Baranov S. N., *Program RTMT for Simulation of Multi-Task Application Behavior*, Certificate of computer program registration number 2016613095 of 16 March 2016 (RU), 2016, http://www1.fips.ru/wps/portal/Registers/].

[11] Forth 200x, 2016, http://www.forth200x.org/forth200x.html.

Baranov S. N., Nikiforov V.V., "Analysis of Real-Time Applications Feasibility through Simulation", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 673–687.

DOI: 10.18255/1818-1015-2016-6-673-687

Abstract. An approach to estimate feasibility of a real-time multi-task application with various combinations of the scheduling mode and the protocol of access to shared informational resources when run on a multi-core platform is described. The application structure is specified through a simple

formalized profile consisting of segments of three types and specifying access to informational resources shared among application tasks, the amount of the required computing resource being estimated for each segment. The approach is based on the notion of application density introduced by the authors, which characterizes the use of computational resource by this application and is derived from estimation of the application feasibility for various values of processor performance and the number of its cores in case of a multi-core platform. The overall structure of a simulation tool for estimation of the task response time (and therefore, application feasibility) is described, which provides more exact data compared to the known analytical methods where they are applicable. Two dissimilar implementations of this tool were developed and run on a number of benchmarks, including Liu-Layland configurations specified in the described formalism for application structure; the results in form of charts are presented along with their analysis and interpretation. The suggested approach allows to indentify an optimal combination of the scheduling mode and access protocol for the given multi-task application structure.

Keywords: simulation, real-time, application density, application feasibility

About the authors:
Sergey N. Baranov, orcid.org/0000-0001-6692-8432, Doc. Nat. Sci.,
SPIIRAS, ITMO University,
49 Kronverksky pr., St.Petersburg 197101, Russia, e-mail: SNBaranov@gmail.com

Victor V. Nikiforov, orcid.org/0009-0203-2514-7755, Doc. Nat. Sci.,
SPIIRAS, 14 liniya 39, St.Petersburg 199178, Russia,
e-mail: nik@iias.spb.su

Acknowledgments:
This work was partially financially supported by Government of the Russian Federation, Grant 074-U01.

Моделирование и анализ информационных систем. Т. 23, №6 (2016), с. 688–702
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 688–702

DOI: 10.18255/1818-1015-2016-6-688-702

УДК 519.172

Применение раскрашенных сетей Петри для верификации конструкций управления сценариями языка UCM

Визовитин Н.В., Непомнящий В.А, Стененко А.А.

получена 15 марта 2016

Аннотация. В данной работе представлен метод анализа и верификации моделей Use Case Maps (UCM) с конструкциями управления сценариями — защищенными компонентами и конструкциями обработки ошибок. Анализ и верификация UCM моделей проводится с помощью раскрашенных сетей Петри (РСП) и верификатора SPIN. Приводятся описания алгоритмов трансляции UCM в РСП и РСП во входной язык Promela системы SPIN. Впервые представлены оценки для количества элементов результирующих РСП моделей в зависимости от количества элементов исходных UCM моделей с конструкциями управления сценариями, а также количества состояний Promela моделей. Представленный метод анализа и верификации демонстрируется на примере верификации программы обновления прошивки маршрутизатора.

Ключевые слова: верификация, трансляция, нотация Use Case Maps, раскрашенные сети Петри, верификатор SPIN, защищенный компонент, обработка ошибок

Для цитирования: Визовитин Н.В., Непомнящий В.А, Стененко А.А., "Применение раскрашенных сетей Петри для верификации конструкций управления сценариями языка UCM", *Моделирование и анализ информационных систем*, **23**:6 (2016), 688–702.

Об авторах:
Визовитин Николай Валерьевич, orcid.org/0000-0001-7420-2711, младший научный сотрудник,
Институт систем информатики им. А.П. Ершова СО РАН,
проспект Академика Лаврентьева, 6, г. Новосибирск, 630090 Россия, e-mail: vizovitin@gmail.com

Непомнящий Валерий Александрович, orcid.org/0000-0003-1364-5281, канд. физ.-мат. наук, доцент,
Институт систем информатики им. А.П. Ершова СО РАН,
проспект Академика Лаврентьева, 6, г. Новосибирск, 630090 Россия, e-mail: vnep@iis.nsk.su

Стененко Александр Александрович, orcid.org/0000-0002-9538-6887, аспирант,
Институт систем информатики им. А.П. Ершова СО РАН,
проспект Академика Лаврентьева, 6, г. Новосибирск, 630090 Россия, e-mail: stirpar@mail.ru

Благодарности:
Работа частично поддержана грантом РФФИ 14-07-00401.

Введение

Предотвращение ошибок на ранних стадиях разработки программных проектов, таких как сбор и анализ требований, является важной проблемой в силу высокой стоимости их исправления на более поздних стадиях. Графическая нотация Use Case Maps (UCM) [12] позволяет формально описывать и анализировать функциональные требования. В то же время она поддерживает возможность контроля

требований заказчиком. UCM модели могут использоваться для решения широкого спектра задач. Они применяются для генерации тестовых сценариев [5, 6], формирования критериев покрытия тестами [4] и как язык спецификации свойств [10] для верификаторов.

UCM модели представляют набор сценариев как совокупность причинно-следственных связей между действиями в моделируемой системе. Действия могут быть связаны с компонентами системы, отражая ее архитектуру. Нотация UCM описывает взаимодействие архитектурных сущностей, уделяя внимание причинно-следственным связям и абстрагируясь от некоторых деталей передачи сообщений и обработки данных.

Средства для анализа и верификации UCM моделей недостаточно развиты. Стандарт UCM [12] описывает процедуру анализа, реализованную в редакторе jUCMNav [13]. Но эта техника анализа является достаточно примитивной и сложна в использовании. Стандарт дает семантику языка в неформальном виде, используя процедуры обхода UCM моделей. Поэтому часть работ, посвященных UCM, рассматривают формальную семантику [8]. В нескольких работах рассматривается проблема верификации UCM моделей. Среди них отметим работу [9], в которой предлагается метод моделирования и валидации UCM моделей с временными конструкциями. Методы верификации UCM моделей были разработаны для отдельных предметных областей. В работе [1] представлены методы тестирования, анализа и верификации для телекоммуникационных систем, основанные на UCM моделях.

В наших работах [16, 17] был представлен новый подход для анализа и верификации UCM моделей, базирующийся на раскрашенных сетях Петри (РСП). В рамках этого подхода UCM модель транслируется в РСП. Полученная РСП может быть верифицирована с помощью разработанного верификатора РСП методом проверки моделей, использующего известную систему SPIN [11], и проанализирована с помощью системы CPN Tools [7]. В работе [17] расширяется набор поддерживаемых конструкций UCM защищенными компонентами и конструкциями для обработки ошибочных ситуаций. В настоящей работе, наряду с описанием метода трансляции UCM моделей в РСП, представлены оценка размера результирующих РСП, а также оценка числа состояний программы на входном языке Promela системы верификации SPIN.

1. Обзор нотации Use Case Maps

Нотация Use Case Maps является одним из языков, описанных в стандарте User Requirements Notation [12]. UCM является высокоуровневым сценарно-ориентированным графическим средством моделирования, описывающим поведение системы через причинно-следственные связи между событиями и действиями, которые могут быть связаны со структурой компонентов системы. Нотация упрощает моделирование и анализ функциональных требований для распределенных и параллельных систем, а также позволяет рассуждать об архитектуре рассматриваемой системы.

Ниже приведен обзор основных элементов нотации UCM. Подробное описание языка, включая графический синтаксис, приведено в [12]. *Диаграмма (map)* может содержать произвольное число путей и компонентов. *Пути (path)* выража-

ют причинно-следственные связи между действиями. Пути являются направленными и могут соединять несколько типов *узлов (Path Node)*. Пути начинаются в *начальных узлах (Start Point)* и заканчиваются в *конечных узлах (End Point)*, которые отражают начальные и результирующие состояния системы соответственно, или предусловия и постусловия. Начальные узлы также могут обозначать начало сценариев для обработки ошибок и исключений. Такие узлы называют *начальными узлами обработчиков ошибок (Failure Start Points)* и *исключений (Abort Start Points)* соответственно. Тип начального узла определяется атрибутом `failureKind`, а атрибут `failureList` определяет список ошибочных и исключительных ситуаций, которым он соответствует. Начальный узел обработчика исключений является начальным узлом обработчика ошибок, который также останавливает все сценарии в своей области действия — диаграмме, на которой он находится, а также всех дочерних диаграммах в соответствии с иерархией узлов расширения (см. ниже). *Узлы действия (Responsibility)* отражают шаги, необходимые для выполнения сценария. *Or-ветвления (Or-Fork)* с условиями выбора исходящих ветвей и *Or-объединения (Or-Join)* используются для моделирования альтернативного выбора и циклов. *And-ветвления (And-Fork)* и *And-объединения (And-Join)* выражают параллелизм. Нотация UCM не накладывает никаких ограничений на порядок и вложенность элементов ветвления и объединения. *Узлы ожидания (Waiting Place)* и *таймеры (Timer)* обозначают точки, в которых исполнение сценария приостанавливается, пока не будет выполнено некоторое условие, или до прибытия сигнала. Как правило, у таймеров два исходящих пути: *основной (regular path или release path)* и *альтернативный (timeout path). Узлы соединения (Connect)* и *пустые узлы (Empty Point)* используются для синхронного или асинхронного соединения двух путей. Эти элементы, как правило, не имеют собственного графического представления. При достижении сценарием *узла прерывания (Failure Point)* он может прерваться, если произойдет ошибка или исключение. Каждый узел прерывания имеет условие срабатывания, а также имя, идентифицирующее произошедшую ошибку или исключение. Имя определяет начальные узлы обработчиков, с которых должно продолжиться исполнение сценария, если условие срабатывания оказалось истинным. UCM модели могут быть иерархически декомпозированы на *дочерние диаграммы (plug-in map)*, содержащие переиспользуемые шаблоны поведения и структуры, с помощью *узлов расширения (Stub)*.

Компоненты (Component) используются для моделирования структурных особенностей системы. Говорят, что элементы диаграммы, изображенные внутри компонента, *связаны (bound)* с ним. UCM модели без компонентов называют *свободными (unbounded)*. Компоненты могут содержать вложенные компоненты и иметь различные типы. Большинство типов компонентов не влияют на семантику и служат лишь для передачи архитектурных особенностей системы. Исключения составляют *объекты (Object)* и *защищенные (protected)* компоненты, ограничивающие количество одновременно исполняемых сценариев внутри них. В стандарте максимальное количество одновременно исполняемых сценариев внутри защищенного компонента равно 1. Следовательно, защищенные компоненты работают как механизм взаимного исключения для одновременно исполняемых сценариев.

2. Метод трансляции UCM моделей в раскрашенные сети Петри

Для анализа и верификации UCM моделей они транслируются в раскрашенные сети Петри [14]. Входные и выходные модели представляются в виде иерархических ориентированных графов с дополнительной информацией, ассоциированной с вершинами и дугами. Алгоритм трансляции переводит одно представление в другое.

2.1. Ограничения на входные UCM модели

На входные UCM модели налагаются следующие ограничения. Считается, что направление всех путей однозначно определено и все узлы модели уникально идентифицируемы. Особая семантика обхода узлов, связанных с компонентами типа объект, не поддерживается. UCM модели с начальными узлами обработчиков исключений отвергаются. Эти узлы редко используются при моделировании и вызывают комбинаторный взрыв количества состояний РСП модели в силу семантики остановки множества сценариев.

Также для трансляции защищенных компонентов введем дополнительное ограничение на использование узлов Or-ветвления, Or-объединения и таймеров. Каждый такой узел должен либо одновременно со всеми своими смежными узлами быть связанным с защищенным компонентом, либо нет. Это ограничение не является существенным, т.к. может быть удовлетворено простой модификацией исходной модели, например, введением дополнительных пустых узлов.

Указанные ограничения позволяют поддерживать наиболее часто используемые элементы и сценарии их использования. Поэтому они не ограничивают спектр поддерживаемых UCM моделей существенным образом.

2.2. Обзор алгоритма трансляции UCM моделей в РСП

Алгоритм трансляции UCM моделей в раскрашенные сети Петри на верхнем уровне состоит из пяти этапов. Алгоритм трансляции подробно описан в [15 – 17].

На первом этапе входная UCM модель подвергается предварительной обработке. В частности, осуществляется проверка удовлетворения ограничениям алгоритма, определяются начальные значения для всех переменных, входная модель преобразуется в свободную (т.е. из нее удаляются все компоненты). Информация о защищенных компонентах сохраняется в атрибутах узлов, связанных с ними. Пользователь уведомляется о любых неявных преобразованиях на данном этапе.

На втором этапе создаются определения на языке CPN ML, общие для всей модели. На этом этапе определяются цвета, константы и некоторые переменные. Вводятся служебные цвета, включая `UNIT` — стандартный «основной» цвет с единственным возможным значением `()`. Фишки цвета `UNIT` в основном используются для моделирования передачи сигналов и выполнения сценариев.

На третьем этапе дуги графа UCM модели, полученной в результате предварительной обработки на первом этапе, кроме дуг, инцидентных узлам соединения, разбиваются вершинами, соответствующими узлам нового типа — *вспомогатель-*

ным узлам. Преобразование, осуществляемое на данном этапе, сохраняет аннотации на исходящих дугах.

На четвертом этапе каждый узел вместе со своей непосредственной окрестностью, определяемой узлами соединения и вспомогательными узлами, обрабатывается по отдельности. Каждый рассматриваемый узел преобразуется во фрагмент модели РСП — аннотированный граф с дополнительными определениями на языке CPN ML. Фрагмент РСП также создается для каждого имени ошибки из UCM модели.

На пятом этапе объединяются фрагменты РСП, полученные на четвертом этапе. Элементы полученной модели с одинаковыми именами либо сливаются, если это возможно, либо представляются в виде объединяющих мест (fusion place).

3. Трансляция узлов, связанных с защищенными компонентами

Для эффективной верификации UCM моделей с помощью РСП необходимо, чтобы количество одновременно исполняющихся сценариев в любой заданной точке было ограничено. Иначе, в транслированной РСП появляются места с неограниченной емкостью, моделирующие транспорт сигналов.

Стандарт UCM предлагает способ для моделирования механизма взаимного исключения исполнения сценариев для подмножества путей модели. Таким механизмом являются защищенные компоненты, изображаемые с двойным контуром. Элементы UCM модели, связанные с любым экземпляром защищенного компонента, попадают под его действие. Исполнение любого сценария может быть продолжено внутрь защищенного компонента, только если никакой другой сценарий не исполняется внутри него.

Заметим, что предлагаемый стандартом механизм является слишком ограниченным для того, чтобы одновременно представлять широкий спектр взаимодействий различных сценариев и ограничивать количество исполняющихся сценариев. Поэтому мы предлагаем расширить стандарт, позволяя задавать максимальное количество одновременно исполняющихся сценариев внутри защищенного компонента. Это можно сделать, либо добавив новый целочисленный атрибут `scenarios` в класс *Component* абстрактной грамматики UCM, либо используя связанные с компонентом элементы комментария. Второй способ может быть использован для того, чтобы избежать необходимости модификации существующих редакторов UCM.

Далее будет рассмотрена трансляция компонентов с атрибутом `protected = true` и положительным значением атрибута `scenarios`, и других элементов UCM модели, связанных с ними. Заметим, что если `scenarios = 1`, то такой компонент является защищенным компонентом в терминах действующего стандарта UCM.

Каждый защищенный компонент, как и другие элементы UCM, имеет имя. Также мы ввели дополнительное ограничение на использование узлов Or-ветвления, Or-объединения и таймеров. Оно позволяет избежать неоднозначности трактовки семантики UCM моделей и существенного усложнения алгоритма трансляции.

На первом этапе алгоритма трансляции дополнительно необходимо проверить ограничение `scenarios > 0` для всех защищенных компонент. Если оно не соблю-

дается, модель считается некорректной. Далее проверим дополнительное ограничение на узлы Or-ветвления, Or-объединения и таймеры. Если оно не выполняется, то модель считается непригодной к верификации и пользователю предлагается исправить ее путем добавления дополнительных пустых узлов на дуги, инцидентные узлам, вызвавшим проблему, или любым другим подходящим способом. Для каждого связанного узла UCM модели с ним ассоциируется информация (имя и значение атрибута `scenarios`) о каждом защищенном компоненте, с которым он связан. Заметим, что один узел может быть связан с несколькими различными защищенными компонентами. После этого все компоненты из модели можно удалить.

Второй и третий этапы алгоритма трансляции не меняются по сравнению с этими этапами, описанными в разделе 2.2.

Защищенные компоненты в РСП будут моделироваться с помощью так называемых антимест. Введение антимест является распространенным шаблоном моделирования в РСП, используемым для ограничения количества фишек в заданном фрагменте РСП. Начальная разметка создаваемых антимест состоит из такого же количества фишек типа `UNIT`, как и максимальное количество сценариев, одновременно исполняющихся в данном компоненте (значение атрибута `scenarios`).

На четвертом этапе алгоритма трансляция каждого узла, связанного в исходной модели, может породить необходимое количество антимест (по количеству связанных с ним защищенных компонентов) и дуги к переходу, соответствующему узлу, или от него. Только узлы, способные породить или остановить сценарии, проходящие через них и защищенный компонент, породят дополнительные дуги и антиместа. К таким узлам относятся And-ветвления, And-объединения, начальные узлы и конечные узлы. Между именами защищенных компонентов и соответствующих им антимест поддерживается взаимно однозначное соответствие.

Пятый этап алгоритма фактически остается без изменений — все дополнительные антиместа будут объединены в соответствии со своими именами, аналогично другим местам объединяемых фрагментов РСП.

Алгоритм трансляции узлов, связанных с защищенными компонентами, рассмотрен подробнее в [17].

4. Трансляция конструкций для обработки ошибок

Для обработки ошибок в UCM моделях используются узлы прерывания и начальные узлы обработчиков ошибок. Узел прерывания представляет собой точку пути, в которой продолжение исполнения сценария зависит от происхождения ошибки или исключения. Если произошла ошибка, то текущий сценарий прерывается и начинаются новые с соответствующих начальных узлов обработчиков ошибок. Соответствие между узлами прерывания и начальными узлами обработчиков ошибок определяется через имя ошибки.

Описание первого, второго и третьего этапов алгоритма трансляции дано в разделе 2.2.

На четвертом этапе алгоритма трансляции каждому имени ошибки в модели сопоставляется фрагмент РСП — переход и место с соответствующими именами. Место имеет тип `UNIT` и пустую начальную разметку. От места к переходу ведет

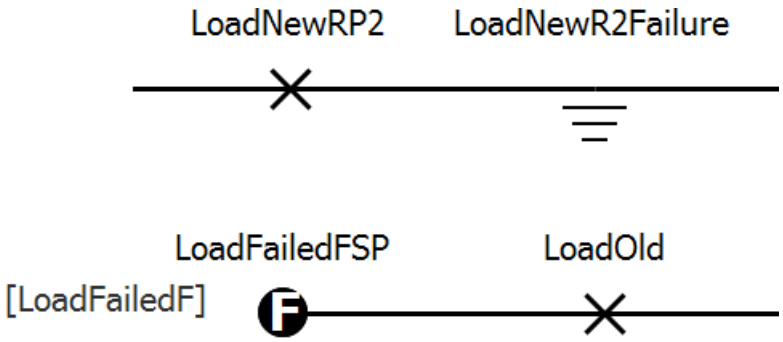


Рис. 1. Фрагмент UCM модели обработчика ошибок загрузки прошивки маршрутизатора
Fig. 1. UCM fragment with router firmware load errors handler

дуга с надписью `()`. Дуги также ведут от перехода ко всем местам, соответствующим начальным узлам обработчиков ошибок с соответствующим именем ошибки в списке ошибок, которые они могут обрабатывать. Эти дуги также имеют надпись `()`. Эта процедура трансляции во многом похожа на трансляцию узлов And-ветвления [16].

Трансляция начальных узлов обработчиков ошибок аналогична трансляции начальных узлов на дочерних диаграммах в случае, если они связаны со входами узла расширения [16]. Переход, соответствующий начальному узлу обработчика ошибок, связывается с местом типа `UNIT` с пустой начальной разметкой. Дуга от места к переходу имеет надпись `()`.

Трансляция узлов прерывания аналогична трансляции узлов Or-ветвления [16] с двумя выходными путями. Дуга, соответствующая одному из этих путей, продолжает штатное исполнение сценария, другая дуга ведет к месту, имя которого соответствует имени ошибки для этого узла прерывания. Условия на исходящих дугах основаны на условии срабатывания ошибки — одна из дуг переносит фишку, если условие истинно, другая – если оно ложно. Поэтому при трансляции узлов прерывания не возникает необходимости в дополнительных местах `*_OrForkWarnings`, которые используются для проверки того, что условия на выходных дугах узла Or-ветвления являются взаимоисключающими.

Заметим, что узел прерывания и начальный узел обработчика ошибок, ему соответствующий, могут быть на разных диаграммах. Этот случай разрешается естественным образом на пятом этапе алгоритма посредством преобразования отдельных мест, смежных с переходом, соответствующим имени ошибки, в объединяющие места при слиянии фрагментов РСП.

На Рис. 1 представлен пример фрагмента UCM модели с узлом обработчика ошибок. На Рис. 2 представлен соответствующий ему фрагмент РСП.

5. Оценка размера результирующих моделей РСП

В [16] были приведены оценки размера результирующей модели РСП в зависимости от размера исходной UCM модели. В этой работе рассматривались UCM модели без конструкций для обработки ошибок и без защищенных компонентов.

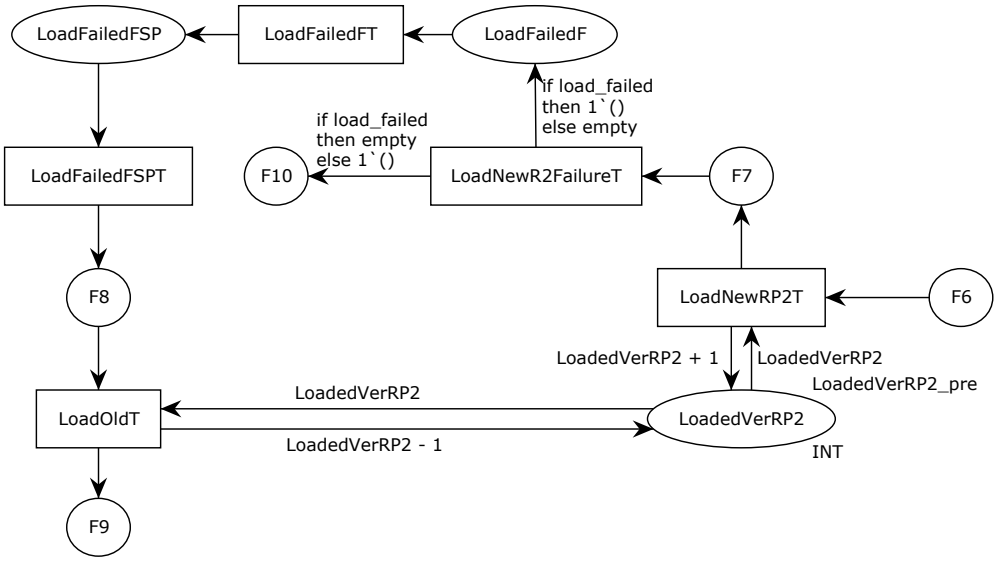


Рис. 2. РСП, полученная трансляцией фрагмента UCM, представленного на Рис. 1
Fig. 2. CPN translated from the UCM fragment presented on Fig. 1

Обозначим множество узлов UCM модели через V, множество дуг через E, множество переменных через $Vars$. Обозначим через $|A|$ количество элементов множества A. Обозначим максимальную степень захода среди всех узлов UCM модели через d^+_{max}. Также пусть d_s — максимальная степень среди узлов расширения, c_s — максимальное количество дочерних диаграмм среди всех узлов расширения, n_s — общее количество узлов расширения. Тогда общее количество переходов РСП ограничено величиной $O(d^+_{max}|V|+n_sc_sd_s)$. Общее количество мест ограничено $O(|E|+|V|+|Vars|+n_sc_sd_s)$. Общее количество дуг ограничено $O(|E|+d^+_{max}|V|+|Vars||V|+n_sd_s(|Vars|+c_s))$.

Пусть C — множество защищенных компонентов, F — множество имен ошибок в исходной UCM модели. Каждый защищенный компонент при трансляции создает одно дополнительное место и количество дуг, ограниченное удвоенным количеством узлов, связанных с этим компонентом. Использование защищенных компонентов не порождает дополнительных переходов. Для каждого имени ошибки при трансляции создается одно дополнительное место и один переход. Также, если V_f — множество начальных узлов обработчиков ошибок с соответствующим именем, то количество дополнительных дуг равно $|V_f|+1$ или $O(|V|)$. Для целей оценки количества элементов РСП узлы прерывания допустимо рассматривать как узлы Or-ветвления, а начальные узлы обработчиков ошибок как начальные узлы.

Поэтому, если в исходной UCM модели были использованы рассмотренные в данной работе новые конструкции — защищенные компоненты и конструкции для обработки ошибок, то общее количество переходов РСП ограничено величиной $O(d^+_{max}|V|+|F|+n_sc_sd_s)$. Общее количество мест ограничено $O(|E|+|V|+|Vars|+|C|+|F|+n_sc_sd_s)$. Общее количество дуг ограничено $O(|E|+|V|(d^+_{max}+|Vars|+|C|+|F|)+n_sd_s(|Vars|+c_s))$.

Таким образом, общее количество вершин РСП (то есть мест и переходов, в том

числе подстановочных) ограничено величиной $O(|E| + d^+_{max}|V| + |Vars| + |C| + |F| + n_s c_s d_s)$.

Заметим, что максимальное количество дочерних диаграмм c_s среди всех узлов расширения часто ограничено небольшой константой, что следует из естественного требования удобного представления исходной UCM модели.

Пусть N — общее количество элементов в исходной UCM модели (включая узлы, переменные, компоненты и имена ошибок). Тогда оценка может быть упрощена до $O(N^2)$ вершин РСП, т.е. является квадратичной в зависимости от количества элементов UCM модели.

Полученная оценка может быть уменьшена при наложении дополнительных ограничений на UCM модель. Так, если в UCM модели нет узлов Or-объединения и узлов расширения, то оценка принимает вид $O(|E| + |V| + |Vars| + |C| + |F|)$, т.е. $O(N)$.

Приведенные оценки показывают эффективность алгоритма трансляции в РСП. Меньший размер РСП модели также упрощает процесс анализа контрпримеров.

6. Верификация раскрашенных сетей Петри

6.1. Метод верификации раскрашенных сетей Петри

Формальная верификация может проводиться для РСП, у которых ограничена ёмкость мест и длина списков. Для проведения формальной верификации свойств РСП была разработана система CPNVer [2], которая включает транслятор из РСП во входной язык Promela системы верификации SPIN [11] и систему SPIN. Система CPNVer позволяет верифицировать свойства, представленные в виде формул линейной темпоральной логики LTL, а также в виде постусловий. Для постусловий проверяется истинность заданного предиката в конечных состояниях модели, то есть в таких состояниях, в которых нет допустимых переходов.

Заметим, что язык логики LTL достаточно выразителен. В частности, на этом языке можно представить многие свойства коммуникационных протоколов. Система SPIN широко применяется как для проведения исследований, так и для решения практических задач. Регулярно выходят новые версии этой системы.

Результатом верификации является либо сообщение системы SPIN о том, что проверяемые свойства выполняются, либо контрпример. Он является последовательностью срабатывания переходов РСП, которая приводит к нарушению проверяемого свойства. Контрпример может быть проанализирован с использованием системы CPN Tools [7]. Симуляция контрпримера в этой системе позволяет во многих случаях локализовать ошибку и внести изменения для её устранения.

6.2. Трансляция раскрашенных сетей Петри в язык Promela

Язык Promela оперирует понятиями процессов и переменных. Значения переменных входят в состояние программы. Процессы могут содержать инструкции недетерминированного выбора, а при верификации система SPIN последовательно выполнит каждую из альтернатив такого выбора. Программа на языке Promela может содержать вставки на языке C, а переменные языка C могут быть включены в состояние

программы. Инструкции процесса могут быть сгруппированы в атомарно выполняющиеся блоки.

Программа на языке Promela, полученная в результате трансляции, содержит набор переменных, каждая из которых хранит значение, соответствующее разметке определённого места исходной РСП, и единственный процесс, изменяющий значения этих переменных. В начале работы этот процесс инициализирует переменные, присваивая им значения, транслированные из начальной разметки мест. Затем процесс начинает выполнять главный цикл, каждая итерация которого состоит из двух этапов. На первом этапе происходит поиск допустимых в текущем состоянии переходов. На втором этапе происходит моделирование срабатывания одного из допустимых переходов, выбранного недетерминированно. Если допустимых переходов в текущем состоянии нет, то программа проверяет постусловие и завершает работу. Непосредственно перед завершением работы программа обнуляет значения переменных, это делается для того, чтобы избежать появления избыточных состояний. Каждая итерация главного цикла выполняется атомарно и не порождает промежуточных состояний.

Алгоритм трансляции состоит из нескольких этапов, на которых последовательно транслируются типы данных, функции, переменные, места и переходы исходной РСП. Типы данных РСП транслируются в типы данных языка C. При этом записи и кортежи транслируются в структуры, содержащие поля соответствующих типов, а списки транслируются в структуры, одно из полей которых — это массив для хранения элементов списка, а другое — это переменная, хранящая длину списка.

Для каждого типа данных РСП на языке C определяется тип «мультимножество элементов такого типа». Представление мультимножеств различается в зависимости от типа его элементов. Для типов `UNIT`, булевских типов и перечислений мультимножество представляется набором счётчиков числа вхождений каждого из значений типа данных. Для прочих типов мультимножество представляется массивом входящих в него элементов, а также переменной, хранящей ёмкость мультимножества. Элементы в указанном массиве хранятся упорядоченно, чтобы обеспечить единственность представления мультимножества и, таким образом, не допустить роста числа состояний программы.

Места РСП транслируются в переменные-мультимножества, а выражения начальной разметки мест транслируются в код на языке C, который выполняется на этапе инициализации программы и присваивает мультимножествам их начальные значения.

Каждый переход РСП транслируется в два фрагмента кода на языке Promela. Первый фрагмент проверяет, является ли данный переход допустимым, второй фрагмент моделирует срабатывание перехода, при условии, что он является допустимым. При генерации этих фрагментов кода, выражения на дугах РСП транслируются в код на языке C.

При проверке допустимости перехода выражения на дугах, инцидентных данному переходу, используются для того, чтобы определить, какие значениях переменных могут сделать его допустимым. С этой целью для каждой входящей в переход дуги производится перебор возможных значений выражения на данной дуге. В качестве возможных значений рассматриваются элементы, которые входят в разметку инцидентного этой дуге места. По значению выражения на дуге программа опреде-

ляет значения некоторых переменных, входящих в выражение. При этом значение каждой из переменных перехода должно определяться во время рассмотрения хотя бы одной из входных дуг перехода. Для найденного таким образом набора значений переменных перехода программа проверяет допустимость перехода следующим образом. Программа вычисляет, что охранное значение истинно, а также что для каждого из входных мест перехода его разметка содержит элемент, значение которого определяется при вычислении значения на соответствующей входной дуге перехода.

Второй фрагмент кода моделирует срабатывание перехода при известных значениях переменных перехода. Моделирование срабатывания перехода заключается в изменении значений переменных-мультимножеств, в соответствии с вычисленными при данных значениях переменных перехода выражениями на инцидентных переходу дугах. Во время моделировния срабатывания перехода программа показывает название данного перехода и значения его переменных для того, чтобы в случае обнаружения ошибки можно было определить, какая последовательность срабатывания переходов привела к нарушению свойств.

Подробное описание алгоритма трансляции дано в препринте [2].

6.3. Оценка числа состояний транслированной модели

Состояние программы на языке Promela включает значения переменных и значения указателей на инструкции каждого из процессов. Программа, полученная в результате трансляции, содержит единственный процесс. Указатель на инструкцию этого процесса может принимать одно из следующих значений: начало инициализации переменных, начало итерации главного цикла, завершение работы процесса. Эти значения не зависят от исходной РСП, так как инструкции процесса сгруппированы в атомарные блоки, которые при верификации исключают промежуточные состояния из рассмотрения. Переменные программы имеют ненулевые значения только в тех состояниях, когда указатель на инструкцию находится в начале итерации главного цикла. Каждое такое состояние, достижимое в процессе работы программы, взаимно однозначно отображается на достижимое состояние исходной РСП. Кроме этих состояний программа может находиться в начальном состоянии, когда переменные не инициализированы, и в конечном состоянии, в которое программа переходит, если нет достижимых переходов. Таким образом, число состояний программы составляет $N + 2$, где N — число достижимых состояний исходной РСП.

При проверке свойств, заданных LTL-формулами, система SPIN добавляет в программу ещё один процесс, представляющий собой автомат Бюхи для этой формулы [11]. В этом случае в состояние программы включается также состояние этого вспомогательного процесса. Если число состояний этого процесса составляет K, то число состояний программы не превосходит $(N + 2)K$.

Таким образом, число состояний программы на языке Promela, которая проверяет заданное свойство, линейно зависит от числа состояний исходной РСП, что позволяет применять систему SPIN для широкого класса моделей.

7. Пример верификации программы обновления прошивки маршрутизатора

Доступность сети становится все более важной проблемой для поставщиков услуг и их клиентов. Основной причиной простоя сетевого оборудования и недоступности сети часто является запланированное техническое обслуживание и обновление программного обеспечения. Многие производители маршрутизаторов, такие как Cisco Systems, предлагают оборудование с возможностью обновления программного обеспечения без прерывания обслуживания (ISSU), т.е. с минимальными перебоями для проходящего сетевого трафика клиентов.

Одна из тактик для обеспечения высокой доступности заключается в использовании избыточности компонент (активного и пассивного обработчика маршрутов). В [15] приведен пример UCM модели обновления маршрутизатора с двумя обработчиками маршрутов (RP) со старой версии прошивки на новую. Маршрутизатор работает в режиме нагруженного резерва, где RP1 является активным обработчиком, а RP2 – резервным. Первым шагом для обновления прошивки является ее копирование на файловые системы обоих обработчиков. Затем новая версия загружается на резервный обработчик RP2, что может привести к ошибке в силу несовместимого или поврежденного программного обеспечения. В случае возникновения ошибок загружается старая версия прошивки. Фрагмент UCM модели, описывающий эту ситуацию, представлен на Рис. 1.

Если новая версия была успешно загружена, RP2 перезагружается и входит в режим переключения с сохранением состояния (SSO). Если переключение происходит успешно и в течение некоторого времени новая версия принимается администратором, то ставший пассивным обработчик RP1 также обновляется на новую версию и процедура обновления успешно завершается. Иначе, происходит возврат к предыдущей версии прошивки и процедура обновления считается завершенной неуспешно.

Для формализации требований к процедуре обновления прошивки маршрутизатора по этим требованиям была построена UCM модель. При верификации UCM модели проверялись следующие свойства:

1. Во всех состояниях модели один обработчик маршрутов является активным. Это свойство, сформулированное в виде LTL-формулы [15], позволяет убедиться, что нет перебоев в обслуживании.
2. В конечном состоянии проверяются постусловия конечных узлов UCM модели. Эти условия позволяют убедиться, что обновление маршрутизатора завершилось либо корректным возвратом к предыдущей версии в случае ошибок, либо корректным обновлением до новой версии прошивки.

В результате верификации с помощью системы CPNVer после трансляции модели из UCM в РСП, была обнаружена ошибка, приводящая к нарушению первого свойства. В результате анализа контрпримера исходная UCM модель была модифицирована и повторно верифицирована. Верификация второй модели прошла успешно. Подробное описание UCM моделей и их верификации приведено в [15]. В полученной UCM модели зафиксированы корректные требования к системе. Эта UCM модель может быть использована для проверки построенного программного обеспечения на соответствие требованиям, а также для автоматического создания тестовых сценариев для проверки программного обеспечения [1, 4].

Заключение

Графическая нотация Use Case Maps является выразительным средством описания функциональных требований к программным системам и протоколам. В данной работе мы представили алгоритмы и средства для трансляции UCM моделей в раскрашенные сети Петри и модели на входном языке Promela известной системы верификации SPIN. Описанный метод позволяет анализировать и верифицировать более сложные UCM модели по сравнению с методом, описанным в работе [16], благодаря поддержке конструкций для обработки ошибочных ситуаций и защищенных компонентов.

Защищенные компоненты с расширенной семантикой особенно полезны для верификации. Они позволяют ограничить количество одновременно исполняемых сценариев, тем самым ограничивая емкость мест в соответствующей РСП модели. Это свойство позволяет обеспечить финитность модели и верифицировать её с помощью системы SPIN.

Текущая версия нашей системы поддерживает трансляцию файлов редактора jUCMNav [13] в файлы CPN Tools [7]. UCM модели, транслированные в РСП, могут быть верифицированы, используя систему CPNVer [2]. Процедура верификации показывает, является ли верифицируемая модель корректной по отношению к заданному свойству. Если это не так, необходимо локализовать ошибку. Для этой цели можно использовать контрпримеры, полученные системой SPIN. Для их анализа можно использовать систему CPN Tools.

Алгоритм трансляции UCM моделей в РСП является эффективным, так как он имеет квадратичную сложность в отношении размера результирующей РСП в зависимости от количества элементов UCM модели.

Важной задачей является обоснование корректности трансляции UCM в РСП. Однако это требует формальной семантики для UCM, которую стандарт [12] не предоставляет.

Мы планируем применить представленный метод для временных расширений [9] нотации UCM.

Список литературы / References

[1] Ануреев И.С., Баранов С.Н., Белоглазов Д.М., Бодин Е.В., Дробинцев П.Д., Колчин А.В., Котляров В.П., Летичевский А.А., Летичевский А.А. мл., Непомнящий В.А., Никифоров И.В., Потиенко С.В., Прийма Л.В., Тютин Б.В., “Средства поддержки интегрированной технологии для анализа и верификации спецификаций телекоммуникационных приложений”, *Тр. СПИИРАН*, **26** (2013), 349–383; [Anureev I.S., Baranov S.N., Beloglazov D.M., Bodin E.V., Drobitsev P.D., Kolchin A.V., Kotlyarov V.P., Letichevsky A.A., Letichevsky A.A. jr., Nepomniaschy V.A., Nikiforov I.V., Potiyenko S.V., Priyma L.V., Tyutin B.V., “Tools of Integrated Technology for Analysis and Verication of Telecom Application Specs”, *SPIIRAS Proceedings*, **26** (2013), 349–383, in Russian].

[2] Стененко А.А., Непомнящий В.А., *Верификация раскрашенных сетей Петри методом проверки моделей*, Препринт 178, http://www.iis.nsk.su/files/preprints/stenenko_nepomniaschy_178.pdf, Ин-т систем информатики СО РАН, Новосибирск, 2015, 28 с.; [Stenenko A.A., Nepomniaschy V.A., *Model Checking Approach to Verifiaction of Coloured Petri Nets*, Preprint 178, http://www.iis.nsk.su/files/preprints/

stenenko_nepomniaschy_178.pdf, Institute of Informatics Systems RAS, Novosibirsk, 2015, 28 pp., in Russian].

[3] Визовитин Н.В., Непомнящий В.А., *Алгоритмы трансляции UCM-спецификаций в раскрашенные сети Петри*, Препринт 168, http://www.iis.nsk.su/files/preprints/168.pdf, Ин-т систем информатики СО РАН, Новосибирск, 2012, 55 с.; [Vizovitin N.V., Nepomniaschy V.A., *UCM-Specifications to Colored Petri Nets Translation Algotithms*, Preprint 168, http://www.iis.nsk.su/files/preprints/168.pdf, Institute of Informatics Systems RAS, Novosibirsk, 2012, 55 pp., in Russian].

[4] Kotlyarov V., Weigert T., "Verifiable Coverage Criteria for Automated Testing", *SDL 2011, LNSC 7083*, 2011, 79–89.

[5] Baranov S.N., Drobintsev P.D., Kotlyarov V.P., Letichevsky A.A., "The Technology of Automated Verification and Testing in Industrial Projects", *Proc. IEEE Russia Northwest Section*, 110 Anniversary of Radio Invention Conference, IEEE Press, St.Petersburg, 2005, 81–89.

[6] Boulet P., Amyot D., Stepien B., "Towards the Generation of Tests in the Test Description Language from Use Case Map Models", *SDL 2015, LNCS 9369*, Springer, 2015, 193–201.

[7] *CPN Tools Homepage*, http://cpntools.org/.

[8] Hassine J., Rilling J., Dssouli R., "Abstract Operational Semantics for Use Case Maps", *FORTE 2005, LNCS 3731*, 2005, 366–380.

[9] Hassine J. Early modeling and validation of timed system requirements using Timed Use Case Maps, *Requirements Engineering*, **20**:2 (2015), 181–211.

[10] Hassine J., Rilling J., Dssouli R, "Use Case Maps as a Property Specification Language", *Software and Systems Modeling*, **8**:2 (2009), 205–220.

[11] Holzmann G.J, *The SPIN model checker. Primer and Reference Manual*, Addison-Wesley, 2004.

[12] *ITU-T, Recommendation Z.151 (10/12), User Requirements Notation (URN) — Language definition*, http://www.itu.int/rec/T-REC-Z.151/en.

[13] *jUCMNav — Eclipse plugin for the User Requirements Notation*, http://jucmnav.softwareengineering.ca/ucm/bin/view/ProjetSEG/WebHome.

[14] Jensen K., Kristensen L.M, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*, Springer, 2009.

[15] Vizovitin N.V., *Application of coloured Petri nets for verification of scenario control structures in UCM notation. Appendix*, http://bitbucket.org/vizovitin/ucm-verification-examples-3.

[16] Vizovitin N.V., Nepomniaschy V.A., Stenenko A.A., "Verifying UCM Specifications of Distributed Systems Using Colored Petri Nets", *Cybernetics and Sys. Anal.*, **51**:2 (2015), 213–222.

[17] Vizovitin N.V., Nepomniaschy V.A., Stenenko A.A., "Verification of UCM Models with Scenario Control Structures Using Colored Petri Nets", *System Informatics*, **7** (2016), 11–22.

Vizovitin N.V., Nepomniaschy V.A., Stenenko A.A., "Application of Coloured Petri Nets for Verification of Scenario Control Structures in UCM Notation", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 688–702.

DOI: 10.18255/1818-1015-2016-6-688-702

Abstract. This article presents a method for the analysis and verification of Use Case Maps (UCM) models with scenario control structures — protected components and failure handling constructs. UCM models are analyzed and verified with the help of coloured Petri nets (CPN) and the SPIN model checker. Algorithms for translating UCM scenario control structures into CPN and CPN into SPIN input language Promela are described. The number of elements of the resulting CPN model and the

number of Promela model states are estimated. The presented algorithm and the verification process are illustrated by the study of a network router firmware update.

Keywords: verification, translation, UCM, coloured Petri nets, SPIN, protected component, error handling

About the authors:
Nikolay V. Vizovitin, orcid.org/0000-0001-7420-2711, junior researcher,
A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences,
6 Acad. Lavrentjev pr., Novosibirsk 630090, Russia, e-mail: vizovitin@gmail.com

Valery A. Nepomniaschy, orcid.org/0000-0003-1364-5281, PhD,
A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences,
6 Acad. Lavrentjev pr., Novosibirsk 630090, Russia, e-mail: vnep@iis.nsk.su

Alexander A. Stenenko, orcid.org/0000-0002-9538-6887, graduate student,
A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences,
6 Acad. Lavrentjev pr., Novosibirsk 630090, Russia, e-mail: stirpar@mail.ru

Acknowledgments:
This work was partially supported by RFBR grant 14-07-00401.

Моделирование и анализ информационных систем. Т. 23, № 6 (2016), с. 703–714
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 703–714

DOI: 10.18255/1818-1015-2016-6-703-714

УДК 004.052, 519.179.2

Подход к верификации семейства мультиагентных систем разрешения конфликтов

Гаранина Н. О.[1,2], **Сидорова Е.А.**[1]

получена 13 марта 2016

Аннотация. В данной работе мы описываем метод верификации для семейств распределенных систем, которые порождаются контекстно-зависимой сетевой грамматикой специального вида. Эта грамматика содержит специальные нетерминальные символы — квази-терминалы. Квази-терминалы однозначно соответствуют терминалам грамматики и могут задавать процессы, которые определяются слиянием базовых процессов системы, в то время как нетерминалы задают сети параллельных композиций этих процессов. Данный метод верификации основан на техниках верификации моделей и абстракции. Абстрактная репрезентативная модель семейства систем зависит от задающей их грамматики и верифицируемых свойств системы. Эта модель симулирует поведение заданных систем таким образом, что свойства, которые выполняются в репрезентативной модели, также выполняются и во всех заданных системах. Проверку свойств репрезентативной модели можно осуществлять с помощью метода проверки моделей. Свойства порождаемых систем специфицируются с помощью универсальной логики ветвящегося времени $\forall CTL$ с конечными детерминированными автоматами в качестве атомарных формул. Мы показываем, что предложенный метод верификации можно применять для проверки некоторых свойств мультиагентных систем разрешения конфликтов, в частности, систем разрешения неоднозначностей при пополнении онтологий. Также показано, что этот подход можно использовать для верификации вычислений на подрешетках, являющихся подграфами решеток вычислений, например, для вычисления четности числа работающих процессов.

Ключевые слова: проверка моделей, контекстно-зависимая сетевая грамматика, мультиагентные системы, абстракция

Для цитирования: Гаранина Н. О., Сидорова Е.А., "Подход к верификации семейства мультиагентных систем разрешения конфликтов", *Моделирование и анализ информационных систем*, **23**:6 (2016), 703–714.

Об авторах:
Гаранина Наталья Олеговна, orcid.org/0000-0001-9734-3808, канд. физ.-мат. наук,
Институт систем информатики им. А.П. Ершова СО РАН,
проспект Лаврентьева, 6, г. Новосибирск, 630090 Россия, e-mail: garanina@iis.nsk.su

Сидорова Елена Анатольевна, orcid.org/0000-0001-8731-3058, канд. физ.-мат. наук,
Институт систем информатики им. А.П. Ершова СО РАН,
проспект Лаврентьева, 6, г. Новосибирск, 630090 Россия, e-mail: lena@iis.nsk.su

Благодарности:
[1]Работа выполнена при финансовой поддержке РФФИ (проект №15-07-04144).
[2]Работа выполнена при финансовой поддержке СО РАН (интеграционный проект №15/10 «Математические и методологические аспекты интеллектуальных информационных систем»).

Введение

Мотивацией нашей работы является задача разрешения неоднозначности в рамках задачи пополнения онтологий из данных, представленных текстами на естественном языке. В работе [7] мы описали алгоритмы анализа текста, порождающие систему информационных агентов, соответствующих экземплярам заданной онтологии и входным данным. Однако особенности естественного языка порождают неоднозначности при пополнении онтологий, и эти агенты предназначены их разрешать. Для разрешения неоднозначностей мы предлагаем оценивать контекст агентов-экземпляров, т.е. насколько агент связан с другими агентами посредством информации, которой он владеет, и выбирать среди альтернатив агента, наиболее интегрированного в текст. Мы разработали обобщенный алгоритм разрешения конфликтов в мультиагентных системах [9], специализацией которого является мультиагентный алгоритм разрешения неоднозначностей [8], удаляющий наименее интегрированных агентов из системы.

В процессе работы алгоритма все агенты в параллельном режиме исполняют довольно сложные протоколы с периодической локальной синхронизацией. Следовательно, необходимо использовать формальные методы для доказательства корректности этого алгоритма. В качестве техники верификации мы выбрали метод проверки моделей. Мы верифицируем довольно специфическую мультиагентную систему разрешения конфликтов. Работы по мультиагентным системам обычно фокусируются на поведении агентов, методах коммуникации между агентами, знании и мнениях агентов о других агентах и окружении и т.п. [6, 13]. Работы, касающиеся процесса разрешения конфликтов, обычно рассматривают этот процесс в терминах поведения агента, зависящего от его внутреннего состояния, методов рассуждений и аргументации, однако динамика связей агентов не исследуется [11]. Есть статьи, связанные с динамикой взвешенных связей, но эти связи однородны, и их изменение не влияет на внутреннее состояние агента [5]. С другой стороны, работы по изучению социальных сетей, в которых агенты связаны типизированными связями, не учитывают их вес [1]. Насколько нам известно, не существует работ по верификации алгоритмов разрешения конфликтов исследуемого нами типа.

Техника проверки моделей широко используется для верификации распределенных и мультиагентных систем [3]. В нашем случае необходимо верифицировать не одну мультиагентную систему, а бесконечное семейство таких систем. Для верификации бесконечных семейств распределенных сетей в работе [2] был предложен особый метод проверки моделей. Этот метод основан на использовании контекстно-свободных сетевых грамматик, порождающих семейства распределенных систем, а также на абстрагировании с помощью конечных автоматов. Идея метода состоит в конструировании инварианта сети, основанного на заданной грамматике. Этот инвариант симулирует поведение всех систем семейства согласованно с абстрактными функциями, ассоциированными с проверяемыми свойствами, выраженными в логике ветвящегося времени $\forall CTL$. Благодаря согласованной симуляции, свойства, выполняющиеся для репрезентативного инварианта, также выполняются для всех систем семейства. Однако авторы [2] исследовали только контекстно-свободные грамматики, тогда как наша модель мультиагентной системы порождается контекстно-зависимой грамматикой специального вида. В данной статье мы определяем та-

кую грамматику, расширив стандартное определение контекстно-свободной сетевой грамматики из [2] понятиями квази-терминалов и оператора слияния. Мы показываем, что метод верификации из [2] также может быть использован для семейств систем, заданных этой новой грамматикой, поскольку новая грамматика обладает свойствами, аналогичными свойствам исходной.

Оставшаяся часть статьи организована следующим образом. В следующем разделе 1 мы даем основные определения. Раздел 2 представляет результаты для нового оператора слияния, использующегося в нашей контекстно-зависимой грамматике с квази-терминалами. В разделе 3 описано использование предложенного метода верификации для системы разрешения конфликтов и вычислений на подрешетках. В заключении 4 обсуждаются направления дальнейших исследований.

1. Основные определения

Дадим необходимые определения из [2] в адаптированном виде. Изменения касаются оператора слияния и квази-терминалов сетевой грамматики.

Определение 1.
Помеченная система переходов (LTS) — это структура $M = (S, R, ACT, S_0)$, где
– S — множество состояний,
– $S_0 \subseteq S$ — множество начальных состояний,
– ACT — множество действий, и
– $R \subseteq S \times ACT \times S$ — тотальное отношение переходов такое, что для каждого $s \in S$ существует действие a и состояния s', для которого $(s, a, s') \in R$ (обозначается как $s \xrightarrow{a} s'$).

Пусть L_{ACT} — это класс систем LTS со множеством действий, являющимся подмножеством ACT, $L_{(S,ACT)}$ — это подмножество систем L_{ACT} со множеством состояний, являющимся подмножеством S. Пусть $ACT_1, ACT_2 \subseteq ACT$ и заданы две LTS $M_1 = (S_1, R_1, ACT_1, S_0^1)$ и $M_2 = (S_2, R_2, ACT_2, S_0^2)$ в классе L_{ACT}.

Определение 2.

– Функция $\|$: $L_{ACT} \times L_{ACT} \mapsto L_{ACT}$ называется *функцией композиции,* и параллельная композиция двух систем $M_1 \parallel M_2$ имеет вид $(S_1 \times S_2, R', ACT_1 \cup ACT_2, S_0^1 \times S_0^2)$.
– Функция $\cup : L_{ACT} \times L_{ACT} \mapsto L_{ACT}$ называется *функцией слияния,* и слияние двух систем $M_1 \cup M_2$ имеет вид $(S_1 \cup S_2, R', ACT_1 \cup ACT_2, S_0^1 \cup S_0^2)$.

Определение R' зависит от точной семантики функций композиции и слияния. Пусть S^i — это слово длины i в алфавите S.

Определение 3.
Для заданного множества состояний S и множества действий ACT всякое подмножество $\bigcup_{i=1}^{\infty} L_{(S^i,ACT)}$ называется *сетью* для пары (S, ACT).

Дадим определение *контекстно-зависимой сетевой грамматики с квази-терминалами (CSNQ-грамматика)* для описания сетей. Множество всех систем LTS,

выводимых в сетевой грамматике, образует сеть, которая также является системой LTS. Пусть S – это множество состояний и ACT — множество действий. *CSNQ-грамматика* $G = (T, Qt, t, N, P, S)$ – это грамматика, где

– T — конечное множество терминалов, каждый из которых является LTS из класса $L_{(S,ACT)}$, эти LTS называют *базовыми процессами*,

– Qt — множество квази-терминалов, каждый из которых является LTS из класса $L_{(S,ACT)}$, их слияние определяет LTS,

– отображение $t : Qt \mapsto T$ ассоциирует квази-терминалы с терминалами,

– N — конечное множество нетерминалов, каждый нетерминал определяет сеть,

– P — множество правил вывода следующего вида:

– $A \longrightarrow B \parallel_k C$, где $A \in N$, и $B, C \in T \cup Qt \cup N$, и $\parallel_i$ — функция композиции;

– $\{V_1, ...V_n\} \longrightarrow t(V_1) \cup_i ... \cup_i t(V_n)$, где для $j \in [1..n]$ $V_j \in Qt$, и $\cup_i$ — функция слияния.

– $S \in N$ представляет сеть, порожденную грамматикой.

Заметим, что такие грамматики контекстно-свободны относительно функций композиции и контекстно-зависимы относительно функций слияния. Отметим также, что, поскольку множество квази-терминалов может быть бесконечно, то, вообще говоря, множество правил вывода, содержащих квази-терминалы, также может быть не ограничено. Неограниченные множества правил могут задаваться, например, регулярными выражениями. Однако очевидно, что такие грамматики порождают конечные сети процессов.

Для задания свойств моделей, составленных из конечного, но неопределенного числа систем LTS, определим конечный автомат над алфавитом S.

Определение 4.
$D = (Q, q_0, \delta, F)$ — *детерминированный автомат* над S, где

– Q — множество состояний автомата,

– $q_0 \in Q$ — начальное состояние,

– $\delta \subseteq Q \times S \times Q$ — отношение переходов,

– $F \subseteq Q$ — множество допустимых состояний, и

– $L(D) \subseteq S^*$ — множество слов, допускаемых автоматом D.

Мы используем конечные автоматы над S для спецификации атомарных свойств состояний. Пусть D — это автомат над S. Состояние s выполнимо в D ($s \models D$) $\Leftrightarrow$ $s \in L(D)$. Язык спецификации — универсальная логика ветвящегося времени $\forall CTL$ [2] с конечными автоматами над S в качестве атомарных формул. Синтаксис $\forall CTL$ включает формулы, построенные из булевских констант, атомарных формул, связок $\neg$, $\vee$, $\wedge$, модальностей ветвящегося времени $\mathbf{AX}\varphi$, $\mathbf{AG}\varphi$, и $\varphi\mathbf{AU}\psi$ со стандартной семантикой.

Напомним определение абстрактной LTS из [2]. Для простоты, пусть язык спецификации содержит единственную атомарную формулу D. Для заданного автомата $D = (Q, q_0, \delta, F)$ и слова $w \in S^*$ функция, индуцированная w на Q, $f_w : Q \mapsto Q$, определяется как $f_w(q) = q' \Leftrightarrow q \xrightarrow{w} q'$. Заметим, что $w \in L(D) \Leftrightarrow f_w(q_0) \in F$. Два состояния s и s' *эквивалентны* $s \equiv s' \Leftrightarrow f_s = f'_s$. Функция f_s называется *абстракцией* s и обозначается как $h(s)$. Отношение $\models$ расширяется на абстрактные состояния: $h(s) \models D \Leftrightarrow f_s(q_0) \in F$. Поэтому $s \models D \Leftrightarrow h(s) \models D$.

Пусть F_D — это множество функций, соответствующих детерминированному автомату D. Функция абстракции h, расширенная на F_D, определяется как $h(f) = f$ для $f \in F_D$ и расширение функций h на $(S \cup F_D)$ — это $h((a_1, a_2, ..., a_n)) = h(a_1) \circ ... \circ h(a_n)$. Далее мы рассматриваем системы LTS в сети N на паре $(S \cup F_D, ACT)$.

Определение 5. *(абстрактной LTS)*
Для данной LTS $M = (S^i, R, ACT, S_0)$ в сети N, соответствующая абстрактная LTS определяется как $h(M) = (S^h, R^h, ACT, S_0^h)$, где
– $S^h = \{h(s)|s \in S^i\}$ — множество абстрактных состояний,
– $S_0^h = \{h(s)|s \in S_0\}$, и
– отношение R^h определено следующим образом. Для всех $h_1, h_2 \in S^h$, и $a \in ACT$: $(h_1, a, h_2) \in R^h \Leftrightarrow \exists s_1, s_2[h_1 = h(s_1) \wedge h_2 = h(s_2) \wedge (s_1, a, s_2) \in R]$.

M' симулирует M (обозначается как $M \preceq M'$), если и только если существует предпорядок симуляции $E \subseteq S \times S'$ ($(s, s') \in E$ обозначается как $s \preceq s'$), удовлетворяющий следующим условиям: для каждого $s_0 \in S_0$ существует $s'_0 \in S'_0$ такой что $s_0 \preceq s'_0$. Для каждого s, s', если $s \preceq s'$, то
– $h(s) = h(s')$, и
– для каждого s_1 такого, что $s \xrightarrow{a} s_1$, существует s'_1 такой, что $s' \xrightarrow{a} s'_1$ и $s_1 \preceq s'_1$.

2. Проверка моделей с оператором слияния

Первые два предложения леммы доказаны в [2], последнее доказано ниже:

Лемма 1.
1. $M \preceq h(M)$, т.е., $h(M)$ симулирует M.
2. Если $M \preceq M'$, тогда $h(M) \preceq h(M')$.
3. $M \cup M' \preceq h(M) \cup h(M')$

Доказательство пункта 3 очевидно: $M \cup M' \preceq h(M \cup M')$ в силу пункта 1, и $h(M \cup M') = h(M) \cup h(M')$.■

Следующая теорема о выполнимости свойств в системе LTS и в системе, ее симулирующей, была доказана в [2]. Она остается верной для нашего метода, поскольку не связана с новой грамматикой.

Теорема 1.
Пусть φ - формула $\forall CTL$ над атомарной формулой D. Пусть M и M' - две системы LTS, такие что $M \preceq M'$. Пусть $s \preceq s'$. Тогда $s' \models \varphi$ влечет $s \models \varphi$.

Определение 6.
Оператор композиции или слияния $\bullet \in \{\cup, \|\}$ называется *монотонным* относительно предпорядка симуляции $\preceq$, если и только если для заданных систем LTS, таких что $M_1 \preceq M_2$ и $M'_1 \preceq M'_2$, верно $M_1 \bullet M'_1 \preceq M_2 \bullet M'_2$. Сетевая грамматика G называется *монотонной*, если и только если все ее правила используют только монотонные операторы композиции и слияния.

Модифицируем определения и результаты для синхронных систем из [2] в соответствии с оператором слияния. Нашей моделью будет специальный вид систем LTS, *машина Мура* $M = (S, R, I, O, S_0)$, такая что множества входов I и выходов O не пересекаются. Кроме того, машины имеют специальное внутреннее действие, обозначаемое τ. Множество действий $ACT = \{\tau\} \cup 2^{I \cup O}$, где каждое не внутреннее действие — это множество входов и выходов. Переход $s \xrightarrow{a} t$ из состояния s в машине M при $a = i \cup o$, так что $i \subseteq I$ и $o \subseteq O$ возможен, только если среда предоставляет входы i и машина M вырабатывает выходы o.

Для оператора слияния множества входов и выходов сливающихся машин также не должны пересекаться. Пусть $I \cap O' = \emptyset$ и $O \cap I' = \emptyset$. Слияние машин M и M', $M'' = M \cup M'$ определено следующим образом:

– $S'' = S \cup S'$,
– $S''_0 = S_0 \cup S'_0$,
– $I'' = I \cup I'$ и $O'' = O \cup O'$, и
– $s'' \xrightarrow{a''} s''_1$ — переход в $R'' \Leftrightarrow$ выполнено следующее: $s'' \xrightarrow{a} s''_1$ — переход в R и $s'' \xrightarrow{a'} s''_1$ — переход в R' для некоторых a, a', таких что $a'' = a$ или $a'' = a'$.

Лемма 2.
Слияние $\cup$ монотонно относительно $\preceq$.

Доказательство. Пусть заданы $M = (S, R, I, O, S_0)$, $M_1 = (S_1, R_1, I_1, O_1, S_{1,0})$, $M' = (S', R', I', O', S'_0)$, $M'_1 = (S'_1, R'_1, I'_1, O'_1, S'_{1,0})$ — четыре машины Мура. Предположим, что $M \preceq M_1$ и $M' \preceq M'_1$. Пусть $E \subseteq S \times S_1$ и $E' \subseteq S' \times S'_1$ — соответствующие отношения симуляции. Докажем, что $M \cup M' \preceq M_1 \cup M'_1$.

Мы считаем, что $(s'', s''_1) \in E'' \Leftrightarrow (s'', s''_1) \in E$ или $(s'', s''_1) \in E'$. Покажем, что E'' имеет требуемое свойство. Из определения следует, что для заданного состояния $s_0 \in S_0 \cup S'_0$, существует $s_{0,1} \in S_{0,1} \cup S'_{1,0}$ такое, что $(s_0, s_{0,1}) \in E \cup E'$. Предположим $(s, s_1) \in E \cup E'$.

(1) По предположению верно, что $h(s) = h(s_1)$.

(2) Пусть $s \xrightarrow{a''} t$ — переход в $M \cup M'$. Это означает, что существует переход $s \xrightarrow{a} t$ в M или переход $s \xrightarrow{a'} t$ в M', такой что $a'' = a$ или $a'' = a'$. По определению существует $t_1 \in S_1 \cup S'_1$ такой, что $s_1 \xrightarrow{a} t_1$ или $s_1 \xrightarrow{a'} t_1$, где $(t, t_1) \in E$ или $(t, t_1) \in E'$. Следовательно, $s_1 \xrightarrow{a''} t_1$ и $(t, t_1) \in E''$, что завершает доказательство. ■

Понятие репрезентатива делает возможным построение симуляционного инварианта. Для заданной CSNQ-грамматики G мы ассоциируем каждый символ A грамматики с *репрезентативным процессом* $rep(A)$. Адаптируем определение свойства монотонности для множества репрезентативных процессов CSNQ-грамматики:

– для каждого терминала и квази-терминала A: $h(rep(A)) \succeq h(A)$, и
– для каждого правила $A \longrightarrow B \parallel C$: $h(rep(A)) \succeq h(h(rep(B)) \parallel h(rep(C)))$.

Дополним доказательство следующей теоремы из [2] для CSNQ-грамматик:

Теорема 2.
Пусть G — монотонная грамматика и найдены репрезентативы для символов G, удовлетворяющие свойству монотонности. Пусть A — символ грамматики G, и a — система LTS, полученная из A с по правилам G. Тогда $h(rep(A)) \succeq a$.

Доказательство. Докажем, что $h(rep(A)) \succeq h(a)$. Поскольку $h(a) \succeq a$, результат следует из транзитивности. Пусть $A \Rightarrow^k a$, т.е. a выводимо из A за k шагов. Индукция по k.

$[(k = 0)]$ Доказано в [2].

$[(k = 1)]$ Пусть A, B — квази-терминалы в правиле $A, B \longrightarrow t(A) \cup t(B)$ и при этом $a = t(A) \cup t(B)$. Тогда результат следует из свойства монотонности и леммы 1.

$[(k \geq 1)]$ Доказано в [2]. ■

Метод верификации совпадает с методом в [2], поскольку верны аналогичные теоремы, необходимые для его обоснования. Пусть задана монотонная грамматика G и φ — формула $\forall CTL$ с атомарными формулами $D_1, ..., D_k$. Чтобы проверить, что каждая LTS, выводимая в грамматике G, удовлетворяет φ, необходимо выполнить следующие шаги:

1. Для каждого символа A в G выбрать репрезентативный процесс $rep(A)$ и построить абстрактную систему LTS $h(rep(A))$ в соответствии с формулами $D_1, ..., D_k$.

2. Проверить, что множество репрезентативов удовлетворяет свойству монотонности. Теорема 2 влечет, что для каждого a, выводимого в грамматике G, верно $h(rep(S)) \succeq a$.

3. Выполнить проверку моделей на $h(rep(S))$ для спецификации φ. По теореме 1, если $h(rep(S)) \models \varphi$, то для всех систем LTS M, выводимых в грамматике G, верно $M \models \varphi$.

Для поиска монотонных репрезентативов используется алгоритм из [2], с $\{t(A)\}$ в качестве начального репрезентативного множества каждого квази-терминала A.

3. Примеры

В данном разделе мы приводим определения грамматик, задающих мультиагентную систему разрешения конфликтов и подрешеток, задаем базовые процессы порождаемых систем и формулы $\forall CTL$, выражающие свойства этих систем. Отметим, что порождающая грамматика для систем разрешения конфликтов имеет конечные множества квази-терминалов и правил, в отличие от грамматики для подрешеток. Описание построения конечных детерминированных автоматов, являющихся базисом для абстрактных функций состояний порождаемых систем, и согласованных репрезентативов для символов грамматик выходит за рамки данной статьи.

3.1. Мультиагентная система разрешения конфликтов

Подробное описание мультиагентного алгоритма разрешения конфликтов при пополнении онтологии приведено в [9], а соответствующая ему специализированная система разрешения неоднозначностей описана в [8]. В этой статье мы кратко опишем коммуникационную структуру системы агентов, без рассмотрения действий агентов по обработке входящих сообщений.

Пусть задано множество *агентов-участников*. Некоторые из агентов находятся *в конфликте*, соответствующем некоторой неоднозначности. *Агент-мастер* конструирует конфликтно-свободное множество агентов, учитывая интегрированность конфликтных агентов в систему. Эта интегрированность оценивается посредством

вычисления весов и конфликтных весов агентов. Конфликт разрешается через удаление либо изменение слабых агентов. Агент-мастер выполняет основной протокол, конструируя конфликтно-свободное множество, тогда как остальные агенты выполняют протоколы вычисления весов.

Каждый агент-участник соединен с мастером двусторонним каналом. Агенты-участники связаны друг с другом помеченными связями типов, соответствующих их реакции на конфликт: удаление (rem-тип) и обновление (upd-тип). Каждая помеченная связь ациклична. Обрабатывание каждой конфликтной реакции, индуцированной определенной связью, рассматривается как отдельный базовый процесс. Агент-участник может быть объединением таких процессов. Этот факт задает структуру грамматики, порождающей семейство наших мультиагентных систем для разного числа агентов, связанных друг с другом различным образом. Агенты соединены двусторонними каналами, соответствующими их помеченным связям. Такая структура мультиагентной сети порождается следующей контекстно-зависимой грамматикой с квази-терминалами $G = (T, Qt, t, N, P, S)$. Пусть задано множество связей $C = \{c_1, ..., c_n\}$ и каждая связь c_i^k имеет конфликтный тип $k \in \{rem, upd\}$. Тогда для грамматики G:

– терминалы $T = \{master\} \cup \bigcup_{i=1}^{n} \{root_i, inter_i, leaf_i\} \cup vrtxs^i$, при этом $vrtxs^i = \{vrtx \mid vrtx = inter_j$ или $vrtx = leaf_j, j \in [1..n]\}$ и $|vrtxs^i| = i$,

– квази-терминалы $Qt = \bigcup_{i=1}^{n} \{Inter_i, Leaf_i\}$,

– ассоциативное отображение $t : Qt \mapsto T$ определяется как $t(Inter_i) = inter_i$, и $t(Leaf_i) = leaf_i$ для каждого $i \in [1..n]$,

– нетерминалы $N = \{S\} \bigcup_{i=1}^{n} \{ROOT_i, SUB_i\}$;

– множество правил вывода P для каждого $i \in [1..n]$:

1. $S \longrightarrow master \parallel_m ROOT_1 \parallel_m \ldots \parallel_m ROOT_n$
2. $ROOT_i \longrightarrow (ROOT_i \parallel_{c_i^k} SUB_i) \bigvee (root_i \parallel_{c_i^k} SUB_i)$
3. $SUB_i \longrightarrow (SUB_i \parallel_{c_i^k} SUB_i) \bigvee (Inter_i \parallel_{c_i^k} SUB_i) \bigvee$
$(SUB_i \parallel_{c_i^k} Leaf_i) \bigvee (Inter_i \parallel_{c_i^k} Leaf_i) \bigvee$
$(inter_i \parallel_{c_i^k} SUB_i) \bigvee (SUB_i \parallel_{c_i^k} leaf_i) \bigvee$
$(inter_i \parallel_{c_i^k} Leaf_i) \bigvee (Inter_i \parallel_{c_i^k} leaf_i) \bigvee (inter_i \parallel_{c_i^k} leaf_i)$
4. $\{V_1, ..., V_m\} \longrightarrow t(V_1) \cup ... \cup t(V_m) = vrtx^m$, где для каждого $j \in [1..m]$ $V_j \in \{Inter_i, Leaf_i\}$, и если $V_j = Inter_i$, то для каждого $l \in [1..m]$ верно $V_l \neq Leaf_i$ $(i \in [1..n])$.

Неформально, правила грамматики позволяют построить ациклические деревья связей с произвольным числом потомков, а затем слить между собой их вершины. Параллельная композиция агентов-процессов синхронна. Протоколы вычислений весов обладают высокой степенью параллелизма, поэтому очень важно доказать их завершаемость и корректную синхронизацию. Выполнимость этих свойств необходима для корректного вычисления весов. Запуск этих вычислений может быть смоделирован пересылкой фишек.

Состояния каждого базового процесса определяются значениями следующих переменных состояний:

– $Name : int$ — имя процесса;

– $Channel$: множество $\{name : int;\ c\_type : bool;\ dir : bool;\ agn : int;\ rmvd : bool\}$, где $name$ — метка связи, c_type ее тип, dir — направление: потомок ($dir = 0$) или предок ($dir = 1$) с именем agn, и $rmvd$ — статус отсутствия;

– $Rmvd:\ bool$ — статус отсутствия;
– $Active:\ bool$ — статус активности;
– $WasActive:\ bool$ — статус предыдущей активности.

Входные и выходные каналы базового процесса соответствуют именам, типам и направлениям $Channel$. При синхронной композиции базовых процессов с различными именами соответствующие каналы с одинаковыми именами объединяются. При слиянии процессов с одним и тем же именем $Name$ множества каналов и множества $Channel$ объединяются. Процессы с разными именами не могут сливаться и процессы с одним и тем же именем $Name$ не могут участвовать в параллельной композиции. Переходы определяются пересылкой и получением фишек через каналы. Начальное состояние — $(i, Channel, 0, 0, 0)$, где $Channel$ — непустое множество каналов с $Channel.rmvd = 0$, число каналов с $dir = 1$ не больше 1 и число каналов с $dir = 0$ может быть равным 0.

Мы хотели бы проверить следующие свойства, выраженные с помощью $\forall CTL$. Для протокола параллельного вычисления весов:
– $\mathbf{AF}(\{wasActive\}^* \wedge \mathbf{AXAF}\{\neg Active\}^*)$
(каждый агент был активен, и после этого все вычисления завершаются).

Для протокола параллельного вычисления конфликтных весов:
– $\mathbf{AF}\{\neg Active\}^*$ (все вычисления завершаются);
– $\mathbf{AG}\{Not2Rmvd\}^*$ (Каналы и агенты не удаляются дважды).

3.2. Подрешетка

Подрешеткой размера $n \times m$ называется граф, в котором не более $n \times m$ вершин, сопоставленных элементам множества $V_G \subseteq \{(i,j) \mid 0 \le i < n, 0 \le j < m\}$ так, что вершины, помеченные парами (i,j) и (i',j'), являются смежными только в том случае, когда $i = i' \wedge j = j' + 1$, или $i = i' \wedge j = j' - 1$, или $i = i' + 1 \wedge j = j'$, или $i = i' - 1 \wedge j = j'$. Нетрудно видеть, что подрешетка размера $n \times m$ — это подграф графа решетки размера $n \times m$, возможно, несвязный.

Сетевая грамматика с квази-терминалами позволяет построить подрешетку посредством задания ее ячеек, каждая из которых является решеткой квази-терминалов 2×2, и последующим слиянием этих ячеек в подрешетку по определенным правилам. Мы выделяем самый левый нижний процесс в подрешетке, который может являться инициатором вычислений, и обозначаем его как ini. Грамматика $G = (T, Qt, t, N, P, S)$, задающая подрешетки SG произвольного размера, определяется следующим образом:

– терминалы $T = \{ini, p_{lb}, p_{rb}, p_{lt}, p_{rt}\}$, где ini — процесс-инициатор и p_* — базовые процессы, ассоциированные с вершинами ячеек с соответствующим направлением каналов,

– квази-терминалы $Qt = \cup_{i \ge 0, j \ge 0}\{l_i b_j, l_i t_j, r_i b_j, r_i t_j\}$ задают вершины ячеек,

– ассоциативное отображение $t : Qt \mapsto T$ определяется для всех $i \ge 0, j \ge 0$ как $t(l_i b_j) = p_{lb}$, $t(l_i t_j) = p_{lt}$, $t(r_i b_j) = p_{rb}$, $t(r_i t_j) = p_{rt}$,

– нетерминалы $N = \{S, ICELL, CELL, IBOT, ITOP, BOT, TOP\}$;

– множество правил вывода P. Мы считаем, что 1) функция $\|_0$ задает параллельную композицию подсетей, не связанных коммуникационными каналами; 2) функция $\|_v$ задает параллельную композицию двух подсетей, состоящих из двух

процессов и попарно связанных двумя "вертикальными" каналами v; 3) функция $\|_g$ задает параллельную композицию двух процессов, связанных "горизонтальными" каналами g; 4) функция слияния $\cup$ объединяет множество каналов сливающихся процессов.

- Правила задания ячеек:
  1. $S \longrightarrow ICELL \|_0 CELL$;
  2. $ICELL \longrightarrow ITOP \|_v IBOT$;
  3. $ITOP \longrightarrow l_0t_0 \|_g r_0t_0$;
  4. $IBOT \longrightarrow ini \|_g r_0b_0$;
  5. $CELL \longrightarrow (CELL \|_0 CELL) \bigvee (TOP \|_v BOT)$;
  6. $TOP \longrightarrow \bigvee_{i\geq 0,j\geq 1}((l_it_j \|_g r_it_j) \vee (l_it_j \|_g p_{rt}) \vee (p_{lt} \|_g r_it_j)) \bigvee (p_{lt} \|_g p_{rt})$;
  7. $BOT \longrightarrow \bigvee_{i\geq 1,j\geq 0}((l_ib_j \|_g r_ib_j) \vee (l_ib_j \|_g p_{rb}) \vee (p_{lb} \|_g r_ib_j)) \bigvee (p_{lb} \|_g p_{rb})$.
- Правила склеивания вершин ячеек:
  1. $\{r_{i-1}t_{j-1}, l_it_{j-1}, r_{i-1}b_j, l_ib_j\} \longrightarrow p_{rt} \cup p_{lt} \cup p_{rb} \cup p_{lb}$;
  2. $\{r_{i-1}t_{j-1}, l_it_{j-1}, r_{i-1}b_j\} \longrightarrow p_{rt} \cup p_{lt} \cup p_{rb}$;
  3. $\{r_{i-1}t_{j-1}, l_it_{j-1}, l_ib_j\} \longrightarrow p_{rt} \cup p_{lt} \cup p_{lb}$;
  4. $\{r_{i-1}t_{j-1}, r_{i-1}b_j, l_ib_j\} \longrightarrow p_{rt} \cup p_{rb} \cup p_{lb}$;
  5. $\{l_it_{j-1}, r_{i-1}b_j, l_ib_j\} \longrightarrow p_{lt} \cup p_{rb} \cup p_{lb}$;
  6. $\{r_{i-1}b_j, l_ib_j\} \longrightarrow p_{rb} \cup p_{lb}$;
  7. $\{r_{i-1}t_j, l_it_j\} \longrightarrow p_{rt} \cup p_{lt}$;
  8. $\{l_it_{j-1}, l_ib_j\} \longrightarrow p_{lt} \cup p_{lb}$;
  9. $\{r_it_{j-1}, r_ib_j\} \longrightarrow p_{rt} \cup p_{rb}$.

Пусть необходимо подсчитать четность числа активных процессов подрешетки SG, достижимых из процесса-инициатора. Эта задача может решаться посредством простого распределенного эхо-алгоритма [12]. Пусть $Dir = \{down, up, left, right\}$ — множество направлений. Состояния каждого базового процесса определяются значениями следующих переменных состояний.

– $Channels:\ set\ of\ Dir$ — множество каналов;
– $Nb:\ Dir$ — имя соседа;
– $Nbs:\ set\ of\ Dir$ — смежные процессы;
– $Active:\ bool$ — статус активности;
– $Token:\ bool$ — четность активности.

Входные и выходные каналы базового процесса соответствуют направлениям каналов $Channels$. При синхронной композиции базовых процессов соединяются каналы $down$ и up, $left$ и $right$. При слиянии процессов множества каналов и множества $Channels$ объединяются. Переходы определяются пересылкой и получением фишек через каналы. Начальное состояние — $(Ch_0, null, Nbs_0, a, 0)$, где Ch_0 и Nbs_0 — непустые множества каналов, и статус активности $a \in \{0, 1\}$. *Процесс-инициатор* ini посылает одновременно всем смежным процессам из $Nbs_{ini} = \{up, right\}$ фишки со своим значением активности $Active_{ini}$: $Token_{ini} := Active_{ini}$. При получении от *процесса-соседа* $Nb_p \in Dir$ фишки $Token_{Nb}$ процесс p, ранее ее не получавший, добавляет к ее значению свое значение активности $Token_p := Token_{Nb} \oplus Active_p$, отсылает фишки с этим значением далее каждому смежному процессу, кроме соседа: $q \in Nbs \setminus \{Nb\}$, и ожидает их возвращения. Получая фишку от смежного процесса q, процесс p учитывает ее значение $Token_p := Token_p \oplus Token_q$ и удаляет q из Nbs_p. Когда множество смежных процессов Nbs_p становится пустым, это

означает, что все фишки вернулись, и процесс возвращает полученный результат $Token_p$ соседу Nb_p. При повторном получении фишки ($Nb_p \neq null$) процесс возвращает отправителю в качестве ее значения значение 0, не меняющее четности включенных процессов. Отметим, что в описании этого алгоритма имена процессов p и q употребляются для читаемости описания, при этом в реальном алгоритме каждому процессу для выполнения действий достаточно имен его каналов. Сумма значений фишек, вернувшихся к процессу-инициатору, будет значением четности числа достижимых активных процессов, что выражается следующим равенством (CV): $Token_{ini} \oplus \bigoplus_{p \in SG} Token_p = 0$. Корректность этого вычисления выражается с помощью следующей формулы $\forall CTL$: $\mathbf{AG}(End \rightarrow CorVal)$, где End — конечный автомат для строк состояний, в которых в первом состоянии строки $Nb_{ini} = \emptyset$, а остальные состояния произвольны, а $CorVal$ — автомат, допускающий строки состояний, удовлетворяющие равенству CV.

4. Заключение

В данной статье представлен метод верификации для семейств распределенных систем, заданных контекстно-зависимой грамматикой с квази-терминалами. Этот метод может быть использован для верификации мультиагентной системы разрешения неоднозначностей при пополнении онтологий. Свойства системы выражены формулами логики $\forall CTL$.

Мы планируем реализовать предложенный метод с использованием инструмента проверки моделей SPIN и дать формальное обоснование корректности мультиагентного алгоритма разрешения неоднозначностей. Однако некоторые свойства, касающиеся взаимодействия агентов, выражаются в рамках такого подхода довольно неэффективно. Это служит причиной для попыток использования более выразительных формализмов. Еще одним направлением дальнейших исследований является развитие предложенного метода для других типов контекстно-зависимых грамматик, в частности, для грамматики с квази-терминалами в комбинации с индексированной грамматикой [10] или грамматикой с управляемыми правилами [4], с помощью которых можно задавать произвольные точные решетки.

Список литературы / References

[1] Bergenti F., Franchi E., Poggi A., “Selected models for agent-based simulation of social networks”, *Proc. of 3rd Symposium on Social Networks and Multiagent Systems (SNAMAS 2011)*, 2011, 27–32.

[2] Clarke E. M., Grumberg O., Jha S., “Verifying Parameterized Networks”, *ACM Transactions on Programming Languages and Systems*, **19**:5 (1997), 726–750.

[3] Clarke E. M., Grumberg O., Peled D., *Model Checking*, MIT Press, 1999.

[4] Dassow J., “Grammars With Regulated Rewriting”, *Formal Languages and Applications*, Studies in Fuzziness and Soft Computing, **148**, 2004, 249–273.

[5] De Gennaro M. C., Jadbabaie A., “Decentralized Control of Connectivity for Multi-Agent Systems”, *Proc. of 45th IEEE Conference on Decision and Control*, 2006, 3628–3633.

[6] Fagin R., Halpern J. Y., Moses Y., Vardi M. Y., *Reasoning about Knowledge*, MIT Press, 1995.

[7] Garanina N. O., Sidorova E .A., "Ontology Population as Algebraic Information System Processing Based on Multi-agent Natural Language Text Analysis Algorithms", *Programming and Computer Software*, **41**:3 (2015), 140–148.

[8] Garanina N., Sidorova E., "An Approach to Ambiguity Resolution for Ontology Population", *Proc. of 24th International Workshop on Concurrency, Specification, and Programming*, (CS&P 2015) (Rzeszow, Poland, 2015, Sep. 28–30), **1**, 2015, 27–32.

[9] Garanina N. O., Sidorova E. A., Anokhin S. A., "Conflict Resolution in Multi-agent Systems with Typed Connections for Ontology Population", *Perspectives of System Informatics*, Lecture Notes in Computer Science, **9609**, 2016, 116–129.

[10] Hopcroft J. E., Ullman J. D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[11] Huhns M. N., Stephens L. M., "Multiagent Systems and Societies of Agents", *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, 1999, 79–120.

[12] Tel G., *Introduction to Distributed Algorithms*, Cambridge University Press, 2000.

[13] Wooldridge M., *An Introduction to Multiagent Systems*, Willey&Sons Ltd, 2002.

Garanina N. O., Sidorova E.A., "An Approach to Verification of a Family of Multi-agent Systems for Conflict Resolution", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 703–714.

DOI: 10.18255/1818-1015-2016-6-703-714

Abstract. In the paper, we describe a verification method for families of distributed systems generated by context-sensitive network grammar of a special kind. This grammar includes special non-terminal symbols, so called quasi-terminals, which uniquely correspond to grammar terminals. These quasi-terminals specify processes which are merging of base system processes, in contrast to simple nonterminals which specify networks of parallel compositions of the processes. The method is based on model checking technique and abstraction. An abstract representative model for a family of systems depends on their specification grammar and system properties to be verified. This model simulates the behaviour of the systems in such a way that the properties which hold for the representative model are satisfied for all these systems. The properties of the representative model can be verified by model checking method. The properties of a generated system are specified by universal branching time logic $\forall CTL$ with finite deterministic automata as atomic formulas. We show the use of this method for verification of some properties of a multiagent system for conflict resolution, in particular, for context-dependent disambiguation in ontology population. We also suggest that this approach should be used for verification of computations on sub-grids which are sub-graphs of computation grids. In particular, we consider the computation of parity of the active processes number in a sub-grid.

Keywords: model checking, context-sensitive network grammar, multi-agent systems, abstractions

About the authors:
Natalia O. Garanina, orcid.org/0000-0001-9734-3808, PhD,
A.P. Ershov Institute of Informatics Systems, SB RAS
6 Acad. Lavrentjev pr., Novosibirsk 630090, Russia, e-mail: garanina@iis.nsk.su

Elena A. Sidorova, orcid.org/0000-0001-8731-3058, PhD,
A.P. Ershov Institute of Informatics Systems, SB RAS
6 Acad. Lavrentjev pr., Novosibirsk 630090, Russia, e-mail: lena@iis.nsk.su

Acknowledgments:
The research has been supported by Russian Foundation for Basic Research (grant 15-07-04144) and Siberian Branch of Russian Academy of Science (Integration Grant n.15/10 "Mathematical and Methodological Aspects of Intellectual Information Systems").

Моделирование и анализ информационных систем. Т. 23, № 6 (2016), с. 715–728
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 715–728

DOI: 10.18255/1818-1015-2016-6-715-728

УДК 519.713

Построение каскадной параллельной композиции временных автоматов с использованием BALM-II

Громов М. Л., Шабалдина Н. В.

получена 5 сентября 2016

Аннотация. В данной работе мы рассмотрели задачу построения каскадной параллельной композиции временных автоматов. Построение такой композиции можно свести к поэтапному построению бинарной параллельной композиции. Известно, что если каждая из компонент бинарной параллельной композиции есть временной автомат с константными задержками выходов, то результатом композиции может быть временной автомат, множество задержек выходных символов которого бесконечно и задано при помощи конечного множества линейных функций. Поэтому задача построения каскадной композиции временных автоматов с константными задержками выходов сводится к построению ряда бинарных параллельных композиций временных автоматов, задержки выходов которых заданы либо в виде констант, либо в виде множества линейных функций. В данной работе мы уточняем определение временного автомата, обращая особое внимание на описание задержки выходного символа. В качестве инструмента для построения композиции мы используем BALM-II, и поэтому рассматриваем переход от временного автомата с задержками выходов в виде множества линейных функций к соответствующему полуавтомату. Мы предлагаем свою процедуру построения полуавтомата, которая, в отличие от известной процедуры, не требует последующей детерминизации полученного полуавтомата. Кроме того, мы пошагово описываем, каким образом построить композицию соответствующих полуавтоматов при помощи BALM-II, а также обсуждаем процедуру обратного преобразования от полуавтомата композиции к временному автомату, отмечая некоторые нюансы, связанные с композицией временных автоматов с задержками выходов в виде множества линейных функций. В работе приведён пример, иллюстрирующий построение каскадной параллельной композиции временных автоматов.

Ключевые слова: временные автоматы, каскадная параллельная композиция, BALM-II

Для цитирования: Громов М. Л., Шабалдина Н. В., "Построение каскадной параллельной композиции временных автоматов с использованием BALM-II", *Моделирование и анализ информационных систем*, **23**:6 (2016), 715–728.

Об авторах:
Громов Максим Леонидович, orcid.org/0000-0002-2990-8245, канд. физ.-мат. наук,
Томский государственный университет,
пр. Ленина, 36, г. Томск, 634050 Россия, e-mail: maxim.leo.gromov@gmail.com

Шабалдина Наталия Владимировна, orcid.org/0000-0002-1958-550X, канд. техн. наук, доцент,
Томский государственный университет,
пр. Ленина, 36, г. Томск, 634050 Россия, e-mail: nataliamailbox@mail.ru

Благодарности:
Работа частично выполнена при финансовой поддержке гранта на проведение фундаментальных научных исследований и поисковых научных исследований № 16-49-03012 Российского научного фонда.

Введение

Большинство современных приложений, таких как веб-сервисы, телекоммуникационные протоколы и т.п., ориентированы на взаимодействие друг с другом. Классической моделью описания поведения дискретных систем является модель конечного автомата [1, 2]. Если поведение каждой системы описывается при помощи конечного автомата, то совместная работа таких систем может быть описана автоматной композицией, которая (при определённых ограничениях) также будет являться конечным автоматом. В данной работе нас интересует так называемая параллельная композиция [3], в которой компоненты взаимодействуют асинхронно в предположении «медленной внешней среды». Для построения такой автоматной композиции есть инструмент BALM-II (Berkeley Automata and Language Manipulation) [4]. Отметим, что мы рассматриваем не бинарную композицию, а каскадную, содержащую более двух компонент.

Иногда бывает необходимо учесть временные аспекты поведения дискретной системы. Вероятно, наиболее общий способ описания поведения такого рода систем – это временной полуавтомат [5]. Однако в нашей работе нас интересуют входо-выходные системы, в которых за каждым входным воздействием обязательно следует выходная реакция, возможно, спустя некоторое время. Класс таких систем был упомянут выше, он включает в себя телекоммуникационные протоколы, последовательностные схемы, веб-сервисы и т.п. В этом случае мы можем использовать в качестве модели временной конечный автомат. Известны различные модели временного конечного автомата, например, с интервалами на переходах [6]. В данной работе мы рассматриваем модель временного конечного автомата с задержками выходов и таймаутами [7, 8]. В данной работе, как и в работе [11], мы продолжаем исследования, начатые в [7], в которой авторы предложили метод построения бинарной параллельной композиции временных конечных автоматов с константными задержками выходов и таймаутами. Для построения композиции двух временных автоматов сначала необходимо перейти к соответствующим полуавтоматам [7]. Таким образом, сначала мы преобразуем оба временных автомата в полуавтоматы, затем строим композицию двух полуавтоматов, и затем нам необходимо вернуться к модели временного конечного автомата. В [7] показано, что композиция двух временных конечных автоматов может иметь бесконечное число задержек выходов для некоторых переходов и эти задержки могут быть описаны конечным способом при помощи следующего множества линейных функций: $\{b + kt | b, k \in \{0\} \cup \mathbb{N}\}$.

Существует ряд инструментов для работы с временными полуавтоматами, в том числе для построения композиций, а также для верификации. Среди этих инструментов наиболее популярным является UPPAAL [9]. В работе [11] мы поясняем, почему для наших целей этот инструмент не подходит, и аргументируем выбор инструмента BALM-II. В данной работе мы продолжаем наши исследования и рассматриваем построение каскадной параллельной композиции временных автоматов с использованием BALM-II.

1. Основные определения и обозначения

Полуавтомат A есть пятёрка $\langle A, X, \lambda_{\mathsf{A}}, \hat{a}, \mathfrak{F}_{\mathsf{A}}\rangle$, где A – конечное множество состояний с выделенным начальным состоянием $\hat{a}$ и множеством финальных состояний $\mathfrak{F}_{\mathsf{A}} \subseteq A$; X – алфавит действий; $\lambda_{\mathsf{A}} \subseteq A \times X \times A$ – отношение переходов, определяющее все возможные переходы в полуавтомате. Язык $\mathfrak{L}_{\mathsf{A}}$ полуавтомата A есть множество всех последовательностей α в алфавите X, таких, что в полуавтомате A есть цепочка переходов (помеченная последовательностью α) из начального состояния в некоторое финальное состояние. Автомат G есть пятёрка $\langle G, I, O, \lambda_{\mathsf{G}}, \hat{g}\rangle$, где G есть конечное непустое множество состояний с выделенным начальным состоянием $\hat{g}$; I и O – входной и выходной алфавиты, соответственно; и $\lambda_{\mathsf{G}} \subseteq G \times I \times O \times G$ – отношение переходов. В автомате все состояния являются финальными.

Пусть $\mathbb{N}$ – множество натуральных чисел, а $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ – множество целых неотрицательных чисел. Обозначим через $\mathbb{F} \stackrel{\text{def}}{=} \{b + kt | b \in \mathbb{N}_0, k \in \mathbb{N}_0\}$ множество всех возможных линейных функций от аргумента t с целыми неотрицательными коэффициентами. Пусть задана некоторая линейная функция $f_1(t) = b_1 + k_1 t$ из множества $\mathbb{F}$. Очевидно, придавая аргументу t этой функции последовательно целые неотрицательные значения (то есть значения 0, 1, 2 и т.д.), получим в результате множество целых неотрицательных чисел: b_1, $b_1 + k_1$, $b_1 + 2k_1$ и т.д. Обозначим это множество как $\mathbb{T}(f_1)$ и назовём его *множество моментов времени функции* $f_1(t)$. Пусть задано конечное множество $D = \{f_1(t), f_2(t), \ldots, f_n(t)\} \subset \mathbb{F}$ линейных функций с целыми неотрицательными коэффициентами. *Множеством моментов времени множества* D назовём следующее множество: $\mathbb{T}(D) \stackrel{\text{def}}{=} \cup_{i=1}^{n} \mathbb{T}(f_i)$.

В данной работе под *временным конечным автоматом* мы понимаем автомат с задержками выходов, то есть кортеж вида $\mathsf{S} = \langle S, I, O, \lambda_{\mathsf{S}}, \hat{s}, \sigma_{\mathsf{S}}\rangle$, для которого выполняются следующие условия:

1. Пятёрка $\langle S, I, O, \lambda_{\mathsf{S}}, \hat{s}\rangle$ есть конечный автомат.

2. Каждый временной автомат оборудован внутренними часами (временно́й переменной), которые меняют своё значение с течением времени и пригодны для измерения дискретных промежутков времени: 1 такт (единица времени), 2 такта и т.д. В этой работе мы полагаем, что выполнение всякого перехода приводит к сбросу внутренних часов в 0.

3. Функция $\sigma_{\mathsf{S}} : \lambda_{\mathsf{S}} \to (2^{\mathbb{F}} \setminus \varnothing)$ есть функция задержки выходного символа, которая определяет для каждого перехода непустое, возможно, бесконечное множество допустимых моментов времени, в которые автоматом может быть произведён выходной символ. Это множество моментов времени задаётся конечным набором линейных функций с целыми неотрицательными коэффициентами. Если $\sigma_{\mathsf{S}}(\langle s_1, i, o, s_2\rangle) = D$, где $D \subset \mathbb{F}$ – конечное множество линейных функций с целыми неотрицательными коэффициентами, то, находясь в состоянии s_1 и получив входное воздействие i, автомат сбрасывает значение внутренних часов в 0 и может выдать выходную реакцию o на это воздействие в любой момент, когда внутренние часы приобретают значение из $\mathbb{T}(D)$. В этот же самый момент состояние автомата станет s_2, а часы вновь сбросятся в 0.

4. Временной автомат начинает свою работу из начального состояния, а внутренние часы установлены в 0.

Введённый нами временной автомат является расширением временного автомата с константными задержками выходов, рассмотренного нами ранее [11], в котором функция задержки выходного символа определяется так: $\sigma_S : \lambda_S \to \mathbb{N}_0$. Получить указанную выше модель достаточно просто, нужно потребовать равенство нулю всех коэффициентов k у всех линейных функций.

В работе [7] показано, что при построении параллельной композиции временных автоматов с константными задержками выходов, в случае их недетерминизма, получается временной автомат, в котором задержки выходов на переходах описываются значениями конечного набора целочисленных неотрицательных линейных функций. Поэтому в данном исследовании мы сразу отталкиваемся от такой, более общей модели, учитывая, что в работе [7] показана замкнутость множества таких временных автоматов относительно операции параллельной композиции. Рассмотрение более общей модели актуально в связи с тем, что на практике встречаются не только бинарные композиции, но и композиции, содержащие более двух компонент, построение которых может быть сведено к каскадному построению бинарных композиций. Тогда, даже если все автоматы, описывающие поведение компонент, являются временными автоматами с константными задержками выходов, на очередном шаге построения такой композиции может возникнуть задача построения бинарной композиции временных автоматов, в которых задержки выходов на переходах описываются значениями конечного набора целочисленных неотрицательных линейных функций.

Для описания поведения временного автомата с учётом временных аспектов вводится понятие *временно́го выходного символа* (*временно́й реакции* автомата).

Полагаем, что внешний наблюдатель, проводящий эксперименты с временным автоматом, имеет доступ к внешним, *глобальным* часам (глобальной временной переменной), по которым и проводит все измерения временных промежутков. Наблюдатель может сбрасывать глобальные часы. В момент, когда начинаются эксперименты с временным автоматом, значение глобальной временной переменной равно 0. Кроме того, полагаем, что внешний наблюдатель достаточно осведомлен о внутренних часах временного автомата и настроил глобальные часы так, чтобы единица времени, измеренная по глобальным часам, соответствовала единице времени, измеренной по внутренним часам временного автомата. Подавая входные воздействия на автомат, наблюдатель сбрасывает глобальные часы в 0, «засекая», сколько времени будет «готовиться» выходная реакция.

Временной выходной символ (*временная выходная реакция*) есть пара $\langle o, u \rangle \in \in O \times \mathbb{N}_0$, указывающая на то, что выходная реакция o получена от автомата в момент, когда значение глобальной временной переменной равно u.

Заметим, что такое определение выходной реакции отличается от классически принятого определения наблюдаемых временных событий (см., например [5]), в котором внешний наблюдатель никогда не сбрасывает значение глобальной временной переменной.

Множество возможных временных реакций временного автомата S, находящегося в состоянии s, на входное воздействие $i \in I$ определяется переходами из этого состояния по входному символу i и функциями задержек выходов σ_S этих перехо-

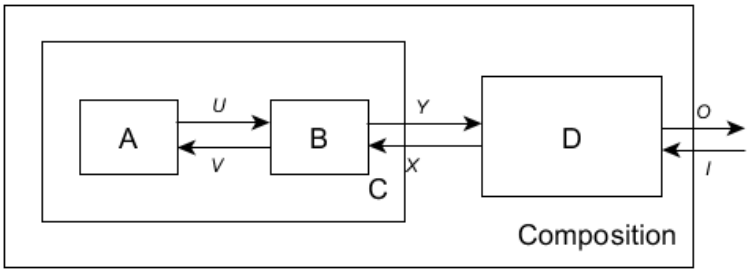


Рис. 1: Структура каскадной параллельной композиции
Fig. 1. Structure of cascade parallel composition

дов. Пусть в состоянии s по входу i в автомате определены переходы $\tau_1 = \langle s, i, o_1, s_1\rangle$, $\ldots$, $\tau_q = \langle s, i, o_q, s_q\rangle$ и пусть $\sigma_S(\tau_1) = D_1$, $\ldots$, $\sigma_S(\tau_q) = D_q$. Тогда наблюдатель в качестве реакции может получить выходной символ o_1 в один из моментов времени из множества $\mathbb{T}(D_1)$, либо выходной символ o_2 в один из моментов времени из множества $\mathbb{T}(D_2)$ и т.д. Всё множество возможных реакций автомата S, находящегося в состоянии s, на входное воздействие i опишется как $\cup_{j=1}^{q}\{o_j\} \times \mathbb{T}(D_j)$.

Рассмотрим входную последовательность $\alpha = i_1 i_2 \ldots i_n$ и временную выходную последовательность $\beta = \langle o_1, u_1\rangle\langle o_2, u_2\rangle \ldots \langle o_n, u_n\rangle$. Если во временном автомате S существует такая последовательность переходов, что при подаче в состоянии s последовательности α автомат может произвести в ответ последовательность β, то последовательность $i_1/\langle o_1, u_1\rangle \ldots i_n/\langle o_n, u_n\rangle$ называется *временной входо-выходной последовательностью* автомата S в состоянии s. Множество всех временных входо-выходных последовательностей автомата S в состоянии s обозначим как $\mathfrak{L}_S(s)$ и назовём языком временного автомата S в состоянии s. Язык временного автомата в начальном состоянии будем просто называть *языком временного автомата* и обозначать $\mathfrak{L}_S$.

Параллельная композиция описывает диалог между двумя компонентами. Структура рассматриваемой в данной работе каскадной параллельной композиции представлена на рисунке 1. Построение такой композиции может быть сведено к построению бинарной параллельной композиции компонент A и B (нахождению компоненты C), а затем построению бинарной параллельной композиции компонент C и D (нахождению поведения системы в целом). Мы предполагаем, что компоненты взаимодействуют между собой в условиях «медленной внешней среды», т.е. следующий входной символ может быть подан на систему только после выдачи системой внешнего выходного символа на предыдущий внешний входной символ. Мы также предполагаем для простоты изложения, что алфавиты различных каналов не пересекаются, и что при работе системы не возникает бесконечных внутренних диалогов (осцилляций). Ещё одно наше предположение состоит в том, что каждая компонента, а также система в целом имеют временные переменные. Значения этих переменных наращиваются синхронно, с одинаковой скоростью. Временная переменная компоненты обнуляется (сбрасывается), когда компонента меняет состояние или когда на вход компоненты поступает входной символ. Временная переменная композиции сбрасываются, когда на систему подаётся входной символ или когда си-

стема (композиция) выдаёт внешний выходной символ. Подробное описание работы компонент и системы в целом можно найти в статье [7].

2. Построение полуавтомата, соответствующего заданному временному конечному автомату

Параллельная композиция в классическом случае строится через композицию соответствующих полуавтоматов [4]. Подобным же образом будем поступать и мы. Для этого мы опишем то, как построить по заданному временному автомату соответствующий полуавтомат. Ключевым моментом в таком построении является введение в алфавите полуавтомата специального символа 1 [10], который трактуется как «истечение» одного такта (одной единицы) времени. Это возможно, поскольку в нашей модели воздействия могут подаваться лишь в дискретные моменты времени (точнее, автомат считывает их только в эти моменты времени). Что касается выходных реакций, они также выдаются только в дискретные моменты времени. Такая модель поведения, конечно же, является частным случаем поведения временного полуавтомата, предложенного в классической работе Алюра и Дилла [5], однако отметим, что в работе [12] Спринхинтфельд и др. показали, что, например, для вопросов тестирования оказывается достаточно рассмотреть поведение временного полуавтомата Алюра и Дилла в дискретные моменты времени, кратные специально рассчитываемой величине. Поэтому введение тактирования входных воздействий и выходных реакций мы считаем вполне оправданным.

Далее задача построения полуавтомата, соответствующего временному автомату, становится достаточно очевидной, нужно лишь показать, как развернуть переход временного автомата в переходы полуавтомата. Однако подходов к реализации такой «развёртки» может быть несколько. В данном разделе мы кратко описываем подход работы [7], опирающийся на использование отдельных «маленьких» полуавтоматов для каждого отдельного перехода, которые затем объединяются, и предлагаем свой подход, который позволяет описать полуавтоматом сразу все переходы временного автомата для заданного состояния и заданного входного символа.

2.1. Применение полуавтоматов вида $1^b.(1^k)^*$

В данной работе мы рассматриваем временные автоматы, у которых задержки появления выходной реакции на выходе описываются множеством функций вида $b + kt$. Для отображения переходов временного автомата в переходы полуавтомата в работе [7] предлагается строить специальный полуавтомат с формулой языка $i.1^b.(1^k)^*.o$ для каждой из этих функций и затем объединять эти полуавтоматы. Однако такой подход приводит к появлению нежелательного недетерминизма. В итоге полученный полуавтомат придётся детерминизировать, что может оказаться достаточно накладным в случае большого количества линейных функций в множестве, описывающем задержки выхода. Особенно это становится заметным, если автомат недетерминированный (по одному и тому же входному символу из одного и того же состояния в автомате имеются несколько различных переходов).

2.2. Непосредственное построение полуавтомата задержек

В данном подразделе мы предлагаем процедуру, которая позволит построить полуавтомат-задержку сразу, опираясь только на коэффициенты заданных линейных функций.

Пусть задан некоторый временной автомат $\mathsf{S} = \langle S, I, O, \lambda_{\mathsf{S}}, \hat{s}, \sigma_{\mathsf{S}} \rangle$. Рассмотрим в нём множество переходов $\{\tau_j = \langle s, i, o_j, s'_j \rangle : j = 1, \ldots, n\}$ из некоторого состояния s по некоторому входному символу i, и пусть $\sigma_{\mathsf{S}}(\tau_j) = \{b_{j,h_j} + k_{j,h_j} t \,|\, h_j = 1, \ldots, N_j\}$ – множество функций задержки выхода o_j при переходе автомата из состояния s в состояние s'_j по входному символу i. При этом, для упрощения рассмотрения, полагаем, что все $k_{j,h_j} > 0$. Обозначим через $\mathbb{B} = \max(b_{j,h_j})$ – наибольший из коэффициентов b_{j,h_j}, $j = 1, \ldots, n$, $h_j = 1, \ldots, N_j$, а через $\mathbb{K} = \text{НОК}(k_{j,h_j})$ – наименьшее общее кратное чисел k_{j,h_j}, $j = 1, \ldots, n$, $h_j = 1, \ldots, N_j$. Обозначим через $\mathbb{M} = \mathbb{B} + \mathbb{K} - 1$ – сумму чисел $\mathbb{B}$ и $\mathbb{K}$ за вычетом единицы. Теперь, при построении соответствующего полуавтомата, будем действовать следующим образом:

1. Добавим в множество состояний полуавтомата и в множество финальных состояний полуавтомата состояние s и все состояния s'_j, $j = 1, \ldots, n$.

2. Добавим в множество состояний полуавтомата (но *не* в множество финальных состояний) состояния вида $\langle s, i, m \rangle$, где $m = 0, \ldots, \mathbb{M}$. Число m назовём *номером состояния.*

3. В множество переходов полуавтомата добавим переход $\langle s, i, \langle s, i, 0 \rangle \rangle$.

4. В множество переходов полуавтомата добавим переходы $\langle \langle s, i, m \rangle, 1, \langle s, i, m + 1 \rangle \rangle$, $m = 0, \ldots (\mathbb{M} - 1)$.

5. В множество переходов полуавтомата добавим переход $\langle \langle s, i, \mathbb{M} \rangle, 1, \langle s, i, \mathbb{B} \rangle \rangle$ (замкнём петлю).

6. Перебирая состояния $\langle s, i, m \rangle$ и функции $b_{j,h_j} + k_{j,h_j} t$, проверяем, если число $m - b_{j,h_j}$ неотрицательное и оно нацело делится на k_{j,h_j}, то добавляем переход $\langle \langle s, i, m \rangle, o_j, s'_j \rangle$.

Утверждение 1. *Пусть с использованием описанной выше процедуры построен полуавтомат* $\mathsf{A} = \langle A, X, \lambda_{\mathsf{A}}, \hat{a}, \mathfrak{F}_{\mathsf{A}} \rangle$, *где* $A = \{s, \langle s, i, 0 \rangle, \ldots, \langle s, i, \mathbb{M} \rangle, s'_1, \ldots, s'_n\}$, $X = \{i, 1, o_1, \ldots, o_n\}$, $\hat{a} = s$, $\mathfrak{F}_{\mathsf{A}} = \{s, s'_1, \ldots, s'_n\}$, *а множество переходов* λ_{A} *получено согласно правилам процедуры. Тогда язык полуавтомата* A *содержит пустую последовательность и последовательности, которые описывают все возможные временные входо-выходные пары всех переходов* $\tau_j = \langle s, i, o_j, s'_j \rangle$, $\sigma_{\mathsf{S}}(\tau_j) = D_j$ *заданного временного автомата и только их.*

Доказательство. Наличие в языке полуавтомата A пустой последовательности с очевидностью следует из пункта 1 процедуры.

Для доказательства остальной части утверждения достаточно показать, что любая временная входо-выходная пара, порождаемая переходами $\tau_j = \langle s, i, o_j, s'_j \rangle$, $\sigma_{\mathsf{S}}(\tau_j) = D_j$ заданного автомата, описывается соответствующим словом из языка

полуавтомата и что всякому слову из языка полуавтомата соответствует некоторая временная входо-выходная пара исходного временного автомата.

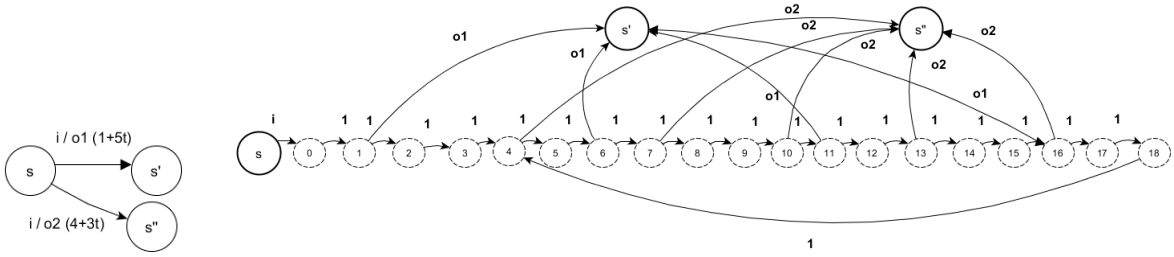
Пусть некоторый переход автомата $\langle s, i, o_j, s'_j \rangle$ может породить временную входо-выходную пару $i/\langle o_j, T \rangle$. Пусть задержка T определяется функцией $b + kt \in D_j$, то есть $T = b + kt_1$, для некоторого целого $t_1 \geqslant 0$.

Пусть $T \leqslant \mathbb{M}$. Рассмотрим состояние полуавтомата с номером T (согласно пункту 2 состояние с таким номером обязательно существует). Попасть в это состояние полуавтомат мог только из состояния $\langle s, i, 0 \rangle$ после выполнения действия 1 T раз. Поскольку $T = b + kt_1$, согласно правилу 6 процедуры ($T - b = kt_1$ – это число не меньше нуля и делится нацело на k) из состояния с этим номером есть переход по действию o_j в состояние s_j. Состояние s_j объявлено финальным. Начальным состоянием является состояние s. Из него возможен единственный переход по действию i в состояние $\langle s, i, 0 \rangle$, из которого после выполнения действия 1 T раз полуавтомат попадает в состояние $\langle s, i, T \rangle$. Далее, из состояния $\langle s, i, T \rangle$ по действию o_j полуавтомат попадает в финальное состояние s_j. Значит, слово $i.1^T.o_j$ принадлежит языку полуавтомата.

Пусть $T > \mathbb{M}$. Отметим, что цикл по действиям 1 в полуавтомате охватывает состояния с номерами от $\mathbb{B}$ (своеобразная «точка входа» в цикл) по $\mathbb{M}$. Из состояния $\mathbb{M}$ по действию 1 полуавтомат возвращается в состояние $\mathbb{B}$. Длина цикла (число переходов по действию 1 до возвращения в состояние, с которого начато движение по циклу) составляет $\mathbb{K}$. Представим число T следующим образом: $T = \mathbb{B} + p \cdot \mathbb{K} + r$, где $0 \leqslant r < \mathbb{K}$ – остаток от деления числа $T - \mathbb{B}$ на $\mathbb{K}$ (очевидно, число $T - \mathbb{B} \geqslant 0$), p – некоторое целое неотрицательное число (результат целочисленного деления числа $T - \mathbb{B}$ на $\mathbb{K}$). Поскольку $T = b + kt_k$, то можно записать $\mathbb{B} + p \cdot \mathbb{K} + r = b + kt_1$. Переписав это выражение иначе, получим $(\mathbb{B} + r - b) + p \cdot \mathbb{K} = kt_1$. Правая часть этого выражения нацело делится на k, а значит, и левая часть тоже нацело делится на k. Так как $\mathbb{K}$ – НОК всех коэффициентов k_{j,h_j}, в том числе и рассматриваемого числа k, то слагаемое $p \cdot \mathbb{K}$ делится на k без остатка, а в таком случае и слагаемое $(\mathbb{B} + r - b)$ также делится на k без остатка. Рассмотрим состояние с номером $\mathbb{B} + r$ (как было отмечено выше, $0 \leqslant r < \mathbb{K}$, а значит, состояние с таким номером существует в полуавтомате). Поскольку это состояние находится внутри цикла, попасть в него из состояния $\langle s, i, 0 \rangle$ можно бесконечным числом способов. Например, это можно сделать следующим способом. Сначала делается $\mathbb{B}$ переходов по 1 и достигается состояние с номером $\mathbb{B}$ – «точка входа» в цикл. Затем делается r переходов и достигается состояние с номером $\mathbb{B} + r$. После этого p раз «прокручивается» цикл (длина которого $\mathbb{K}$ действий) и вновь достигается состояние с номером $\mathbb{B} + r$. Таким образом, это состояние достигается за $\mathbb{B} + r + p \cdot \mathbb{K}$ переходов по действию 1. По доказанному выше, число $(\mathbb{B} + r - b)$ делится на k без остатка, а значит, согласно правилу 6 процедуры из состояния с номером $\mathbb{B} + r$ есть переход по действию o_j в состояние s_j. И вновь слово $i.1^T.o_j$ принадлежит языку полуавтомата.

Обратное доказывается аналогично. Всякое слово языка полуавтомата, за исключением пустого, имеет вид $i.1^T.o_j$, причём среди всех функций из σ_j найдётся такая функция $b + kt$, что число T представимо в виде $T = b + kt_1$ для некоторого неотрицательного целого t_1. А значит, оно описывает некоторую временную входо-выходную пару $i/\langle o_j, T \rangle$ исходного временного автомата. □

Пример 1. *Рассмотрим переходы временного автомата, представленные на рисунке 2a.*



(a) (b)

Рис. 2: Пример временных переходов временного автомата и соответствующий им полуавтомат
Fig. 2. Timed transitions example and corresponding automaton transitions

Для этих переходов $\mathbb{B} = \max(1, 4) = 4$, $\mathbb{K} = (5, 3) = 15$, *тогда* $\mathbb{M} = 4 + 15 - 1 = 18$. *Значит, нефинальных состояний будет* 18, *петля будет замыкаться переходом по символу* 1 *из состояния с номером* $\mathbb{M} = 18$ *в состояние с номером* $\mathbb{B} = 4$ *(рисунок 2b).*

Рассмотрим, например, состояние с номером 16. *Так как* $16 - 1 = 15$ *делится нацело на* 5 *(функция* $1 + 5t$ *описывает задержки выхода* o_1*), то из состояния с номером* 16 *есть переход по символу* o_1 *в финальное состояние* s'.

Аналогично для выхода o_2 *(функция* $4+3t$ *описывает задержки выхода* o_2*): число* $16-4 = 12$ *нацело делится на* 3, *значит, из состояния* 16 *есть переход по символу* o_2 *в финальное состояние* s''.

Рассмотрев все нефинальные состояния, мы получим все возможные переходы (рисунок 2b).

2.3. «Сборка» полуавтомата, описывающего временной автомат

После того как построены все полуавтоматы $\mathsf{A}_j = \langle A_j, X_j, \lambda_{\mathsf{A}_j}, \hat{a}_j, \mathfrak{F}_{\mathsf{A}_j} \rangle$ для каждого перехода $\langle s_j, i_j, o_j, s'_j \rangle$, $j = 1, \ldots, |\lambda_\mathsf{S}|$, временного автомата $\langle S, I, O, \lambda_\mathsf{S}, \hat{s}, \sigma_\mathsf{S} \rangle$ (с помощью любого из указанных выше способов), полуавтомат, описывающий весь временной автомат, собирается следующим образом. Нужно построить следующие множества: $A = \cup_{j=1}^{|\lambda_\mathsf{S}|} A_j$, $X = \cup_{j=1}^{|\lambda_\mathsf{S}|} X_j$, $\lambda_\mathsf{A} = \cup_{j=1}^{|\lambda_\mathsf{S}|} \lambda_{\mathsf{A}_j}$, $\mathfrak{F}_\mathsf{A} = \cup_{j=1}^{|\lambda_\mathsf{S}|} \mathfrak{F}_{\mathsf{A}_j}$. Добавим в множество переходов λ_A для каждого состояния $s \in \mathfrak{F}_\mathsf{A}$ переходы вида $\langle s, 1, s \rangle$ – переход из финального состояния в само себя по специальному действию 1. Эти петли нужно добавить обязательно, чтобы в дальнейшем правильно построить параллельную композицию. Они отражают тот факт, что, находясь в некотором состоянии, временной автомат сколь угодно долго ожидает входное воздействие. Отметим, что если бы мы рассматривали временные автоматы с задержками и *таймаутами*, то это уже были бы не петли, а более сложные группы переходов. Рассмотрение таких временных автоматов мы предполагаем в будущем как развитие данного исследования.

Теперь полуавтомат $A = \langle A, X, \lambda_A, \hat{s}, \mathfrak{F}_A \rangle$ будет описывать поведение временного автомата $S = \langle S, I, O, \lambda_S, \hat{s}, \sigma_S \rangle$, то есть язык полуавтомата A будет совпадать с языком временного автомата S, если временные выходы вида $\langle o, u \rangle$ автомата S воспринимать как последовательность $1^u.o$ (то есть прошло u тактов и появился выходной символ o, что соответствует принятой нами семантике временного выхода), и будет пригоден для построения параллельной композиции.

2.4. Преобразование глобального полуавтомата параллельной композиции к временному автомату

Пусть даны два временных автомата R и S, для которых построены полуавтоматы A_R и A_S соответственно, как описывалось в разделе 2. Далее, построив параллельную композицию этих полуавтоматов [4], получим (глобальный) полуавтомат A, описывающий параллельную композицию исходных автоматов R и S. Естественно возникает вопрос, соответствует ли глобальный полуавтомат некоторому временному автомату. Согласно утверждению в работе [7], множество рассматриваемых нами временных автоматов замкнуто относительно операции параллельной композиции. Решению задачи обратного преобразования полуавтомата к временному автомату был посвящён один из разделов нашей предыдущей работы [11]. Ввиду ограниченного объёма статьи мы не станем приводить эту процедуру, заметим лишь, что согласно правилам построения параллельной композиции [4], полуавтомат $A = \langle A, X, \lambda_A, \hat{a}, \mathfrak{F}_A \rangle$ должен быть сначала спроецирован на внешние алфавиты (действия по внутренним символам заменяются на специальный «пустой» символ), а затем детерминизирован. Под *внешними алфавитами (символами)* мы понимаем алфавиты, в которых композиция «общается» с внешней средой, а под *внутренними алфавитами (символами)* – алфавиты, в которых компоненты «общаются» между собой. После детерминизации полуавтомат будет удовлетворять необходимым условиям для применения процедуры из работы [11].

3. Применение BALM-II для построения параллельной композиции автоматов

В этом разделе мы описываем, как построить бинарную параллельную композицию двух полуавтоматов при помощи BALM-II, и иллюстрируем эту процедуру на примере построения каскадной композиции.

Временной автомат левой компоненты этой композиции представлен на рисунке 3a и является результатом параллельной композиции двух временных автоматов, описание построения этой композиции можно найти в работе [11]. Соответствующий полуавтомат представлен на рисунке 3b.

Временной автомат и соответствующий полуавтомат правой компоненты представлены на рисунках 3c и 3d. BALM-II поддерживает формат файла AUT для описания полуавтомата [4]. Этот формат является упрощённым вариантом формата BLIF_MV. В связи с ограниченным объёмом статьи, мы упомянем здесь лишь самые важные моменты.

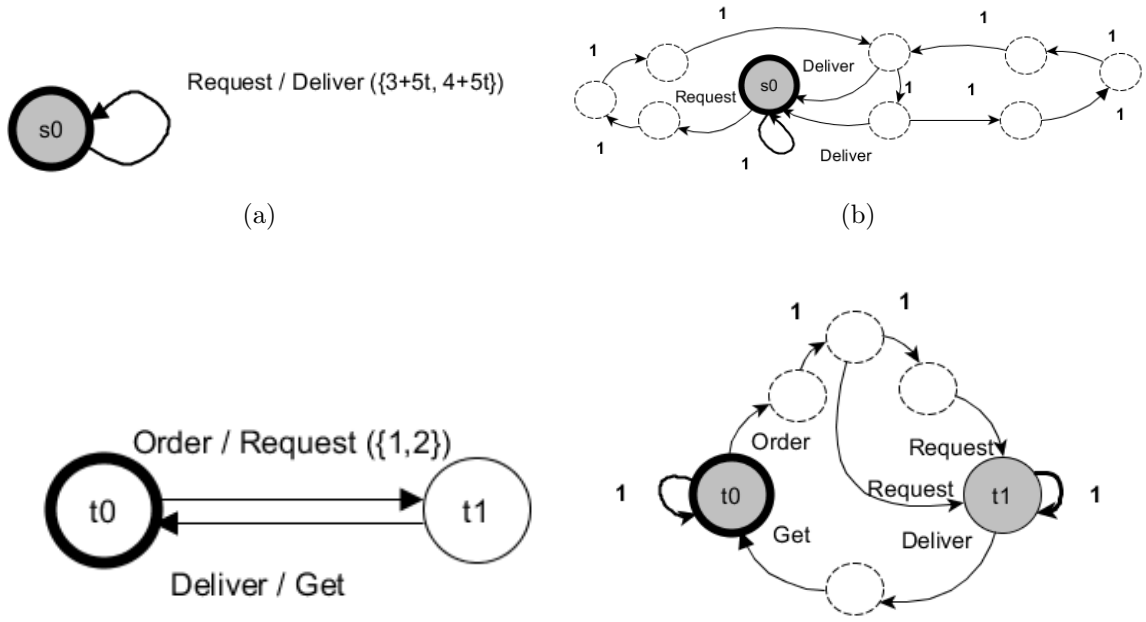


Рис. 3: Временной автомат левой компоненты (a) и соответствующий полуавтомат (b), временной автомат правой компоненты (c) и соответствующий полуавтомат (d)
Fig. 3. TFSM of the left component (a) and corresponding automaton (b), TFSM of the right component (c) and corresponding automaton (d)

Прежде всего, мы должны задать каналы компоненты (каналы, через которые компонента взаимодействует с внешней средой и с другой компонентой). В формате AUT для правой компоненты это будет выглядеть следующим образом:

```
.inputs x i o y t E
```

Мы обращаем внимание, что в дополнение к тем каналам левой компоненты, которые можно видеть на рисунке 1, добавлен специальный канал для моделирования времени (временной канал `t`), который соответствует временной переменной (или нашему специальному действию 1). Что касается канала `E`, это также специальный канал, который определяет, какой из каналов `x`, `i`, `o`, `y` и `t` сейчас активен (в то время как остальные каналы не активны).

Для временного канала `t` мы должны помимо специального действия 1 добавить ещё одно действие (в связи с тем, что для каждого канала должно быть хотя бы два значения в алфавите):

```
.mv t 2 1 none
```

После того, как мы запишем оба полуавтомата в формате AUT, первое, что мы должны сделать, – это синхронизировать каналы композируемых полуавтоматов:

```
chan_sync x|y|t|E x|i|o|y|t|E comp_timed.aut addition_part.aut \
  C.aut D.aut
```

Затем, согласно алгоритму построения параллельной композиции двух полуавтоматов [3, 4], мы должны расширить поведение полуавтомата левой компоненты на алфавиты каналов I и O:

```
expansion E3,E4 C.aut C_exp.aut
support x,y,t,i,o,E(5) C_exp.aut C_support.aut
```

Следующий шаг – построение пересечения двух полуавтоматов:

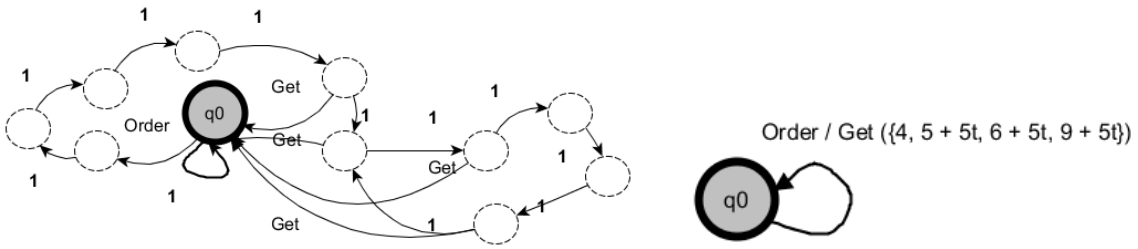
```
product D.aut C_support.aut product_cascade.aut
```

Теперь у нас построен полуавтомат, который описывает общее поведение левой и правой компонент (так называемый «глобальный полуавтомат»). Однако поведение такого полуавтомата не всегда соответствует ограничению «медленной внешней среды». Если данное ограничение не выполняется, то необходимо построить пересечение глобального полуавтомата с полуавтоматом, представляющим язык $(I(XT^*Y)^*T^*O)^*$. В нашем примере ограничение «медленной внешней среды» выполняется.

Следующий шаг – ограничить поведение полуавтомата на алфавиты внешних каналов и временного канала:

```
restriction E2,E3,E4 product_cascade.aut restriction_cascade.aut
support t,i,o,E(3) restriction_cascade.aut comp_cascade.aut
```

Итоговый полуавтомат для нашего примера представлен на рисунке 4a. Можно видеть, что на входной символ `Order` композиция может выдать выходной символ `Get` через 4 или $5 + 5t$, или $6 + 5t$, или $9 + 5t$ единиц времени, где t – произвольное неотрицательное целое число. Соответствующий временной конечный автомат представлен на рисунке 4b.



(a) (b)

Рис. 4: Полуавтомат, описывающий каскадную композицию (a), и соответствующий временной автомат (b)

Fig. 4. Automaton of the cascade parallel composition (a) and corresponding TFSM (b)

Заключение

В данной работе мы продолжили исследования, начатые нами в работе [11], и рассмотрели задачу построения каскадной параллельной композиции временных автоматов. Построение такой композиции можно свести к поэтапному построению бинарной параллельной композиции. Поскольку результатом бинарной параллельной композиции временных автоматов с константными задержками выходов может оказаться временной автомат, множество задержек выходных символов которого бесконечно и задано при помощи конечного множества линейных функций [7], в этой работе мы рассматриваем построение бинарной параллельной композиции таких автоматов, уточняя прежде всего определение данной модели. В качестве

инструмента для построения композиции мы выбрали BALM-II, который требует перехода к полуавтоматам. Предложенный в работе [7] способ построения по временному автомату с задержками выходов в виде множества линейных функций соответствующего полуавтомата требует последующей детерминизации полученного полуавтомата, поэтому мы предлагаем свою процедуру построения полуавтомата. В работе описано, каким образом построить композицию полученных полуавтоматов при помощи BALM-II. Также обсуждается процедура обратного преобразования от полуавтомата композиции к временному автомату, которая была введена нами в работе [11], с учётом нюансов, связанных с рассматриваемым типом временных автоматов. В дальнейшем мы планируем рассмотреть модель временного автомата, в которой помимо задержек выходов в виде констант и линейных функций присутствуют также переходы по таймаутам, и исследовать построение параллельной композиции для такой модели.

Список литературы / References

[1] Moore E. F., “Gedanken-experiments on Sequential Machines”, *Automata Studies, Annals of Mathematical Studies*, **34** (1956), 129–153.

[2] Gill A., *Introduction to the theory of finite-state machines*, McGraw-Hill, New-York, 1962, 207 pp.

[3] Yevtushenko N., Villa T., Brayton R., Petrenko A. and Sangiovanni-Vincentelli A., “Solution of parallel language equations for logic synthesis”, *The Proceedings of the International Conference on Computer-Aided Design*, 2001, 103–110.

[4] Castagnetti G., Piccolo M., Villa T., Yevtushenko N., Mishchenko A. and Brayton R. K., *Solving Parallel Equations with* BALM-II, Technical Report No UCB/EECS-2012-181, Electrical Engineering and Computer Sciences University of California at Berkeley, 2012, http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-181.pdf.

[5] Alur R. and Dill D. L., “A theory of timed automata”, *Theoretical computer science*, **126**:2 (1994), 183–235.

[6] El-Fakih K., Gromov M, Shabaldina N. and Yevtushenko N., “Distinguishing Experiments for Timed Non-Deterministic Finite State Machines”, *Acta Cybernetica*, **21**:2 (2013), 205–222.

[7] Kondratyeva O., Yevtushenko N. and Cavalli A., “Parallel composition of nondeterministic finite state machines with timeouts”, *Journal of Control and Computer Science of Tomsk State University*, **2**:27 (2014), 73–81.

[8] Kondratyeva O., Yevtushenko N. and Cavalli A., “Solving parallel equations for Finite State Machines with Timeouts”, *The Proceedings of ISP RAS*, **26**:6 (2014), 85–98.

[9] http://www.uppaal.com/.

[10] Zhigulin M., Yevtushenko N., Maag S. and Cavalli A. R., “FSM-based test derivation strategies for systems with timeouts.”, *Proceedings of the international conference QSIC*, 2011, 141–149..

[11] Gromov M. and Shabaldina N., “Using BALM-IIfor deriving parallel composition of timed finite state machines with outputs delays and timeouts: work-in-progress”, *System Informatics*, **8** (2016), 33–42.

[12] Springintveld J., Vaandrager F. and D’Argenio P. R., “Testing timed automata”, *Theoretical Computer Science*, **254**:1–2 (March 2001), 225–257.

Gromov M. .L, Shabaldina N. V., "Using BALM-II for Deriving Cascade Parallel Composition of Timed Finite State Machines", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 715–728.

DOI: 10.18255/1818-1015-2016-6-715-728

Abstract. In this paper, we consider the problem of deriving a cascade parallel composition of timed finite state machines (TFSMs). In order to build such a composition we can derive the corresponding binary parallel compositions step-by-step. It is known that if each component of a binary parallel composition is TFSM with output delays that are natural numbers or zero, the result of the composition can be TFSM with output delays that are sets of linear functions. So, the problem of deriving a cascade composition of TFSMs with constant output delays is reduced to the problem of deriving several binary parallel compositions of TFSMs with output delays that are sets of constants or linear functions. In this work, we refine the definition of TFSM and pay special attention to the description of an output delay function. As a tool for deriving the composition we consider BALM-II, and in consequence we study the ways of constructing the corresponding automaton for TFSM with output delays as a set of linear functions. We suggest a new procedure for constructing such an automaton, and unlike the known procedure our procedure does not require further determinization of the derived automaton. Moreover, we describe step by step how to build the composition of the derived automaton by using BALM-II, and discuss a procedure of reverse transformation from the global automaton to TFSM in case when the components are TFSMs with output delays that are sets of linear functions. We use an application example in order to illustrate derivation of the cascade parallel composition of TFSMs.

Keywords: timed finite state machines, cascade parallel composition, BALM-II

About the authors:
Maxim L. Gromov, orcid.org/0000-0002-2990-8245, PhD,
Tomsk State University,
36 Lenin av., Tomsk 634050, Russia, e-mail: maxim.leo.gromov@gmail.com

Natalia V. Shabaldina, orcid.org/0000-0002-1958-550X, PhD,
Tomsk State University,
36 Lenin av., Tomsk 634050, Russia, e-mail: nataliamailbox@mail.ru

Acknowledgments:
This work is partially supported by the grant for the basic research № 16-49-03012 of Russian Scientific Fund.

Моделирование и анализ информационных систем. Т. 23, №6 (2016), с. 729–740
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 729–740

DOI: 10.18255/1818-1015-2016-6-729-740

УДК 004.415.53

Метод синтеза тестов с гарантированной полнотой по модели расширенного автомата

Ермаков А.Д., Евтушенко Н.В.[1]

получена 2 сентября 2016

Аннотация. Расширенные автоматы активно используются при построении тестов для программного обеспечения на основе формальных моделей. Однако полнота тестов, построенных по расширенному автомату на основе покрытия путей, переменных и т.п., остается практически неизвестной; более того, как известно, такие тесты не обнаруживают большое количество часто встречающихся функциональных ошибок в программных реализациях системы, поведение которой описано таким расширенным автоматом. В данной работе для построения тестовых последовательностей мы предлагаем использовать шаблонную реализацию расширенного автомата в языке Java. Поскольку программа составлена по шаблону, то ошибки в программе напрямую переносятся на ошибки в расширенном автомате. В работе предлагается метод построения множества тестовых последовательностей, обнаруживающих функциональные ошибки в шаблонной реализации расширенного автомата. На первом шаге тест, построенный по расширенному автомату одним из известных методов, проверяется на полноту относительно ошибок, сгенерированных инструментом μJava в шаблонной программной реализации. После этого для каждого необнаруженного тестом программного мутанта строится мутант эталонного расширенного автомата; на следующем шаге по некоторой конечно-автоматной абстракции генерируется последовательность, различающая два расширенных автомата (если такая последовательность существует), которая добавляется в строящийся тест. Построенный таким образом тест является полным относительно ошибок, сгенерированных инструментом μJava. Если соответствующий конечный автомат, построенный посредством моделирования расширенного автомата, получается слишком сложным, или построить такой конечный автомат не представляется возможным, то полнота построенного теста не гарантируется. Однако экспериментально показывается, что исходный тест, расширенный такими различающими последовательностями, обнаруживает значительно больше функциональных ошибок в программных реализациях системы, для которой расширенный автомат используется в качестве спецификаци.

Ключевые слова: мутационное тестирование, расширенный автомат, конечно-автоматная абстракция, μJava

Для цитирования: Ермаков А. Д., Евтушенко Н.В., "Метод синтеза тестов с гарантированной полнотой по модели расширенного автомата", *Моделирование и анализ информационных систем*, **23**:6 (2016), 729–740.

Об авторах:
Ермаков Антон Дмитриевич, orcid.org/0000-0003-0838-8014, аспирант,
Томский государственный университет,
пр. Ленина, 36, г. Томск, 634050 Россия, e-mail: antonermak@inbox.ru

Евтушенко Нина Владимировна, orcid.org/0002-0007-1896-0876, профессор, д-р. техн. наук,
Томский государственный университет,
пр. Ленина, 36, г. Томск, 634050 Россия, e-mail: nyevtush@gmail.com

Благодарности:
[1]Работа выполнена при финансовой поддержке проекта РНФ № 16-49-03012.

Введение

Программные реализации, используемые, в том числе, в критических системах, становятся более сложными, и гарантировать полноту их тестирования без использования математических моделей практически невозможно. При синтезе тестов с гарантированной полнотой активно используются модели с конечным числом переходов [1,2], в частности модель расширенного автомата, которая достаточно близка к программной реализации в языке высокого уровня. Несмотря на большое количество публикаций об автоматическом построении такой модели, в опубликованных статьях все примеры обычно ограничиваются небольшими программами, и достаточно часто такой расширенный автомат строится инженером по тестированию «вручную», исходя из неформальных требований к функционированию тестируемой программы. Соответственно, практически невозможно сопоставить ошибки в построенной модели, относительно которых и будет гарантирована полнота построенного теста, с ошибками в исходной программе, и для генерации тестовой последовательности, обнаруживающей такую функциональную ошибку, приходится сравнивать программу-спецификацию с программой-мутантом для построения соответствующей различающей последовательности [3,4]. В настоящей работе мы предлагаем метод построения по расширенному автомату множества тестовых последовательностей, которые обнаруживают все функциональные ошибки в шаблонной реализации расширенного автомата в языке Java.

Расширенный автомат [5] расширяет классическую автоматную модель внутренними (контекстными) переменными, входными и выходными параметрами. Известно, что тесты, построенные на основе таких расширенных систем переходов, оказываются достаточно качественными, однако остается много функциональных ошибок в тестируемых программных реализациях, которые построенными тестами не обнаруживаются [4,6]. Поэтому мы предлагаем повысить полноту тестов, построенных по расширенному автомату, рассматривая наиболее часто встречаемые ошибки в программных реализациях, на основе «шаблонной» реализации расширенного автомата в языке Java. С помощью инструмента μJava [7] генерируется множество мутантов для «шаблонной» программной реализации, на которые подается проверяющий тест, построенный к настоящему моменту; первоначальный тест может быть построен одним из известных методов, в том числе это может быть множество последовательностей, сгенерированных случайным образом. Если мутант тестом не обнаруживается, то соответствующая ошибка легко преобразуется в ошибку в расширенном автомате, и, таким образом, различающая последовательность строится не для двух программных реализаций, что, как известно, достаточно сложно [4], а для двух автоматных моделей, что значительно проще [8]. Частично результаты по подобному подходу были опубликованы в [9]; в настоящей работе мы обсуждаем возможности использования предложенного подхода для обнаружения функциональных ошибок в расширенных автоматах.

Структура работы следующая. В разделе 1 вводятся необходимые определения и обозначения. В разделе 2 обсуждаются способы построения различающих последовательностей для двух расширенных автоматов, в нашем случае для автомата-спецификации и интересующего нас мутанта. Предлагаемый метод построения проверяющего теста описывается в разделе 3. В разделе 4 рассматривается пример те-

стирования программной реализации протокола SCP (Simple Connection Protocol), для которого и иллюстрируется предлагаемый подход. В заключении мы кратко обсуждаем возможные направления дальнейшей работы.

1. Определения и обозначения

Под *конечным автоматом* понимается пятёрка $S = (S,\ I,\ O,\ T_S,\ s_0)$, где S – непустое конечное множество *состояний* с выделенным начальным состоянием s_0, I – непустое конечное множество *входных* символов, называемое входным алфавитом, O – непустое конечное множество *выходных* символов, называемое выходным алфавитом, $T_S \subseteq I \times S \times S \times O$ – отношение *переходов* [8]. Расширенный автомат дополняет конечный автомат контекстными (внутренними) переменными, входными и выходными параметрами, и условиями, при которых переход может быть выполнен. Формально [5] под расширенным автоматом M понимается пятерка $M = (S,\ s_0,\ X,\ Y,\ T,\ V)$, где S – непустое конечное множество состояний автомата, X – непустое конечное множество входных символов, Y – непустое конечное множество выходных символов, V – конечное, возможно, пустое множество контекстных переменных, T – множество переходов между состояниями из S. Каждый переход в расширенном автомате есть семёрка $(s,\ x,\ P,\ o_M,\ y,\ u_M,\ s')$, где s и s' суть начальное и финальное состояния перехода; $x \in X$ есть входной символ и D_{inp-x} обозначает множество входных векторов, компонентами которых являются значения параметров, соответствующих входному символу x (далее *входные параметры*); $y \in Y$ – выходной символ, и D_{out-y} обозначает множество выходных векторов, компонентами которых являются значения параметров, соответствующих выходному символу y (далее *выходные параметры*); P, o_M и u_M – функции, определенные над входными параметрами и контекстными переменными из V. Предикат P: $D_{inp-x} \times D_V \to \{0,\ 1\}$, где D_V – множество контекстных векторов, то есть векторов, компонентами которых являются значения контекстных переменных, описывает условия, при которых данный переход может быть выполнен; функция $o_M : D_{inp-x} \times D_V \to D_{out-y}$ описывает значения выходных параметров в результате выполнения перехода; функция $u_M : D_{inp-x} \times D_V \to D_V$ описывает значения контекстных переменных в результате выполнения перехода.

Пара «состояние, вектор значений контекстных переменных» называется *конфигурацией*, пары «входной символ, вектор значений входных параметров» и «выходной символ, вектор значений выходных параметров» называются *параметризированными* входным и выходным символами соответственно. Начальная конфигурация расширенного автомата обычно предполагается известной. Переход в расширенном автомате может быть выполнен, если только соответствующий предикат принимает значение «Истина» в данной конфигурации на данном параметризированном входном символе. Таким образом, в отличие от классических систем с конечным числом состояний, в расширенном автомате не каждый переход может быть выполнен в текущем состоянии (хорошо известная проблема выполнимости последовательности переходов в расширенном автомате). Вполне возможно, что для выполнения конкретного перехода понадобится сначала выполнить ряд других переходов, на-

пример, для достижения переменной-счетчиком нужного значения, и только потом можно будет выполнить требуемый переход.

Под *функциональными ошибками* в расширенном автомате понимаются ошибки переходов / выходов, присвоения значений переменным, ошибки в предикатах и т.п. Известные методы построения проверяющих тестов по модели расширенного автомата ориентируются на покрытие путей, переменных, условий и т.п. [4, 6], но как показано в диссертации С. Ники [4], полнота таких тестов относительно функциональных ошибок очень низкая, не превышает 70 %. На основе компьютерных экспериментов с различными протокольными реализациями в [6] показывается, что полнота достаточно длинных случайных тестов относительно функциональных ошибок практически совпадает с полнотой тестов, построенных на основе покрытия множеств различных переходов расширенного автомата.

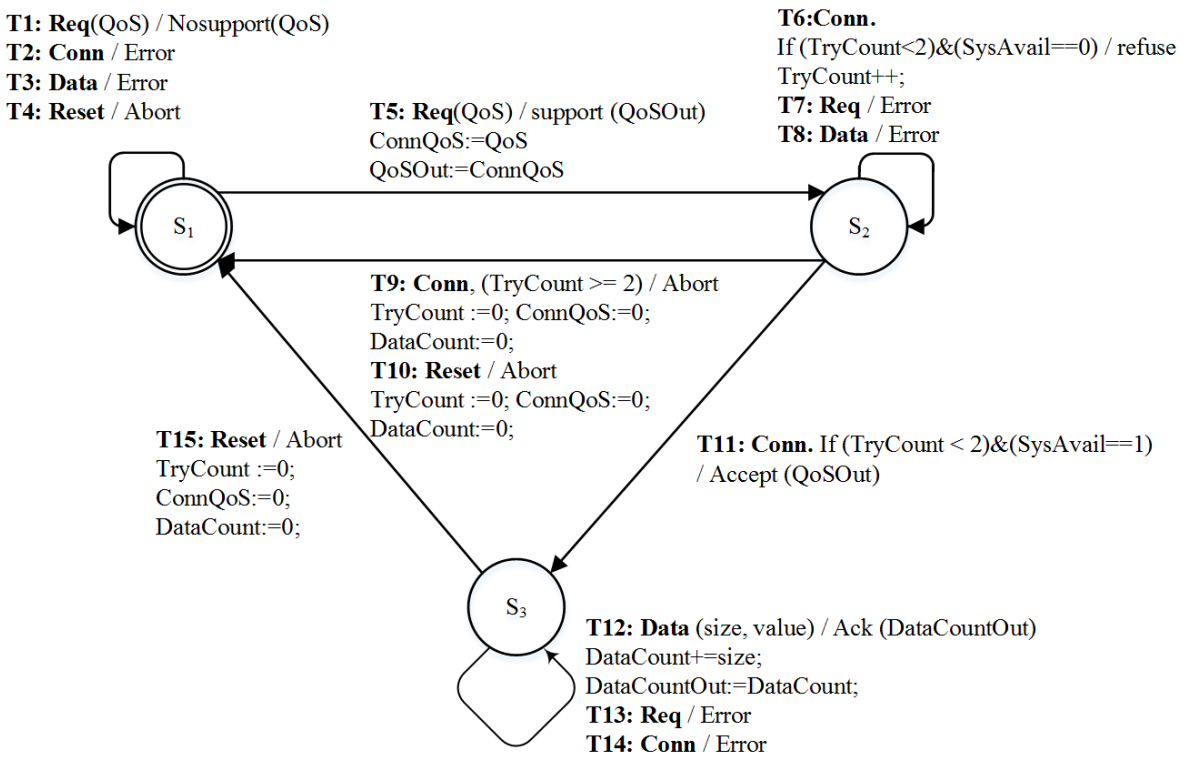


Рис. 1. Расширенный автомат для SCP

Fig. 1. EFSM for SCP

В качестве примера рассмотрим расширенный автомат, представляющий собой описание протокола SCP (Simple Connection Protocol) [10,11]; расширенный автомат (рис. 1) имеет 3 состояния, которые, вообще говоря, описывают различные режимы функционирования. Состояние S_1 описывает режим ожидания запроса на соединение, S_2 соответствует состоянию процесса установки соединения, а состояние S_3 соответствует передаче данных. Входные символы описывают стандартные команды протокола: *Req* – запрос, *Conn* – подключение, *Data* – передача данных, *Reset* – сброс; мы также используем входной параметр *Support*, который равен 1 в случае готовности установить соединение уровня QoS, и 0 при отсутствии готовности. Вход-

ной параметр *SysAvail* соответствует 1, если система свободна для подключения, и 0, если занята. Выходные параметры суть *Nosupport, Error, Abort, Support, Refuse, Accept, Ack*. Контекстная переменная *TryCount* соответствует счетчику неудачных попыток установки соединения. Несмотря на то, что этот протокол является в некоторой степени «игрушечным», он достаточно хорошо иллюстрирует многие аспекты протокольных реализаций.

Заметим, что, в отличие от построения различающих последовательностей для мутантов программного обеспечения, построение различающих последовательностей для конечных автоматов намного проще, и в следующем разделе мы кратко обсуждаем известные методы построения различающей последовательности для двух расширенных автоматов на основе различных конечно-автоматных абстракций, отмечая, для каких функциональных ошибок какие абстракции являются наиболее подходящими.

2. Построение различающих последовательностей для расширенных автоматов

В настоящем разделе мы обсуждаем, каким образом можно построить различающую последовательность для двух расширенных автоматов. Реализация общего метода из [5], который гарантирует построение различающей последовательности для двух неэквивалентных детерминированных расширенных автоматов, является достаточно трудоемкой, и отметим, авторы ничего не говорят о решении проблемы выполнимости. Все остальные методы построения различающих последовательностей для расширенных автоматов являются достаточно эффективными эвристиками, использующими конечно-автоматные абстракции [11–15], для которых отсутствует проблема выполнимости, и построение различающей последовательности достаточно просто осуществляется на основе построения пересечения двух конечных автоматов, которые могут быть детерминированными или недетерминированными, частичными или полностью определенными.

Построение различающей последовательности на основе пересечения двух расширенных автоматов. В [5] рассматривается задача построения различающей последовательности для двух различных расширенных автоматов; в нашем случае спецификации и мутанта. Для описания всех возможных различающих последовательностей строится *различающий автомат*, в котором специально введенное состояние *fail* представляет все последовательности, различающие начальные конфигурации автомата-спецификации и автомата-мутанта, не различимого со спецификацией уже построенным тестом. Поскольку рассматривается различимость двух расширенных автоматов, в которых множества контекстных переменных не пересекаются, в нашем случае контекстные переменные мутанта необходимо переименовать. В этом случае все различающие последовательности представляются специальным состоянием *fail*, и тем самым задача построения различающей последовательности сводится к хорошо изученной задаче достижимости. С практической точки зрения использование такого различающего автомата имеет ряд недостатков. Во-первых, речь идет только о детерминированных полностью определенных спецификациях, что не всегда имеет место, в частности, для протокольных

спецификаций. Во-вторых, не всякая последовательность переходов, переводящая различающий автомат в состояние *fail*, выполнима, т.е. проблема выполнимости в работе не решается. В-третьих, остается вопрос о равенстве параметризированных выходных символов в случае, когда такие символы являются даже простыми функциями, а не просто назначаются равными некоторой константе. Таким образом, несмотря на общность предлагаемого подхода, использование различающего автомата-пересечения для построения различающих последовательностей для двух расширенных автоматов является достаточно трудоемким.

Построение различающей последовательности на основе конечно-автоматных абстракций двух расширенных автоматов. Наиболее известными можно считать следующие две конечно-автоматные абстракции для расширенных автоматов. В первом случае поведение расширенного автомата в начальной конфигурации моделируется на входных параметризированных последовательностях до достижения в конечном автомате заданного числа состояний или на всех таких последовательностях длины не более l. Если известно, на каком переходе отличаются автомат-спецификация и исследуемый автомат-мутант, то на каждом шаге моделирования можно выбирать следующий переход с использованием некоторого «жадного» алгоритма, приближаясь как можно ближе к мутированному переходу. Проведенные эксперименты показывают, что при условии, что два расширенных автомата отличаются между собой в небольшом количестве переходов (один-два мутированных перехода в спецификации), обычно достаточно рассматривать $l = 2$, 3 для построения различающей последовательности. Кроме того, при достаточно большом ограничении на число состояний обход графа переходов соответствующего конечного автомата обнаруживает одиночные ошибки в предикатах и функциях для контекстных переменных и выходных параметров [13].

Во втором случае из расширенного автомата просто удаляются все предикаты, входные и выходные параметры и соответствующие функции. Как показано в [14], в этом случае (адаптивная) различающая последовательность строится для двух недетерминированных, возможно ненаблюдаемых автоматов. Методы построения различающих последовательностей для таких автоматов существуют [14,15], и как показывают проведенные эксперименты, несмотря на экспоненциальные общие верхние оценки длины таких последовательностей, в большинстве случаев длина таких последовательностей близка к числу состояний автомата. Более того, адаптивные различающие последовательности существуют чаще и оказываются обычно короче; однако в этом случае и проверяющие тесты также должны быть адаптивными. Следует отметить, что поскольку в такой абстракции удаляются все предикаты, контекстные переменные, входные и выходные параметры, то такая абстракция может быть эффективно использована только при построении различающих последовательностей для ошибок в выходных символах и переменных состояний. Например, для расширенного автомата на рис. 1 ошибка в финальном состоянии перехода T5 (вместо S_2 состояние S_1) немедленно обнаруживается при подаче входного символа *Req*.

Построение различающей последовательности на основе предикатных абстракций двух расширенных автоматов. Недетерминированные автоматы появляются и при рассмотрении предикатных абстракций расширенных автома-

тов [16]; при построении различающих последовательностей такие абстракции оказываются наиболее эффективными при отсутствии входных параметров.

Пусть β есть множество k предикатов, определенных относительно контекстных переменных и входных параметров расширенного автомата M. Если S – множество состояний расширенного автомата и D_W – множество всех конфигураций и параметризированных входных векторов, то предикатная β-абстракция определяется как a_β: $S \times D_W \to S \times \{(0,1)\}^k$, где $a_\beta(s, \mathbf{w}) = (s, b_1, \ldots, b_k)$, $\mathbf{w} \in D_W, b_i \in \{0,1\}, i = 1..k$. По определению, $b_i = 1$, если и только если предикат $B_i(\mathbf{w})$ принимает значение «Истина». Для выбранного множества $\mathbf{P_x}$ (параметризированных) входных символов множество входных символов предикатной абстракции содержит пары $(x, \mathbf{p_x}), \mathbf{p_x} \in \mathbf{P_x}$, и все непараметризированные входные символы. Для каждой конфигурации $(s, \mathbf{v})$, входного символа $(x, \mathbf{p_x}) \in \mathbf{P_x}$ и предиката P перехода (s, x, P, y, u_p, s') из состояния s, такого что $(\mathbf{v}, \mathbf{p_x})$ обращает P в истину и $u_M(\mathbf{v}) = \mathbf{v}'$, в множество переходов предикатной абстракции включается переход $((s, \mathbf{a}), x, y, (s', \mathbf{a}'))$, $(s, \mathbf{a}) = a_\beta(s, \mathbf{v})$ и $(s', \mathbf{a}') = a_\beta(s', \mathbf{v}')$. В качестве примера рассмотрим предикатную абстракцию для расширенного автомата на рис. 1 относительно множества предикатов $(TryCount < 2)$ и $(SysAvail = 0)$. При ошибке в условии $(SysAvail = 0)$ предикатная абстракция не останется в состоянии S_2, а из состояния S_2 перейдет в состояние S_3, что достаточно просто обнаружится при подаче входного символа *Data*.

В общем случае предикатная абстракция является недетерминированным автоматом, в котором степень недетерминизма существенно зависит от выбора множества предикатов и множества параметризированных входных символов. Упрощая утверждение из работы [16] для случая различимости начальных конфигураций двух расширенных автоматов, получаем, что два расширенных автомата различимы (адаптивной) входной последовательностью α, если начальные состояния соответствующих предикатных абстракций разделимы последовательностью α (адаптивно различимы), т.е. множества выходных реакций на α в этих состояниях не пересекаются (существует адаптивный различающий тестовый пример [15]). В работе [16] обсуждаются возможности выбора предикатов (для построения предикатной абстракции) с целью уменьшения недетерминизма, чтобы увеличить вероятность существования разделяющей последовательности. Кроме того, можно строить адаптивную различающую последовательность, т.е. говорить об адаптивной различимости двух расширенных автоматов. Насколько нам известно, адаптивные различающие последовательности для расширенных автоматов нигде не исследовались.

Отметим, что процесс построения предикатной абстракции достаточно трудоемкий, поскольку исследуются конфигурации расширенного автомата. Однако при наличии ошибки только на одном переходе (как получается при использовании инструмента μJava), можно просто отметить переход с ошибкой в предикатной абстракции спецификации, т.е. не строить заново предикатную абстракцию для автомата-мутанта.

Построение различающей последовательности на основе различных срезов двух расширенных автоматов достаточно подробно обсуждается в работах [13,17]. Как показывают проведенные авторами эксперименты, тесты, построенные как обход графа переходов таких срезов, достаточно хорошо обнаруживают одиночные функциональные ошибки, такие как ошибки переходов и выходов,

а также ошибки типа присвоения неправильного значения контекстной переменной/выходному параметру или замены некоторой арифметической/логической операции.

3. Метод построения проверяющего теста с использованием инструмента μJava

Проверяющие тесты, построенные по расширенному автомату одним из известных (достаточно простых) методов, дополняются различающими последовательностями для соответствующих мутантов, которые строятся с использованием специального инструмента μJava [7]. Инструмент μJava имеет достаточно обширные функциональные возможности и согласно документации способен генерировать 34 вида мутаций программного кода, среди которых выделяют традиционные ошибки (замена математических, логических операторов, операторов сравнения и т.д.) и ошибки объектно-ориентированного программирования (ошибки наследования, полиморфизма, и т.д.), достаточно хорошо соответствующие функциональным ошибкам программного обеспечения. Последнее и является основной причиной выбора такого подхода для повышения полноты тестов, построенных по расширенному автомату. Поскольку известно, что наиболее трудно обнаружимыми являются одиночные ошибки, то именно такие мутанты программных реализаций и рассматриваются. Для генерации мутаций необходимо запустить графическую оболочку μJava, выбрать проект программы и типы мутаций для генерации. В результате в директории Results появятся все сгенерированные мутанты в поддиректориях с именами, соответствующими типу мутации и её порядковому номеру. С использованием библиотеки для модульного тестирования JUnit [18] тест может быть подан сразу на все мутанты с целью определения, какие из мутантов не обнаруживаются тестом. Таким образом, построение теста по расширенному автомату с использованием инструмента μJava содержит следующие шаги.

Шаг 1. Первоначальный проверяющий тест TS строится одним из известных методов по расширенному автомату-спецификации M. Можно использовать обход графа переходов соответствующего расширенного автомата, различные методы покрытия путей, переменных, условий и т.п., а также случайно сгенерированные тесты определенной длины.

Шаг 2. По заранее определенному шаблону строится программная реализация расширенного автомата-спецификации таким образом, что ошибки программной реализации жестко связаны с ошибками в расширенном автомате. В частности, в программной реализации состояния расширенного автомата используются как метки для описания соответствующего режима работы. Контекстные переменные, входные и выходные параметры соответствуют таковым в программной реализации; предикаты описывают условия выполнения инструкций.

Шаг 3. Проверяющий тест TS проверяется на полноту для построенной программной реализации с использованием ошибок, вносимых генератором инструмента μJava. Те ошибки, которые не были обнаружены, вносятся в расширенный автомат, и строится множество Mut расширенных автоматов-мутантов, не обнаружимых тестом TS.

Шаг 4. Для каждого автомата *Imp* из множества *Mut* строится подходящая конечно-автоматная абстракция и определяется последовательность, различающая конечно-автоматные абстракции двух расширенных автоматов, автомата-спецификации и автомата *Imp*. Если такая последовательность существует, то она добавляется в тест *TS*; при отсутствии такой последовательности делается вывод о неразличимости текущего мутанта и спецификации.

Утверждение 1. Если существуют детерминированные конечные автоматы, моделирующие поведение расширенных автомата-спецификации и каждого построенного мутанта, то алгоритм, содержащий вышеописанные шаги, возвращает полный проверяющий тест относительно квазиэквивалентности, т.е. построенный тест обнаруживает всякий мутант, поведение которого отличается от спецификации на некоторой определенной в спецификации (параметризированной) входной последовательности.

В ряде случаев для расширенного автомата-спецификации или некоторого мутанта такой конечный автомат нельзя построить в силу вычислительных проблем или, например, бесконечного множества допустимых значений для некоторой контекстной переменной или входного параметра. Соответственно мы не можем гарантировать различимость мутанта и спецификации в случае ненахождения различающей последовательности, но, как показывают эксперименты, проведенные с рядом расширенных автоматов, описывающих поведение протоколов и технических систем, такие ситуации встречаются достаточно редко. Если построенные автоматы являются недетерминированными, то полнота построенного теста определяется не относительно эквивалентности, а относительно отношений неразделимости или адаптивной неразличимости (при использовании адаптивной различающей последовательности). Мы подчеркиваем, что для произвольных расширенных автоматов отсутствуют необходимые и достаточные условия для проверки отношений эквивалентности, редукции, разделимости и адаптивной различимости двух автоматов. В наших экспериментах рассматривались достаточно простые расширенные автоматы; поэтому для проверки этих отношений мы использовали только достаточные условия. Одним из таких условий является внесение ошибки, при которой появляется дополнительная компонента связности в эталонном расширенном автомате, причем в исходной компоненте связности не происходит никаких изменений.

4. Анализ проведенных экспериментов на примере SCP

Эксперименты проводились с расширенными автоматами, описывающими такие протоколы как SCP, «Time» protocol, SMTP, POP3, TFTP, калькулятор, Audio CD плеер. Практически во всех случаях, за исключением самых простых протоколов, исходный обход графа переходов расширенного автомата приходилось доопределять различающими последовательностями. Более детально мы иллюстрируем процесс для протокола SCP (Simple Connection Protocol), для которого расширенный автомат был реализован программно, и в качестве начального теста использовался обход графа переходов. В результате запуска инструмента μJava на программной

реализации было сгенерировано 245 традиционных (арифметических) мутантов и 7 мутантов объектного типа (см. таблицу 1).

Таблица 1. Сгенерированные мутанты

Table 1. Generated mutants

Наименование Name	Описание мутантов Mutant description	Количество мутантов Number of mutants
AOIS	Инкремент/декремент случайной переменной	96
AOIU	Отрицание переменной	5
LOI	Побитовое отрицание	24
ROR	Замена знаков операторов сравнения >, <, =, <=, >=, ==	91
COR	Замена знаков логических операторов ^, ||, &&, &, |&	4
COI	Внедрение в код логического отрицания условий !(true), !(false)	17
ASRS	Модификация операций составного присваивания +=, /=, -=, %=	8
JSI	Добавление служебного слова Static в объявлении член-данных класса	7
Всего		252

Запуск начального теста *TS* на данных мутантах показал, что 62 мутанта (24,6%) сработали идентично исходной программе. С использованием конечно-автоматных абстракций выяснилось, что 9 (3,6%) из них не эквивалентны спецификации. Остальные 53 (21%) мутантов не вносили изменений в поведение расширенного автомата-спецификации. Далее, путем возврата к расширенному автомату были выделены переходы, на которых произошли неэквивалентные мутации. Благодаря этому тест был достроен путем добавления трех различающих последовательностей параметризированных входных символов суммарной длины 11. Таким образом, длина теста увеличилась с 18 до 29 входных символов. Повторный запуск теста различил 9 неэквивалентных мутантов от спецификации. Таким образом, неотличимыми остались только 53 эквивалентных мутанта. В результате полнота теста выросла с 75,4% до 100 % (относительно мутантов, сгенерированных с использованием инструмента μJava).

5. Заключение

В данной работе мы предложили метод повышения полноты тестов, построенных по расширенному автомату, который описывает поведение программной реализации, за счет использования мутационного тестирования для программной реализации,

выполненной по специальному шаблону. Тесты, построенные как обход графа переходов спецификации, оказались неполными относительно ошибок, вносимых инструментом μJava, т.е. были функциональные ошибки, которые не обнаруживались таким тестом. Причина этого, в первую очередь, заключается в том, что расширенный автомат-спецификация чаще всего строится не непосредственно по тестируемой программной реализации, а на основе некоторых неформально описанных требований к такой реализации. Для полного обнаружения ошибок, вносимых инструментом μJava, тест дополнялся различающими последовательностями для конечно-автоматных абстракций спецификации и ее мутанта. Как показывают проведенные эксперименты, такой подход с использованием конечно-автоматных абстракций позволил выявить мутанты, эквивалентные спецификации, а также достроить проверяющий тест различающими последовательностями для неэквивалентных мутантов. Интересным вопросом является исследование возможности введения наблюдаемости для контекстных переменных на основе такого подхода. Если некоторые мутации контекстных переменных не обнаруживаются при использовании конечно-автоматных абстракций, то, возможно, имеет смысл сделать эти переменные наблюдаемыми при отладке программы, т.е. объявить их выходными параметрами. Построение различающих последовательностей дает возможность минимизировать список таких наблюдаемых переменных. Особый интерес представляет построение различающих последовательностей для автоматов специальных классов, например, древовидных и временных автоматов. Еще одно направление нашей дальнейшей работы относится к использованию полученных результатов не только для обнаружения, но и для локализации ошибок в программных реализациях.

Список литературы / References

[1] Kaur M., Singh R., “A Review of Software Testing Techniques”, *International Journal of Electronic and Electrical Engineering*, **7**:5 (2014), 463–474.

[2] Jorgensen P.C., *Software Testing: A Craftsman’s Approach, Third Edition*, Auerbach Publications, 2008.

[3] Nica M., Nica S., Wotawa F., “On the use of mutations and testing for debugging”, *Software – practice and experience*, **43**:9 (2013), 1121–1142.

[4] Nica S., *On the Use of Constraints in Program Mutations and its Applicability to Testing*, PhD thesis, Graz Technical University, 2013.

[5] Petrenko A., Boroday S., Groz R., “Confirming Configurations in EFSM Testing”, *IEEE Trans. Software Eng*, **30**:1 (2004), 29–42.

[6] El-Fakih K., Salameh T., Yevtushenko N., “On Code Coverage of Extended FSM Based Test Suites: An Initial Assessment”, *LNCS*, **8763** (2014), 198–204.

[7] “μJava documentation. μJava home page. URL: http://cs.gmu.edu/~offutt/mujava/”, 2014, (access date: 10.04.2016).

[8] Villa, T. et al., *The Unknown Component Problem: Theory and Applications*, Springer, 2012, 313 pp.

[9] Ermakov A., Yevtushenko N., “Increasing the fault coverage of tests derived against Extended Finite State Machines”, *System informatics*, **7** (2016), 23–32.

[10] Alcalde B. et al., “Network Protocol System Passive Testing for Fault Management: A Backward Checking Approach”, *Proc. of the 24th IFIP WG 6.1 Intern. Conf. on Formal Techniques for Networked and Distributed Systems, FORTE’2004*, 150–166.

[11] Kushik N. et al., "Optimizing Protocol Passive Testing through 'Gedanken' Experiments with Finite State Machines", *Proc. of the Intern conference on Sofware Quality and Reliability, QSR'2016*, August, 2016.

[12] Kushik N., Yenigün H., "Heuristics for Deriving Adaptive Homing and Distinguishing Sequences for Nondeterministic Finite State Machines", *Lecture Notes in Computer Science*, **9447** (2015), 243–248.

[13] Коломеец А. В., *Алгоритмы синтеза проверяющих тестов для управляющих систем на основе расширенных автоматов*, дис. ... канд. техн. наук, Томский государственный университет, 2010; [Kolomeec A. V., *Algoritmy sinteza proverjajushhih testov dlja upravljajushhih sistem na osnove rasshirennyh avtomatov*, dis. ... kand. tekhn. nauk, Tomsk State University, 2010, (in Russian).]

[14] Kushik N., Yevtushenko N., Cavalli A., "On Testing against Partial Non-observable Specifications", *Proc. of the Intern. Conf. on the Quality of Information and Communications Technology*, 2014, 230–233.

[15] Kushik N. et al., "On adaptive experiments for nondeterministic finite state machines", *The Intern. Journal on Software Tools for Technology Transfer*, **18**:3 (2016), 251–264.

[16] El-Fakih K. et al., "Distinguishing extended finite state machine configurations using predicate abstractions", *Journal on Software Research and Development (published online)*, 2016.

[17] Михайлов Ю. В., Коломеец А. В., "Проверка переходов в расширенном автомате на основе срезов", *Вестник ТГУ. Серия: Управление, вычислительная техника и информатика*, **3**:4 (2008); [Mikhaylov Yu. V., Kolomeets A. V., "Proverka perekhodov v rasshirennom avtomate na osnove srezov", *Vestnik TGU. Seriya: Upravlenie, vychislitel'naya tekhnika i informatika*, **3**:4 (2008), (in Russian).]

[18] "JUnit 4, documentation. URL: http://cs.gmu.edu/~offutt/mujava/", (access date: 10.04.2016).

Ermakov A. D., Yevtushenko N. V., "Deriving Test Suites with the Guaranteed Fault Coverage for Extended Finite State Machines", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 729–740.

DOI: 10.18255/1818-1015-2016-6-729-740

Abstract. Extended Finite State Machines (EFSMs) are widely used when deriving tests for checking functional requirements for software implementations. However, the fault coverage of tests covering appropriate paths, variables, etc. of the specification EFSM, remains rather obscure and such tests do not detect many functional faults in EFSM implementations. In this paper, an approach is proposed for deriving complete tests with respect to functional faults of a proper Java EFSM implementation. First, an initial test suite derived against the specification EFSM is checked with respect to faults generated by a μJava tool. Since the EFSM software implementation is template based, each undetected fault can be easily mapped into a mutant EFSM of the specification machine. Thus, a distinguishing sequence is derived for two Finite State Machines modeling two EFSMs instead of deriving such a sequence for two programs. If the corresponding FSMs are too complex or cannot be completely derived, a test suite can be incomplete. However, the performed experiments clearly show that a test suite extended by such distinguishing sequences detects much more functional faults in software implementations of a system whose behaviour is described by the given EFSM.

Keywords: Mutation testing, Extended Finite State Machine (EFSM), finite automata, μJava

About the authors:
Anton D. Ermakov, orcid.org/0002-0007-1896-0951, researcher, Tomsk State University,
36 Lenina ave., Tomsk 634050, Russia, e-mail: antonermak@inbox.ru

Nina V. Yevtushenko, orcid.org/0000-0002-4006-1161, Professor, Doctor of Technical Sciences, Tomsk State University,
36 Lenina ave., Tomsk 634050, Russia, e-mail: nyevtush@gmail.com

Acknowledgments:
This work was supported by the project of RSF № 16-49-03012.

Моделирование и анализ информационных систем. Т. 23, № 6 (2016), с. 741–753
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 741–753

DOI: 10.18255/1818-1015-2016-6-741-753

УДК 517.9

О минимизации конечных автоматов-преобразователей над полугруппами

Захаров В. А.[1], **Темербекова Г.Г.**

получена 15 августа 2016

Аннотация. Автоматы-преобразователи над полугруппами можно использовать в качестве модели последовательных реагирующих программ, работающих в постоянном взаимодействии со своим окружением. Получив очередную порцию данных, реагирующая программа выполняет некоторую последовательность действий и предъявляет результат. Такие программы возникают при проектировании компьютерных драйверов, алгоритмов, работающих в оперативном режиме, сетевых коммутаторов. Во многих случаях проблема верификации программ такого рода может быть сведена к задачам минимизации и проверки эквивалентности конечных автоматов-преобразователей. Минимизация преобразователей над полугруппами проводится в три этапа. Вначале для всех состояний преобразователя вычисляются наибольшие общие левые делители. Затем все вычисленные делители "поднимаются вверх" по переходам преобразователя, и в результате образуется приведенный преобразователь. Наконец, для минимизации приведенных преобразователей применяются методы минимизации классических конечных автоматов-распознавателей.

Ключевые слова: реагирующая система, автомат-преобразователь, полугруппа, минимизация, проверка эквивалентности

Для цитирования: Захаров В. А., Темербекова Г.Г., "О минимизации конечных автоматов-преобразователей над полугруппами", *Моделирование и анализ информационных систем*, **23**:6 (2016), 741–753.

Об авторах:
Захаров Владимир Анатольевич, orcid.org/0000-0002-3794-9565, доктор физ.-мат. наук, профессор,
Московский государственный университет им. М.В. Ломоносова,
факультет ВМК, Ленинские горы, д. 1, стр. 52, ГСП-1, Москва, 119991, Россия, e-mail: zakh@cs.msu.ru

Темербекова Гульгайша Габдуловна, orcid.org/0000-0002-5856-8788, магистр,
Московский государственный университет им. М.В. Ломоносова,
факультет ВМК, Ленинские горы, д. 1, стр. 52, ГСП-1, Москва, 119991, Россия, e-mail: gulgaisha93@mail.ru

Благодарности:
[1]Работа выполнена при финансовой поддержке исследовательской программы НИУ ВШЭ в 2016 г. и гранта РФФИ №16-01-00546

Введение

Автоматы-преобразователи (трансдьюсеры) представляют собой расширения конечных автоматов, предназначенные для моделирования функций над строками и списками. Область их применения охватывает разделы от тестирования и оптимизации программ [1, 15] до компьютерной лингвистики [10]. В программировании автоматы-преобразователи служат удобной моделью для разнообразных драйверов,

проводящих манипуляции со строками, преобразование изображений, коммутацию и сортировку потоков данных и др. С помощью этой модели можно конструировать композиции драйверов, анализировать их поведение, проводить тестирование. Преобразователи нашли применение в некоторых методах верификации параметризованных моделей распределенных систем: конфигурации системы с произвольным неограниченным числом процессов представимы в виде слов некоторого конечного алфавита, и отношения переходов между конфигурациями можно задать при помощи автоматов-преобразователей, работающих над словами этого алфавита [18]. Понятно, что чем проще устроены эти преобразователи, тем эффективнее работают алгоритмы верификации регулярных моделей такого рода. В статье [14] предложено использовать в качестве модели коммуникационных протоколов автоматы-преобразователи, работающие над битовыми строками, и решать проблему верификации протоколов как задачу проверки эквивалентности двух преобразователей, один из которых моделирует протокол, а другой описывает его спецификацию. Приведенные примеры свидетельствуют о том, что алгоритмы проверки эквивалентности и уменьшения размеров автоматов-преобразователей востребованы для решения задач проектирования, верификации и оптимизации некоторых программ.

Автоматы-преобразователи могут служить простой формальной моделью последовательных реагирующих программ, работающих во взаимодействии со своим окружением. После получения очередной порции данных или запроса реагирующая программа выполняет некоторую последовательность действий. При достижении определенных контрольных точек программа выдает сложившийся к этому моменту результат вычисления в качестве отклика на запросы. Поскольку разные последовательности действий могут приводить к одному и тому же результату, нам потребуется более изощренная интерпретация действий программы, нежели простое представление о них как о словах в некотором алфавите. Базовые действия программы можно рассматривать как порождающие элементы некоторой полугруппы; тогда результатом вычисления будет элемент полугруппы, представленный композицией выполненных простейших действий.

Рассмотрим в качестве примера радиоуправляемый робот, способный совершать шаги в любом из четырех направлений N, E, S, W. При получении управляющего сигнала syg в состоянии q он должен выполнить последовательность шагов (например N, N, W, S) и перейти в следующее состояние q'. Достигнув некоторого особого состояния q_{fin}, робот сообщает о своем местоположении. Наиболее естественная модель вычислений, которую можно использовать при проектировании такого робота и исследовании его поведения, — это автомат-преобразователь, оперирующий над свободной абелевой группой ранга 2. Другим примером может служить сетевой коммутатор, на вход которого поступают потоки пакетов, перемежающиеся с командами управления. Руководствуясь таблицей коммутации пакетов, это сетевое устройство отправляет модифицированные копии каждого пакета в тот или иной выходной порт. Команды управления вносят изменения в таблицу коммутации. Модификация и коммутация пакетов — это элементарные действия коммутатора. Когда два пакета из разных потоков коммутируются в разные выходные порты, соответствующие действия могут выполняться в произвольном порядке. Поэтому такой коммутатор можно моделировать конечным автоматом-преобразователем, работающим над частично коммутативной полугруппой (множеством трасс [6]).

В данной статье исследуются задачи минимизации и проверки эквивалентности конечных автоматов-преобразователей, работающих над полугруппами. Изучение этих задач в основном проводилось для классических преобразователей, работающих над словами. В [9] удалось установить, что проблема эквивалентности неразрешима для недетерминированных преобразователей. Но этот эффект возникает лишь тогда, когда входные слова могут иметь неограниченно много образов. Для ограниченно недетерминированных преобразователей проблема эквивалентности разрешима; решение достигается для детерминированных преобразователей [3] за полиномиальное время [8], а для функциональных [2] и k-значных преобразователей [5] за экспоненциальное время [17]. Более общий метод проверки эквивалентности [19] позволил получить аналогичные результаты для автоматов-преобразователей, которые работают над полугруппами, вложимыми в разрешимые группы.

Начало исследованию задачи минимизации конечных автоматов-преобразователей было положено в статье [12], но приемлемое решение было впервые получено в работе [11]. Предложенный в ней алгоритм минимизации был исправлен и улучшен в статьях [4, 13]. В работе [7] была предпринята попытка адаптировать этот алгоритм к взвешенным преобразователям, используемым в компьютерной лингвистике. Другой подход к решению задачи минимизации был предложен в статье [20]: при помощи алгоритма проверки эквивалентности, разработанного в статье [19], задачу минимизации удалось решить для преобразователей, работающих над группами. Для этого метода требуется, чтобы все элементы полугруппы были обратимы.

В данной статье показано, как обобщить метод минимизации, предложенный в статье [11], применительно к преобразователям, работающим над упорядоченными полугруппами. Минимизация преобразователя π, оперирующего над полугруппой S, проводится в три этапа. Вначале для каждого состояния q находится наибольший общий левый делитель $gcd(\pi, q)$ всех элементов полугруппы S, которые вычисляются на прогонах π из состояния q. Затем все $gcd(\pi, q)$ "поднимаются вверх" по переходам преобразователя; в результате образуется приведенный преобразователь π', у которого все $gcd(\pi, q)$ равны нейтральному элементу e полугруппы S. И, наконец, для минимизации приведенных преобразователей применяются методы минимизации классических конечных автоматов-распознавателей (см. [16]).

1. Автоматы-преобразователи над полугруппами

Пусть заданы два конечных множества $\mathcal{C}$ и $\mathcal{A}$. Элементы множества $\mathcal{C}$ будем называть *входными сигналами*; их нужно рассматривать как сообщения (команды управления, показания датчиков, пакеты данных и др.), поступающие на вход реагирующей системы от окружающей среды. Мы абстрагируемся от природы и структуры сигналов. Конечные последовательности входных сигналов (слова в алфавите $\mathcal{C}$) будем называть *потоками сигналов*. Обозначим записью $\mathcal{C}^*$ множество всевозможных потоков сигналов, а записью uv — конкатенацию потоков сигналов u и v.

Элементы множества $\mathcal{A}$ будем называть *простыми действиями*; к их числу относятся операции обработки данных, отправления сообщений и др. Конечные по-

следовательности простых действий, представляющие собой слова в алфавите $\mathcal{A}$, будем называть *составными действиями*.

Для интерпретации действий воспользуемся полугруппами. Рассмотрим полугруппу $(S, e, \circ)$, порожденную множеством простых действий $\mathcal{A}$, в которой e обозначает нейтральный элемент, а $\circ$ — полугрупповую операцию. Элементы полугруппы S выступают в роли *состояний данных*. Применив простое действие a к состоянию данных s, получаем результат $s \circ a$. Составное действие $h = a_1 a_2 \dots a_k$ представляет собой композицию $[h]_S = a_1 \circ a_2 \circ \dots \circ a_k$ простых действий системы. Индекс S может быть опущен, если из контекста понятно, о какой полугруппе идет речь.

Детерминированный *автомат-преобразователь* с конечным числом состояний над множеством сигналов $\mathcal{C}$ и множеством базовых действий $\mathcal{A}$ — это размеченная система переходов $\pi = (\mathcal{C}, \mathcal{A}, Q, q_0, F, T, h_0, E)$, состоящая из

- конечного множества *состояний управления* Q,
- *начального состояния* $q_0 \in Q$,
- множества *состояний выхода* $F \subseteq Q$,
- *функции переходов* $T : Q \times \mathcal{C} \to Q \times \mathcal{A}^*$,
- *инициализирующего действия* $h_0 \in \mathcal{A}^*$,
- *функции финализации* $E : F \to \mathcal{A}^*$.

Каждая четверка (q, c, q', h), удовлетворяющая равенству $T(q, c) = (q', h)$, называется *переходом* и традиционно обозначается записью $q \xrightarrow{c,h} q'$. *Размером* $|\pi|$ автомата-преобразователя π называется число $|Q|$ его состояний управления.

Автоматы-преобразователи можно использовать в качестве формальных моделей последовательных реагирующих систем. В начале работы реагирующей системы происходит ее инициализация, которая состоит в выполнении последовательности действий h_0. Далее реагирующая система может функционировать неограниченно долго, обрабатывая поступающие на ее вход сообщения (сигналы). Обработка сообщений сопряжена с выполнением переходов. Каждый переход $q \xrightarrow{c,h} q'$ соответствует элементарному шагу вычислений: система, пребывающая в состоянии q, приняв на входе сообщение c, выполняет последовательность действий h и переходит в новое состояние q'. Стороннему наблюдателю доступны лишь те результаты вычислений системы, которые образуются при достижении состояний выхода. В них реагирующая система выполняет финализирующие действия и формирует отклик: мобильный робот сообщает о своем местоположении, блочный шифратор отправляет очередной блок шифртекста, сетевой драйвер завершает обработку пакета данных. По откликам можно судить о функции, реализуемой реагирующей программой. Более формально это можно определить при помощи следующих понятий.

Прогоном автомата-преобразователя π на потоке сигналов $w = c_1 c_2 \dots c_n$ из состояния управления q называется последовательность переходов

$$q \xrightarrow{c_1,h_1} q_1 \xrightarrow{c_2,h_2} q_2 \xrightarrow{c_3,h_3} \dots \xrightarrow{c_n,h_n} q' \ , \tag{1}$$

которую будем обозначать записью $q \xrightarrow{w,h}_* q'$, где $h = h_1 h_2 \dots h_n$. Если состояние управления q является инициальным, то прогон также называется *инициальным*, а если $q' \in F$, то прогон называется *финальным*. Прогон, являющийся одновременно инициальным и финальным, называется *полным*. Для полного прогона

$q \xrightarrow{w,h}_* q'$ автомата-преобразователя π, действия которого интерпретируются в полугруппе S, состояние данных $[h_0 hE(q')]_S$ считается *результатом* этого прогона. Таким образом, поведение реагирующей системы, которая моделируется автоматом-преобразователем π, характеризуется частичной функцией $\pi : \mathcal{C}^* \to S$; ее значения для потока сигналов w определяются соотношением

$$\pi(w) = \begin{cases} [h_0 hE(q')], \text{ если преобразователь } \pi \text{ имеет полный прогон } q \xrightarrow{w,h}_* q', \\ \text{не определено, в противном случае.} \end{cases}$$

Состояние управления q преобразователя π считается *полезным*, если через него проходит хоть один полный прогон. Бесполезные состояния не влияют на функцию, вычисляемую преобразователем, и могут быть удалены. Далее мы будем полагать, что все состояния рассматриваемых преобразователей являются полезными.

Для заданной полугруппы $(S, e, \circ)$ преобразователи π_1 и π_2 называются *S-эквивалентными*, если равенство $\pi_1(w) = \pi_2(w)$ выполняется для любого потока сигналов w. Отношение S-эквивалентности условимся обозначать записью $\pi_1 \sim_S \pi_2$. Задача проверки эквивалентности автоматов-преобразователей над полугруппой S состоит в том, чтобы для произвольной заданной пары преобразователей π_1 и π_2 выяснить, являются ли они S-эквивалентными. Автомат-преобразователь π' называется *S-минимальным*, если неравенство $|\pi'| \leq |\pi|$ выполняется для любого преобразователя π, S-эквивалентного преобразователю π'. Задача минимизации автоматов-преобразователей над полугруппой S состоит в том, чтобы для произвольного заданного преобразователя π построить S-эквивалентный ему S-минимальный преобразователь π'.

Для некоторых моделей вычислений задачи проверки эквивалентности и минимизации взаимосвязаны. Так, например, разделив при помощи алгоритма проверки эквивалентности все состояния детерминированного конечного автомата-распознавателя (автомата Рабина-Скотта) на классы эквивалентности, можно легко построить минимальный автомат, а для проверки эквивалентности двух детерминированных конечных автоматов достаточно провести их минимизацию и проверить изоморфность полученных минимальных автоматов. Аналогичный эффект имеет место для конечных автоматов-преобразователей, работающих над свободными полугруппами [11], над группами [20], и, как показано в данной статье, над полугруппами действий, которые обладают некоторыми свойствами, «естественными» как с алгебраической, так и с вычислительной точки зрения.

2. Унифицированные автоматы-преобразователи

Вначале при помощи простого приема унифицируем действия инициализации и финализации преобразователей. Пусть задан конечный автомат-преобразователь $\pi = (\mathcal{C}, \mathcal{A}, Q, q_0, F, T, h_0, E)$. Расширим множество сигналов $\mathcal{C}$, введя два новых сигнала $init$ и fin; положим $\mathcal{C}_0 = \mathcal{C} \cup \{init, fin\}$. Расширим множество состояний управления Q, введя два новых состояния *start* и *stop*; положим $Q_0 = Q \cup \{start, stop\}$. Доопределим функцию переходов T на новых элементах области определения; по-

ложим

$$T_0(q,x) \;=\; \begin{cases} T(q,x), \text{ если } q \in Q \text{ и } x \in \mathcal{A}, \\ (q_0, h_0), \text{ если } q = start \text{ и } x = init, \\ (stop, E(q)), \text{ если } q \in F \text{ и } x = fin, \\ \text{не определено, в остальных случаях.} \end{cases}$$

В итоге получим автомат-преобразователь $\pi_0 = (\mathcal{C}_0, \mathcal{A}, Q_0, start, \{stop\}, T_0, \varepsilon, E_0)$, в котором определена новая функция финализации E_0, принимающая значение $E_0(stop) = \varepsilon$. Такой преобразователь назовем *унифицированным*. Как видно из приведенного описания, для любого потока сигналов $w = c_1c_2\ldots c_n$ преобразователь π имеет полный прогон (1) тогда и только тогда, когда унифицированный преобразователь π_0 для потока сигналов $w' = init, c_1c_2\ldots c_n, fin$ имеет полный прогон

$$start \xrightarrow{init,h_0} q \xrightarrow{c_1,h_1} q_1 \xrightarrow{c_2,h_2} q_2 \xrightarrow{c_3,h_3} \cdots \xrightarrow{c_n,h_n} q' \xrightarrow{fin,E(q')} stop\ . \tag{2}$$

Отсюда следует

Утверждение 1. *Для любой пары автоматов-преобразователей π' и π'' верно соотношение $\pi' \sim_S \pi'' \iff \pi'_0 \sim_S \pi''_0$.*

Это утверждение позволяет ограничиться исследованием задач проверки эквивалентности и минимизации только для унифицированных автоматов-преобразователей, которые не выполняют действий инициализации и финализации. Каждому такому преобразователю $\pi_0 = (\mathcal{C}_0, \mathcal{A}, Q_0, start, \{stop\}, T_0, \varepsilon, E_0)$, работающему над полугруппой $(S, e, \circ)$, можно сопоставить детерминированный автомат-распознаватель $A_{\pi_0} = (\mathcal{C}_0 \times S, Q_0, start, \{stop\}, \varphi)$ над алфавитом (в общем случае бесконечным) пар $\mathcal{C}_0 \times S$. Функция переходов $\varphi : Q_0 \times (\mathcal{C}_0 \times S) \to Q_0$ этого автомата определяется следующим соотношением: $\varphi(q,(c,s)) = q' \iff T_0(q,c) = (q',h) \wedge s = [h]$. На вход автомата, пребывающего в начальном состоянии $start$, поступает конечная последовательность пар $\alpha = (c_1,s_1),(c_2,s_2),\ldots,(c_n,s_n)$. Она допускается автоматом, если после ее прочтения автомат переходит в состояние выхода $stop$. Нетрудно заметить, что автомат A_{π_0} допускает последовательность пар α в том и только том случае, когда преобразователь π_0 имеет такой полный прогон (2), для которого равенство $[h_i] = s_i$ выполняется для каждого $i, 1 \le i \le n$. Унифицированные преобразователи π'_0 и π''_0 назовем *строго эквивалентными* на полугруппе S (и обозначим это отношение записью $\pi'_0 \approx_S \pi''_0$), если автоматы-распознаватели $A_{\pi'_0}$ и $A_{\pi''_0}$ допускают одно и то же множество слов. Из определения строгой эквивалентности следует

Утверждение 2. *Для любой пары унифицированных автоматов-преобразователей π'_0 и π''_0 верно соотношение $\pi'_0 \approx_S \pi''_0 \Rightarrow \pi'_0 \sim_S \pi''_0$.*

Обратное соотношение в общем случае неверно. Ключевая идея решения задачи минимизации для автоматов-преобразователей состоит в том, чтобы для некоторого класса полугрупп S суметь выделить семейство приведенных преобразователей, удовлетворяющих следующим двум требованиям:

1) для каждого преобразователя π можно эффективно построить S-эквивалентный приведенный преобразователь π' того же самого размера, т.е. $|\pi| = |\pi'|$,

2) для каждой пары приведенных преобразователей π' и π'' верно соотношение $\pi' \sim_S \pi'' \iff \pi' \approx_S \pi''$.

Тогда для минимизации преобразователя π нужно построить эквивалентный приведенный преобразователь π', а затем, применив любой из известных методов минимизации автоматов-распознавателй (см., например, [16]), минимизировать детерминированный конечный автомат-распознаватель $A_{\pi'}$. Тот преобразователь π'', который соответствует построенному минимальному автомату-распознавателю, и будет являться результатом минимизации исходного преобразователя π. А для проверки выполнимости отношения $\pi_1 \sim_S \pi_2$ достаточно построить S-эквивалентные приведенные преобразователи π_1' и π_2', а затем проверить эквивалентность детерминированных конечных автоматов-распознавателей $A_{\pi_1'}$ и $A_{\pi_2'}$.

3. Упорядоченные левосократимые полугруппы

В этом разделе перечислены требования, которым должна удовлетворять полугруппа $(S, e, \circ)$, чтобы для нее можно было осуществить описанную выше стратегию минимизации детерминированных автоматов-преобразователей.

На множестве S элементов полугруппы определим бинарное отношение $\preceq_S$ следующим образом: для любой пары элементов s_1, s_2 отношение $s_1 \preceq_S s_2$ выполняется тогда и только тогда, когда для некоторого элемента s имеет место равенство $s_1 \circ s = s_2$. Полугруппа называется *упорядоченной*, если $\preceq_S$ является отношением частичного порядка на множестве S. Далее мы будем опускать индекс в обозначении отношения $\preceq_S$, если из контекста ясно, о какой полугруппе идет речь.

Первое требование, предъявляемое к рассматриваемой полугруппе, таково.

Req1: Частично упорядоченное множество $(S, \preceq)$ является фундированной решеткой, в которой для каждой пары элементов $[h_1]$ и $[h_2]$ эффективно вычислима их точная нижняя грань.

Будем использовать записи $s_1 \vee s_2$ и $s_1 \wedge s_2$ для обозначения соответственно точной нижней грани и точной верхней грани элементов s_1 и s_2. По сути дела, $s_1 \vee s_2$ — это наибольший общий левый делитель элементов s_1 и s_2, а $s_1 \wedge s_2$ — это наименьшее общее кратное элементов s_1 и s_2. Из определения отношения $\preceq$ следует, что справедлив закон левой дистрибутивности операции композиции действий относительно операции взятия точной нижней грани: $s \circ s_1 \vee s \circ s_2 = s \circ (s_1 \vee s_2)$. Нейтральный элемент полугруппы e является наименьшим элементом решетки $(S, \preceq)$, но у решетки может не оказаться наибольшего элемента. Мы добавим к множеству S еще один мнимый элемент τ, для которого будем полагать справедливыми равенства $s \circ \tau = \tau \circ s = \tau$ для всякого элемента s из S. Ясно, что $s \preceq \tau$ для любого s из S. Положим $S_\tau = S \cup \{\tau\}$. Таким образом, если рассматриваемая полугруппа удовлетворяет требованию **Req1**, то частично упорядоченное множество $(S_\tau, \preceq)$ является полной фундированной решеткой. Для всякого множества S' элементов этой решетки условимся обозначать записью $\bigvee S'$ точную нижнюю грань множества S'.

Второе требование касается разрешимости линейных уравнений в рассматриваемой полугруппе $(S, e, \circ)$.

Req2: Существует алгоритм решения уравнений вида $[g] \circ X = [h]$ для любой заданной пары действий $g, h \in \mathcal{A}^*$.

Заметим, что если рассматриваемая полугруппа удовлетворяет требованиям **Req1** и **Req2**, то проблема тождества $[g] \stackrel{?}{=} [h]$ в этой полугруппе разрешима.

Последнее требование касается свойства сократимости. Полугруппа называется *левосократимой*, если всякое уравнение вида $s \circ X = \widehat{s}$ имеет не более одного решения, т.е. соотношение $s \circ s' = s \circ s'' \;\Rightarrow\; s' = s''$ верно для любой тройки s, s', s''.

Req3: $(S, e, \circ)$ — левосократимая полугруппа.

Многие полугруппы, используемые в тех или иных моделях в теории вычислений, удовлетворяют перечисленным выше требованиям. Например, требованиям **Req1**–**Req3** удовлетворяют частично коммутативные моноиды (трассы) [6], при помощи которых можно описывать поведение систем взаимодействующих процессов.

4. Наибольшие общие делители состояний

Минимизация автоматов-преобразователей проводится в три этапа. Вначале для каждого состояния управления q униформного преобразователя π_0 нужно найти наибольшие общие делители всех состояний данных, которые вычисляются на финальных прогонах из q.

Предположим, что униформный преобразователь π_0 имеет множество состояний управления $Q_0 = \{q_1, q_2, \ldots, q_n\}$. Для каждого состояния управления q_i преобразователя π_0 сформируем множество $S(\pi_0, q_i) = \{[h] : q_i \xrightarrow{w,h}_* stop\}$ результатов финальных прогонов, начинающихся из q_i. Точную нижнюю грань $gcd(\pi_0, q_i) = \bigvee S(\pi_0, q_i)$ будем называть *наибольшим общим делителем* состояния управления q_i; обозначим записью $GCD(\pi_0)$ набор $(gcd(\pi_0, q_1), gcd(\pi_0, q_2), \ldots, gcd(\pi_0, q_n))$ наибольших общих делителей всех состояний преобразователя. Чтобы вычислить наибольшие общие делители всех состояний управления преобразователя π_0, введем оператор $\Psi_{\pi_0} : S_\tau^n \to S_\tau^n$, значения которого определяются так: для каждого набора элементов $(s_1, s_2, \ldots, s_n)$ из S_τ^n будем полагать, что $\Psi_{\pi_0}(s_1, s_2, \ldots, s_n) = (s'_1, s'_2, \ldots, s'_n)$, где

$$s'_i = \begin{cases} e, \text{ если } q_i = stop, \\ \bigvee\{[h] \circ s_j : T_0(q_i, c) = (q_j, h), c \in \mathcal{C}_0\}, \text{ в противном случае,} \end{cases}$$

для каждого $i, 1 \le i \le n$. Частичный порядок $\preceq$ можно естественным образом распространить на множество наборов S_τ^n: $(s_1, s_2, \ldots, s_n) \preceq (s'_1, s'_2, \ldots, s'_n) \iff \forall i : s_i \preceq s'_i$. Максимальным набором здесь является набор $\top = (\tau, \tau, \ldots, \tau)$.

Утверждение 3. *Если полугруппа $(S, e, \circ)$ удовлетворяет требованию **Req1**, то оператор Ψ_{π_0} является монотонным.*

Справедливость утверждения непосредственно следует из определения оператора Ψ_{π_0}. Поскольку решетка $(S_\tau, \preceq)$ полна, по теореме Кнастера–Тарского оператор Ψ_{π_0} имеет наибольшую неподвижную точку $gfp(\Psi_{\pi_0})$. По теореме Клини она является пределом убывающей последовательности наборов $\top \succeq_S \Psi_{\pi_0}(\top) \succeq_S \Psi_{\pi_0}(\Psi_{\pi_0}(\top)) \succeq_S \ldots$. Так как согласно требованию **Req1** решетка $(S, \preceq)$ является фундированной, существует такое k, для которого $\Psi_{\pi_0}^k(\top) = \Psi_{\pi_0}^{k+1}(\top) = gfp(\Psi_{\pi_0})$. Значит, $gfp(\Psi_{\pi_0})$ вычисляется эффективно.

Утверждение 4. *Если полугруппа $(S, e, \circ)$ удовлетворяет требованию **Req1**, то $gfp(\Psi_{\pi_0}) = GCD(\pi_0)$.*

Доказательство. 1). Если $q_i = stop$, то $S(\pi_0, q_i) = \{e\}$, и поэтому $gcd(\pi_0, q_i) = e$. В противном случае $S(\pi_0, q_i) = \bigcup\limits_{c \in \mathcal{C}_0} \{[h] \circ [g] : T_0(q_i, c) = (q_j, h), g \in S(\pi_0, q_j)\}$, и по закону левой дистрибутивности $\circ$ относительно $\vee$ имеем

$$gcd(\pi_0, q_i) = \bigvee \{[h] \circ gcd(\pi_0, q_j) : T_0(q_i, c) = (q_j, h), c \in \mathcal{C}_0\}.$$

Следовательно, $GCD(\pi_0)$ — это неподвижная точка оператора Ψ_{π_0}.

2). Допустим, что $gfp(\Psi_{\pi_0}) = (s'_1, s'_2, \ldots, s'_n)$. Рассмотрим какой-либо финальный прогон $q_i \xrightarrow{w,h}_* stop$ преобразователя π_0. Индукцией по длине этого прогона можно показать, что $s'_i \preceq [h]$. Если $q_i = stop$, то по определению оператора Ψ_{π_0} неравенство $s'_i = e \preceq [h]$ справедливо для любого действия h. Предположим, что прогон имеет вид $q_i \xrightarrow{c,g} q_k \xrightarrow{w',h'}_* stop$. Тогда согласно индуктивной гипотезе и определению оператора Ψ_{π_0} справедливы неравенства $s'_i \preceq [g] \circ s'_k \preceq [g] \circ [h'] = [gh'] = [h]$.

Так как неравенство $s'_i \preceq [h]$ соблюдается для каждого состояния данных $[h]$ из множества $S(\pi_0, q_i)$, верно неравенство $s'_i \preceq gcd(\pi_0, q_i)$. Таким образом, $gfp(\Psi_{\pi_0}) \preceq GCD(\pi_0)$, и, значит, $gfp(\Psi_{\pi_0}) = GCD(\pi_0)$. □

5. Редукция преобразователей

На следующем этапе минимизации проводится редукция автомата-преобразователя. Униформный преобразователь π_0, который работает над полугруппой, удовлетворяющей требованию **Req1**, назовем *приведенным*, если $gcd(\pi_0, q) = e$ для любого состояния q, отличного от начального состояния $start$.

Теорема 1. *Если полугруппа $(S, e, \circ)$ удовлетворяет требованиям **Req1**–**Req3**, то для каждого унифицированного преобразователя π_0 можно эффективно построить S-эквивалентный приведенный преобразователь π'_0 такого же размера.*

Доказательство. Для каждого перехода $q_i \xrightarrow{c,h} q_j$ в преобразователе π_0 рассмотрим уравнение $gcd(\pi_0, q_i) \circ X = [h] \circ gcd(q_j)$. Так как $S(\pi_0, q_i) \supseteq \{[h] \circ s : s \in S(\pi_0, q_j)\}$, из определения наибольшего общего делителя следует неравенство $gcd(\pi_0, q_i) \preceq [h] \circ gcd(\pi_0, q_j)$. Значит, указанное уравнение обязательно имеет решение $X = g_{c,i}$, которое согласно требованию **Req2** можно вычислить эффективно. Заметим, что для перехода $start \xrightarrow{init,h_0} q_1$ соответствующее уравнение имеет решение $X = e$. Образуем преобразователь π'_0 из преобразователя π_0 заменой каждого перехода $q_i \xrightarrow{c,h} q_j$ переходом $q_i \xrightarrow{c,g_{c,i}} q_j$ в случае $q_i \neq start$ или переходом $q_i \xrightarrow{init,g_0} q_j$, где $g_0 = gcd(\pi_0, start)$, в случае $q_i = start$.

Равенство $gcd(\pi_0, q_i) \circ [g_{c,i}] = [h] \circ gcd(\pi_0, q_j)$, обеспечивающее взаимосвязь между переходами $q_i \xrightarrow{c,h} q_j$ и $q_i \xrightarrow{c,g_{c,i}} q_j$ преобразователей π_0 и π'_0, можно распространить на прогоны этих преобразователей. Рассмотрим пару соответствующих прогонов преобразователей π_0 и π'_0 на одном и том же потоке сигналов $w = c_1 c_2 \ldots c_{m-1} c_m$:

$$q_1 \xrightarrow{c_1,h_1} q_2 \xrightarrow{c_2,h_2} \cdots q_{m-1} \xrightarrow{c_{m-1},h_{m-1}} q_m \xrightarrow{c_m,h_m} q_{m+1},$$
$$q_1 \xrightarrow{c_1,g_1} q_2 \xrightarrow{c_2,g_2} \cdots q_{m-1} \xrightarrow{c_{m-1},g_{m-1}} q_m \xrightarrow{c_m,g_m} q_{m+1}.$$

Учитывая устройство преобразователя π'_0, можно получить цепочку равенств

$$[h_1 \dots h_{m-1}h_m] \circ gcd(\pi_0, q_m) = [h_1h_2 \dots h_{m-1}] \circ [h_m] \circ [gcd(\pi_0, q_{m+1})] =$$
$$= [h_1h_2 \dots h_{m-1}] \circ [gcd(\pi_0, q_m)] \circ [g_m] = [h_1h_2 \dots h_{m-2}] \circ gcd(\pi_0, q_{m-1}) \circ [g_{m-1}g_m] = \cdots$$
$$\cdots = [h_1] \circ [gcd(\pi_0, q_2)] \circ [g_2 \dots g_{m-1}g_m] = gcd(\pi_0, q_1) \circ [g_1 \dots g_{m-1}g_m] \ .$$

Чтобы убедиться в том, что $\pi_0 \sim_S \pi'_0$, прежде всего заметим, что обе функции $\pi_0(\cdot)$ и $\pi'_0(\cdot)$ определены на одном и том же множестве потоков сигналов. Рассмотрим произвольный поток сигналов w, на котором определено значение $\pi_0(w)$, и полные прогоны $start \xrightarrow{init,h_0} q_1 \xrightarrow{w,h}_* stop$ и $start \xrightarrow{init,g_0} q_1 \xrightarrow{w,g}_* stop$ преобразователей π_0 и π'_0 на w. Так как $gcd(\pi_0, stop) = e$, имеет место следующая цепочка равенств:

$$\pi_0(w) = [h_0h] = [h_0] \circ [h] \circ [gcd(\pi_0, stop)] = [h_0] \circ [gcd(\pi_0, q_1)] \circ [g] =$$
$$= [gcd(\pi_0, start)] \circ [g] = [g_0] \circ [g] = \pi'_0(w) \ .$$

Следовательно, $\pi_0(w) = \pi'_0(w)$ для каждого потока сигналов w.

Чтобы убедиться в том, что π'_0 — приведенный преобразователь, рассмотрим произвольное состояние управления q_i, отличное от начального, и элемент $gcd(\pi_0, q_i) = \bigvee\{[h] : q_i \xrightarrow{w,h}_* stop\}$. Опираясь на установленную взаимосвязь между соответствующими прогонами преобразователей π_0 и π'_0, а также принимая во внимание очевидное равенство $gcd(\pi_0, stop) = e$, можно заметить, что

$$gcd(\pi_0, q_i) \circ gcd(\pi'_0, q_i) = \bigvee\{gcd(\pi_0, q_i) \circ [g] : q_i \xrightarrow{w,g}_* stop\} =$$
$$= \bigvee\{[h] \circ gcd(\pi_0, stop) : q_i \xrightarrow{w,h}_* stop\} = gcd(\pi_0, q_i) \ .$$

Поскольку S — левосократимая полугруппа, равенство $gcd(\pi_0, q_i) \circ gcd(\pi'_0, q_i) = gcd(\pi_0, q)$ влечет $gcd(\pi'_0, q_i) = e$. □

6. Минимизация приведенных преобразователей

На последнем этапе для минимизации приведенных преобразователей применяются методы минимизации детерминированных конечных автоматов-распознавателей на основании взаимосвязи между приведенными преобразователями и конечными автоматами, которая устанавливается в следующем утверждении.

Утверждение 5. *Пусть π'_0 и π''_0 — пара приведенных S-эквивалентных преобразователей, которые работают над полугруппой $(S, e, \circ)$, удовлетворяющей требованиям **Req1, Req3**, и пусть $start \xrightarrow{w,h'}_* q'_1 \xrightarrow{c,g'} q'_2$ и $start \xrightarrow{w,h''}_* q''_1 \xrightarrow{c,g''} q''_2$ — пара начальных прогонов этих преобразователей на некотором потоке сигналов wc, где $w \in \mathcal{C}_0^*, c \in \mathcal{C}_0$. Тогда $[g'] = [g'']$.*

Доказательство. Применим индукцию по длине потока сигналов wc. Обоснование базиса индукции и индуктивного перехода проводится по одной и той же схеме.

Так как все состояния преобразователей π'_0 и π''_0 полезные, преобразователь π'_0 имеет финальный прогон $q'_2 \xrightarrow{u,f'}_* stop$. Значит, преобразователь π'_0 имеет полный прогон $start \xrightarrow{w,h'}_* q'_1 \xrightarrow{c,g'} q'_2 \xrightarrow{u,f'}_* stop$. Поскольку $\pi'_0 \sim_S \pi''_0$, преобразователь π''_0 также имеет полный прогон $start \xrightarrow{w,h''}_* q''_1 \xrightarrow{c,g''} q''_2 \xrightarrow{u,f''} stop$, и при этом

$[h'g'f'] = [h''g''f'']$. Последнее равенство означает, что $[h'g'] \wedge [h''g''] \neq \tau$. Следовательно, существует тройка элементов s, s', s'', для которых выполняются равенства $[h'g'] \wedge [h''g''] = [h'g'] \circ s' = [h''g''] \circ s''$ и $[h'g'f'] = [h''g''f''] = ([h'g'] \wedge [h''g'']) \circ s$. Таким образом, $[h'g'] \circ s' \circ s = [h'g'f'] = [h''g''f''] = [h''g''] \circ s'' \circ s$.

Если проводится обоснование базиса индукции, то $h' = h'' = \varepsilon$. Если же проводится обоснование индуктивного перехода, то по индуктивному предположению $[h'] = [h'']$. И в том, и в другом случае закон левого сокращения приводит к равенству $[g'] \circ s' = [g''] \circ s''$ и неравенствам $s' \preceq [f']$ и $s'' \preceq [f'']$.

Заметим, что элементы s' и s'' не зависят от f' и f'', и поэтому указанные неравенства верны для любых элементов f' из $S(\pi'_0, q'_2)$ и f'' из $S(\pi''_0, q''_2)$. Отсюда следуют неравенства $s' \preceq gcd(\pi'_0, q'_2)$ и $s'' \preceq gcd(\pi''_0, q''_2)$. Состояния управления q'_2 и q''_2 не являются начальными, и поэтому согласно определению приведенных преобразователей верны равенства $gcd(\pi'_0, q'_2) = gcd(\pi''_0, q''_2) = e$. Таким образом, $s' = s'' = e$, и, следовательно, $[g'] = [g'']$. □

Из утверждения 5 следует

Теорема 2. *Если полугруппа $(S, e, \circ)$ удовлетворяет требованиям **Req1, Req3**, то для любой пары приведенных преобразователей π' и π'' справедливо соотношение*

$$\pi' \sim_S \pi'' \iff \pi' \approx_S \pi'' .$$

Теоремы 1 и 2 дают способ решения задач минимизации и проверки эквивалентности для детерминированных автоматов-преобразователей, работающих над полугруппами, которые удовлетворяют требованиям **Req1**–**Req3**. Для этого достаточно редуцировать анализируемые преобразователи, воспользовавшись теоремой 1, а затем обратиться к соответствующим конечным автоматам-распознавателям и алгоритмам решения упомянутых задач для этих автоматов.

7. Заключение

В данной статье предложена общая схема решения задач минимизации конечных автоматов-преобразователей, работающих над полугруппами специального вида. Так как преобразователи над полугруппами служат формальными моделями последовательных реагирующих программ, эта схема предоставляет один из подходов к решению задач оптимизации и верификации таких программ. В этом состоит теоретическая значимость полученных в статье результатов.

Вместе с тем, остаются вопросы, требующие дальнейшего исследования. Мы не оценивали сложность алгоритмов минимизации, построенных на основе предложенной схемы. Это объясняется тем, что сложность этих алгоритмов зависит не только от сложности решения вычислительных задач, упомянутых в требованиях **Req1**–**Req3** (вычисление наибольших общих делителей, решение линейных уравнений), но также от выбора подходящих структур данных для представления элементов рассматриваемых полугрупп. Эти вопросы относятся, прежде всего, к области вычислительной алгебры.

Особого внимания заслуживает вопрос о необходимых условиях, которым должна удовлетворять полугруппа, для того чтобы задача минимизации автоматов-преобразователей имела единственное решение. Как видно из результатов работы [20],

свойство упорядоченности полугруппы, фигурирующее в **Req1**, к числу таковых условий не относится.

Список литературы / References

[1] Alur R., Cerny P., "Streaming transducers for algorithmic verification of single-pass list-processing programs", *Proc. of 38-th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, 2011, 599–610.

[2] Blattner M, Head T., "Single-valued a-transducers", *J. of Comput. and Syst. Sci.*, **15**:3 (1977), 310–327.

[3] Blattner M, Head T., "The decidability of equivalence for deterministic finite transducers", *J. of Comput. and Syst. Sci.*, **19**:1 (1979), 45–49.

[4] Beal M.-P., Carton O., "Computing the prefix of an automaton", *Theoretical Informatics and Applications*, **34**:6 (2000), 503–514.

[5] Culik K., Karhumaki J., "The equivalence of finite-valued transducers (on HDTOL languages) is decidable", *Theor. Comput. Sci.*, **47** (1986), 71–84.

[6] Diekert V., Metivier Y., "Partial commutation and traces", *Handbook of formal languages*, **3**, 1997, 457–533.

[7] Eisner J., "Simpler and more general minimization for weighted finite-state automata", *Proc. of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, **1**, 2003, 64–71.

[8] Friedman E.P., Greibach S.A., "A polynomial time algorithm for deciding the equivalence problem for 2-tape deterministic finite state acceptors", *SIAM J. Comput.*, **11**:1 (1982), 166–183.

[9] Griffiths T., "The unsolvability of the equivalence problem for ε-free nondeterministic generalized machines", *J. of the ACM*, **15** (1968), 409–413.

[10] Mohri M., "Finite-state transducers in language and speech processing", *Comput. Ling*, **23**:2 (1997), 269–311.

[11] Mohri M., "Minimization algorithms for sequential transducers", *Theor. Comput. Sci.*, **234** (2000), 177–201.

[12] Reutenauer C., Schuzenberger M. P., "Minimization of rational word functions", *SIAM J. of Comput.*, **20**:4 (1991), 669—685.

[13] Shofrutt C., "Minimizing subsequential transducers: a survey", *Theor. Comput. Sci.*, **292**:1 (2003), 131—143.

[14] Thakkar J., Kanade A., Alur R., "A transducer-based algorithmic verification of retransmission protocols over noisy channels", *Proc. of IFIP Joint International Conference on Formal Techniques for Distributed Systems, LNCS*, **7892** (2013), 209–224.

[15] Veanes M., Hooimeijer P., Livshits B., et al., "Symbolic finite state transducers: algorithms and applications", *Proc. of the 39th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. ACM SIGPLAN Notices*, **147** (2012), 137–150.

[16] Watson B. W., "A taxonomy of finite automata minimization algorithm", *Computing Science Report. Eindhoven University of Technology*, **93/44** (2005), 1–32.

[17] Weber A., "Decomposing finite-valued transducers and deciding their equivalence", *SIAM Journal on Computing*, **22**:1 (1993), 175–202.

[18] Wolper P., Boigelot B., "Verifying systems with infinite but regular state spaces", *Proc. 10th Int. Conf. on Computer Aided Verification (CAV-1998), LNCS*, **1427** (1998), 88—97.

[19] Zakharov V. A., "Equivalence checking problem for finite state transducers over semigroups", *Proc. of the 6-th International Conference on Algebraic Informatics (CAI-2015). LNCS*, **9270** (2015), 208-221.

[20] Захаров В. А., Подымов В. В., “Применение алгоритмов проверки эквивалентности для оптимизации программ”, *Труды Института системного программирования*, **27**:3 (2015), 145–174; [Zakharov V. A., Podymov V. V., “Primenenije algoritmov proverki ekvivalentnisti dlya optimizacii program”, *Trudy insituta sistemnogo programmirovaniya*, **27**:3 (2015), 145–174, (in Russian).]

Zakharov V. A., Temerbekova G. G., "On the Minimization of Finite State Transducers over Semigroups", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 741–753.

DOI: 10.18255/1818-1015-2016-6-741-753

Abstract. Finite state transducers over semigroups are regarded as a formal model of sequential reactive programs that operate in the interaction with the environment. At receiving a piece of data a program performs a sequence of actions and displays the current result. Such programs usually arise at implementation of computer drivers, on-line algorithms, control procedures. In many cases verification of such programs can be reduced to minimization and equivalence checking problems for finite state transducers. Minimization of a transducer over a semigroup is performed in three stages. At first the greatest common left-divisors are computed for all states of the transducer, next the transducer is brought to a reduced form by pulling all such divisors ”upstream”, and finally a minimization algorithm for finite state automata is applied to the reduced transducer.

Keywords: reactive system, transducer, semigroup, minimization, equivalence checking

About the authors:
Vladimir A. Zakharov, orcid.org/0000-0002-3794-9565, PhD, professor
Lomonosov Moscow State University,
Faculty of Computational Mathematics and Cybernetics, GSP-1, 1-52 Leninskiye Gory, Moscow 119991, Russia, e-mail: zakh@cs.msu.ru

Gulgaysha G. Temerbekova, orcid.org/0009-0203-2514-7755, graduate student,
Lomonosov Moscow State University,
Faculty of Computational Mathematics and Cybernetics, GSP-1, 1-52 Leninskiye Gory, Moscow 119991, Russia, e-mail: gulgaisha93@mail.ru

Acknowledgments:
This work is supported by the Basic Research Program at the National Research University Higher School of Economics in 2016 and by RFBR grants №16-01-00546.

Моделирование и анализ информационных систем. Т. 23, №6 (2016), с. 754–766
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 754–766

DOI: 10.18255/1818-1015-2016-6-754-766

УДК 004.423.4+004.415.5

Формализм и языковые инструменты для описания семантики программных библиотек

Ицыксон В. М.

получена 5 сентября 2016

Аннотация. Статья посвящена вопросам спецификации структуры и поведения программных библиотек. Описываются существующие проблемы спецификации библиотек. Дается краткий обзор состояния дел в области формализации спецификации библиотек и библиотечных функций. Формулируются требования к создаваемому формализму. На основе требований предлагается формализм, позволяющий специфицировать все необходимые свойства библиотек, требуемые для автоматизации нескольких классов задач: обнаружение дефектов в программном обеспечении, миграция приложений в новое окружение, генерация программной документации. На базе формализма формулируются требования к языковым средствам спецификации библиотек. В заключении определяются дальнейшие направления исследований.

Ключевые слова: формальная спецификация, программная библиотека, поведенческое описание, программный дефект, язык спецификаций

Для цитирования: Ицыксон В. М., "Формализм и языковые инструменты для описания семантики программных библиотек", *Моделирование и анализ информационных систем*, **23**:6 (2016), 754–766.

Об авторах:
Ицыксон Владимир Михайлович, orcid.org/0000-0003-0276-4517, канд. техн. наук, доцент,
Санкт-Петербургский политехнический университет Петра Великого,
ул. Политехническая, д. 29, г. Санкт-Петербург, 195251, Россия, e-mail: vlad@icc.spbstu.ru

1. Введение

Программные библиотеки – ставший стандартом «де-факто» способ реализации компонентно-ориентированного подхода, при котором производитель программного обеспечения инкапсулирует определенную функциональность в виде набора функций, типов данных и пользовательского программного интерфейса доступа. Современные библиотеки являются чрезвычайно сложными объектами, функциональность которых зачастую сложнее, чем у использующих их приложений.

Ключевым отличием библиотек от обычных приложений является способ их использования. Приложения применяют пользователи, следуя инструкциям, руководствам по эксплуатации и встроенным справкам, им не требуются формальные спецификации, описывающие приложения. В то же время основными пользователями

библиотек являются другие программисты, которым для интеграции функциональности приложений и библиотек требуется абсолютно четкое понимание, как работает библиотека, как ее можно использовать, как она влияет на приложение, какие изменения вносятся в нее от версии к версии и т.п.

Каким образом производитель библиотеки обычно специфицирует библиотеку? Обычно используется один или несколько следующих способов:

- заголовочные файлы с комментариями;
- словесное описание интерфейса библиотеки;
- словесное описание поведения отдельных функций;
- словесное описание некоторых допустимых последовательностей вызовов функций;
- поставляемые разработчиком примеры использования.

Однако ни один из указанных способов не решает задачи формальной спецификации семантики библиотеки. Семантика библиотеки состоит из двух составляющих: семантика отдельных функций и допустимых способов совместного использования разных функций библиотеки. Семантика отдельных функций определяется условиями вызова функции, получаемым результатом, побочными действиями и влиянием на окружение. Обычно семантика функций описывается неформально в виде текстовых описаний. Допустимые способы совместного использования разных функций библиотеки в лучшем случае описываются авторами неформально в сопроводительной документации к библиотеке, иногда с помощью сопутствующих примеров использования.

То есть, на настоящий момент в индустрии программной инженерии отсутствует аппарат формализованного описания семантики программных библиотек. В условиях отсутствия формальной спецификации библиотек существует несколько классов задач, которые невозможно удовлетворительно решить в настоящее время:

- автоматическая проверка корректности использования библиотеки каким-либо приложением. Здесь под корректностью мы будем понимать выполнение приложением протокола доступа к библиотеке, предусмотренного разработчиком;
- обнаружение программных ошибок в многофайловых проектах, использующих сторонние библиотеки в случае недоступности исходного кода;
- анализ совместимости приложения и новой версии библиотеки;
- портирование приложения в новое библиотечное окружение.

Таким образом, целью данной работы является создание формализма, позволяющего строго описывать все необходимые аспекты библиотек.

2. Обзор предметной области

Проблема спецификации библиотек и сервисов исследуется довольно давно. Имеется достаточное число публикаций, предлагающих разные подходы к описанию спецификаций. Первые исследования были связаны в первую очередь с обеспечением интероперабельности, а основная цель создаваемых спецификаций была создать самодостаточное описание интерфейсов библиотек и сервисов, для возможности использовать их в разных языках программирования и различных операционных системах. Примерами таких спецификаций является язык IDL [1], а также многие его расширения MIDL [2] или, например, ADL [3]. Основное ограничение этих языков для спецификации библиотек – ограниченность детальным описанием API, без фокусировки на валидных вариантах использования библиотек. То есть основной упор делается на описание сигнатур функций и типов данных, а спецификации семантики всей библиотеки не уделяется должного внимания.

Одним из первых исследований в области спецификации интерфейсов компонентов является работа Алена и Гарлана [4], в которой авторы сводят задачу взаимодействия компонентов программной системы к задаче спецификации протоколов взаимодействия наподобие протоколов компьютерных сетей. В качестве основы формализма был взят аппарат взаимодействующих последовательных процессов (CSP) Ч. Хоара [5], который они соответствующим образом видоизменили. Введя в формализм специальные элементы – порты, коннекторы и роли, они сделали возможным отдельно специфицировать разные аспекты возможного взаимодействия компонентов между собой. Использование формализма позволяет частично решать задачу совместимости компонентов с помощью верификатора FDR [6].

Альфаро и Хензингер в статье 7 предлагают свой формализм для описания взаимодействующих компонентов, названный интерфейсными автоматами. В работе используется оптимистическое определение совместимости компонентов, основанное на использовании модели окружения. Авторами предлагаются формальные методы проверки оптимистической совместимости двух интерфейсных автоматов.

Отдельные исследования посвящены не столько спецификациям интерфейсов и библиотек, сколько механизмам их автоматизированного построения на основе анализа существующего программного обеспечения. Так, в статье [8] авторами предлагается подход к выводу спецификаций библиотек на основе статического извлечения предикатов. Авторы используют анализ потоков данных (data flow analysis) и анализ потока управления для извлечения предикатов, характеризующих функции интерфейса. Другой подход изложен в работе [9], где авторы предлагают использование динамического вывода спецификаций библиотек на основе модульного тестирования. Для этого используется формализм Datalog, в терминах которого определяются функции библиотеки, интерфейсы, типы данных и транзакции. Валидные последовательности вызовов функций задаются с помощью специальных предикатов. Вывод спецификаций основывается на анализе и обобщении (генерализации) результатов случайного модульного тестирования функций библиотеки.

Один из наиболее интересных подходов к описанию API библиотек и правил их применения используется в подходе SLAM, предложенном Microsoft Research для верификации драйверов. SLAM использует язык SLIC [10] для спецификации библиотек и правил взаимодействия программ и API. SLIC-спецификация используется

для инструментирования программы и/или библиотеки для дальнейшего динамического или статического контроля соответствия. Отсутствие семантических описаний внутренностей библиотек не позволяет использовать его для автоматизации миграции.

Г. Ливинс в статье «The future of library specification» [11], кроме известных подходов, связанных с неформальным документированием и формальным специфицированием, выделяет несколько косвенных подходов: спецификация через примеры использования, спецификация посредством исходных кодов библиотек и спецификация через модульные тесты. Основной вывод автора – спецификация библиотек должна объединять все перечисленные подходы.

В наших предыдущих работах [12], [13] мы также предлагали формализм для спецификации библиотек и язык, поддерживающий задание таких спецификаций. Однако в этом подходе варианты использования библиотек (поведение) описываются неявно, а язык не позволяет в полной мере задавать контракты функций и влияние функций на окружение. Таким образом, на настоящий момент не существует универсального подхода к спецификации библиотек, позволяющего:

- детально описывать внешний интерфейс библиотеки;
- задавать возможные протоколы использования библиотеки;
- специфицировать побочные действия библиотеки – влияние ее на окружение;
- явно вводить семантические описания поведения библиотек.

3. Особенность организации библиотек

Особенность использования библиотек заключается в том, что библиотека не является чисто функциональным объектом, может иметь внутреннее состояние и различные побочные эффекты, существенно влияющие на возможность вызова отдельных функций.

Введем классификацию библиотек функций с точки зрения их внутреннего состояния.

1. Библиотеки без внутреннего состояния
2. Библиотеки с внутренним состоянием библиотеки
3. Библиотеки с внутренним состоянием создаваемого объекта
4. Комбинированные библиотеки

К первому классу относятся библиотеки, содержащие чистые функции без побочных эффектов. К таким библиотекам относится, например, библиотека математических функций `math.h` стандартной библиотеки языка программирования C.

Ко второму классу относятся библиотеки, которые хранят свое состояние. То есть поведение отдельных функций зависит от того, в каком состоянии находится библиотека. Примером такой библиотеки может служить, например, часть библиотеки `stdlib.h`, обеспечивающая генерацию случайных чисел. Вызов `srand()` задаёт

начальное значение генератора, а `rand()` выдает очередное случайное число из последовательности, построенной на основе начального значения генератора.

К третьему классу относятся библиотеки, которые сохраняют контекст внутри создаваемого функциями библиотеки объекта. Таким объектом может быть, например, созданный сокет или дескриптор файла. В первом случае контекст содержит параметры сокета (IP-адреса, номера портов, состояние), во втором случае контекст содержит параметры файла, режим открытия и указатель на следующие читаемые данные.

Четвертый класс – наиболее общий, к нему относятся библиотеки, объединяющие особенности второго и третьего класса.

4. Формальная спецификация библиотек

Полная формальная спецификация библиотек должна описывать:

- сигнатуру всех функций, входящих в библиотеку;
- контракт каждой библиотечной функции (предусловия, постусловия, влияние на окружение);
- поведенческую модель библиотеки, учитывающую все возможные варианты использования функций библиотеки, специфицирующую, в том числе, поведение библиотеки при не валидном ее использовании.

Исходя из вышеперечисленного предлагается полную спецификацию библиотек определять как $\langle F, L\rangle$, где

- $F = \{F_i\}$ – множество функций библиотеки;
- L – поведенческое описание библиотеки.

Отдельная функция библиотеки F_i определяется как

$$< Name, Args, Res, Pre, Post, A, CondA, D, CondD, E, CondE >, \quad \text{где}$$

- $Name$ – имя функции;
- $Args$ – множество формальных аргументов функции;
- Res – результат функции;
- Pre – предусловия функции, выраженные формулой в логике первого порядка от аргументов $Args$;
- $Post$ – постусловия функции, выраженные формулой в логике первого порядка от аргументов $Args$ и Res;
- $A = \{A_i\}$ – множество семантических действий, производимых функцией;

- $CondA = \{CondA_i\}$ – множество условий выполнения семантических действий. Действие A_i выполняется во время работы функции тогда, когда выражение $CondA_i$ истинно.

- $D = \{D_i\}$ – множество запускаемых дочерних автоматов;

- $CondD = \{CondD_i\}$ – множество условий запуска дочерних автоматов. Автомат D_i запускается во время работы функции тогда, когда выражение $CondD_i$ истинно;

- $E = \{E_i\}$ – множество объектов окружения, изменяемых функцией (то есть объектов программы, использующей библиотеку);

- $CondE = \{CondE_i\}$ – множество условий изменения объектов окружения. Объект E_i изменяется во время работы функции тогда, когда выражение $CondE_i$ истинно.

Объединение всех множеств E объектов окружения, изменяемых всеми функциями, показывает множество всех объектов программы на которые влияет библиотека.

Поведенческое описание библиотеки будем представлять с помощью множества параметризируемых расширенных конечных автоматов (EFSM):

$$L = \{B, S_1(q, P), \ldots, S_n(q, P), I\}, \quad \text{где}$$

- B – основной расширенный конечный автомат, описывающий поведение всей библиотеки;

- S_i – i-й дочерний расширенный конечный автомат, запускаемый при достижении определенных условий;

- q – параметр – начальное состояние дочернего автомата;

- P – дополнительный параметр дочернего автомата;

- I – множество созданных при инициализации библиотеки экземпляров дочерних автоматов (например, для библиотеки работы с файлами это могут быть автоматы, соответствующие файловым потокам `sysin`, `sysout` и `syserr`).

Состояние основного автомата соответствует состоянию библиотеки, состояния дочерних автоматов соответствуют состоянию созданных объектов. Стимулами автоматов, форсирующими переход автомата из одного состояния в другое, являются вызовы функций API библиотеки. Отдельный автомат определяется как модифицированный расширенный конечный автомат

$$\langle Q, Q_0, X, V, C, C^A, C^D, T\rangle, \quad \text{где}$$

- Q – множество управляющих состояний автомата – состояния библиотеки или объектов библиотеки;

- Q_0 – непустое множество начальных состояний автомата. Начальных состояний может быть несколько для дочерних автоматов, так как при создании экземпляра автомата начальные состояния могут быть разные, для автомата B начальное состояние всегда одно;

- X – множество состояний завершения. Дочерние автоматы после попадания в такие состояния уничтожаются;

- V – множество внутренних переменных автомата;

- $C = \{C_i\}$ – множество стимулов – вызовов функций, C_i – вызов i-й функции; $C_i \in F$;

- $C^A = \{C_i^A\}$ – множество семантических действий автомата, C_i^A – семантические действия, инициируемые запуском i-й функции при условии истинности C_i^{CondA};

- $C^D = \{C_i^D\}$ – множество дочерних автоматов, C_i^D – дочерние автоматы, запускаемые i-й функцией при условии C_i^{CondD};

- T – отношение перехода.

Из-за ограниченности объема статьи формализм представлен без глубокой детализации. За рамками описания остались такие вопросы, как спецификация невалидного поведения, действия по умолчанию, типы данных и т.п. Эти вопросы более подробно будут рассмотрены в других работах.

Вообще для разработчика поведенческое описание библиотек более наглядно представлять не в теоретико-множественном виде, а в графическом виде.

Пример описания в графическом виде клиентской части библиотеки TCP-сокетов представлен на рис. 1. На рисунке сплошной линией показаны переходы автоматов, пунктирной линией – запуски дочерних автоматов, состояния завершения выделены темно-серым цветом. Автомат **L** описывает общее поведение библиотеки `bsd-socket`, которая в отличие от `WinSock` не требует инициализации. Побочным результатом вызова `socket()` является создание нового автомата **P**, соответствующего созданному сокету, обладающего своим жизненным циклом. При этом стоит отметить, что создаваемых автоматов может быть несколько, отличаться они будут только параметром запуска дочернего автомата (элемент, соответствующий вызову `socket()`).

Более сложный пример представлен на рис. 2. Здесь приведена графическая модель серверной части библиотеки протокола TCP библиотеки `bsd-socket`. Кроме автомата верхнего уровня, соответствующего библиотеке (**L**), на рисунке представлены два семейства автоматов: одно (**P**) – инкапсулирует свойства слушающих сокетов, второе (**S**) – созданных серверных сокетов.

Оба примера демонстрируют только поведенческое описание библиотек, без спецификации множества функций.

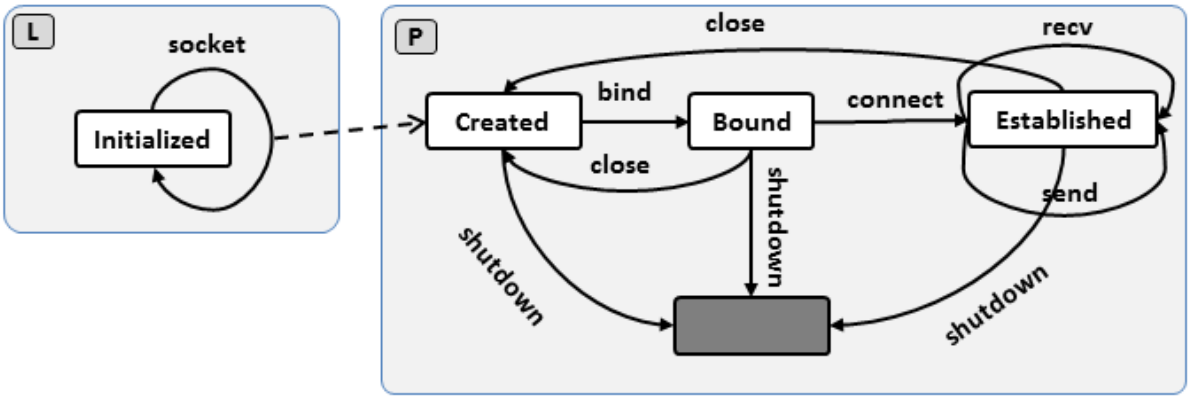


Рис. 1. Пример простого автомата, соответствующего клиентской части протокола TCP библиотеки `bsd-socket`
Fig. 1. An example of a simple FSM representing the client side of the TCP protocol of the bsd-socket library

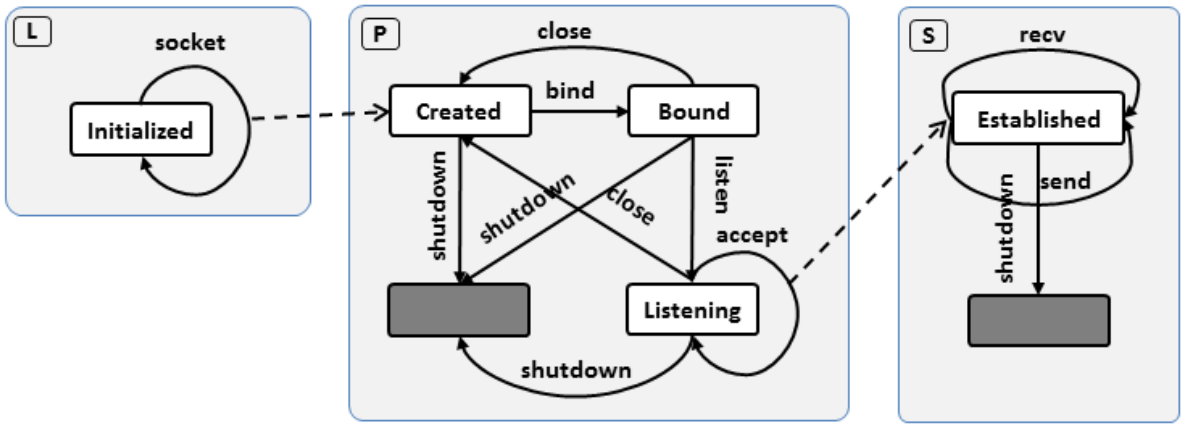


Рис. 2. Пример автомата, описывающего серверную часть протокола TCP библиотеки `bsd-socket`
Fig. 2. An example of a simple FSM representing the server side of the TCP protocol of the bsd-socket library

5. Перспективы использования разработанного формализма

5.1. Построение спецификаций

Предполагается два пути построения формальных спецификаций: с помощью авторов библиотек и сообществами разработчиков.

В первом случае спецификацию библиотеки создает ее разработчик. Для этого формируется язык описания спецификации библиотек (наподобие разработанного ранее языка PanLang [13]), средствами которого можно будет задать все свойства библиотеки, выражаемые разработанным формализмом. Требования к такому языку представлены в следующем разделе.

Во втором случае будут использованы методы вывода, базирующиеся на эксплуатации мирового программистского опыта (Empirical Software Engineering), когда структурная и часть поведенческой составляющих спецификаций строятся на основе анализа программных репозиториев (Mining Software Repositories). В этом случае формируется не вся спецификация, а только скелет, оставшуюся часть необходимо будет доработать вручную.

Методы анализа программных репозиториев для излечения скелетов спецификаций в настоящее время разрабатываются в научной группе автора статьи, их описание выходит за рамки данной работы.

5.2. Использование формальных спецификаций библиотек

Разработанный и описанный в данной статье формализм может использоваться в дальнейшем для решения целого ряда научно-технических задач. К ним относятся: автоматизация поиска дефектов в сложных многокомпонентных программных проектах, автоматизация портирования приложений на новые библиотеки и автоматизация создания программной документации.

В рамках решения задачи поиска программных дефектов спецификации библиотек применяются для сокращения размерности задачи обнаружения. Это достигается путем аппроксимации поведения библиотек и библиотечных функций укрупненным видимым поведением, заданным в спецификации. В этом случае библиотечная функция заменяется системой предикатов, основанных на контрактах и на спецификации ошибочных состояний, заданных в спецификации. Такой подход используется в разрабатываемом в лаборатории анализа и верификации программ СПбПУ анализаторе Borealis [14], основанном на методе ограниченной проверки моделей (Bounded Model Checking, BMC). Похожий подход используется в основанном на абстрактной интерпретации инструменте Aegis, разрабатываемом в той же лаборатории [15].

Задача автоматизации миграции программ на новые библиотеки требует не только наличия внешней спецификации поведения библиотеки, но и частичного описания внутренней семантики библиотеки. На основе описания внутренней семантики строится семантический домен библиотеки, который может использоваться для проверки совместимости библиотеки и автоматического построения процедуры миграции [16].

6. Требования к языковым средствам задания спецификаций

Описанный в предыдущих разделах формализм удобен для доказательства свойств библиотек, а также для выполнения различных формальных процедур. Авторам библиотек и программистам нужны более удобные языковые инструменты, которые дадут возможность описывать поведение функций и библиотек в более привычном виде.

С этой целью в лаборатории верификации и анализа программ СПбПУ разрабатывается новое поколение языковых средств, полностью соответствующих разработанному формализму. Комплект языковых средств состоит из собственно языка

задания спецификаций и графического инструментария, позволяющего визуально описывать примитивы спецификаций – автоматы в виде графов переходов.

За основу для проектирования нового языка описания взят разработанный автором ранее язык описания библиотечных функций PanLang [13], используемый в статическом анализаторе Aegis [15]. При этом учитывается опыт других существующих языков, таких, например, как SLIC [10] и ACSL [17].

Основные функциональные требования к языку являются следствием представленного в данной работе формализма и его компонентов. Исходя из этого в язык должны входить следующие компоненты:

1. Описание основного расширенного конечного автомата, задающего поведение всей библиотеки (B).

2. Описание дочерних расширенных конечных автоматов, запускаемых при создании ресурсов библиотеки (S_i) с возможностью задания начального состояния q и параметра P.

3. Описания поведений функций (F_i), входящих в библиотеку.

4. Описание семантических доменов библиотеки. Каждый семантический домен вводит семантику решаемой высокоуровневой задачи. Такой задачей может быть проверка совместимости библиотек, миграция приложений, обнаружение дефектов или аппроксимация функций.

5. Описание семантических типов – специальных типов данных, унаследованных от классических типов данных, но имеющих проаннотированные имена и значения.

6. Описание процесса инициализации библиотеки, в том числе объектов и переменных, создаваемых на этапе инициализации.

Спецификация каждой функции должна содержать:

- описание сигнатуры функции, включающее, кроме названия функции, спецификацию аргументов ($Args$) и возвращаемого результата (Res), тип которых может быть в том числе и семантическим типом, обогащенным аннотациями;

- предусловия функции ($Pred$), заданные в виде предикатов от аргументов ($Args$), описывающие условия, при которых функция гарантирует корректность своего поведения (постусловий);

- постусловия функции ($Post$), заданные в виде предикатов от аргументов ($Args$) и возвращаемого результата (Res), описывающие гарантии поведения функции при условии выполнения предусловий;

- описания влияний функции на окружение. Каждое влияние описывается внешним изменяемым объектом (E_i) и предикатом ($CondE_i$), задающим условия, при котором объект изменяется;

- описания создаваемых при работе функции дочерних автоматов. Каждый создаваемый автомат характеризуется начальным состоянием и параметром. Создание дочернего автомата D_i происходит в зависимости от предиката создания $CondD_i$;

- описания влияний функции на окружение. Каждое влияние описывается внешним изменяемым объектом (E_i) и предикатом ($CondE_i$), задающим условия, при котором объект изменяется;

- описание действий (A). Каждое действие (A_i) является способом задания семантики поведения функции путем отображения на какой-либо семантический домен, зависящий от решаемой задачи. Таким доменом, например, может быть домен программных дефектов, детектируемых внутри библиотечной функции или библиотеки, или домен видимого поведения, требуемый для решения задачи проверки совместимости библиотек при миграции.

Одним из важных отличий создаваемого языка от языка Panlang является отсутствие задаваемого явно в императивном виде поведенческого описания функции (тела функции). Вместо этого все составляющие поведения (контракты, влияние на окружение, создание ресурсов, совершение семантических действий) задаются декларативно с помощью логических предикатов, представленных в виде формул логики первого порядка. Указанная особенность позволит, с одной стороны, обеспечить достаточную мощность описаний, а с другой – формально проверять различные свойства библиотек.

Нефункциональные требования к языку предъявляются следующие:

- простая грамматика и, как следствие, возможность построения эффективного парсера;

- прозрачный синтаксис и, следовательно, простота освоения пользователями.

На основании вышеперечисленных требований в настоящее время разрабатывается новый язык спецификации библиотек. Детальное описание синтаксиса и семантики разрабатываемого языка выходит за рамки настоящей статьи и является предметом отдельной публикации.

7. Заключение

В работе представлены результаты исследования по созданию формализма для спецификации программных библиотек. Формализм построен с учетом целого спектра задач, которые можно будет решать с его помощью. Основная идея – использовать одну и ту же формальную спецификацию как основу для нескольких методов программной инженерии: обнаружение программных дефектов, автоматизированная миграция программ и генерация программной документации. Из-за ограничений объема статьи формализм представлен без необходимой детализации. На основе разработанного формализма определены требования к языковым средствам спецификации библиотек.

Направление дальнейших исследований – разработка языковой поддержки для предложенного формализма и реализация конвертеров языковых описаний для существующих средств обнаружения ошибок и миграции программ.

Список литературы / References

[1] Lamb D., "IDL: sharing intermediate representations", *ACM Trans. Program. Lang. Syst*, **9**:3 (1987), 297–318.

[2] Exton C., Watkins D., Thompson D., "Comparisons between CORBA IDL and COM/DCOM MIDL: Interfaces for Distributed Computing", *Proceedings of the Technology of Object-Oriented Languages and Systems – Tools-25 (TOOLS '97)*, IEEE Computer Society, Washington, DC, USA, 1997, 15–23.

[3] Sankar S., Hayes R., "ADL—an interface definition language for specifying and testing software", *SIGPLAN*, **29**:8 (1994), 13–21.

[4] Allen R., Garlan D., "Formalizing architectural connection", *Proceedings of the 16th international conference on Software engineering (ICSE '94)*, IEEE Computer Society Press, Los Alamitos, CA, USA, 1994, 71–80.

[5] Хоар Ч., *Взаимодействующие последовательные процессы*, М.: Мир, 1989; [Hoar C.A.R., *Communicating sequential processes*, Mir, Moscow, 1989, (in Russian).]

[6] Roscoe A.W., "Modelling and verifying key-exchange protocols using CSP and FDR", *Proceedings of 1995 IEEE Computer Security Foundations Workshop*, IEEE Computer Society Press, 1995.

[7] de Alfaro L, Henzinger T., "Interface automata", *Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE-9)*, ACM, New York, NY, USA, 2001, 109–120.

[8] Ramanathan M., Grama A., Jagannathan S., "Static specification inference using predicate mining", *Proceedings of the 28th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '07)*, ACM, New York, USA, 2007, 123–134.

[9] Sankaranarayanan S.,Ivancic F.,Gupta A., "Mining library specifications using inductive logic programming", *Proceedings of the 30th international conference on Software engineering (ICSE '08)*, ACM, New York, USA, 2008, 131–140.

[10] Ball T., Rajamani S.K., *SLIC: a Specication Language for Interface Checking (of C)*, Microsoft Research, Technical Report, MSR-TR-2001-21, 2002.

[11] Leavens G.T., "The future of library specification", *Proceedings of the FSE/SDP workshop on Future of software engineering research (FoSER '10)*, ACM, New York, USA, 2010, 211–216.

[12] Ицыксон В.М., Зозуля А.В., "Формализм для описания частичных спецификаций компонентов программного окружения", *Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление*, **4** (2011), 81–90; [Itsykson V.M., Zozulya A.V., "The formalism for description of the partial specifications of program envinroment components", *St. Petersburg State Polytechnical University Journal. Computer Science. Telecommunication and Control Systems*, **4** (2011), 81–90, (in Russian).]

[13] Ицыксон В.М., Глухих М.И., "Язык спецификаций поведения программных компонентов", *Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление*, **3** (2010), 63–71; [Itsykson V.M., Glukhikh M.I., "A program component behavior specification language", *St. Petersburg State Polytechnical University Journal. Computer Science. Telecommunication and Control Systems*, **3** (2010), 63–71, (in Russian).]

[14] Akhin M.Kh., Belyaev M.A., Itsykson V.M., "Software defect detection by combining bounded model checking and approximations of functions", *Automatic Control and Computer Sciences*, **48**:7 (2014), 389–397.

[15] Itsykson V., etc, "Automatic defects detection in industrial C/C++ software", *Proceeding of 5th Central and Eastern European Software Engineering Conference in Russia (CEE-SECR)*, IEEE, 2009, 50–55.

[16] Ицыксон В.М., Зозуля А.В., "Автоматизированная трансформация программ при миграции на новые библиотеки", *Программная инженерия*, **6** (2012), 8–14; [Itsykson V.M., Zozulya A.V., "Automated Program Transformation for Migration to New Libraries", *Software Engineering*, **6** (2012), 8–14, (in Russian).]

[17] Kirchner F., etc, "Frama-C: A software analysis perspective", *Formal Aspects of Computing*, **27**:3 (2015), 573—609.

Itsykson V. M., "The Formalism and Language Tools for Semantics Specification of Software Libraries", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 754–766.

DOI: 10.18255/1818-1015-2016-6-754-766

Abstract. The paper is dedicated to the specification of the structure and the behaviour of software libraries. It describes the existing problems of libraries specifications. A brief overview of the research field concerned with formalizing the specification of libraries and library functions is presented. The requirements imposed on the formalism designed are established; the formalism based on these requirements allows specifying all the properties of the libraries needed for automation of several classes of problems: defects detection in the software, migration of applications into a new environment, generation of software documentation. The requirements on the language tools based on the developed formalism are proposed. The conclusion defines potential directions for further research.

Keywords: formal specification, software library, behavioral description, software defect, specification language

About the authors:
Vladimir M. Itsykson, orcid.org/0000-0003-0276-4517, PhD,
Peter the Great St. Petersburg Polytechnic University,
29 Polytechnicheskaya str., Saint-Petersburg 195251, Russia, e-mail: vlad@icc.spbstu.ru

Моделирование и анализ информационных систем. Т. 23, № 6 (2016), с. 767–776
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 767–776

DOI: 10.18255/1818-1015-2016-6-767-776

УДК 004.056.53

Разработка и анализ защищенности фрагмента информационно-телекоммуникационной системы, реализующей концепцию Интернета вещей

Александров В. А., Десницкий В.А.[1,2]**, Чалый Д.Ю.**[1]

получена 17 октября 2016

Аннотация. В работе исследуются вопросы разработки и реализации систем, использующих концепцию Интернета вещей. В условиях активного развития отраслей, использующих концепцию Интернета вещей, актуальна проблема информационной безопасности. Для того чтобы определить актуальные угрозы, необходимо использовать детальный анализ рисков в соответствии с действующими стандартами ГОСТ. Выбирая защитные меры, необходимо учитывать все идентифицированные актуальные угрозы информационной безопасности. В статье определяются актуальные угрозы и защитные меры, необходимые для разработки и внедрения защищенного фрагмента программно-аппаратной системы Умный дом в части контроля доступа в помещение. Решены следующие задачи: описание системы Умный дом, описание этапов оценки и обеспечения безопасности системы Умный дом; осуществление аппаратной сборки и написания программного кода для выбранного фрагмента системы; оценка безопасности выбранного фрагмента Умного дома и определение актуальных угроз; выработка рекомендаций по противодействию актуальным угрозам; программная реализация одной из актуальных угроз и программная реализация защитных мер для выбранной угрозы. Особенностью работы является комплексный подход к проектированию с использованием моделей нарушителя, анализа активов системы и оценки их защищенности.

Ключевые слова: Интернет вещей, информационная безопасность

Для цитирования: Александров В. А., Десницкий В.А., Чалый Д.Ю., "Разработка и анализ защищенности фрагмента информационно-телекоммуникационной системы, реализующей концепцию Интернета вещей", *Моделирование и анализ информационных систем*, **23**:6 (2016), 767–776.

Об авторах:
Александров Владислав Андреевич, orcid.org/0000-0003-1169-6034, магистрант,
СПБНИУ ИТМО, Кронверский пр-т, д. 49, г. Санкт-Петербург, 197101 Россия, e-mail: 2-q2@mail.ru

Десницкий Василий Алексеевич, orcid.org/0000-0002-3748-5414, канд. техн. наук, старший научный сотрудник,
ФГБУН Санкт-Петербургский институт информатики и автоматизации Российской академии наук,
14 линия, 39, г. Санкт-Петербург, 199178 Россия, e-mail: vasily.desnitsky@mail.ru

Чалый Дмитрий Юрьевич, orcid.org/0000-0003-0553-7387, канд. физ.-мат. наук,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150003, Россия, e-mail: dmitry.chaly@gmail.com

Благодарности:
[1]Работа выполнена при финансовой поддержке РФФИ (проект №16-37-50035).
[2]Работа выполнена при финансовой поддержке РФФИ (проекты №14-07-00697, 14-07-00417, 15-07-07451, 16-29-09482 офи_м, 16-37-50035).

Введение

В работе исследуются вопросы разработки и реализации систем, использующих концепцию Интернета вещей. Понятие Интернета вещей включает системы аппаратных устройств специализированного назначения, в которые встраиваются электронные модули для управления такими устройствами и организации внешних коммуникаций. Количество подключенных устройств растет с каждым годом. Так, близкий к линейному, по оценкам компании Cisco, прогнозируемый рост числа встроенных устройств обусловливает важность исследования вопросов информационной безопасности систем Интернета вещей [1]. Множество устройств Интернета вещей можно увидеть в потребительской сфере – устройства бытовой электроники, контрольные устройства физической и информационной безопасности, игровые устройства, имплантируемые медицинские устройства и другие. Такие устройства, соединенные между собой, значительно упрощают их использование и расширяют выдаваемый функционал. В целом концепция Интернета вещей, внедренная на предприятии, позволяет эффективней использовать его ресурсы путем повышения скорости реагирования на изменения. Взаимосвязь и анализ датчиков и объектов предприятия происходит с минимальным участием человека или без него. Это также способствует повышению производительности и уменьшает влияние человеческого фактора.

Цель работы состоит в разработке и исследовании защищенного фрагмента системы Умный дом, являющегося типовым примером системы Интернета вещей. Решены следующие задачи: описание системы Умный дом, описание этапов оценки и обеспечения безопасности системы Умный дом; осуществление аппаратной сборки и написания программного кода для выбранного фрагмента системы; оценка безопасности выбранного фрагмента Умного дома и определение актуальных угроз; выработка рекомендаций по противодействию актуальным угрозам; программная реализация одной из актуальных угроз и программная реализация защитных мер для выбранной угрозы. Особенностью работы является комплексный подход к проектированию с использованием моделей нарушителя, анализа активов системы и оценки их защищенности.

1. Методология разработки защищенных систем Умного дома

В настоящее время концепция Умного дома набирает все больший охват: в системах взаимосвязанных между собой программно-аппаратных устройств и сенсоров, применяемых для повышения автоматизации, физической и информационной безопасности, энергоэффективности и улучшения целевой функциональности. При этом использование концепции Интернета вещей позволяет получить такие преимущества, как модульность системы, масштабируемость системы, расширение функциональности и гибкость системы.

К основным недостаткам систем, реализующих концепцию Интернета вещей, можно отнести: (1) подверженность устройств системы множеству различных киберфизических атак, включающих сочетание в чистом виде программно-информа-

ционных воздействий и атак с использованием физических характеристик устройств и сенсоров системы, что определяет необходимость учета повышенных требований к защищенности; (2) разнородность устройств, их компонентов, используемых протоколов и технологий, что увеличивает сложность интеграции системы из отдельных компонентов; (3) высокую зависимость реализуемой защиты от бизнес-функций и особенностей системы, что затрудняет разработку универсальных средств и методик проектирования механизмов защиты для таких систем.

В соответствии с ГОСТ Р ИСО/МЭК ТО 13335-3-2007 [2] для обеспечения безопасности и выбора защитных мер для фрагмента системы Умного дома применяется детальный анализ рисков. Детальный анализ рисков включает идентификацию активов, оценку возможных угроз, которым подвержены активы, а также оценку их уязвимости. По результатам этих операций выполняется оценка рисков и последующее определение обоснованных защитных мер [3]. Результаты анализа рисков позволяют идентифицировать объекты системы или этапы в организации и использовании с высоким уровнем риска и выбрать меры обеспечения безопасности. Применение выбранных мер повышения безопасности снижает уровень идентифицированного риска до некоторого приемлемого.

2. Реализация и оценка фрагмента защищенной системы Умного дома

Разработан фрагмент системы Умного дома, нацеленный на проверку помещения на наличие в нем движения. В частности, анализируется ликвидность нахождения в контролируемом помещении при помощи политики безопасности с последующим оповещением в случае незаконного проникновения. Система включает следующие объекты: (1) центральное управляющее устройство на основе одноплатного компьютера; (2) датчик движения; (3) визуальное оповещение (светодиод).

В качестве датчика движения используется инфракрасный сенсор движения DFRobot. Микроконтроллер Raspberry Pi (RPi) используется в качестве центрального управляющего устройства системы. Небольшие размеры RPi определяют широкие возможности по его встраиванию в информационно-техническое окружение, тогда как аппаратные возможности RPi позволяют реализовать управление бизнес-функциями системы Умного дома с возможностью подключения различных сенсоров к стандартизированным пинам GPIO [4]. Наличие операционной системы Linux позволяет осуществлять программирование системы Умного дома на языках программирования высокого уровня, поддерживающих эту операционную систему.

На RPi установлен серверный модуль, написанный на Python 2.7, который получает данные с датчика движения каждые 10 миллисекунд. В случае обнаружения движения в помещении, контроллер сверяет точное значение времени и дату события с политикой безопасности, которая хранится в специальном файле. В данном файле устанавливается время и день недели, в пределах которых пользователю разрешается легально находиться в помещении.

В случае если действие было обнаружено в запрещенное в соответствии с политикой безопасности время, то RPi (1) включает предупреждающий светодиод; (2) записывает в log-файл информацию о событии (указывая дату и время); (3) от-

правляет сообщение о тревоге всем подключенным к серверу клиентам. Связь с клиентом происходит при помощи сокетов по протоколу TCP/IP. Подключение может осуществляться как в локальной сети, так и через сеть Интернет. Клиентская сторона написана также на языке Python 2.7 [5].

Клиент подключается к серверу. После прохождения аутентификации, посредством ввода пары логин/пароль, можно получить информацию с датчика движения. Пользователь может просмотреть действующую политику безопасности. Также имеется возможность изменять файл политики безопасности через клиентское приложение. В случае как законного, так и незаконного проникновения клиент получает оповещение от сервера.

Выбранный фрагмент Умного дома включает следующие активы, которые подлежат учету: (1) центральное управляющее устройство на основе одноплатного компьютера; (2) датчик движения; (3) программное обеспечение – клиент-серверное приложение, приложения для выполнения бизнес-функций системы и функций защиты; (4) файл политики безопасности Умного дома; (5) log-файл, хранящий информацию о случившихся изменениях в системе – данные об изменениях файла политики безопасности, подключенных клиентах и полученных данных от датчика.

К фрагменту Умного дома имеют доступ авторизованные пользователи, которые являются сотрудниками организации. Также имеется администратор, который имеет возможность добавлять пользователей и должен контролировать корректность работы системы. Допускается, что система используется в офисном помещении организации, в пределах одного этажа. Система предназначена для контроля периметра помещения и оповещения в случае проникновений. Все дальнейшие этапы выполняются в рамках установленных границ.

После получения перечня активов производится оценка ценности каждого из них. Ценность актива определяется его важностью для функционала системы Умного дома. Результаты проведенной оценки активов приведены в таблице 1.

Таблица 1. Оценка активов
Table 1. Evaluation of assets

Название актива	Оценка актива
центральное управляющее устройство	высокая
датчик движения	средняя
программное обеспечение	высокая
файл политики безопасности	высокая
log-файл	средняя

Для идентифицированных активов используется перечень угроз и их классификация с учетом известных типовых разновидностей угроз [3]. Далее классифицируемые угрозы сопоставляются с минимальным уровнем нарушителя, необходимым для реализации угрозы в соответствии с моделью Арбахама [6], [7]. На основе этого определяется вероятность возникновения угрозы, выраженная оценкой: «низкая», «средняя» или «высокая». Для угроз, не обусловленных преднамеренной деятельностью, основным фактором оценки будут служить данные о частоте появления угрозы.

На основании таблицы 1 составим перечень уязвимостей для исследуемой программно-аппаратной системы. При составлении перечня будем использовать примеры

общих уязвимостей [3]. Перечень уязвимостей представлен в таблице 2. Перечень уязвимостей будет неполным, но достаточным для проведения всех этапов анализа риска. Для оценки вероятности реализации покажем, какие угрозы можно осуществить, используя данную уязвимость.

Таблица 2. Оценка вероятности реализации уязвимости
Table 2. Assessment of the probability of vulnerability

Вид уязвимости	Угрозы, использующие данную уязвимость	Оценка вероятности реализации
отсутствие резервных копий	изменение целостности переданной информации (У7)	средняя
незащищенные линии связи	ошибки передачи (У1); повреждение линий (У2); перехват информации (У4); анализ трафика (У5); изменение целостности переданной информации (У7); сбои в функционировании услуг связи (например, сетевых услуг) (У8)	высокая
передача ценной информации без применения шифрования	перехват информации (У4); изменение целостности переданной информации (У7)	высокая
пересылка паролей открытым текстом	перехват информации (У4); изменение целостности переданной информации (У7)	высокая
отсутствие подтверждений посылки или получения сообщения	перехват информации (У4); изменение маршрута направления сообщений (У6); изменение целостности переданной информации (У7)	средняя
неадекватное управление сетью	ошибки передачи (У1); перегруженный трафик (У3); сбои в функционировании услуг связи (например, сетевых услуг) (У8)	высокая
недостаточная подготовка персонала	ошибки пользователей (У9)	низкая
отсутствие механизмов отслеживания	перегруженный трафик (У3); ненадлежащее использование ресурсов (10)	средняя
отказ системы вследствие отказа одного из элементов	ошибки передачи (У1); сбои в функционировании услуг связи (например, сетевых услуг) (У8)	средняя
неадекватные результаты проведения технического обслуживания	сбои в функционировании услуг связи (например, сетевых услуг) (У8)	высокая

В рамках идентификации существующих защитных мер предполагается, что фрагмент системы Умного дома будет встраиваться в организацию с уже суще-

ствующей системой безопасности. Данная программно-аппаратная система предназначена для развертывания в помещении организации, и предполагается, что физический доступ к центру управления Умного дома будет ограничен, а также что на объекте уже существует защищенная сеть. Персонал предприятия, работающий с объектами системы Умного дома, должен быть ознакомлен с правилами работы с этими объектами. Предполагается, что на предприятии имеется квалифицированный сотрудник, который будет администрировать систему Умного дома.

Для оценки рисков составим таблицу ранжирования угроз по мерам риска. Для этого определим оценку воздействия как оценку актива, на которую направлена угроза (если активов несколько, то берется значение самого ценного актива), информацию об активе получим из таблицы 1, где высокая оценка соответствует оценке 1, средняя – 2, низкая – 3. Вероятность возникновения угрозы и перечень идентифицированных угроз получим из таблицы 2. Высокая вероятность возникновения угрозы соответствует оценке 1, средняя – 2, низкая – 3. В таблице 3 проранжируем опасности. Цифрой 1 обозначена угроза с самым низким рангом, т.е. угроза с самым малым воздействием и самой низкой вероятностью возникновения.

Таблица 3. Ранжирование угроз по мерам риска
Table 3. Ranking vulnerabilities by risks

Дескриптор Угроза	Оценка воздействия (ценности актива)	Вероятность возникновения угрозы	Мера риска	Ранг угрозы
У1	2	2	4	2
У2	2	1	2	4
У3	2	1	2	4
У4	2	1	2	4
У5	2	1	2	4
У6	1	2	2	4
У7	1	1	1	5
У8	2	2	4	2
У9	2	2	4	2
У10	2	2	4	2

В таблице 3 каждой идентифицированной угрозе соответствует её ранг, в зависимости от метрики риска. Угрозы с рангом два или один считаются угрозами, риск от которых приемлем. Также необходимо учитывать существующие меры защиты, которые могут снизить ранг угрозы.

Исходя из идентифицированных защитных мер, можно считать, что риски от угрозы кражи, повреждения линий связи и нелегального проникновения злоумышленника можно считать допустимыми, так как предполагается, что за сохранностью физических объектов на предприятии/организации следит служба охраны или другая служба, на которую возложена функция охраны объекта.

Защитные меры выбираются исходя из списка угроз, уровень риска которых считается недопустимым. Защитные меры можно разделить на организационные и технические. В защитные меры для технической части необходимо включить периодическую проверку работоспособности всех элементов системы. Это необходимо

для уменьшения вероятности аппаратных сбоев. Исходя из существующих защитных мер, на предприятии должна быть организованна защищенная корпоративная сеть. Это уменьшит вероятность перегрузки трафика, изменения конфиденциальности, целостности, доступности информации, передаваемой или обрабатываемой в программно-аппаратной системе. Также во избежание изменения целостности информации, передаваемой между клиентом и сервером, необходимо применять шифрование и хеширование. Для уменьшения вероятности использования программного обеспечения несанкционированными пользователями необходимо передавать и хранить пару логин/пароль в виде результата хеш-функции.

3. Реализация

В рамках программно-аппаратной системы Умного дома рассмотрим более детально угрозу перехвата информации для получения пары логин/пароль, отправляемых в процессе аутентификации клиент-серверного взаимодействия. Способ моделирования атаки: для снифинга сетей обычно используют сетевые карты в режиме прослушивания. Предположим, что злоумышленник имеет доступ в сеть, в которой функционирует приложение клиент системы Умного дома. Для перехвата процесса сетевого взаимодействия между двумя хостами A и B подменим IP адреса взаимодействующих хостов своим IP адресом, направив сетевым хостам A и B фальсифицированные ARP-сообщения с использованием дистрибутива Debian Linux и утилит Ettercap и Wireshark. Данные средства используются для сканирования сети, задания цели атаки, отправки ARP-сообщений и анализа трафика между атакуемым компьютером и маршрутизатором. В качестве результата атаки получены пара логин/пароль и пользовательские данные (Рис. 1), определяющие правила доступа в помещения Умного дома в зависимости от времени суток и роли пользователя.

Для выбора защитных мер необходимо проследить, через какие уязвимости можно реализовать выбранную угрозу. Используя таблицу 2, можно определить, что угрозу перехвата данных можно осуществить через такие уязвимости, как наличие незащищенных линий связи и передачу ценной информации без применения шифрования.

Устранение этих уязвимостей можно осуществить несколькими способами. Основным условием проведения такой атаки является нахождение нарушителя в сети, в которой работает атакуемый компьютер. Поэтому если сеть будет защищена и злоумышленник не сможет подключиться к ней, то он не сможет провести атаку. Отметим, что при внедрении Умного дома на предприятии, как правило, нет возможности такого контроля состояния сети, поэтому этот способ не подходит. Возможно использовать маршрутизаторы с поддержкой защиты и фильтрации ARP-пакетов. Также можно использовать сети VPN или VLAN. Но все перечисленные способы требуют дополнительных настроек сети или компьютеров, что делает систему Умного дома менее гибкой. В рамках реализованного прототипа для защиты от атаки «Человек посередине» используется шифрование передаваемой информации между клиентом и сервером. При этом даже если линии связи будут не защищены, то получить информацию из перехваченных пакетов у злоумышленника не получится. Используемый протокол SSL базируется на основе асимметричной криптографии

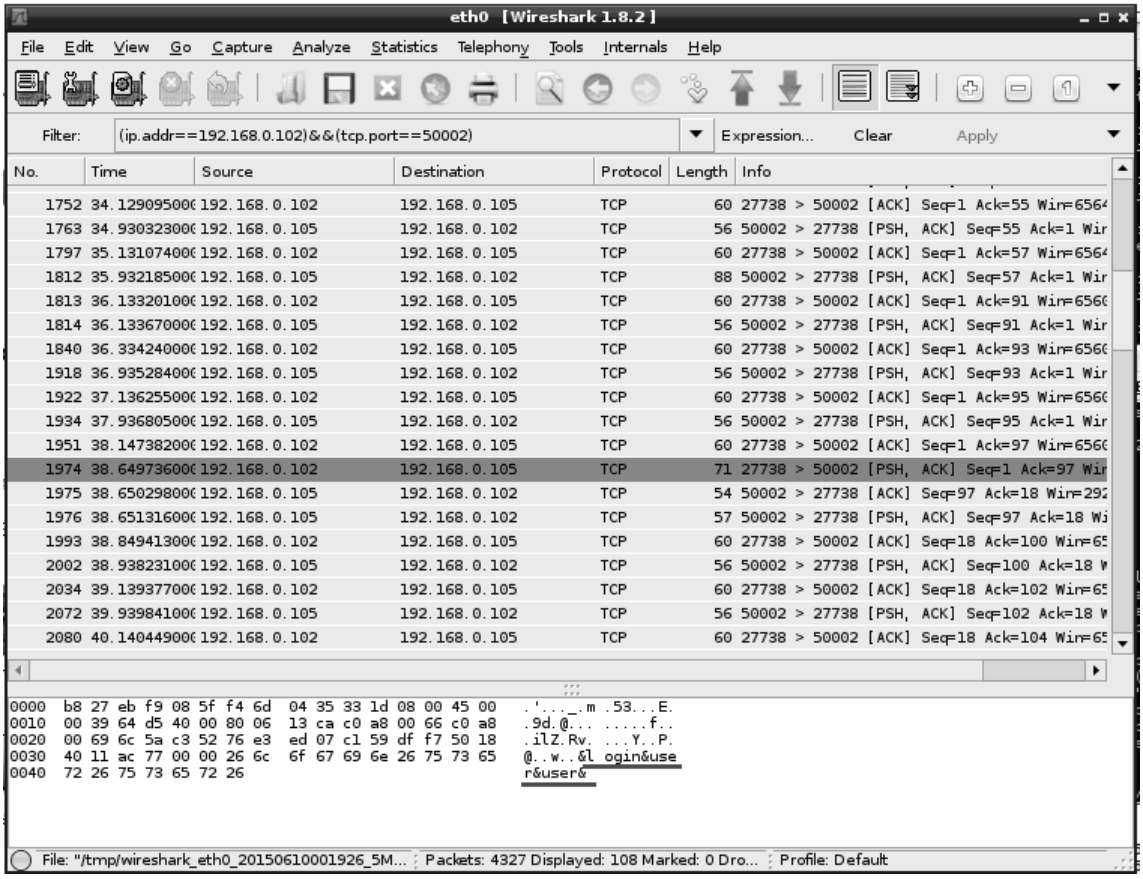


Рис. 1. Перехваченный пакет с парой логин/пароль

Fig. 1. Login/password packet captured

для аутентификации ключей обмена, используется также симметричное шифрование для обеспечения конфиденциальности, коды аутентификации сообщений – для целостности сообщений.

Для подключения протокола SSL воспользуемся библиотекой языка Python ssl. При помощи программы openSSL создадим самоподписанный сертификат для сервера, который он будет предоставлять клиентам. В openSSL сгенерируем приватный ключ с размером 1024 бит. Далее с помощью этого ключа сгенерируем самоподписанный сертификат. Теперь с помощью этого сертификата сервер может подтверждать валидность клиентов. Затем с использованием библиотеки ssl и самоподписанного сертификата создадим защищенное соединение между клиентом и сервером. Теперь вся информация, передаваемая между клиентом и сервером, будет зашифрована. После внедрения мер защиты – протокола SSL – необходимо проверить его работоспособность. Для этого смоделируем такую атаку на компьютер с использованием клиентского приложения. После отправки фальсифицированных ARP-сообщений мы получим пакеты, передаваемые между клиентом и сервером. Однако теперь сообщения зашифрованы и злоумышленник уже не может получить пару логин/пароль из этих пакетов без применения криптографического анализа.

Получить данные расписания из политики безопасности таким способом у злоумышленника теперь также не получится. При этом можно утверждать, что от угрозы перехвата информации, а в частности атаки «Человек посередине», система Умный дом защищена.

Заключение

Разработан и исследован фрагмент защищенной системы Умный дом, который является типовым примером системы Интернета вещей. Практическим результатом работы является разработанный фрагмент защищенной системы Умного дома в части функций контроля доступа в помещение. В соответствии с ГОСТ Р ИСО/МЭК 13335-1-2006 [3] и ГОСТ Р ИСО/МЭК ТО 13335-3-2007 [2] проведены и детально описаны основные этапы обеспечения безопасности построенной системы: установление границ рассмотрения; идентификация активов; оценка угроз; оценка уязвимостей; идентификация существующих мер безопасности; оценка рисков; выбор защитных мер.

Для заданного фрагмента системы проведена оценка обеспечения безопасности и определены актуальные угрозы. Также для идентифицированных актуальных угроз были разработаны защитные меры, для противодействия актуальным угрозам. Реализована одна из актуальных угроз – угроза перехвата критически важной информации системы. Реализация осуществлена путем моделирования атаки типа «Человек посередине». Далее реализованы программные меры защиты для противодействия установленной угрозе. Путём повторной попытки реализации угрозы было зафиксировано, что система Умный дом защищена от нее.

Список литературы / References

[1] Morgan S., *Internet Trends: 2007*, http://www.slideshare.net/rmesquita/morgan-stanley-technology-internet-trends.

[2] *ГОСТ Р ИСО/МЭК ТО 13335-3-2007. Информационная технология. Методы и средства обеспечения безопасности. Часть 3. Методы менеджмента безопасности информационных технологий*, 2007, http://ohranatruda.ru/ot_biblio/normativ/data_normativ/51/51065.html;
[*GOST R ISO/MJeK TO 13335-3-2007. Informacionnaja tehnologija. Metody i sredstva obespechenija bezopasnosti. Chast 3. Metody menedzhmenta bezopasnosti informacionnyh tehnologij*, 2007, URL: http://ohranatruda.ru/ot_biblio/normativ/data_normativ/51/51065.html, (in Russian)].

[3] *ГОСТ Р ИСО/МЭК 13335-1-2006. Методы и средства обеспечения безопасности. Часть 1. Концепция и модели менеджмента безопасности информационных и телекоммуникационных технологий*, 2006, http://www.gosthelp.ru/gost/gost271.html;
[*Information technology. Security techniques. Part 1. Concepts and models for information and communications technology security management*, 2006, http://www.gosthelp.ru/gost/gost271.html, (in Russian)].

[4] Ричардсон М., Уоллес Ш., *Заводим Raspberry Pi*, Амперка, 2013, 230 с.; [Richardson M., Uolles S., *Zavodim Raspberry Pi*, Amperka, 2013, 230 c., (in Russian).]

[5] Лутц М., *Программирование на Python*, **2**, Символ-Плюс, 2011, 992 с.; [Lutc M., *Programmirovanie na Python*, **2**, Simvol-Pljus, 2011, 992 c., (in Russian).]

[6] Abraham D. G., Dolan G. M., Double G. P., “Transaction Security System”, *IBM Systems Journal*, **30**:2 (1991), 230–243.

[7] Десницкий В. А., Чечулин А. А., “Обобщенная модель нарушителя и верификация информационно-телекоммуникационных систем со встроенными устройствами”, *Технические науки — от теории к практике*, 2014, № 39, 7–21; [Desnitsky V. A., Chechulin A. A., “Obobshhennaja model narushitelja i verifikacija informacionno-telekommunikacionnyh sistem so vstroennymi ustrojstvami”, *Tehnicheskie nauki — ot teorii k praktike*, 2014, № 39, 7–21, (in Russian).]

Alexandrov V. A., Desnitsky V.A., Chaly D.Y., "Design and Security Analysis of a Fragment of Internet of Things Telecommunication System", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 767–776.

DOI: 10.18255/1818-1015-2016-6-767-776

Abstract. This paper comprises the development and implementation of systems using the concept of Internet of Things. In terms of active development of industries, use the concept of the Internet of Things, the information security problem is urgent. To create a protected module of information-telecommunication system which implements the Internet of Things concept, it is important to take into account all its aspects. To determine relevant threats, it is necessary to use the detailed risk analysis according to existing GOST standards when choosing protection measures, one must rely on identified relevant threats. Actual threats and necessary protective actions are determined in this paper for implementation of Smart House computer appliance module, in order to develop a protected part of Smart House, which is necessary for realization of room access control. We solved the following tasks in the work, namely, a description of the system Smart Home, a description of steps and evaluation system security Smart Home; implementation of hardware assembly and writing a code for the selected fragment of the system; safety evaluation of the selected fragment Smart House and identification of actual threats; make recommendations to counter current threats; software implementation of one of the most urgent threats and software implementation of protective measures for a selected threat. A feature of the work is an integrated approach to the design with the use of the intruder models, analysis of the system's assets and evaluation of their security.

Keywords: Internet of Things, information security

About the authors:
Vladislav A. Alexandrov, orcid.org/0000-0002-3748-5414, a candidate for a Master's degree,
ITMO University, 49 Kronverksky Pr., St. Petersburg 197101, Russia, e-mail: 2-q2@mail.ru

Vasily A. Desnitsky, orcid.org/0000-0002-3748-5414, PhD, senior researcher,
St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences,
39 Liniya 14-ya, Saint-Petersburg 199178, Russia, e-mail: desnitsky@mcomsec.spb.ru

Dmitry Y. Chaly, orcid.org/0000-0003-0553-7387, PhD, head of department,
P.G. Demidov Yaroslavl Stat University, 14 Str. Sovetskaya, Yaroslavl 150003, Russia, e-mail: dmitry.chaly@gmail.com

Acknowledgments:
[1]This research was financially supported by grants of RFBR (project №16-37-50035).
[2]This resear was financially supported by grants of RFBR (projects №14-07-00697, 14-07-00417, 15-07-07451, 16-29-09482 ofi_m, 16-37-50035).

Моделирование и анализ информационных систем. Т. 23, № 6 (2016), с. 777–783
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 777–783

DOI: 10.18255/1818-1015-2016-6-777-783

УДК 004.9

Генерация графа социальной сети с использованием Apache Spark

Белов Ю. А., Вовчок С.И.

получена 24 октября 2016

Аннотация. Планируется создать метод кластеризации графа социальной сети. Для тестирования будущего метода возникла необходимость в генерации графа, по своей структуре схожего с лежащими в основе существующих социальных сетей. В статье представлен алгоритм для распределенной генерации такого графа. Учитываются основные свойства социальной сети: степенное распределение количества сообществ для пользователей, плотные пересечения сообществ и другие. В данном алгоритме учтены проблемы, присутствующие в подобных работах других авторов, например, проблема кратных ребер при генерации. Особенностью созданного алгоритма стала реализация, зависящая от такого параметра как количество сообществ, а не от количества пользователей, как это делается в других работах. Это связано с особенностью развития структуры реальной существующей социальной сети. В работе перечислены свойства ее графа. Описана таблица, содержащая необходимые для алгоритма переменные. Составлен пошаговый алгоритм генерации. Для него определены соответствующие математические параметры. Генерация происходит распределенно с помощью фреймворка Apache Spark. Подробно описано, каким образом происходит разделение задач с помощью данного фреймворка. В алгоритме используется модель Эрдеша–Реньи для случайных графов как наиболее подходящая и достаточно простая для реализации. Основными преимуществами созданного метода являются использование малого количества ресурсов, по сравнению с другими подобными генераторами, и скорость выполнения. Быстрота достигается за счет распределенной работы и того, что при распределенной работе алгоритма в любой момент пользователи сети имеют свои уникальные номера и упорядочены по этим номерам, поэтому не требуется их сортировка. Разработанный алгоритм будет способствовать не только созданию эффективного метода кластеризации. Он может быть полезен в других областях, связанных, например, с поисковыми системами социальных сетей.

Ключевые слова: социальная сеть, генерация

Для цитирования: Белов Ю. А., Вовчок С.И., "Генерация графа социальной сети с использованием Apache Spark", *Моделирование и анализ информационных систем*, **23**:6 (2016), 777–783.

Об авторах:
Белов Юрий Анатольевич, orcid.org/0000-0002-4221-6470, канд. физ.-мат. наук, доцент,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская 14, г. Ярославль, 150003 Россия, e-mail: belov45@yandex.ru

Вовчок Сергей Иванович, orcid.org/0000-0003-0535-5786, аспирант,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская 14, г. Ярославль, 150003 Россия, e-mail: vof4ok@gmail.com

Введение

В настоящее время уже существуют методы для генерации графа социальной сети, многие из которых рассматривались при подготовке данной работы. Большинство методов генерируют граф, не схожий по перечисленным далее в работе свойствам со структурой реальной социальной сети. Внимание привлекла работа 2014 года [1]. В ней разработан процесс генерации графа, основанный на модели AGM [2] и названный CKB. Нами была поставлена цель создать алгоритм для генерации графа социальной сети, усовершенствовав CKB, в частности минимизировав количество необходимой для хранения структур памяти, решив проблему кратных ребер при реализации.

Итоговый алгоритм стал интересен своей реализацией на основе количества сообществ сети, а не количества пользователей. Такая идея лучше передает процесс развития структуры социальной сети. Первоначально в сети появляется группа знакомых людей, образующих сообщество. Далее, благодаря рекламе, в сети регистрируются новые группы знакомых людей и отдельные пользователи из различных городов и стран. Постепенно часть из них образуют большие по количеству членов сообщества. Этот процесс и стал основой первой части алгоритма, в которой генерируются независимые сообщества с пользователями. Связи между пользователями образуются по модели Эрдеша–Реньи [4], как наиболее подходящей [3] для быстрого выполнения алгоритма.

Уже внутри социальной сети на следующем этапе ее развития ранее незнакомые друг другу пользователи из различных групп начинают контактировать и тем самым образуют новые связи. В результате пользователь сети становится участником нескольких сообществ. Социальная сеть получает более реалистичную структуру. Это стало идеей второй части алгоритма.

В алгоритме использована идея модели Чунг-Лу [5], в которой существует возможность выбора показателя степенного распределения вершин. Итоговый метод пошагово описывается далее. Программная реализация производится на фреймворке Apache Spark [6]. Планируется рядом экспериментов подтвердить эффективность алгоритма по отношению к существующим и подтвердить выполнение в полученном графе свойств социальной сети.

1. Постановка задачи

Необходимо сгенерировать граф $G = (V, E)$, где $|V| = A_1$ и $|E| = B$. Сообщество C_i является подграфом графа G, размер сообщества $|C_i| = r_{C_i}$. Количество сообществ равно A_2. Все вместе они образуют вершинное покрытие графа. Количество вхождений j-й вершины в разные сообщества — b_j.

Вершина $j \in C_i$ имеет внутреннюю степень d^{int}_{j,C_i} и внешнюю степень d^{ext}_{j,C_i}. Внутренняя — определяется как количество ребер, соединяющих j с другими вершинами в C_i. Внешняя степень — соответственно количество ребер, соединяющих j с остальной частью графа. Общая степень вершины j равна $d_j = d^{int}_{j,C_i} + d^{ext}_{j,C_i}$.

Количество ребер внутри сообщества C_i равно $B_{C_i} = \frac{1}{2}\Sigma_{\forall j \in C_i} d^{int}_{j,C_i}$.

Задача состоит в том, чтобы сгенерировать граф социальной сети G со следующими свойствами:

1. Размеры сообществ графа распределены по степенному закону с экспонентой $\alpha > 0$:
$$\forall i \in \mathbb{N} \Rightarrow p(r_{C_i}) \sim r_{C_i}^{-\alpha}; \tag{1}$$

2. В графе присутствует большая компонента связности:
$$\exists G^* \subset G \colon |V^*| \sim A_1, \forall i, j \in V^* \exists \omega_1, \omega_2, \ldots, \omega_k \in E^* \exists t_l \in V \colon \\ \omega_0 = (i, t_0), \omega_1 = (t_0, t_1), \ldots, \omega_D = (t_{D-1}, j); \tag{2}$$

3. Граф имеет диаметр $D \approx \frac{\ln A_1}{\ln \ln A_1}$:
$$\forall 0 < \epsilon < 1 \colon (1-\epsilon)\frac{\ln A_1}{\ln \ln A_1} \leq D \leq (1+\epsilon)\frac{\ln A_1}{\ln \ln A_1}, \tag{3}$$
где
$$D = \max_{i,j \in V^*}(d_{i,j}),$$
$$d_{i,j} = min_{\omega_s \in E^*}(|\{\omega_1, \omega_2, \ldots, \omega_k \mid \omega_0 = (i, t_0), \omega_1 = (t_0, t_1), \ldots, \omega_k = (t_k, j)\}|);$$

4. Пользователи могут не иметь связей и/или не входить ни в одно сообщество:
$$\forall i \in N \Rightarrow d_i \geq 0, b_i \geq 0; \tag{4}$$

5. Сообщества могут пересекаться:
$$\exists i, j \in \mathbb{N} \colon C_i \bigcap C_j \neq \emptyset; \tag{5}$$

6. С большой вероятностью каждое сообщество C_i является связным графом:
$$\forall k \in \mathbb{N} \Rightarrow P(C_k = (V_k, E_k) \colon \forall i, j \in C_k \exists \omega_t \in E_k \colon \\ \omega_0 = (i, t_0), \omega_1 = (t_0, t_1), \ldots, \omega_l = (t_l, j)) \geq 1 - \frac{1}{r_{C_i}}; \tag{6}$$

7. Плотность ребер внутри сообществ больше, чем средняя плотность ребер во всем графе G:
$$\forall i \in \mathbb{N} \Rightarrow \frac{d_{C_i}}{r_{C_i}(r_{C_i} - 1)} > \frac{b}{A_1(A_1 - 1)}; \tag{7}$$

8. Количество ребер внутри сообщества больше, чем количество ребер, соединяющих вершины сообщества с остальной частью графа:
$$\forall i \in \mathbb{N} \Rightarrow d_{C_i} > \Sigma_{\forall j \in C_i} d_{j,C_i}^{ext}; \tag{8}$$

9. Количество ребер в сообществе растет суперлинейно с размером сообщества:
$$\forall i \in \mathbb{N} \Rightarrow d_{C_i} \propto r_{C_i}^{1+\gamma}, \gamma \in (0, 1); \tag{9}$$

10. Пересечение сообществ более плотно, чем их непересекающаяся часть:
$$\forall i, j \in \mathbb{N}(i \neq j), C_{i \bigcap j} = C_i \bigcap C_j \Rightarrow \\ \frac{d_{C_{i \bigcap j}}}{r_{C_{i \bigcap j}}(r_{C_{i \bigcap j}} - 1)} > \frac{d_{C_i}}{r_{C_i}(r_{C_i} - 1)}. \tag{10}$$

Свойства взяты из результатов исследований структуры реальной социальной сети [1]

2. Предложенный метод

Основные шаги алгоритма следующие:

1. Распределенно и независимо генерируются сообщества пользователей в виде списков связанных друг с другом пользователей.

2. Объединяя пары пользователей из различных сообществ в одну вершину, получаем граф социальной сети.

3. Подробный алгоритм предложенного метода

Таблица 1. Входные параметры для генерации
Table 1. Input parameters for generation

Параметры Parameters	**Значение Value**	**По умолчанию By default**
A_2	Количество сообществ	—
b_{min}	Минимальное число сообществ, в которое входит один пользователь	1
b_{max}	Максимальное число сообществ, в которое входит один пользователь	10000
x_{min}	Минимальный размер сообщества	2
x_{max}	Максимальный размер сообщества	10000
$\alpha > 1$	Экспонента степенного распределения размеров сообщества	$2,5$
$\beta > 1$	Экспонента степенного распределения числа вхождений пользователя в сообщества	$2,5$

1. Генерируем последовательность для степенного распределения размеров сообществ: ставим в соответствие каждому j-му сообществу d_j.

2. Вводим вспомогательные переменные D_j: $D_0 = 1$, $D_1 = d_1$, $D_2 = D_1 + d_2$, $\cdots$, $D_{A_2} = D_{A_2-1} + d_{A_2}$. Таким образом, D_j равно количеству пользователей для первых j сообществ. Т. е. если $d_1 = 5$; $d_2 = 4$; $d_3 = 3$; $d_4 = 5$, то $D_1 = 5$; $D_2 = 9$; $D_3 = 12$; $D_4 = 17$.

3. Вычисляем количество пользователей A_1 создаваемой социальной сети, а также — X_{A_1}, равное поправке на объединение в одну вершину пар пользователей из различных сообществ. В данной работе предполагается, что X_{A_1} — это математическое ожидание числа вхождений пользователей в сообщества ($E[b]$),

которое вычисляется по формуле

$$E[b] = \int_{b_{min}}^{b_{max}} bp(b)db = \frac{(1-\beta)(b_{max}^{2-\beta} - b_{min}^{2-\beta})}{(b_{max}^{1-\beta} - b_{min}^{1-\beta})(2-\beta)}. \tag{11}$$

A_1 рассчитывается исходя из уравнения:

$$A_1 \cdot E[b] = A_2 \cdot E[x], \tag{12}$$

где $E[x]$ — математическое ожидание размеров сообществ, рассчитывается аналогично $E[b]$.

4. Распределенно (с использованием нескольких компьютеров, объединенных в параллельную вычислительную машину) генерируем сообщества и составляем для каждого из них «списки» пользователей при помощи инструментов фреймворка Apache Spark. В терминологии Apache Spark в вычислительном кластере существует главный узел, называемый *мастер*, и ведомые узлы, называемые *слэйвами*. Для генерации «списка» пользователей для сообщества передаем на слэйвы данные о каждом i-м сообществе. Данными являются номера пользователей от $D_{i-1}+1$ до D_i для i-го сообщества. В сгенерированном «списке» для i-го сообщества содержатся уникальные номера пользователей от $D_{i-1}+1$ до D_i. Таким образом, все пользователи распределены в «списки» сообществ без повторения. Количество сообществ равно A_2. Их размер распределен по степенному закону.

5. Распределенно для каждого «списка» на слэйвах запускаем модель Эрдеша–Реньи [4] для случайных графов, в результате чего получаем для каждого пользователя в «списке» связанных с ним пользователей из данного «списка». Так как пользователи имеют уникальные номера, на этом шаге имеем возможность проверять при добавлении связи, нет ли уже добавленной идентичной. Тем самым в дальнейшем убираем возможность существования кратных ребер в сообществах. После выполнения данного шага алгоритма у нас есть независимые сообщества пользователей со списками связей между пользователями.

6. «Сцепляем» полученные на предыдущем шаге списки связей в один упорядоченный общий список пользователей. Список каждого i-го сообщества «сцепляется» с $i+1$-м. Эта операция возможна благодаря уникальным номерам пользователей. Она не требует дополнительной сортировки пользователей по номерам, так как изначально списки в сообществах упорядочены. В результате получается общий список сообществ, каждое из которых сгенерировано с помощью алгоритма Эрдеша–Реньи.

7. Вводим значение «*», необходимое для того, чтобы помечать элемент списка, который не будет использован в дальнейшем.

8. Из двух разных сообществ выбираем двух пользователей. Для этого случайным образом задаем i и j : $i \neq j$ и $0 \leq i, j \leq A_2$. Выбираем пользователей

Z_1 в отрезке от $(D_{i-1}+1)$-го элемента списка до D_i-го и Z_2 — от $(D_{j-1}+1)$-го до D_j-го, не равные «∗». Нам необходимо объединить данных пользователей, сделав из них одного. Для этого в общем списке добавляем к Z_1 все связи Z_2. А также выполняем проход от $(D_{j-1}+1)$-го до D_j-го элемента, заменяя значения смежных с элементами вершин равных Z_2 на Z_1. Сам элемент Z_2 заменяем на «∗».

9. Предыдущий шаг повторяем X_{A_1} раз, т. е. количество поправок, указанное ранее, на объединение в одну вершину пар пользователей из различных сообществ.

10. Создаем пустой граф $G(V, E)$ на A_1 вершин. Переносим данные, не равные «∗», из полученного списка в данный граф.

В результате выполнения алгоритма получаем граф социальной сети.

Список литературы / References

[1] Чихрадзе К. К. и др., “Использование модели социальной сети с сообществами пользователей для распределенной генерации случайных социальных графов”, *Машинное обучение и анализ данных*, **1**:8 (2014), 1027–1047; [Chikhradze K. K. et al., “On a model of social network with user communities for distributed generation of random social graphs”, *Machine Learning and Data Analysis*, **1**:8 (2014), 1027–1047, (in Russian).]

[2] Yang J., Leskovec J., “Community-affiliation graph model for overlapping network community detection”, *IEEE 12th Conference (International) on Data Mining*, 2012.

[3] Raigorodskii A., “Models of Random Graphs and Their Applications to the Web-Graphs Analysis”, *RUSSIR-2015*, Lomonosov Moscow University, Moscow Istitute of Physics and Technology, Yandex, Moscow, Russia, 25–29 August, 2015.

[4] Erdos P., Renyi A., “On the evolution of random graphs”, *Bull. Inst. Int. Statist. Tokyo*, **38** (1961), 343–347.

[5] Aiello W., Chung F., Lu L., “On the evolution of random graphs”, *A random graph model for power law graphs*, http://people.math.sc.edu/lu/papers/power.pdf.

[6] Карау Х., и др., *Изучаем Spark: молниеносный анализ данных*, ДМК Пресс, М., 2015, 304 с.; [Karau K. et al., *Learning Spark: Lightning-Fast Data Analysis*, DMK Press, Moscow, 2015, 304 pp., (in Russian).]

[7] Вовчок С. И., “Создание метода кластеризации графа социальной сети”, *Новые информационные технологии в науке: сборник статей Международной научно-практической конференции МЦИИ ОМЕГА САЙНС. Ч. 2*, 2016, 34–36; [Vovchok S. I., “Sozdanie metoda klasterizatsii grafa sotsialnoy seti”, *Novye informatsionnye tekhnologii v nauke: sbornik statey Mezhdunarodnoy nauchno-prakticheskoy konferentsii MTsII OMEGA SAYNS. Part 2*, 2016, 34–36, (in Russian).]

Belov Y. A., Vovchok S.I., "Generation of a Social Network Graph by Using Apache Spark", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 777–783.

DOI: 10.18255/1818-1015-2016-6-777-783

Abstract. We plan to create a method of clustering a social network graph. For testing the method there is a need to generate a graph similar in structure to existing social networks. The article presents an algorithm for the graph distributed generation. We took into account basic properties such as

power-law distribution of the users communities number, dense intersections of the social networks and others. This algorithm also considers the problems that are present in similar works of other authors, for example, the multiple edges problem in the generation process. A special feature of the created algorithm is the implementation depending on the communities number parameter rather than on the connected users number as it is done in other works. It is connected with a peculiarity of progressing the existing social network structure. There are properties of its graph in the paper. We described a table containing the variables needed for the algorithm. A step-by-step generation algorithm was compiled. Appropriate mathematical parameters were calculated for it. A generation is performed in a distributed way by Apache Spark framework. It was described in detail how the tasks division with the help of this framework runs. The Erdos-Renyi model for random graphs is used in the algorithm. It is the most suitable and easy one to implement. The main advantages of the created method are the small amount of resources in comparison with other similar generators and execution speed. Speed is achieved through distributed work and the fact that in any time network users have their own unique numbers and are ordered by these numbers, so there is no need to sort them out. The designed algorithm will promote not only the efficient clustering method creation. It can be useful in other development areas connected, for example, with the social networks search engines.

Keywords: social network, generation

About the authors:
Yuriy A. Belov, orcid.org/0000-0002-4221-6470, PhD,
P.G. Demidov Yaroslavl State University,
4 Sovetskaya ul., Yaroslavl 150003, Russia, e-mail: belov45@yandex.ru

Sergei I. Vovchok, orcid.org/0000-0003-0535-5786, graduate student,
P.G. Demidov Yaroslavl State University,
14 Sovetskaya ul., Yaroslavl 150003, Russia,
e-mail: vof4ok@gmail.com

Моделирование и анализ информационных систем. Т. 23, №6 (2016), с. 784–803
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 784–803

DOI: 10.18255/1818-1015-2016-6-784-803

УДК 517.994

Бифуркации периодических решений уравнения Мэкки–Гласса

Кубышкин Е. П., Морякова А.Р.[1]

получена 15 марта 2016

Аннотация. В работе изучаются бифуркации периодических решений из состояния равновесия известного уравнения Мэкки–Гласса, предложенного в качестве математической модели изменения плотности белых клеток крови. Уравнение, записанное в безразмерных переменных, содержит малый параметр при производной, что делает его сингулярным. К уравнению применяется метод равномерной нормализации, позволяющий свести исследование поведения решений в окрестности состояния равновесия к анализу счетной системы обыкновенных дифференциальных уравнений, из которых выделяются уравнения «быстрой» и «медленных» переменных. Показано, что состояния равновесия уравнений «медленных» переменных определяют периодические решения. Анализ состояний равновесия позволяет изучить бифуркации периодических решений в зависимости от параметров уравнения и их устойчивость. Показана возможность одновременной бифуркации большого числа устойчивых периодических решений. Это явление носит название бифуркации мультистабильности.

Ключевые слова: уравнение Мэкки–Гласса, периодические решения, бифуркация мультистабильности

Для цитирования: Кубышкин Е. П., Морякова А.Р., "Бифуркации периодических решений уравнения Мэкки–Гласса", *Моделирование и анализ информационных систем*, **23**:6 (2016), 784–803.

Об авторах:
Кубышкин Евгений Павлович, orcid.org/0000-0003-1796-0190, д-р. физ.-мат. наук, профессор,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150003 Россия, e-mail: kubysh.e@yandex.ru

Морякова Алена Романовна, orcid.org/0000-0003-2529-6277, аспирант,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150003 Россия, e-mail: alyona_moryakova@mail.ru

Благодарности:
[1]Работа выполнена при частичной поддержке проекта № 984 в рамках базовой части государственного задания на НИР ЯрГУ.

Введение

Уравнение Мэкки–Гласса

$$\dot{x} = -\gamma x + \beta x_\tau \theta^n (\theta^n + x_\tau^n)^{-1}, x_\tau = x(t-\tau), x(t) > 0, \tag{1}$$

в котором $\gamma,\ \beta,\ \theta,\ \tau$ – положительные параметры, $n \geq 3$ – натуральное число, является одним из простейших дифференциально-разностных уравнений с нелинейной

запаздывающей обратной связью. Уравнение (1) предложено в монографии [1] в качестве математической модели изменения плотности ($x(t)$) циркулирующих в организме человека нейтрофилов (белых клеток крови). Входящие в (1) параметры имеют четкий биологический смысл. В [1] на основе численного интегрирования уравнения (1) показано существование различных периодических решений, из чего сделан вывод о периодическом изменении плотности нейтрофилов в организме человека.

Уравнение (1) обладает богатой динамикой. Оно изучалось в ряде работ ([2] – [7]), в которых анализировались различные свойства решений и на основе численного интегрирования показано существование разнообразных периодических решений, а также сложных, в том числе хаотических колебаний. В [5] рассмотрены некоторые обобщения уравнения (1). В работе [7] приведены результаты моделирования динамики уравнения (1) посредством электронного устройства. Отмечено существование хаотических колебаний, изучаются их спектральные свойства.

В настоящей работе изучаются периодические решения уравнения (2), бифурцирующие из его единственного состояния при изменении параметров уравнения, исследуется их устойчивость. Получены строгие теоремы об условиях бифуркации периодических решений, а также построены асимптотические формулы периодических решений. Показана возможность бифуркации одновременно большого числа устойчивых периодических решений – бифуркация мультистабильности. В качестве метода исследования используется метод равномерной нормализации, предложенный в [8].

1. Математическая постановка задачи, анализ устойчивости состояния равновесия

Перейдем в (1) к безразмерным переменным и параметрам, положив $x'(t) = \theta x(t)$, $t' = t/\tau$, $\varepsilon_1 = (\gamma\tau)^{-1}$, $\beta' = \beta/\gamma$. В результате, опустив штрих, получим уравнение

$$\varepsilon_1 \dot{x}(t) = -x(t) + \beta x(t-1)/(1 + x^n(t-1)), \ x(t) > 0, \tag{2}$$

в котором по физике задачи параметр β принимает значение порядка единицы, а $0 < \varepsilon_1 \ll 1$.

Уравнение (2) имеет единственное положительное состояние равновесия

$$x_* = (\beta - 1)^{1/n}, \tag{3}$$

устойчивость которого определяется значениями β и n. Ниже изучаются периодические решения уравнения (2), бифурцирующие из состояния равновесия (3) при изменении параметров ε_1 и β, исследуется их устойчивость.

Выполним в уравнении (2) замену $x(t) = x_* + y(t)$ и представим нелинейную часть уравнения (2) рядом Тейлора. В результате в окрестности состояния равновесия (3) поведение решений уравнения (2) будет определяться следующим уравнением:

$$\varepsilon_1 \dot{y}(t) + y(t) + b_1 y(t-1) + f(y(t-1)) = 0, \tag{4}$$

в котором $f(y) = b_2y^2 + b_3y^3 + o(y^3)$ аналитическая в окрестности $y = 0$ функция,

$$b_1 = n - 1 - n/\beta,\ b_2 = n(1 + (1-n)(\beta-1))(\beta-1)^{(n-1)/n}/(2\beta),$$
$$b_3 = (6n^2 - (5n^2+1)\beta)(\beta-1)^{(n-2)/n}/(6\beta^2). \quad (5)$$

Устойчивость нулевого решения (4) определяется поведением решений его линейной части

$$\varepsilon_1\dot{y}(t) + y(t) + b_1y(t-1) = 0, \quad (6)$$

которое, в свою очередь, определяется расположением корней характеристического уравнения

$$P(\lambda;\varepsilon_1) = \varepsilon_1\lambda + 1 + b_1\exp(-\lambda) = 0 \quad (7)$$

уравнения (6).

Уравнение (7) при фиксированном $b_1 > 0$ и $0 < \varepsilon_1 \le \varepsilon_0$, где ε_0 мало, имеет корни, лежащие в правой комплексной полуплоскости, при фиксированном $b_1 < 0$ и $0 < \varepsilon_1 \ll \varepsilon_1$ все корни уравнения лежат в левой комплексной полуплоскости [9]. В соответствии с этим, нулевое решение уравнения (4) в первом случае будет неустойчиво, в во втором – асимптотически устойчиво. Пограничным является случай $b_1 = 1$. Это определяет согласно (5) последовательность критических значений $\beta_n = n/(n-2)$.

При

$$\beta = \beta_n(1 + \varepsilon_2/(n-2-\varepsilon_2)),\ |\varepsilon_2| \ll 1 \quad (8)$$

имеем $b_1 = 1 + \varepsilon_2, b_2 = b_2(\varepsilon_2), b_3 = b_3(\varepsilon_2)$.

Изучим расположение корней характеристического уравнения $P(\lambda;\varepsilon) = 0$ и структуру решений уравнения (6) при $|\varepsilon| \le \varepsilon_0$ ($\varepsilon = (\varepsilon_1,\ \varepsilon_2)$, $|\varepsilon| = (\varepsilon_1^2 + \varepsilon_2^2)^{1/2}$). Заметим, что в рассматриваемом случае характеристическое уравнение (7) не имеет корней, лежащих на вещественной оси, поэтому достаточно рассмотреть область $Im\lambda > 0$.

Запишем уравнение (7) в виде

$$e^{\lambda}(\varepsilon_1\lambda + 1) = -(1 + \varepsilon_2),$$

что эквивалентно последовательности уравнений

$$e^{\lambda + \ln(1+\varepsilon_1\lambda)} = e^{\ln(1+\varepsilon_2) + i\pi k},\ k = 1, 3, 5, \ldots,$$

где $\ln z = \ln|z| + iargz$ $(-\pi < argz < \pi)$, $i = \sqrt{-1}$.

Для определения корней уравнения (7) достаточно рассмотреть последовательность уравнений

$$\lambda + \ln(1 + \varepsilon_1\lambda) = \ln(1 + \varepsilon_2) + i\pi k,\ k = 1, 3, 5, \ldots \quad (9)$$

Отметим, так как левая часть уравнения (7) является аналитической функцией комплексного переменного λ, то в любой ограниченной области уравнение (7) может иметь лишь конечное число корней конечной кратности. При этом цепочка корней вида $\lambda_p(\varepsilon) = \gamma_p(\varepsilon) + i\sigma_p(\varepsilon)$ $(\sigma_p(\varepsilon) > 0)$, $|\lambda_p(\varepsilon)| \to \infty$ $(p = 1, 2, \ldots)$ может существовать в уравнении (7) лишь при условии $\lim_{p\to\infty}\gamma_p(\varepsilon) = -\infty$, $\lim_{p\to\infty}|\sigma_p(\varepsilon)/\gamma_p(\varepsilon)| = \infty$

(см., например, [9]). В связи с этим существует такое $\varphi_0 > 0$, позволяющее определить область существования корней уравнения как

$$\Lambda_{\varphi_0} = \{\lambda : 0 < arg\lambda < \pi - \varphi_0\}.$$

Рассмотрим уравнение

$$F(\lambda;\varepsilon_1) \equiv \lambda + \ln(1+\varepsilon_1\lambda) = w, \tag{10}$$

считая $\lambda \in \Lambda_{\varphi_0}$, $w \in W_{x_0} = \{w : -x_0 < Rew < x_0,\ Imw \geq \pi,\ x_0 > 0-$ некоторое малое фиксированное число $\}$. Так как при $0 \leq \varepsilon_1 \leq \varepsilon_0 < \sin\varphi_0$ и $\lambda \in \Lambda_{\varphi_0}$ $\varepsilon_1/|1+\varepsilon_1\lambda| \leq q < 1$, то равномерно относительно ε_1 и λ $m_0 < |F_\lambda(\lambda;\varepsilon_1)| < m_\infty$, где m_0, m_∞ – фиксированные положительные постоянные. Отсюда следует, что уравнение (10) при $0 \leq \varepsilon \leq \varepsilon_0$ имеет единственное решение $\lambda(w;\varepsilon)$, которое может быть получено посредством итерационного процесса

$$\lambda_p + \ln(1+\varepsilon_1\lambda_{p-1}) = w,\ p = 1,2,\ldots,\lambda_0 = w. \tag{11}$$

Итерационный процесс сходится равномерно относительно $0 \leq \varepsilon \leq \varepsilon_0$ и w из любой ограниченной подобласти W_{x_0}. Предельная функция

$$\lambda(w;\varepsilon_1) = w + \lambda^1(w;\varepsilon_1),$$
$$\lambda^1(w;\varepsilon_1) = -\ln(1+\varepsilon_1(w-\ln(1+\varepsilon_1(w-\ln(1+\varepsilon_1(w-\ldots)))))) \tag{12}$$

будет непрерывной по совокупности переменных, аналитической по w при каждом $0 < \varepsilon_1 \leq \varepsilon_0$ и аналитической по ε_1 при каждом фиксированном $w \in W_{x_0}$.

Отметим следующее:

$$\lambda(w;\varepsilon) - \lambda_1(w;\varepsilon) = -\ln(1+\varepsilon_1 w^*) + \ln(1+\varepsilon_1 w) = \varepsilon_1^2(1+\varepsilon_1\lambda^*)^{-1}\ln(1+\varepsilon_1 w^*), \tag{13}$$

где согласно (11)–(12) $\lambda(w;\varepsilon) = w - \ln(1+\varepsilon_1 w)$, $w^* = w - \ln(1+\varepsilon_1(w-$ $-\ln(1+\varepsilon_1(w-\ldots))))$, точка λ^* находится на прямой, соединяющей точки w и w^*. С учетом того, что выражение $|(1+\varepsilon_1\lambda^*)^{-1}\ln(1+\varepsilon_1 w^*)|$ ограничено равномерно относительно $0 < |\varepsilon| \leq \varepsilon_0$, $w \in W_{x_0}$, имеем оценку

$$|\lambda(w;\varepsilon) - \lambda_1(w;\varepsilon)| < K\varepsilon_1^2\ (K > 0). \tag{14}$$

Отсюда согласно (11), (13) множество корней характеристического уравнения (7) может быть записано в виде

$$\lambda_k(\varepsilon) = i\pi k + \ln(1+\varepsilon_2) + \lambda^1(i\pi k + \ln(1+\varepsilon_2);\varepsilon_1),\ \lambda_{-k}(\varepsilon) = \bar{\lambda}_k(\varepsilon),\ k = 1,3,5,\ldots \tag{15}$$

При этом на основании (14) равномерно относительно k

$$\lambda_k(\varepsilon) = \gamma_k(\varepsilon) + i(\pi k + \sigma_k(\varepsilon)) = i\pi k + \ln(1+\varepsilon_2) - \ln(1+\varepsilon_1(i\pi k + \ln(1+\varepsilon_2))) + O(|\varepsilon|^2),$$
$$\gamma_k(\varepsilon) = \ln(1+\varepsilon_2) - \ln((1+\varepsilon_1\ln(1+\varepsilon_2))^2 + \varepsilon_1^2\pi^2k^2)/2 + O(|\varepsilon|^2),$$
$$\sigma_k(\varepsilon) = -\arccos((1+\varepsilon_1\ln(1+\varepsilon_2))/((1+\varepsilon_1\ln(1+\varepsilon_2))^2 + \varepsilon_1^2\pi^2k^2)^{1/2}) + O(|\varepsilon|^2). \tag{16}$$

Таким образом, вопрос устойчивости решений уравнения (6) сводится к анализу функций $\gamma_k(\varepsilon)$, являющихся аналитическими в точке $\varepsilon = 0$ и имеющих радиус сходимости соответствующих рядов $r_k = O(k^{-1})$. При этом имеем

$$\gamma_k(\varepsilon) = \varepsilon_2 - \varepsilon_1^2(\pi k)^2/2 - \varepsilon_1\varepsilon_2 - \varepsilon_2^2/2 + o(|\varepsilon|^3),\ k = 1,3,5,\ldots,$$

т.е. при малых ε и выполнении неравенства $\varepsilon_2 > \varepsilon_1^2(\pi k)^2/2$ k-й корень характеристического уравнения (7) имеет положительную вещественную часть.

2. Структура решений уравнения (6)

Опишем теперь структуру решений уравнения (6). Фазовым пространством уравнения (6) является пространство непрерывных вещественных функций $C[-1;0]$, норму в котором определяем стандартным способом и обозначим $||\cdot||_C$. Под решением уравнения (6)(уравнения (4)), определенным при $t \geq 0$, с начальным условием $y_0(s) \in C[-1;0]$ будем понимать функцию $y(t+s;\varepsilon)(-1 \leq s \leq 0)$, которая обращает уравнение (6)(уравнение (4)) в тождество и удовлетворяет начальному условию $y(s;\varepsilon) \equiv y_0(s)$.

Перейдем от уравнения (6) к эквивалентной начально-краевой задаче в полосе $-1 \leq s \leq 0, t \geq 0$, положив $u(s,t) = y(t+s)$:

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial s}, \tag{17}$$

$$\varepsilon_1 \frac{\partial u}{\partial s}\bigg|_{s=0} = -u(0,t) - (1+\varepsilon_2)u(-1,t),\ u(s,0) = y_0(s). \tag{18}$$

Производящим оператором полугруппы линейных вполне непрерывных операторов $T(t;\varepsilon)$ $(T(t_1+t_2;\varepsilon) = T(t_1;\varepsilon)T(t_2;\varepsilon) = T(t_2;\varepsilon)T(t_1;\varepsilon),\ T(0;\varepsilon) = I$ – единичный оператор), действующих в пространстве $C[-1,0]$ и определяющих решение задачи (17)–(18), будет оператор

$$A(\varepsilon)v = \begin{cases} dv/ds, -1 \leq s < 0, \\ -\varepsilon_1^{-1}(v(0) + (1+\varepsilon_2)v(-1)), s = 0 \end{cases} \tag{19}$$

с областью определения $D(A) = \{v(s) \in C^1[-1,0], \varepsilon v'(0) + v(0) + (1+\varepsilon_2)v(-1) = 0\}$.

Собственными значениями оператора $A(\varepsilon)$ являются величины $\lambda_k = \lambda_k(\varepsilon)$, а соответствующими собственными функциями будут функции
$e_k(s;\varepsilon) = e^{\lambda_k(\varepsilon)s}/P'(\lambda_k(\varepsilon);\varepsilon) = e^{\lambda_k(\varepsilon)s}/(1+\varepsilon_1+\varepsilon_1\lambda_k(\varepsilon)),\ k = \pm 1, \pm 3, \ldots\ ||e_k(s;\varepsilon)||_C \sim 1$ при $k \to \infty$.

Наряду с (19) введем в рассмотрение оператор

$$A^*(\varepsilon)h = \begin{cases} -dh/ds, 0 < s \leq 1, \\ -\varepsilon_1^{-1}(h(0) + (1+\varepsilon_2)h(1)), s = 0 \end{cases} \tag{20}$$

действующий в пространстве $C[0,1]$, с областью определения $D(A^*) = \{h(s) \in C^1[0,1],\ -\varepsilon h'(0) + h(0) + (1+\varepsilon_2)v(1) = 0\}$. Оператор (20) является сопряженным с (19) в смысле скалярного произведения С.Н. Шиманова [10], которое для краевой задачи (17)–(18) принимает вид

$$< v(s), h(s) >= \varepsilon_1 v(0)\bar{h}(0) - (1+\varepsilon_2)\int_{-1}^{0} \bar{h}(\xi+1)v(\xi)d\xi.$$

Заметим, что собственными значениями оператора $A^*(\varepsilon)$ являются величины $-\bar{\lambda}_n(\varepsilon)$, а соответствующими собственными функциями будут функции $h_k = h_k(s;\varepsilon) = e^{-\bar{\lambda}_k(\varepsilon)s}$. Между функциями $e_k(s;\varepsilon)$ и $h_k(s;\varepsilon)$ выполнены следующие условия ортогональности

$$< e_{k_1}(s;\varepsilon), h_{k_2}(s;\varepsilon) >= \delta_{k_1 k_2}, \tag{21}$$

где $\delta_{n_1 n_2}$ – символ Кронекера. Отметим, что выражения

$$p_k(v;\varepsilon) =< v(s), h_k(s,\varepsilon) >= \int_{-1}^{0} v(s) dr_k(s,\varepsilon),\ p_{-k}(v;\varepsilon) = \bar{p}_k(v;\varepsilon),$$

$$r_k(s;\varepsilon) = \begin{cases} \lambda_k^{-1}(\varepsilon),\ s = 0, \\ -(\varepsilon_1^{-1} + \lambda_k^{-1}(\varepsilon))e^{-\lambda_k(\varepsilon)s}, -1 \le s < 0,\ k = \pm 1, \pm 3, \dots, \end{cases}$$

определяют последовательность линейных непрерывных комплекснозначных функционалов в пространстве $C[-1;0]$, нормы которых

$$||p_k(v;\varepsilon)|| = \sup_{||v(s)||_C = 1} \le \overset{0}{V}_{-1} |r_k(s;\varepsilon)| = O(1), k \to \infty \tag{22}$$

Теорема 1. *Система функций $e_k(s,\varepsilon)$ полна в $C[-1,0]$ в следующем смысле: не существует функции $q(s) \in C[-1,0]$, $q(s) \not\equiv 0$, для которой $< q(s), h_k(s) >= 0$, $k = \pm 1, \pm 3, \dots$*

Доказательство теоремы 1. Оператор $A^{-1}(\varepsilon)$, действующий в $C[-1,0]$, имеет следующий вид:

$$A^{-1}(\varepsilon) f(s) = \int_{-1}^{0} f(s_1) d_{s_1} R(s, s_1; \varepsilon), \tag{23}$$

где

$$R(s, s_1; \varepsilon) = \frac{1}{2 + \varepsilon_2} \begin{cases} (1 + \varepsilon_2)(s + 1), -1 \le s_1 < s, \\ 1 + \varepsilon_2 - s_1, s \le s_1 < 0, \\ 1 - \varepsilon_1 + \varepsilon + 2, s_1 = 0, \end{cases}$$

является вполне непрерывным оператором, имеющим однократные собственные значения $\lambda_k^{-1}(\varepsilon)$, $k = \pm 1, \pm 3, \dots$, $|\lambda_1^{-1}(\varepsilon)| > |\lambda_2^{-1}(\varepsilon)| > \dots$, которым отвечают собственные функции $e_k(s;\varepsilon)$, $k = \pm 1, \pm 3, \dots$. Из вида $A^{-1}(\varepsilon)$ следует, что $A^{-1}(\varepsilon) f(s) \equiv 0$ тогда и только тогда, когда $f(s) \equiv 0$. Для оператора (23) в $C[-1,0]$ существует эквивалентная норма $||\ ||_C^*$, в которой $||A^{-1}(\varepsilon)||_C^* = \sup_{||f||_C^* = 1} ||A^{-1}(\varepsilon) f(s)||_C^* = \max_k |\lambda_k^{-1}(\varepsilon)| = |\lambda_1^{-1}(\varepsilon)|$ (см., например, [11], стр. 15–16).

Предположим, что нашлась $q(s) \in C[-1,0]$, $q(s) \not\equiv 0$, $< q(s), h_k(s) >= 0$ $k = \pm 1, \pm 3, \dots$ Нормируем $q(s) \to q(s)/||q(s)||_C^*$, т.е. $||q(s)||_C^* = 1$. Обозначим $||A^{-1}(\varepsilon) q(s)||_C^* = \delta(\varepsilon) > 0$. Рассмотрим все собственные значения $\lambda_k^{-1}(\varepsilon)$, для которых $|\lambda_k^{-1}(\varepsilon)| \ge \delta(\varepsilon)/2$, $|\lambda_{k+1}^{-1}(\varepsilon)| < \delta(\varepsilon)/2$, а их может быть конечное число $k = \pm 1, \pm 3, \dots, \pm l$. Обозначим через $C_{l+1}[-1,0]$ подпространство функций $C[-1,0]$, для которых $< v(s), h_k(s,\varepsilon) >= 0$, $k = \pm 1, \pm 3, \dots, \pm l$. При этом $\sup_{||f(s)||_C^* = 1, f(s) \in C_{l+1}[-1,0]} ||A^{-1}(\varepsilon) f(s)||_C^* = |\lambda_{l+1}^{-1}(\varepsilon)| < \delta(\varepsilon)/2$, но $q(s) \in C_{l+1}[-1,0]$ и $||A^{-1}(\varepsilon) q(s)||_C^* = \delta(\varepsilon)$. Получили противоречие. Теорема доказана. □

Обозначим через l_2 пространство комплексных последовательностей вида $z = (z_1, z_{-1}, z_3, z_{-3}, \dots)$, $z_k \in C$, $z_{-k} = \bar{z}_k$, $||z||_{l_2}^2 = \sum_{k=1}^{\infty} |z_k|^2 < \infty$. Через l_2^1 обозначим подпространство l_2 комплексных последовательностей $z = (z_1, z_{-1}, z_3, z_{-3}, \dots)$, для которых $||z||_{l_2^1}^2 = \sum_{k=1}^{\infty} |\lambda_k(\varepsilon)|^2 |z_k|^2 < \infty$ и $|| \sum_{k=-\infty}^{\infty} z_k \lambda_k(\varepsilon) e_k(s;\varepsilon)||_C < \infty$. Подпространство $C[-1,0]$ функций указанного вида обозначим $D_*(A)$.

В дальнейшем $s(r_0) = \{z \in l_2,\ ||z||_{l_2} \le r_0\}$, $s^1(r_0) = \{z \in l_2^1,\ ||z||_{l_2^1} \le r_0\}$.

Теорема 2. *Решение $u(s,t;\varepsilon)$ начально-краевой задачи (17)–(18) с начальным условием $u(s,0;\varepsilon) = y_0(s) \in C[-h,0]$ при $t \geq t_0$ $(t_0 \geq 3)$ может быть представлено в виде*

$$u(s,t;\varepsilon) = \sum_{k=\pm1,\pm3,\dots} z_k(t;\varepsilon)e_k(s;\varepsilon),\ \ z(t;\varepsilon) = (z_1(t;\varepsilon), z_{-1}(t;\varepsilon),\dots) \in l_2^1, \tag{24}$$

где $z(t;\varepsilon)$ решение уравнения

$$\dot{z} = \Lambda(\varepsilon)z,\ \ \Lambda(\varepsilon)z = (\lambda_1(\varepsilon)z_1, \lambda_{-1}(\varepsilon)z_{-1},\dots), \tag{25}$$

в пространстве l_2 с начальным условием
$z(t_0;\varepsilon) = (z_1(t_0;\varepsilon), z_{-1}(t_0;\varepsilon),\dots),\ \ z_k(t_0;\varepsilon) =< u(s,t_0;\varepsilon), h_k(s;\varepsilon) >, k = \pm1,\pm3,\dots$

Обратно, если $z(t;\varepsilon)$ при $t \geq t_0$ решение уравнения (25) с начальным условием $z(t_0;\varepsilon) \in l_2^1$, то выражение (24) определяет при $t \geq t_0$ решение начально-краевой задачи (17)–(18) с начальным условием $u(s,t_0;\varepsilon)$, определяемым (24).

Доказательство теоремы 2. При $t_0 \geq 3$ решение начально-краевой задачи (17)–(18) (уравнения (6)) $u(s,t_0;\varepsilon) = y(t_0+s;\varepsilon) \in C^3[-1,0]$ и $\varepsilon_1 y'''(t_0) + y''(t_0) + (1+\varepsilon_2)y''(t_0) = 0$. С учетом этого и равенств (15), (22) имеем

$$z_k(t_0;\varepsilon) =< y'''(t_0+s;\varepsilon), h_k(s;\varepsilon) > \lambda_k(\varepsilon)^{-3},\ \ |z_k(t_0;\varepsilon)| = O(k^{-3}), k \to \infty.$$

Таким образом, $z(t_0;\varepsilon) \in l_2^1$ и $z_k(t,\varepsilon) = e^{\lambda_k(\varepsilon)(t-t_0)}z_k(t_0;\varepsilon), k = \pm1,\pm3,\dots$ Подставив $z_k(t;\varepsilon)$ в (24), замечаем, что ряд и его производная по t при $t \geq t_0$ на основании теоремы 1 равномерно относительно $-1 \leq s \leq 0$ сходится и удовлетворяет начально-краевой задаче (17)–(18) при $t \geq t_0$. Обратно, если $z(t;\varepsilon)$ является решением (25) с начальным условием $z_k(t_0;\varepsilon) \in l_2^1$, то определяемая рядом (24) функция $y(t_0+s) \in D_*(A)$ и при $t > t_0$ удовлетворяет начально-краевой задаче (17)–(18), в чем легко убедиться непосредственной подстановкой. Теорема доказана. □

3. Нормальная форма уравнения (4)

Отметим сначала следующие свойства функций $\lambda_k(\varepsilon)$.

Теорема 3. *Существуют такие $\varepsilon_0 > 0, c_0 > 0$, что при $0 \leq \varepsilon \leq \varepsilon_0$ равномерно относительно $k_j = \pm1,\pm3,\dots$ $(j = 1,...,4)$ выполнены неравенства*

$$|\lambda_{k_1}(\varepsilon) + \lambda_{k_2}(\varepsilon) + \lambda_{k_3}(\varepsilon)|, |\lambda_{k_1}(\varepsilon) + \lambda_{k_2}(\varepsilon) + \lambda_{k_3}(\varepsilon) + \lambda_{k_4}(\varepsilon)| \geq c_0, \tag{26}$$

вторые при условии $k_1 + k_2 + k_3 + k_4 \neq 0$.

Доказательство теоремы 3. Доказательство непосредственно следует из формул (15), (16). Рассмотрим сначала первое неравенство (26). Согласно (16) $|\lambda_{k_1}(0) + \lambda_{k_2}(0) + \lambda_{k_3}(0)| \geq \pi$. Рассмотрим случай равенства, и пусть для определения $k_1, k_2 > 0$, а $k_3 = -k_1 - k_2 + 1$. При $\varepsilon \neq 0$ согласно (16) $\pi k_j - \pi/2 < Im\lambda_j(\varepsilon) < \pi k_j$ и $Im\lambda_{k_j} \to \pi k_j - \pi/2$ при $k_j \to \infty (j = 1,2)$. Но тогда $\inf_{k_1,k_2,|\varepsilon|\leq\varepsilon_0}(Im(\lambda_{k_1}(\varepsilon)+\lambda_{k_2}(\varepsilon)+\lambda_{k_3}(\varepsilon)))^2 > 0$. Тем самым существует $c_0 > 0$ с указанными в (26) свойствами. Строгое неравенство и другие комбинации k_1, k_2, k_3 рассматриваются аналогично. Для второго неравенства (26) при $k_1+k_2+k_3+k_3 \neq 0$ $|\lambda_{k_1}(0)+\lambda_{k_2}(0)+\lambda_{k_3}(0)+\lambda_{k_4}(0)| \geq 2\pi$. С учетом этого и согласно (16) имеем $inf_{k_1,k_2,k_3,k_4,|\varepsilon|<\varepsilon_0}(Im(\lambda_{k_1}(\varepsilon) + \lambda_{k_2}(\varepsilon) + \lambda_{k_3}(\varepsilon) + \lambda_{k_4}(\varepsilon)))^2 \geq \pi^2$. Теорема доказана. □

Перейдем от уравнения (4) к эквивалентной начально-краевой задаче

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial s}, \tag{27}$$

$$\varepsilon_1 \frac{\partial u}{\partial s}\bigg|_{s=0} = -u(0,t) - (1+\varepsilon_2)u(-1,t) - f(u(1,t)),\ u(s,0) = y_0(s) \tag{28}$$

в полосе $-1 \le s \le 0, t \ge 0$, положив $u(t,s) = y(t+s)$.

Введем в рассмотрение функцию-оператор

$$u(s,z;\varepsilon) = \sum_{k=\pm1,\pm3,\dots} z_k e_k(s;\varepsilon) + \sum_{(k_1,k_2)=\Omega_2} z_{k_1} z_{k_2} u_{k_1k_2}(s;\varepsilon) + \sum_{(k_1,k_2,k_3)=\Omega_3} z_{k_1} z_{k_2} z_{k_3} u_{k_1k_2k_3}(s;\varepsilon), \tag{29}$$

где $\Omega_2 = \{(k_1,k_2) : k_1,k_2 = \pm1,\pm3,\dots,k_1 \le k_2\}$, $\Omega_3 = \{(k_1,k_2,k_3) : k_1,k_2,k_3 = \pm1,\pm3,\dots,k_1 \le k_2 \le k_3\}$, действующую из $s^1(r_0) \otimes \{|\varepsilon| < \varepsilon_0\}$ в $C[-1,0]$ и гладко зависящую от своих переменных, и систему дифференциальных уравнений

$$\dot z_k = \lambda_k(\varepsilon) z_k + \sum_{(k_1k_2k_3)\in\Omega_k^3} d_{k_1k_2k_3}(\varepsilon) z_{k_1} z_{k_2} z_{k_3} \tag{30}$$

в пространстве l_2 с областью определения правой части $s^1(r_0)$, гладко зависящей от ε. Функции $u_{k_1k_2}(s;\varepsilon)$, $u_{k_1k_2k_3}(s;\varepsilon)$, $d_{k_1k_2k_3}(\varepsilon)$ подлежат определению.

Условие принадлежности траекторий системы уравнений (30) в силу (29) краевой задачи (27)–(28) примут вид

$$\sum_{k=\pm1,\pm3,\dots} \frac{\partial u(s,z;\varepsilon)}{\partial z_k}(\lambda_k(\varepsilon) z_k + \sum_{(k_1k_2k_3)\in\Omega_k^3} d_{k_1k_2k_3}(\varepsilon) z_{k_1} z_{k_2} z_{k_3}) = \frac{\partial u(s,z;\varepsilon)}{\partial s}, \tag{31}$$

$$\varepsilon_1 \frac{\partial u(s,z;\varepsilon)}{\partial s}\bigg|_{s=0} = -u(0,z;\varepsilon) - (1+\varepsilon_2)u(-1,z;\varepsilon) - f(u(-1,z;\varepsilon)). \tag{32}$$

Соотношения (31)–(32) определяют тождества, которые должны равномерно по $|\varepsilon| < \varepsilon_0$ выполняться с точностью до величины $o(|z|_{l_2}^3)$. При первых степенях z_k они выполняются в силу определения функций $e_k(s;\varepsilon)$ и $\lambda_k(\varepsilon)$. Соотношения (31)–(32) позволяют последовательно определять функции $u_k(s;\varepsilon)$ и $d_k(\varepsilon)$, приравнивая коэффициенты справа и слева при одинаковых степенях z_k.

Приравняем коэффициенты в (31)–(32) при $z_{k_1} z_{k_2}$. В результате получим краевую задачу

$$(\lambda_{k_1}(\varepsilon) + \lambda_{k_2}(\varepsilon)) u_{k_1k_2}(s;\varepsilon) = \frac{du_{k_1k_2}(s;\varepsilon)}{ds}, \tag{33}$$

$$\varepsilon_1 \frac{du_{k_1k_2}(s;\varepsilon)}{ds}\bigg|_{s=0} = -u_{k_1k_2}(0;\varepsilon) - (1+\varepsilon_2)u_{k_1k_2}(-1;\varepsilon) - p_{k_1k_2} b_2(\varepsilon_2) e_{k_1}(-1;\varepsilon) e_{k_2}(-1;\varepsilon) \tag{34}$$

для определения $u_{k_1k_2}(s;\varepsilon)$. В (34) $p_{k_1k_2} = 1$ при $k_1 = k_2$ и 2 при $k_1 \ne k_2$. Решение (33)–(34) определяется однозначно

$$u_{k_1k_2}(s;\varepsilon) = -p_{k_1k_2} b_2(\varepsilon_2) e^{(\lambda_{k_1}(\varepsilon)+\lambda_{k_2}(\varepsilon))s} / P(\lambda_{k_1}(\varepsilon) + \lambda_{k_2}(\varepsilon);\varepsilon). \tag{35}$$

Приравняем в (31)–(32) коэффициенты при $z_{k_1}z_{k_2}z_{k_3}$. В результате получим краевую задачу

$$e_k(s;\varepsilon)d_{k_1k_2k_3}(\varepsilon)(\lambda_{k_1}(\varepsilon)+\lambda_{k_2}(\varepsilon)+\lambda_{k_3}(\varepsilon))u_{k_1k_2k_3}(s;\varepsilon)=\frac{du_{k_1k_2k_3}(s;\varepsilon)}{ds},\ k=k_1+k_2+k_3 \tag{36}$$

$$\varepsilon_1\frac{du_{k_1k_2k_3}(s;\varepsilon)}{ds}\Big|_{s=0}=-u_{k_1k_2k_3}(0;\varepsilon)-(1+\varepsilon_2)u_{k_1k_2k_3}(-1;\varepsilon)-f_{k_1k_2k_3}(\varepsilon), \tag{37}$$

в которой

$$f_{k_1k_2k_3}=-p_{k_1k_2k_3}b_3(\varepsilon_2)e_{k_1}(-1;\varepsilon)e_{k_2}(-1;\varepsilon)e_{k_3}(-1;\varepsilon)-2b_2(\varepsilon_2)*$$
$$*\begin{cases} e_{k_1}(-1;\varepsilon)u_{k_2k_3}(-1;\varepsilon), \text{если } k_1=k_2=k_3 \\ e_{k_j}(-1;\varepsilon)u_{k_mk_p}(-1;\varepsilon)+e_{k_m}(-1;\varepsilon)u_{k_jk_p}(-1;\varepsilon), \text{если } k_j\neq k_m=k_p, j,m,p=1,2,3 \\ e_{k_1}(-1;\varepsilon)u_{k_2k_3}(-1;\varepsilon)+e_{k_2}(-1;\varepsilon)u_{k_1k_3}(-1;\varepsilon)+e_{k_3}(-1;\varepsilon)u_{k_1k_2}(-1;\varepsilon), k_1\neq k_2\neq k_3, \end{cases},$$

где $p_{k_1k_2k_3}=1$, если $k_1=k_2=k_3$, $p_{k_1k_2k_3}=3$, если $k_1=k_2\neq k_3$, либо $k_1=k_3\neq k_2$, либо $k_2=k_3\neq k_1$, $p_{k_1k_2k_3}=6$, если $k_1\neq k_2\neq k_3$.

Общее решение уравнения (36) имеет вид

$$u_{k_1k_2k_3}(s;\varepsilon)(s;\varepsilon)=e^{\lambda_{k_1k_2k_3}(\varepsilon)s}(c+d_{k_1k_2k_3}(\varepsilon)(1+\varepsilon_1+\varepsilon_1\lambda_k(\varepsilon))(1-e^{(\lambda_k(\varepsilon)-\lambda_{k_1k_2k_3}(\varepsilon))s})*$$
$$*(\lambda_k(\varepsilon)-\lambda_{k_1k_2k_3}(\varepsilon))^{-1}), \tag{38}$$

где использовано обозначение $\lambda_{k_1k_2k_3}(\varepsilon)=\lambda_{k_1}(\varepsilon)+\lambda_{k_2}(\varepsilon)+\lambda_{k_3}(\varepsilon)$, c – произвольная постоянная. Подставляя (38) в краевое условие (37), имеем

$$d_{k_1k_2k_3}(s;\varepsilon)(s;\varepsilon)=(\varepsilon_1+(1+\varepsilon_2)(e^{-\lambda_{k_1k_2k_3}(\varepsilon)}-e^{-\lambda_k(\varepsilon)})/(\lambda_{k_1k_2k_3}(\varepsilon)-\lambda_k(\varepsilon)))^{-1}*$$
$$*(1+\varepsilon_1+\lambda_k(\varepsilon))f_{k_1k_2k_3}(\varepsilon). \tag{39}$$

Условие непрерывности функции $u_{k_1k_2k_3}(\varepsilon)$ по ε при $0\leq|\varepsilon|\leq|\varepsilon_0|$ дает $c=0$, что однозначно ее определяет равенством (39).

Ниже будет показано, что "грубым", т.е. экспоненциально устойчивым (неустойчивым) периодическим решениям нормальной формы (30) в краевой задаче соответствуют периодические решения того же характера устойчивости с близкими периодами. Эти решения связаны посредством оператора (29).

4. Анализ периодических решений нормальной формы (30) и уравнения (4)

Перейдем в системе уравнений (30) к полярным координатам, положив $z_k=\rho_ke^{i\tau_k}$ ($\rho_k\geq 0, -\infty<\tau_k<\infty$). В результате получим систему уравнений вида

$$\dot\rho_k=\gamma_k(\varepsilon)\rho_k+R_k(\rho,\ \tau;\ \varepsilon), \tag{40}$$

$$\dot\tau_k=\pi n+\sigma_n(\varepsilon)+T_k(\rho,\tau;\varepsilon),\ k=1,3,\ldots, \tag{41}$$

в которой $\gamma_k(\varepsilon)$ и $\sigma_k(\varepsilon)$ определены в (16), $\rho = (\rho_1, \rho_3, \dots)$, $\rho_k \geq 0, \sum_{k=1}^{\infty} k^2\rho_k^2 < \infty, \tau = (\tau_1, \tau_3, \dots)$, функционалы $R_k(\cdot), T_k(\cdot)$ гладко зависят от входящих переменных и параметров, 2π-периодические по τ_j.

Структура системы (40)–(41) позволяет ввести одну "быструю" переменную и счетное число "медленных" переменных. Как это сделать, покажем сначала на примере "усеченной" системы. Рассмотрим нормальную форму (30), в которой положим $z_k = 0,\ k = \pm 5, \pm 7, \dots$ В результате имеем систему уравнений

$$\dot{z}_1 = \lambda_1(\varepsilon)z_1 + d_{-111}(\varepsilon)z_{-1}z_1^2 + d_{-313}(\varepsilon)z_{-3}z_1z_3 + d_{-1-13}(\varepsilon)z_{-1}^2z_3, \tag{42}$$

$$\dot{z}_3 = \lambda_3(\varepsilon)z_3 + d_{-113}(\varepsilon)z_{-1}z_1z_3 + d_{-333}(\varepsilon)z_{-3}z_3^2 + d_{111}(\varepsilon)z_1^3. \tag{43}$$

Уравнения для z_{-1}, z_{-3} получаются сопряжением (42)–(43) с учетом равенств $\bar{\lambda}_k(\varepsilon) = \lambda_{-k}(\varepsilon), \bar{z}_k(\varepsilon) = z_{-k}(\varepsilon)$. Обозначив $d_*(\varepsilon) = a_*(\varepsilon) + ib_*(\varepsilon), A_*(\varepsilon) = |d_*(\varepsilon)|, \beta_*(\varepsilon) = arg d_*(\varepsilon)$, перейдем к полярным переменным $\rho_1, \rho_3, \tau_1, \tau_3$. В результате получим систему уравнений

$$\dot{\rho}_1 = (\gamma_1(\varepsilon)+a_{-111}(\varepsilon)\rho_1^2+a_{-313}(\varepsilon)\rho_3^2)\rho_1+A_{-1-13}(\varepsilon)\cos(-3\tau_1+\tau_3+\beta_{-1-13}(\varepsilon))\rho_1^2\rho_3, \tag{44}$$

$$\dot{\rho}_3 = (\gamma_3(\varepsilon) + a_{-1-13}(\varepsilon)\rho_1^2 + a_{-333}(\varepsilon)\rho_3^2)\rho_3 + A_{111}(\varepsilon)\cos(3\tau_1 - \tau_3 + \beta_{111}(\varepsilon))\rho_1^3, \tag{45}$$

$$\dot{\tau}_1 = \pi + \sigma_1(\varepsilon) + b_{-111}(\varepsilon)\rho_1^2 + b_{-313}(\varepsilon)\rho_3^2 + A_{-1-13}(\varepsilon)\sin(-3\tau_1+\tau_3+\beta_{-1-13}(\varepsilon))\rho_1\rho_3, \tag{46}$$

$$\dot{\tau}_3 = 3\pi + \sigma_3(\varepsilon) + b_{-1-13}(\varepsilon)\rho_1^2 + b_{-333}(\varepsilon)\rho_3^2 + A_{111}(\varepsilon)\sin(3\tau_1 - \tau_3 + \beta_{111}(\varepsilon))\rho_1^3/\rho_3. \tag{47}$$

Перейдем в (44)–(47) к переменным $\rho_1, \rho_3, \theta_1 = -3\tau_1 + \tau_3$ и $\tau = \tau_1$. В результате получим систему уравнений

$$\dot{\rho}_1 = (\gamma_1(\varepsilon)+a_{-111}(\varepsilon)\rho_1^2+a_{-313}(\varepsilon)\rho_3^2)\rho_1+A_{-1-13}(\varepsilon)\rho_1^2\cos(-3\tau_1+\tau_3+\beta_{-1-13}(\varepsilon))\rho_1^2\rho_3, \tag{48}$$

$$\dot{\rho}_3 = (\gamma_3(\varepsilon) + a_{-1-13}(\varepsilon)\rho_1^2 + a_{-33-3}(\varepsilon)\rho_3^2)\rho_3 + A_{111}(\varepsilon)\rho_1^2\cos(3\tau_1 - \tau_3 + \beta_{111}(\varepsilon))\rho_1^3, \tag{49}$$

$$\begin{aligned}\dot{\theta}_1 = \delta_1(\varepsilon) + (-3b_{-111}(\varepsilon) + b_{-1-13}(\varepsilon))\rho_1^2 + (-3b_{-313}(\varepsilon) + b_{-333}(\varepsilon))\rho_3^2 - \\ - 3A_{-1-13}(\varepsilon)\sin(\theta_1 + \beta_{-1-13}(\varepsilon))\rho_1\rho_3 + A_{111}(\varepsilon)\sin(-\theta_1 + \beta_{111}(\varepsilon))\rho_1^3/\rho_3,\end{aligned} \tag{50}$$

"медленных" переменных, где $\delta_1(\varepsilon) = -3\sigma_1(\varepsilon) + \sigma_3(\varepsilon)$, и уравнение "быстрой" переменной

$$\dot{\tau} = \pi + \sigma_1(\varepsilon) + b_{-111}(\varepsilon)\rho_1^2 + b_{-313}(\varepsilon)\rho_3^2 + A_{-1-13}(\varepsilon)\sin(\theta_1 + \beta_{-1-13}(\varepsilon))\rho_1\rho_3. \tag{51}$$

Заметим, что правая часть (48)–(50) не зависит от τ.

Пусть теперь в (30) все $z_k(t) \neq 0$. Ввести переменные θ_k можно не единственным способом, однако они все связаны между собой линейными соотношениями. В качестве одного из возможных способов введения θ_k может быть предложен следующий. В качестве "быстрой" переменной берем τ_1 и рассматриваем "усеченные" конечномерные системы, последовательно полагая в (30) $z_k = 0, k = \pm 5, \pm 7, \dots$, затем $k = \pm 7, \pm 9, \dots$ и т.д. Первый случай рассмотрен выше. Во втором случае к системе (48)–(50) добавляются два новых уравнения для переменных z_5 и z_{-5}. При этом в правой части уравнения для z_1 появляется резонансный моном $z_5z_{-3}z_{-1}$(в

уравнении для z_{-1} появляется резонансный моном $z_{-5}z_3z_1$). При переходе к полярным координатам это приводит к появлению слагаемых, зависящих от выражения $\tau_5-\tau_3-2\tau_1=\theta_3$, которое берем в качестве новой "медленной" переменной. В результате имеем две дополнительные "медленные" переменные ρ_5,θ_3. Система уравнений для $\rho_1,\rho_3,\rho_5,\theta_1,\theta_3$ будет иметь вид, аналогичный (48)–(50), правая часть которой также не будет зависеть от τ. В правой части уравнения (51) для τ появятся новые слагаемые. При рассмотрении следующей "усеченной" системы в правой части уравнения для z_1 появится резонансный моном $z_7z_{-5}z_{-1}$, а при переходе к полярным координатам слагаемое, зависящее от выражения $\tau_7-\tau_5-2\tau_1=\theta_5$. Имеем две новые "медленные" переменные ρ_7,θ_5. В общем случае на очередном шаге, добавив уравнения для z_{k_0} и z_{-k_0}, получим в уравнениях для z_1 и z_{-1} появление новых мономов $z_{k_0}z_{-k_0+2}z_{-1}$ и $z_{-k_0}z_{k_0-2}z_1$ соответственно, а при переходе к полярным координатам появятся слагаемые, зависящие от $\tau_{k_0}-\tau_{k_0-2}-2\tau_1=\theta_{k_0-2}$. Имеем две новые "медленные" переменные $\rho_{k_0},\theta_{k_0-2}$. Продолжая этот процесс, осуществим переход к "медленным" переменным $\rho=(\rho_1,\rho_3,\dots),\theta=(\theta_1,\theta_3,\dots)$ и "быстрой" τ, а соответствующая система дифференциальных уравнений для их определения будет иметь вид

$$\dot\rho_k=\gamma_k(\varepsilon)\rho_k+R_k(\rho,\ \theta;\ \varepsilon),\tag{52}$$

$$\dot\theta_k=\delta_k(\varepsilon)+\Theta_k(\rho,\theta;\ \varepsilon),\ \ k=1,3,\dots,\tag{53}$$

$$\dot\tau=\pi+\sigma_1(\varepsilon)+T(\rho,\theta;\ \varepsilon),\tag{54}$$

в которой функционалы $R_k(\cdot),\ \Theta_k(\cdot),\ T(\cdot)-\ 2\pi-$ периодические по θ_k, остальные их свойства и функции $\delta_k(\varepsilon)(\delta_k(0)=0)$ определяются свойствами функций и функционалов, входящих в (40)–(41). Особо отметим, что правая часть (54) не зависит от переменной τ.

Фазовым пространством системы уравнений (52)-(54) будет произведение пространств $l_2\otimes c\times R$, здесь $l_2=\{\rho=(\rho_1,\rho_3,\dots)\},\rho_k\geq 0,||\rho||^2_{l_2}=\sum_{k=1}^{\infty}\rho_k^2<\infty,\ c=\{\theta=(\theta_1,\theta_3,\dots)\},||\theta||_c=\sup_k|\theta_k|<\infty$. Областью определения правой части (52)–(54) является произведение пространств $l_2^1\otimes c_0\times R$, здесь $l_2^1=\{\rho\in l_2,||\rho||^2_{l_2^1}=\sum_{k=1}^{\infty}k^2\rho_k^2<\infty\},c_0=\{\theta=(\theta_1,\theta_3,\dots)\in c,0\leq\theta_k<2\pi\}$.

Введем в области $\{(\varepsilon_1,\varepsilon_2),\varepsilon_1>0,|\varepsilon|<\varepsilon_0\}$ переменные $\zeta\geq 0$ и $\pi/2<\psi<\pi/2$, положив

$$\zeta=(\varepsilon_1^2+|\varepsilon_2|)^{1/2},\varepsilon_1=\zeta\cos\psi,\varepsilon_2=\zeta^2\sin^2\psi sign\psi.\tag{55}$$

Подставим (55) в (52)–(54), нормировав также $\rho_k\to\zeta\rho_k$. В результате получим систему уравнений

$$\dot\rho_k=\gamma_k(\psi,\zeta)\rho_k+\zeta^2R_k(\rho,\ \theta;\ \psi,\zeta),\tag{56}$$

$$\dot\theta_k=\delta_k(\psi,\zeta)+\zeta^2\Theta_k(\rho,\theta;\ \psi,\zeta),\ \ k=1,3,\dots,\tag{57}$$

$$\dot\tau=\pi+\sigma_1(\psi,\zeta)+\zeta^2T(\rho,\theta;\ \psi,\zeta),\tag{58}$$

в которой свойства функций и функционалов определяются свойствами функций и функционалов системы (52)–(55).

Отметим, что ввиду того, что согласно (16), (56) равномерно относительно k $\gamma(\psi,\zeta)=O(\zeta^2),\delta_k(\psi,\zeta)=O(\zeta^3)$, уравнения (56), (57) оправдывают название уравнений "медленных" переменных, а (58) – уравнения "быстрой" переменной.

Рассмотрим главную часть системы уравнений "медленных" переменных (56)–(57), предварительно нормировав время $t \to t/\zeta^2$. В результате имеем

$$\dot{\rho}_k = \gamma_k^*(\psi,\zeta)\rho_k + R_k(\rho,\ \theta;\ \psi,\zeta), \tag{59}$$

$$\dot{\theta}_k = \delta_k^*((\psi,\zeta) + \Theta_k(\rho,\theta;\ \psi,\zeta),\ k = 1,3,\dots, \tag{60}$$

где $\gamma_k^*(\psi,\zeta) = \gamma_k(\psi,\zeta)/\zeta^2$, $\delta_k^*(\psi,\zeta) = \delta_k(\psi,\zeta)/\zeta^2$ согласно (15)–(16) непрерывные функции $\pi/2 \le \psi \le -\pi/2, 0 \le \zeta \le \zeta_0$. При этом

$$\gamma_k^*(\psi,0) = \gamma_k(\psi) = \sin^2\psi sign\psi - \pi^2 k^2 \cos^2\psi/2, \quad \delta_k^*(\psi,0) = 0. \tag{61}$$

Рассмотрим в $l_2 \otimes c$ систему нелинейных уравнений

$$\gamma_k^*(\psi)\rho_k + R_k(\rho,\theta) = 0\ (R_k(\rho,\theta) \equiv R_k(\rho,\theta;\psi,0) = 0), \tag{62}$$

$$\Theta_k(\rho,\theta) = 0\ (\Theta_k(\rho,\theta) \equiv \Theta_k(\rho,\theta;\psi,0) = 0),\ k = 1,3,\dots, \tag{63}$$

считая $(\rho,\theta) \in E_0^1 = l_2^1 \otimes c_0$. В (62) оператор $\gamma(\psi)\rho = (\gamma_1(\psi)\rho_1, \gamma_3(\psi)\rho_3, \dots)$ рассматривается на основании (61) симметрично расширенным на l_2^1.

Пусть $(\rho^*(\psi),\theta^*(\psi)) \in l_2^1 \otimes c_0$ решение системы уравнений (62)–(63).

Введем в рассмотрение бесконечную матрицу

$$B(\psi) = \begin{pmatrix} \gamma_k(\psi)\delta_{kj} + \partial R_k/\partial\rho_j & \partial R_k/\partial\theta_j \\ \partial\Theta_k/\partial\rho_j & \partial\Theta_k/\partial\theta_j \end{pmatrix} (k,j = 1,3,\dots), \tag{64}$$

вычисленную в точке $\rho^*(\psi),\theta^*(\psi)$, где δ_{k_j} – символ Кронекера. Матрица определяет линейный оператор

$$B(\psi)v\ (v = (\rho,\theta)) : E^1 = l_2^1 \otimes c \to E = l_2 \otimes c, \tag{65}$$

$||v||_E = ||\rho||_{l_2} + ||\theta||_c$, $||v||_{E^1} = ||\rho||_{l_2^1} + ||\theta||_c$.

Пусть $\mu_k(\psi)$ ($B(\psi)v_k(\psi) = \mu_k(\psi)v_k(\psi)$) – собственное значение оператора (65). Покажем, что оператор (65) имеет счетное число собственных значений $\mu_k(\psi)$, которые могут быть пронумерованы в порядке возрастания их модулей, и любой ограниченной области комплексной плоскости может принадлежать лишь конечное число собственных значений конечной кратности. Предельной может быть лишь точка бесконечность, при этом $\lim_{k\to\infty} Re\mu_k(\psi) = -\infty$.

Действительно, заметим, что функции

$$z_1(t;\psi,\zeta) = \zeta z_1^*(\tau;\psi) = \zeta\rho_1^*(\psi)e^{i\tau}, z_3(t;\psi,\zeta) = \zeta z_3^*(\tau;\psi) = \zeta\rho_3^*(\psi)e^{i(3\tau+\theta_1^*(\psi))},$$

$$z_5(t;\psi,\zeta) = \zeta z_5^*(\tau;\psi) = \zeta\rho_5^*(\psi)e^{i(5\tau+\theta_1^*(\psi)+\theta_3^*(\psi))}, \dots, z_{-k}(t;\psi,\zeta) = \bar{z}_k(t;\psi,\zeta)\ k = 1,3,\dots,$$

$$\dot{\tau} = \pi(1 - \zeta\cos\psi), \tag{66}$$

определяют периодические решения следующей системы дифференциальных уравнений:

$$\dot{z}_k = (i\pi k(1 - \zeta\cos\psi) + \zeta^2\gamma_k(\psi)) - p_{k_1k_2k_3} \sum_{(k_1,k_2,k_3)\in\Omega_3^k} z_{k_1}z_{k_2}z_{k_3},\ k = \pm1,\pm3,\dots, \tag{67}$$

в чем легко убедиться непосредственной подстановкой (66) в (67), с учетом выполненных для $\rho^*(\psi)$, $\theta^*(\psi)$ замен и равенств (62), (63). Система уравнений (67) является абстрактно параболической в комплексном пространстве l_2 с областью определения l_2^1. Для таких уравнений оператор монодромии, построенный для периодического решения (66), является вполне непрерывным оператором, содержащим счетное число собственных значений (мультипликаторов) $\nu_j(\psi)$, $\lim_{j\to\infty}|\nu_j(\psi)| = 0, \nu_j(\psi) = e^{\mu_j(\psi)}$. Оператор (65) определяет характеристические показатели периодического решения (66). Отсюда следует все перечисленное.

Рассмотрим теперь систему уравнений (56), (57), которую запишем в операторной форме

$$\frac{d\rho}{dt} = \zeta^2(\gamma(\psi,\zeta)\rho + R(\rho,\theta;\psi,\zeta)), \tag{68}$$

$$\frac{d\theta}{dt} = \zeta^2(\delta(\psi,\zeta)\rho + \Theta(\rho,\theta;\psi,\zeta)) \tag{69}$$

в пространстве E, где функция $\delta(\psi,\zeta) = (\delta_1^*(\psi,\zeta), \delta_{-1}^*(\psi,\zeta),\dots)$, операторы $\gamma(\psi,\zeta)\rho = (\gamma_1^*(\psi,\zeta)\rho_1, \gamma_{-1}^*(\psi,\zeta)\rho_{-1},\dots)$, $R(\rho,\theta;\psi,\zeta) = (R_1(\rho,\theta;\psi,\zeta), R_{-1}(\rho,\theta;\psi,\zeta),\dots)$, $\Theta(\rho,\theta;\psi,\zeta) = (\Theta_1(\rho,\theta;\psi,\zeta), \Theta_{-1}(\rho,\theta;\psi,\zeta),\dots)$. Областью определения правой части (68)–(69) является пространство $E_0^1 = l_2^1 \otimes c_0$.

Рассмотрим в E систему нелинейных операторных уравнений

$$\gamma(\psi,\zeta)\rho + R(\rho,\theta;\psi,\zeta) = 0, \tag{70}$$

$$\delta(\psi,\zeta) + \Theta(\rho,\theta;\psi,\zeta) = 0 \tag{71}$$

с областью определения E_0^1. При $\zeta = 0$ (70)-(71) имеет решение $\rho^*(\psi), \theta^*(\psi)$. Предположим, что построенная по этому решению матрица (64) определяет оператор (65), который не имеет собственных значений, лежащих на мнимой оси комплексной плоскости, т.е. выполнено условие $|Re\mu_j(\psi)| > \mu_0 > 0$. Следовательно, по теореме о неявной функции для операторных уравнений в банаховых пространствах (см., например, [11]) существует такое $\zeta_0 > 0$, что при $0 \le \zeta \le \zeta_0$ система уравнений (70), (71) имеет решение $\rho^*(\psi,\zeta), \theta^*(\psi,\zeta)$ $(\rho^*(\psi,0) = \rho^*(\psi), \theta^*(\psi,0) = \rho^*(\psi))$, гладко зависящее от параметра ζ.

Линеаризуем (70)–(71) на решении $\rho^*(\psi,\zeta), \theta^*(\psi,\zeta)$. В результате получим бесконечную матрицу, аналогичную матрице (64) и обращающуюся в нее при $\zeta = 0$, которая определяет линейный оператор

$$B(\psi,\zeta)v \ (v = (\rho,\theta), B(\psi,0) = B(\psi)) : E^1 \to E. \tag{72}$$

Собственные значения $\mu_j(\psi,\zeta)(\mu_j(\psi,0) = \mu_j(\psi))$ оператора (72) при $0 \le \zeta \le \zeta_0$ удовлетворяют условию $|Re\mu_j(\psi,\zeta)| > \mu_0 > 0$.

Подставим $\rho^*(\psi,\zeta), \theta^*(\psi,\zeta)$ в правую часть уравнения (58). В результате имеем

$$\dot\tau = \pi + \sigma_1(\psi,\zeta) + \zeta^2 T_1(\rho^*(\psi,\zeta), \theta^*(\psi,\zeta);\psi,\zeta) = \pi + \sigma_1(\psi,\zeta) + \zeta^2\Delta_2(\psi,\zeta) = \pi + \sigma(\psi,\zeta), \quad (\sigma(\psi,0) \equiv 0). \tag{73}$$

Положим

$$z_1^*(\tau;\psi,\zeta) = \rho_1^*(\psi,\zeta)e^{i\tau}, z_3^*(\tau;\psi,\zeta) = \rho_3^*(\psi,\zeta)e^{i(3\tau+\theta_1^*(\psi,\zeta))},$$
$$z_5^*(\tau;\psi,\zeta) = \rho_5^*(\psi,\zeta)e^{i(5\tau+\theta_1^*(\psi,\zeta)+\theta_3^*(\psi,\zeta))}, z_{-k}^*(\tau;\psi,\zeta) = \bar z_k^*(\tau;\psi,\zeta) \ k = 1,3,\dots \tag{74}$$

Теорема 4. *Пусть при некотором ψ система уравнений (59),(60) имеет решение $(\rho^*(\psi), \theta^*(\psi)) \in E_0^1$, а построенная по этому решению матрица (64) определяет оператор (65), который не имеет собственных значений, лежащих на мнимой оси комплексной плоскости. Тогда существует такое $\zeta_0 > 0$, что при $0 < \zeta \le \zeta_0$ краевая задача (27),(28), в которой ε_1 и ε_2 определены согласно (55), имеет периодическое решение $u^*(s,\tau;\psi,\zeta)$, допускающее представление*

$$u^*(s,\tau;\psi,\zeta) = u_0^*(s,\tau;\psi,\zeta) + O(\zeta^4) =$$
$$= \zeta \sum_{k=\pm1,\pm3} e_k(s;\psi,\zeta) z_k^*(\tau;\psi,\zeta) + \zeta^2 \sum_{(k_1,k_2)\in\Omega_2} u_{k_1k_2}(s;\psi,\zeta) z_{k_1}^*(\tau;\psi,\zeta) z_{k_2}^*(\tau;\psi,\zeta) +$$
$$+ \zeta^3 \sum_{(k_1,k_2,k_3)\in\Omega_3} u_{k_1k_2k_3}(s;\psi,\zeta) z_{k_1}^*(\tau;\psi,\zeta) z_{k_2}^*(\tau;\psi,\zeta) z_{k_3}^*(\tau;\psi,\zeta) + O(\zeta^4), \quad (75)$$

в котором Ω_2, Ω_3 определены в (29), функции $e_k(\cdot), u_{k_1k_2}(\cdot), u_{k_1k_2k_3}(\cdot)$ определены согласно (19), (35), (38) с учетом замены (55), переменная τ определяется в соответствии с (73), функции $z_k^*(\cdot)$ определены в (74).

Доказательство теоремы 4. Решения системы уравнений (30) в силу (15)–(16) обладают свойствами нарастающей гладкости. В связи с этим функции (74) с учетом (73), являясь периодическими решениями системы уравнений (30), будут бесконечно дифференцируемыми по τ. Отсюда следует, что функция $u_0^*(s,\tau;\psi,\zeta)$ будет также бесконечно дифференцируемой по τ, непрерывно дифференцируемой по s и достаточно гладко зависящей от ψ и ζ. Обозначим $Q = \{(s,\tau) : -1 \le s \le 0, -\infty < \tau < \infty\}$. Введем в рассмотрение пространство $C^{1\infty}(\bar\Omega)$ вещественных функций $g(s,\tau) \equiv g(s,\tau+2\pi)$, определенных в $\bar\Omega$, непрерывно дифференцируемых по s и бесконечно дифференцируемых по τ. Норму в $C^{1\infty}(\bar\Omega)$ определим

$$||g(s,\tau)||_{C^{1\infty}} = \sup_{(s,\tau)\in\bar\Omega, k=0,1,\dots} ||\frac{\partial^k g(s,\tau)}{\partial\tau^k}|| + \sup_{(s,\tau)\in\bar\Omega, k=1,2,\dots} ||\frac{\partial^k g(s,\tau)}{\partial s\partial\tau^{k-1}}||$$

Введем в рассмотрение также пространство $C^\infty(R)$ вещественных бесконечно дифференцируемых функций $q(\tau) \equiv q(\tau+2\pi)(-\infty < \tau < \infty)$, определив норму $||q(\tau)||_{C^\infty(R)} = \sup_{\tau,k=0,1,\dots} |\partial^k q(\tau)/\partial\tau^k|$. Пространства $C^{1\infty}(\Omega), C^\infty(R)$ являются банаховыми пространствами.

Рассмотрим в области $\bar\Omega$ краевую задачу

$$g(s,\tau) + i(\pi + \sigma(\psi;\zeta))u_\tau = u_s, \quad (76)$$

$$u(0,\tau) + u(-1,\tau) = q(s), \quad (77)$$

в которой $g(s,\tau) \in C^{1\infty}(\bar\Omega), q(\tau) \in C^\infty(R)$, и выявим условия существования 2π-периодического решения $u_*(s,\tau;\psi,\zeta)$, непрерывно зависящего от $0 \le \zeta \le \zeta_0$.

Представим в виде равномерно сходящихся рядов

$$g(s,\tau) = \sum_{n=-\infty}^{\infty} g_n(s) e^{in\tau}, g_n(s) = \frac{1}{2\pi}\int_0^{2\pi} g(s,\tau_1) e^{in\tau_1} d\tau_1,$$

$$q(\tau) = \sum_{n=-\infty}^{\infty} q_n e^{in\tau}, q_n = \frac{1}{2\pi}\int_0^{2\pi} g(\tau_1)e^{in\tau_1}d\tau_1.$$

Определяя 2π-периодическое решение (76)–(77) в виде ряда

$$u_*(s,\tau;\psi,\zeta) = \sum_{n=-\infty}^{\infty} u_n(s;\psi,\zeta)e^{in\tau}, \tag{78}$$

рассмотрим отдельно случай $n = 2k, k = 0, \pm1, \dots$, и случай $n = 2k+1, k = 0, \pm1, \dots$

В случае $n = 2k$ для определения $u_{2k}(s;\psi,\zeta)$ имеем краевую задачу

$$g_{2k}(s) + i2k(\pi + \sigma(\psi;\zeta))u_{2k} = \frac{du_{2k}}{ds},$$

$$u_{2k}(0) + u_{2k}(-1) = q_{2k},$$

из которых однозначно находим

$$u_{2k}(s,\tau;\psi,\zeta) = e^{i2k(\pi+\sigma(\psi,\zeta))s}(c_{2k} + \int_0^s e^{-i2k(\pi+\sigma(\psi,\zeta))s_1}g_{2k}(s_1)ds_1), \tag{79}$$

где

$$c_{2k} = (1 + e^{-i2k\sigma(\psi,\zeta)})^{-1}(q_{2k} + e^{-i2k\sigma(\psi;\zeta)}\int_{-1}^0 e^{-2ki(\pi+\sigma(\psi,\zeta))s_1}g_{2k}(s_1)ds_1).$$

Рассмотрим теперь случай $n = 2k + 1$. Здесь краевая задача для определения $u_{2k}(s;\psi,\zeta)$ будет иметь вид

$$g_{2k+1}(s) + i(2k+1)(\pi + \sigma(\psi;\zeta))u_{2k+1} = \frac{du_{2k+1}}{ds}, \tag{80}$$

$$u_{2k+1}(0) + u_{2k+1}(-1) = q_{2k+1}, k = 0, \pm1, \dots \tag{81}$$

Подставив общее решение уравнения (80)

$$u_{2k+1}(s;\psi,\zeta) = e^{i(2k+1)(\pi+\sigma(\psi,\zeta))s}(c_{2k+1} + \int_0^s e^{-i(2k+1)(\pi+\sigma(\psi,\zeta))s_1})g_{2k+1}(s_1)ds_1 \tag{82}$$

в краевое условие (81), получим для определения c_{2k+1} следующее уравнение:

$$c_{2k+1}(1-e^{-i(2k+1)\sigma(\psi,\zeta)}) = q_{2k+1} - e^{-i(2k+1)\sigma(\psi,\zeta)}\int_{-1}^0 e^{-i(2k+1)(\pi+\sigma(\psi,\zeta))s_1}q_{2k+1}(s_1)ds_1. \tag{83}$$

Так как коэффициент при c_{2k+1} обращается в нуль при $\zeta = 0$, уравнение (83) в общем случае не имеет непрерывного по ζ при $0 \le \zeta \le \zeta_0$ решения. Потребуем, чтобы правая часть уравнения (83) обращалась в нуль при $\zeta = 0$, т.е.

$$q_{2k+1} - \int_{-1}^0 e^{-i(2k+1)\pi s_1}g_{2k+1}(s_1)ds_1 = 0. \tag{84}$$

При выполнении (84) уравнение (83) можно записать в виде

$$c_{2k+1}(1 + O(\sigma(\psi,\zeta))) = \int_{-1}^0 e^{-i(2k+1)\pi s_1}(1+s_1)g_{2k+1}(s_1)ds_1 + O(\sigma(\psi,\zeta)).$$

Отсюда $c_{2k+1} = c_{2k+1}(\psi, \zeta)$ однозначно определяется. Подставив это выражение в (81), имеем решение $u_{2k+1}(s; \psi, \zeta)$.

Таким образом, при выполнении условий (84) для $k = 0, \pm 1, \dots$, краевая задача (76)–(77) имеет единственное периодическое решение, определяемое рядом (78). На основании свойств $g_n(s)$ и q_n и вида функций $u_n(s; \psi, \zeta)$, определяемых (79), (82), периодическое решение (78) $u_*(s, \tau; \psi, \zeta) \in C^{1\infty}(\bar{\Omega})$.

Линейное подпространство функций $(g(s, \tau), q(\tau)) \in C^{1\infty}(\bar{\Omega}) \oplus C^{\infty}(R)$ удовлетворяющих условиям (84) при $k = 0, \pm 1, \dots$, обозначим $C_0^{1\infty}(\bar{\Omega}) \oplus C_0^{\infty}(R)$. Формула (78) с учетом (79), (82) определяет линейный ограниченный оператор

$$\Phi : C_0^{1\infty} \oplus C_0^{\infty}(R) \to C^{1\infty}(\Omega), \Phi(g, q; \psi, \zeta) = u, (\Phi(0, 0; \psi, \zeta) \equiv 0), ||u||_{C^{1\infty}(\bar{\Omega})} \leq$$
$$\leq K(||g||_{C^{1\infty}(\bar{\Omega})} + ||q||_{C^{\infty}(R)})(K > 0), \quad (85)$$

гладко зависящий от ψ и ζ.

Если подставить функцию $u_0^*(s, \tau; \psi, \zeta)$ в краевую задачу (27)–(28) с учетом (55), то коэффициенты при ζ, ζ^2, ζ^3 в левой и правой частях равенств взаимно сократятся. С учетом этого положим

$$\dot{\tau} = \pi + \sigma(\psi, \zeta) + \zeta^4 \Delta_4 \quad (86)$$

и заменим в (74)

$$\rho_k^*(\psi, \zeta) \to \rho_k^*(\psi, \zeta) + \zeta^2 \tilde{\rho}_k, \theta_k^*(\psi, \zeta) \to \theta_k^*(\psi, \zeta) + \zeta^2 \tilde{\theta}_k, \ k = 1, 3, \dots, \quad (87)$$

где Δ_4 и $\tilde{\rho} = (\tilde{\rho}_1, \tilde{\rho}_2, \dots), \tilde{\theta} = (\tilde{\theta}_1, \tilde{\theta}_2, \dots)$ – подлежащие определению функции.

Представим (75) с учетом (86), (87) в виде

$$u^*(s, \tau, \tilde{\rho}, \tilde{\theta}, \psi, \zeta) = u_0^*(s, \tau, \tilde{\rho}, \tilde{\theta}, \psi, \zeta) + \zeta^4 w(s, \tau),$$

где ρ, θ и w – искомые функции, и подставим в краевую задачу (27)–(28) с учетом (55), (86). В результате, учитывая аналитичность функции $f(\cdot)$, получим следующую краевую задачу:

$$g(s, \tau, \tilde{\rho}, \tilde{\theta}, \Delta_4; \psi, \zeta) + \zeta^4 \Delta_4 w_\tau(s, \tau) + i(\pi + \sigma(\psi, \zeta)) w_\tau = w_s, \quad (88)$$

$$w(0, \tau) + w(-1, \tau) = q(\tau, \tilde{\rho}, \tilde{\theta}, w(-1, \tau); \psi, \zeta) \quad (89)$$

для определения $w(s, \tau) \in C^{1\infty}(\bar{\Omega})$. В (88)–(89) функции (функционалы) $g(\cdot) \in C^{1\infty}(\bar{\Omega}), q(\cdot) \in C^1(R)$. Условие разрешимости (84) краевой задачи (88)–(89) дают следующую последовательность функциональных уравнений:

$$\int_0^{2\pi} (q(\tau, \tilde{\rho}, \tilde{\theta}, w(-1, \tau); \psi, \zeta) + \int_{-1}^{0} e^{-i(2k+1)\pi s} (i(2k+1)\pi \zeta^4 \Delta_4 w(s, \tau) -$$
$$- g(s, \tau, \tilde{\rho}, \tilde{\theta}, \Delta_4; \psi, \zeta)) ds) e^{-i(2k+1)\pi \tau} d\tau \equiv a_{2k+1}(\tilde{\rho}, \tilde{\theta}, \Delta_4, w; \psi, \zeta) +$$
$$+ i b_{2k+1}(\tilde{\rho}, \tilde{\theta}, \Delta_4, w; \psi, \zeta) = 0 (a_{2k+1}(0, 0, 0, 0; \psi, \zeta) + i b_{2k+1}(0, 0, 0, 0; \psi, \zeta) \equiv 0),$$
$$k = 0, \pm 1, \dots,$$

которую запишем в следующей эквивалентной форме:

$$a_{2k+1}(\tilde\rho,\tilde\theta,\Delta_4,w;\psi,\zeta)=0,\tilde b_{2k-1}(\tilde\rho,\tilde\theta,\Delta_4,w;\psi,\zeta)\equiv$$
$$\equiv b_{2k+1}(\tilde\rho,\tilde\theta,\Delta_4,w;\psi,\zeta)-b_{2k-1}(\tilde\rho,\tilde\theta,\Delta_4,w;\psi,\zeta)-2b_1(\tilde\rho,\tilde\theta,\Delta_4,w;\psi,\zeta)=0,\ k=1,3,\dots,$$
$$b_1(\tilde\rho,\tilde\theta,\Delta_4,w;\psi,\zeta)=0\ (b_1(0,0,\Delta_4,0;\psi,\zeta)=\rho_1^*(\psi,\zeta)\Delta_4). \quad (90)$$

Отметим, что бесконечная матрица

$$\begin{pmatrix}\partial a_k/\partial\rho_j & \partial a_k/\partial\theta_j\\ \partial\tilde b_k/\partial\rho_j & \partial\tilde b_k/\partial\theta_j\end{pmatrix}(k,j=1,3,\dots), \quad (91)$$

вычисленная в точке $\tilde\rho=0,\tilde\theta=0,\Delta=0,w=0$ совпадает с матрицей $B(\psi;\zeta)$, вычисленной по системе уравнений (70), (70) в точке $\rho^*(\psi,\zeta),\theta^*(\psi,\zeta)$. Это следует из приведенного выше замечания и того обстоятельства, что матрица $B(\psi;\zeta)$ и (91) получены в результате "малого шевеления" величин $\rho_k^*(\psi;\zeta)$ и $\theta_k^*(\psi;\zeta)$.

Из уравнений (90) в силу сделанных предположений относительно свойств оператора (72), определяемого матрицей (91) по теореме о неявной функции для операторных уравнений [11], находим функционалы:

$$\tilde\rho_k=\tilde\rho_k(w;\psi,\zeta),\tilde\theta_k=\tilde\theta_k(w;\psi,\zeta),\Delta_4=\Delta_k(w;\psi,\zeta)$$
$$(\tilde\rho_k(0;\psi,\zeta)=\tilde\theta_k(0;\psi,\zeta)=\Delta_4(0;\psi,\zeta)=0). \quad (92)$$

Подставив теперь (92) в (88), (89) с учетом ограниченности оператора дифференцирования $w_\tau(s,\tau)$ в пространстве $C^{1\infty}(\bar\Omega)$ сведем задачу нахождения периодического решения краевой задачи (88),(89) на основании (85) к разрешимости операторного уравнения

$$w=\Phi(g(s,\tau,\tilde\rho(w;\psi,\zeta),\tilde\theta(w;\psi,\zeta),\Delta_4(w;\psi,\zeta);\psi,\zeta)+\zeta^4\Delta_4(w;\psi,\zeta)w_\tau(s,\tau);$$
$$q(s,\tau,\tilde\rho(w;\psi,\zeta),\tilde\theta(w;\psi,\zeta),w(-1,\tau;\psi,\zeta));\psi,\zeta) \quad (93)$$

операторного уравнения в пространстве $C^{1\infty}(\bar\Omega)$. Осталось применить к уравнению (93) теорему о неявной функции [11]. Правая часть (93) позволяет это сделать. В результате имеем решение $w^*(s,\tau;\psi,\zeta)\in C^{1\infty}(\bar\Omega)$. □

Решение (75) будет асимптотически орбитально устойчиво, если все собственные значения оператора (65) лежат в левой комплексной полуплоскости, и неустойчиво, если m корней с учетом кратностей принадлежат правой комплексной полуплоскости. В последнем случае периодическое решение (75) имеет $m-$ мерное неустойчивое инвариантное многообразие.

Для получения периодического решения непосредственно уравнения (4) достаточно в (75) положить $s=0$. В качестве первого приближения периодического решения следует взять приближение, определяемое решением $\rho^*(\psi),\theta^*(\psi)$ системы уравнений (62)–(63), которое будет иметь вид

$$y^*(\tau;\psi,\zeta)=\zeta\sum_{k=\pm1,\pm3}z_k^*(\tau,\psi)-\zeta^2b_2(0)/2p_{k_1k_2}\sum_{(k_1,k_2)\in\Omega_2}z_{k_1}^*(\tau,\psi)z_{k_2}^*(\tau,\psi)+O(\zeta^3), \quad (94)$$

$$\tau = \pi - \zeta \cos\psi + \zeta^2 \cos\psi + O(\zeta^3), \tag{95}$$

где функции $z_k^*(\tau, \psi)$ определены в (66).

Ниже на рис. 1 приведены графики периодических решений, построенных по формулам (94)–(95) (жирная линия) для случая $n = 3$, $\psi = 1.55$ и $\zeta = 0.1$. Построение решений системы нелинейных уравнений (62)–(63), определяющих (94)–(95), проводилось методом последовательного увеличения числа уравнений по k до тех пор, пока относительный прирост нормы $||\rho^*(\psi)||_{l_2^1}$ оставался более 0.01. Таких устойчивых решений удалось выявить семь. На рисунке также приведены графики точных решений (тонкая линия), полученные численным интегрированием уравнения (4) при этих же значениях параметров, методом Эйлера с постоянным выбором шага. Шаг интегрирования уменьшался до полной стабилизации результатов вычислений. В качестве начальных значений при интегрировании уравнения (4) выбирались функции, полученные согласно (94)–(95). Таким образом, в поведении решений уравнения (4) наблюдается мультистабильность.

Список литературы / References

[1] Glass L., Mackey M., *From Clocks to Chaos. The Rhythms of Life*, Princeton: Princeton University Press, 1988.

[2] Liz E., Trofimchuk E., Trofimchuk S., “Mackey–Glass type delay differential equations near the boundary of absolute stability”, *Journal of Mathematical Analysis and Applications*, **275**:2 (2002), 747–760.

[3] Su H., Ding X., Li W., “Numerical bifurcation control of Mackey–Glass system”, *Applied Mathematical Modelling*, **35**:27 (2011), 3460–3472.

[4] Berezansky L., Braverman E., “Mackey-glass equation with variable coefficients”, *Computers & Mathematics with Applications*, **51**:1 (2006), 1–16.

[5] Wu X.-M., Li J.-W., Zhou H.-Q., “A necessary and sufficient condition for the existence of positive periodic solutions of a model of hematopoiesis”, *Computers & Mathematics with Applications*, **54**:6 (2007), 840–849.

[6] Junges L., Gallas J., “Intricate routes to chaos in the Mackey–Glass delayed feedback system”, *Physics Letters A*, **376**:30–31 (2012), 2109–2116.

[7] Amil P., Cabeza C., Masoller C., Marti A., “Organization and identification of solutions in the time-delayed Mackey-Glass model”, *Chaos: An Interdisciplinary Journal of Nonlinear Science*, **25**:4 (2015), 043112.

[8] Кубышкин Е. П., Назаров А. Ю., “Анализ колебательных решений одного нелинейного сингулярно возмущенного дифференциально-разностного уравнения”, *Вестник Нижегородского университета им. Н.И. Лобачевского*, **5**:2 (2012), 118–125; [Kubyshkin E. P., Nazarov A. Yu., “Analysis of oscillatory solutions of a nonlinear singularly perturbed differential-difference equation”, *Vestnik Nizhegorodskogo universiteta im. N.I. Lobachevskogo*, **5**:2 (2012), 118–125, (in Russian).]

[9] Bellman R., Cooke K. L., *Differential Difference Equations*, New York: Academic Press, 1963.

[10] Шиманов С. Н., “К теории колебаний квазилинейных систем с запаздыванием”, *Прикл. математика и механика*, **23**:5 (1959), 836–844; [Shimanov S. N., “To theory of oscillations of quasi-linear systems with delay”, *J. Appl. Math. Mech.*, **23**:5 (1959), 836–844, (in Russian).]

[11] Krasnoselskii M. A., Vainikko G. M., Zabreyko R. P., Ruticki Ya. B., Stetsenko V. Ya, *Approximate solution of operator equations*, Springer, 1972.

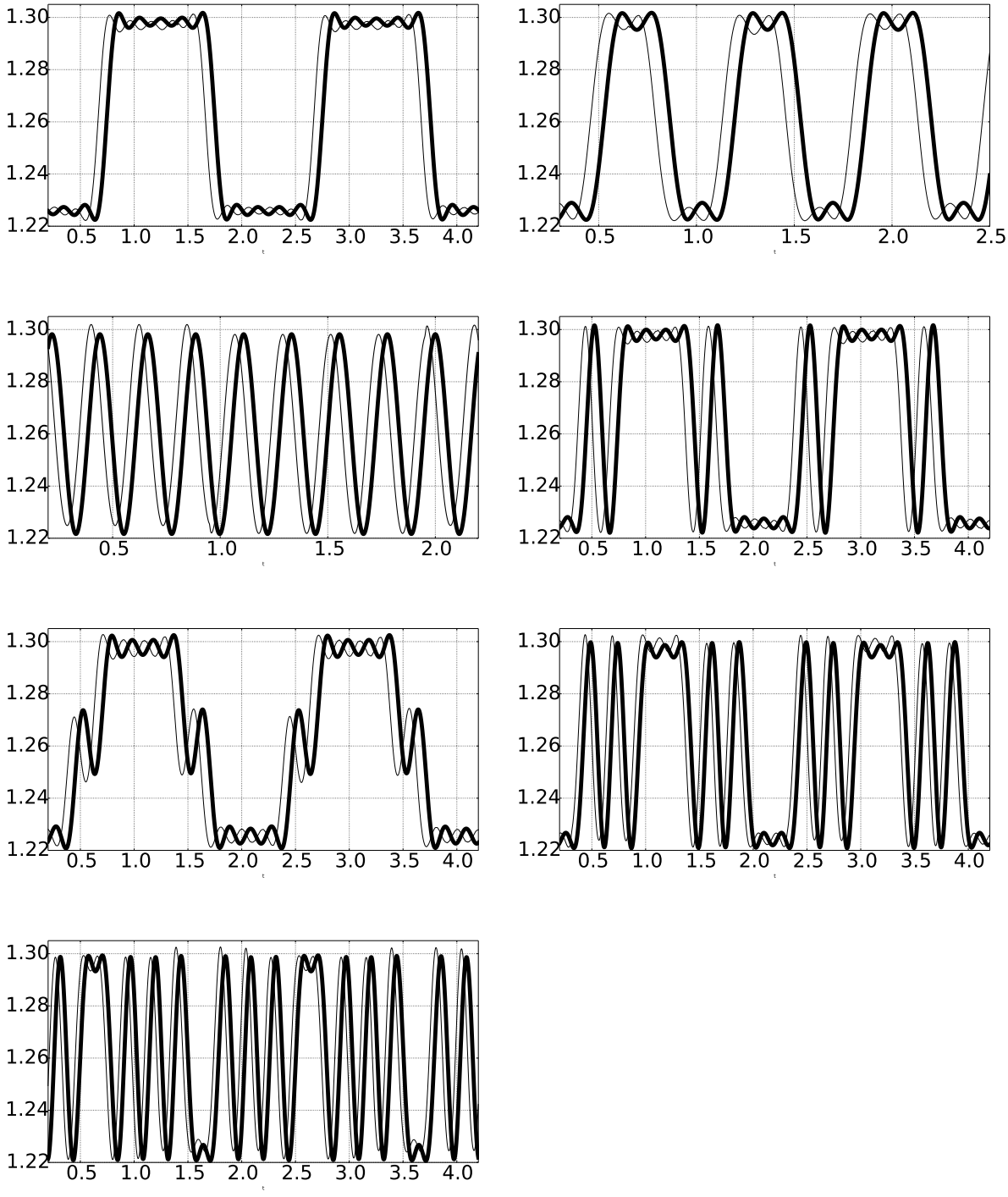


Рис. 1. Периодические решения уравнения Мэкки–Гласса для $\zeta = 0.1$, $\psi = 1.55$
Fig. 1. Periodic solutions of the equation Mackey–Glass for $\zeta = 0.1$, $\psi = 1.55$

Kubyshkin E. P., Moryakova A.R., "Bifurcation of Periodic Solutions of the Mackey–Glass Equation", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 784–803.

DOI: 10.18255/1818-1015-2016-6-784-803

Abstract. We study the bifurcation of the equilibrium states of periodic solutions for the Mackey–Glass equation. This equation is considered as a mathematical model of changes in the density of white blood cells. The equation written in dimensionless variables contains a small parameter at the derivative, which makes it singular. We applied the method of uniform normalization, which allows us to reduce the study of the solutions behavior in the neighborhood of the equilibrium state to the analysis of the countable system of ordinary differential equations. We poot out the equations in "fast" and "slow" variables from this system. Equilibrium states of the "slow" variables equations determine the periodic solutions. The analysis of equilibrium states allows us to study the bifurcation of periodic solutions depending on the parameters of the equation and their stability. The possibility of simultaneous bifurcation of a large number of stable periodic solutions is shown. This situation is called the multistability phenomenon.

Keywords: the Mackey–Glass equation, periodic solutions, multistability bifurcation

About the authors:
Evgenii P. Kubyshkin, orcid.org/0000-0003-1796-0190, Doctor, Professor,
P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl 150003, Russia, e-mail: kubysh.e@yandex.ru

Alena R. Moryakova, orcid.org/0000-0003-2529-6277, graduate student,
P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl 150003, Russia,
e-mail: alyona_moryakova@mail.ru

Acknowledgments:
This research was supported by project № 984 within the base part of state assignment on research in YarSU.

Моделирование и анализ информационных систем. Т. 23, №6 (2016), с. 804–825
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 804–825

DOI: 10.18255/1818-1015-2016-6-804-825

УДК 519.987

Реконфигурирование компонентно-ориентированных систем на базе графовых грамматик

Кушнаренко О. Б., Вебер Ж-Ф.

получена 15 октября 2016

Аннотация. Динамические реконфигурирования могут изменять архитектуру компонентно-ориентированных систем, не подвергаясь никакому системному простою. В этом контексте основной вклад данной статьи – доказательство результатов корректности реконфигурирования систем, используя графовые грамматики. В этой статье предложены новые охраняемые реконфигурирования на базе логики Хоара, которые построены на основе примитивных операций по реконфигурированию и включают последовательности реконфигурирований, альтернативные и повторяющиеся конструкции, сохраняя при этом непротиворечивость конфигураций. Практический вклад состоит в описании имплементации компонентно-ориентированной модели, используя программный инструмент *GROOVE* для преобразования графов. После обогащения модели интерпретированными конфигурациями и реконфигурированиями, совместимого с непротиворечивостью, отношение симуляции используется для доказательства корректности имплементации, выполненной под *GROOVE*. Эта имплементация иллюстрирована на примере многоуровневого облачно-ориентированного приложения.

Ключевые слова: компонентно-ориентированные системы, динамическое реконфигурирование, непротиворечивость, соответствие, реализация, *GROOVE*

Для цитирования: Кушнаренко О. Б., Вебер Ж-Ф., "Реконфигурирование компонентно-ориентированных систем на базе графовых грамматик", *Моделирование и анализ информационных систем*, **23**:6 (2016), 804–825.

Об авторах:
Кушнаренко Ольга Борисовна, orcid.org/0000-0003-1482-9015, профессор информатики, доктор,
Университет Бургундия Франш-Комтэ,
ул. Рут де Грей, 16, г. Безансон, 25000 Франция, e-mail: olga.kouchnarenko@univ-fcomte.fr

Вебер Жан-Франсуа, аспирант,
Университет Бургундия Франш-Комтэ,
ул. Рут де Грей, 16, г. Безансон, 25000 Франция, e-mail: jfweber@femto-st.fr

Благодарности:
Работа выполнена при финансовой поддержке французской целевой программы «Labex ACTION, ANR-11-LABEX-0001-01».

1. Введение

Динамические реконфигурирования, которые изменяют архитектуру самонастраивающихся [1] компонентно-ориентированных систем, не подвергаясь никакому системному времени простоя, должны происходить не только при подходящих обстоятельствах, но также должны сохранять непротиворечивость систем. В то время,

как первое может быть обеспечено политиками адаптации, второе непосредственно связано с определением реконфигурирований.

Для определения свойств поведения компонентно-ориентированных систем темпоральная логика линейного времени, основанная на работе Двайера над шаблонами и рамками их применения [2], была предложена в [3]. Эта логика является расширением JML-спецификаций временными шаблонами, введенными в [4]. Эта логика, названная FTPL[1], используется, чтобы инициировать политики адаптации в [5]. Кроме того, децентрализованное вычисление свойств FTPL по наборам компонентов было изучено в [6]. Что касается условий на непротиворечивость компонентно-ориентированных систем, определенных в [7], доказать их сохранение для систем под наблюдением было непросто, главным образом из-за отсутствия точной семантики для примитивных операций по реконфигурированию.

Для более сложных реконфигурирований, составленных из примитивных операций в виде последовательностей, повторений или выборов, предварительные условия реконфигурирований и их выходные условия должным быть выражены точным и кратким способом. Поэтому в данной работе используется понятие *самого слабого предварительного условия*, введенное в [8], чтобы выразить непримитивные *охраняемые реконфигурирования*.

Далее, используя программный инструмент *GROOVE* преобразования графов [9], предложена реализация для выполнения динамического реконфигурирования над моделями компонентно-ориентированных систем, использующая графовые грамматики. Это практическое внедрение позволяет нам не только имитировать процесс реконфигурирования системы, но также генерировать комбинации возможных реконфигурирований. Так как данная работа имеет целью формализацию реконфигурирования на базе графовых грамматик, основной результат этой работы состоит в доказательстве корректности интерпретируемых систем по отношению к нашей модели реконфигурирования, используя для их выполнения продукции над графами. Это также демонстрирует правильность нашей реализации.

Отметим, что эта работа мотивирована приложениями в многочисленных моделях и платформах, поддерживающих разработку компонентов вместе с их мониторами и контроллерами, такими как, например, Fractal [10], CSP‖B [11], Frascati [12] и т.д.

Статья организована следующим образом. Пример окружения для приема облачно-ориентированных многоуровневых приложений, рассматриваемый в качестве компонентно-ориентированной системы, представлен в разделе 2. Необходимая информация о нашей компонентно-ориентированной модели реконфигурирования, а также элементы ее операционной семантики даны в разделе 3. Реализация модели, использующая программный инструмент *GROOVE* для выражения реконфигурирований посредством графовых грамматик, представлена на базе примера в разделе 4. Наконец, результаты корректности даны в разделе 5. Библиографические заметки и заключение содержатся в разделе 6.

[1]FTPL обозначает TPL (Temporal Pattern Language) с приставкой 'F', чтобы обозначить ее связь к компонентам Fractal и к условиям первого порядка для их целостности.

2. Пример системы с компонентами: многоуровневое облачно-ориентированное приложение

Интернет-провайдеры и телекоммуникационные операторы склонны все больше и больше определять себя поставщиками "облачной" инфраструктуры. В этом контексте автоматизация программного обеспечения и виртуальной установки и конфигурации оборудования является важной задачей. Однако этого недостаточно, чтобы приложение было облачно-применяемым; оно должно быть масштабируемым, и механизмы для этого должны быть интегрированы в систему управления облаком.

Рассмотрим типичное веб-приложение с компонентами трёхуровневых приложений, использующее клиентскую часть веб-сервера, сервер приложений промежуточного программного обеспечения и серверное приложение для работы с данными. На рис. 1 изображена управляемая виртуальная машина *VM*, принимающая такое приложение.

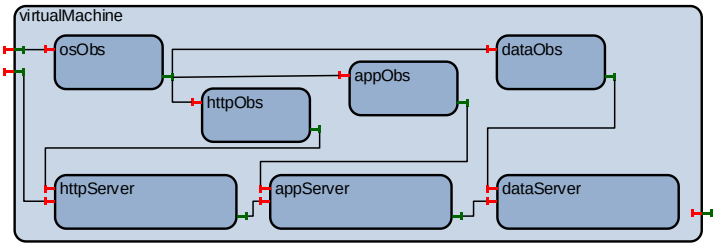


Рис. 1: Управляемая виртуальная машина с компонентами трёхуровневых приложений
Fig. 1. Managed virtual machine with three-tier application compoments

VM представлена составным компонентом *virtualMachine* с подкомпонентами *httpServer*, *appServer* и *dataServer* служб приложения. Подкомпоненты служб имеют два предоставляемых интерфейса: один для предоставления услуги, а другой для её мониторинга.

Кроме того, VM на рис. 1 также содержит четыре *наблюдателя*, которые являются подкомпонентами для мониторинга служб. Подкомпонент *osObs* используется для мониторинга операционной системы VM. Также он связан с подкомпонентами *httpObs*, *appObs* и *dataObs*, используемыми соответственно для мониторинга служб *httpServer*, *appServer* и *dataServer* подкомпонентов. Наконец, отметим, что у самого составного компонента VM есть два предоставляемых интерфейса: один для предоставления сервисов и второй для мониторинга.

На рисунке 2 изображена *Облачная среда cloudEnv*, содержащая VM для использования в целях разработки (*vmDev*). В ней имеются трёхуровневые приложения без мониторинга; назовем такую VM *неконтролируемой*. Три другие VM – *управляемые*, и каждая содержит часть приложения. Читатель может отметить, что каждая из управляемых VM содержит только наблюдателей для мониторинга операционной системы и типа предоставляемой услуги. У облачной среды есть три предоставляемых интерфейса: два, чтобы предоставить услугу, будь то версия разработки или нет, а третий для мониторинга, подключенный к подкомпоненту *monitorObs*, связанному со всеми контролируемыми интерфейсами управляемых VM.

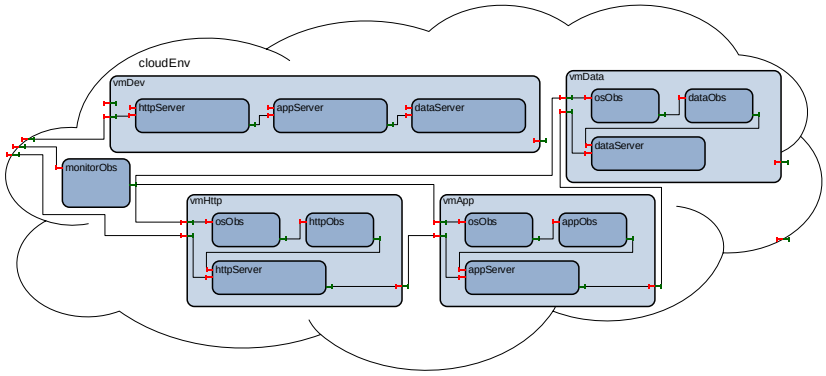


Рис. 2: Пример облачной среды
Fig. 2. Cloud environment example

Поставщик "облачной" инфраструктуры должен быть в состоянии обеспечить по требованию одну или несколько VM, сконфигурированных с правильными компонентами службы и соответствующим мониторингом. В этом контексте мы изучаем подготовку к эксплуатации VM, показанной на рис. 1. В зависимости от предоставляемых услуг и состояния мониторинга (для управляемой или неуправляемой VM) должны быть добавлены необходимые компоненты. Во время жизненного цикла VM могут произойти некоторые изменения конфигурации; мы рассматриваем их как реконфигурирование компонентно-ориентированной системы.

3. Компонентно-ориентированные системы: семантическая модель

3.1. Конфигурации и реконфигурации

Компонентные модели могут быть очень разнородными. В большинстве из них компоненты программного обеспечения рассматриваются как черные ящики с полностью специфицированными интерфейсами (или как серые ящики, если некоторые их внутренние особенности видимы). Определения компонентов и их интерфейсы используются для описания взаимодействий и сложного поведения. В этом разделе мы возвращаемся к модели реконфигурирования архитектуры компонентно-ориентированных систем, введенной в [13, 7]. В общем случае конфигурация системы – это определение элементов, из которых состоит система, в то время как реконфигурирование может быть рассмотрено как переход от одной конфигурации к другой.

Следуя за [13], конфигурация определена как множество архитектурных элементов (компоненты, предоставляемые и требуемые интерфейсы, параметры) и отношений, чтобы структурировать и связать их.

Определение 1 (Конфигурация). *Конфигурацией c называется набор* $\langle Elem, Rel \rangle$, *где*

- $Elem = Components \uplus Interfaces \uplus Parameters \uplus Types$ – *множество архитектурных элементов, таких что*

– $Components$ – *непустое множество компонентов;*

– $Interfaces = RequiredInts \uplus ProvidedInts$ – *конечное множество предоставляемых и требуемых интерфейсов;*

– $Parameters$ – *конечное множество параметров компонентов;*

– $Types = ITypes \uplus PTypes$ – *конечное множество типов интерфейсов и типов данных для параметров;*

- $Rel = \left\{ \begin{array}{l} Container \uplus ContainerType \uplus Contingency \\ \uplus\, Parent \uplus Depth \uplus Binding \uplus Delegate \uplus State \uplus Value \end{array} \right.$

– множество архитектурных отношений, которые связывают элементы, такие что

– $Container : Interfaces \uplus Parameters \to Components$ – всюду определенная функция, *дающая компоненту с рассматриваемым интерфейсом или с рассматриваемым параметром;*

– $ContainerType : Interfaces \uplus Parameters \to Types$ – всюду определенная функция, *дающая тип каждого интерфейса или параметра;*

– $Contingency : RequiredInts \to \{mandatory, optional\}$ – всюду определенная функция, *указывающая, каким (mandatory или optional) является каждый требуемый интерфейс;*

– $Parent \subseteq Components \times Components$ – отношение, *связывающее подкомпонент с соответствующим составным компонентом* [2]*;*

– $Depth : Components \to \mathbb{N}$ – всюду определенная функция, *определяющая глубину компонентов;*

– $Binding : ProvidedInts \to RequiredInts$ – частичная функция, *связывающая вместе предоставляемый и требуемый интерфейсы;*

– $Delegate : Interfaces \to Interfaces$ – частичная функция *для выражения делегируемых связей;*

– $State : Components \to \{started, stopped\}$ – всюду определенная функция, *дающая статус инициированных компонентов;*

– $Value : Parameters \to \{t | t \in PType\}$ – всюду определенная функция, *дающая текущую величину каждого параметра.*

Определим также множество CP свойств конфигурации, которые являются ограничениями на архитектурные элементы и отношения между ними. Эти свойства определяются формулами логики первого порядка [14]. Интерпретация функций, отношений и предикатов над $Elem$ дается согласно базовым определениям в [14] и определению 1 (см. [7] для дополнительной информации).

Пусть $\mathcal{C} = \{c, c_1, c_2, \ldots\}$ – множество конфигураций. *Интерпретация* $l : \mathcal{C} \to CP$ дает самую широкую конъюнкцию $cp \in CP$, которая является истинной на $c \in \mathcal{C}$.

[2] Для каждого $(p, q) \in Parent$ p называется подкомпонентом q. Разделяемые компоненты (подкомпоненты с множественным вложением в составные компонеты) могут иметь более одного родителя.

Мы говорим, что конфигурация $c = \langle Elem, Rel \rangle$ удовлетворяет формуле $cp \in CP$, когда $l(c) \Rightarrow cp$; в этом случае cp действительна на c, иначе c не удовлетворяет cp.

Среди свойств конфигураций архитектурные *условия на непротиворечивость* CC в таблице 1 выражают требования на составление компонентов. Эти требования являются общими для всех компонентно-ориентированных архитектур [7]. Неформально,

- компонент *поставляет*, по крайней мере, один предоставляемый интерфейс (**CC.1**);
- сложные компоненты не имеют параметров (**CC.2**);
- подкомпонент не должен включать своего собственного родителя (**CC.3**);
- два связанных интерфейса должны быть одного и того же типа (**CC.4**), и содержащие их компоненты являются подкомпонентами одного и того же составного компонента (**CC.5**);
- связывая два интерфейса, необходимо гарантировать, что они не были еще вовлечены в делегированную связь (**CC.6**); точно так же, устанавливая делегированную связь между двумя интерфейсами, спецификатор должен гарантировать, что они не были еще связаны между собой (**CC.7**);
- предоставляемый (соответственно требуемый) интерфейс подкомпонента делегируется самое большее одному предоставляемому (соотв. требуемому) интерфейсу родительского компонента (**CC.8**), (**CC.9**) и (**CC.11**); интерфейсы, вовлеченные в делегацию, должны быть одного и того же типа (**CC.10**);
- компонент находится в состоянии *started*, только если его обязательные требуемые интерфейсы связаны или делегированы (**CC.12**).

Определение 2 (Непротиворечивая конфигурация)**.** *Пусть* $c = \langle Elem, Rel \rangle$ *– конфигурация, и* CC *– условия на непротиворечивость. Конфигурация* c *называется* непротиворечивой*, что обозначено* $\texttt{consistent}(c)$*, если* $l(c) \Rightarrow CC$*. Определим* $\texttt{consistent}(\mathcal{C})$*, когда* $\forall c \in \mathcal{C}.\texttt{consistent}(c)$*.*

Дополнительная информация о непротиворечивых конфигурациях может быть найдена в [7].

3.2. Операционная семантика

Реконфигурирования заставляют компонентно-ориентированную архитектуру изменяться динамично. Они состоят из примитивных операций, таких как инициирование/разрушение (*new*/*destroy*) компонентов; добавление/удаление (*add*/*remove*) компонентов; связывание/развязывание (*bind*/*inbind*) интерфейсов; старт/остановка (*start*/*stop*) компонентов; присваивание значений параметрам компонентов (*update*). Эти примитивные операции удовлетворяют pre/post-предикаты. Например, прежде чем добавить $comp_1$ подкомпонента к составному $comp_2$, нужно проверить,

Таблица 1: Условия на непротиворечивость

Table 1: Consistency constraints

$$\forall c.(c \in Components \Rightarrow (\exists ip.(ip \in ProvidedInts \wedge Container(ip) = c))) \quad \textbf{(CC.1)}$$

$$\forall c, c' \in Components.(c \neq c' \wedge (c, c') \in Parent \Rightarrow \forall\, p.(p \in Parameters \Rightarrow Container(p) \neq c')) \quad \textbf{(CC.2)}$$

$$\forall c, c' \in Components.((c, c') \in Parent \Rightarrow Depth(c') < Depth(c)) \quad \textbf{(CC.3)}$$

$$\begin{matrix}\forall ip \in ProvidedInts, \\ \forall ir \in RequiredInts\end{matrix} \cdot \left(Binding(ip) = ir \Rightarrow \begin{matrix} ContainerType(ip) = ContainerType(ir) \\ \wedge Container(ip) \neq Container(ir) \end{matrix} \right) \quad \textbf{(CC.4)}$$

$$\begin{matrix}\forall ip \in ProvidedInts, \\ \forall ir \in RequiredInts\end{matrix} \cdot \left(Binding(ip) = ir \Rightarrow \exists c \in Components. \left(\begin{matrix} (Container(ip), c) \in Parent \\ \wedge (Container(ir), c) \in Parent \end{matrix} \right) \right) \quad \textbf{(CC.5)}$$

$$\begin{matrix}\forall ip \in ProvidedInts, \\ \forall ir \in RequiredInts\end{matrix} \cdot \left(Binding(ip) = ir \Rightarrow \begin{matrix} ip \notin dom(Delegate) \\ \wedge ir \notin dom(Delegate) \end{matrix} \right) \quad \textbf{(CC.6)}$$

$$\forall i, i' \in Interfaces. \left(Delegate(i) = i' \Rightarrow \begin{matrix} i \notin dom(Binding) \\ \wedge i \notin codom(Binding)) \end{matrix} \right) \quad \textbf{(CC.7)}$$

$$\forall i, i' \in Interfaces.(Delegate(i) = i' \wedge i \in ProvidedInts \Rightarrow i' \in ProvidedInts) \quad \textbf{(CC.8)}$$

$$\forall i, i' \in Interfaces.(Delegate(i) = i' \wedge i \in RequiredInts \Rightarrow i' \in RequiredInts) \quad \textbf{(CC.9)}$$

$$\forall i, i' \in Interfaces. \left(Delegate(i) = i' \Rightarrow \begin{matrix} ContainerType(i) = ContainerType(i') \\ \wedge\ (Container(i), Container(i')) \in Parent \end{matrix} \right) \quad \textbf{(CC.10)}$$

$$\forall i, i', i" \in Interfaces. \left(\begin{matrix} (Delegate(i) = i' \wedge Delegate(i) = i" \Rightarrow i' = i") \\ \wedge (Delegate(i) = i" \wedge Delegate(i') = i" \Rightarrow i = i') \end{matrix} \right) \quad \textbf{(CC.11)}$$

$$\forall ir \in RequiredInts. \left(\begin{matrix} State(Container(ir)) = started \\ \wedge Contingency(ir) = mandatory \end{matrix} \Rightarrow \exists i \in Interfaces. \left(\begin{matrix} Binding(i) = ir \\ \vee Delegate(i) = ir \\ \vee Delegate(ir) = i \end{matrix} \right) \right) \quad \textbf{(CC.12)}$$

Таблица 2: Предварительные условия примитивной операции *add*

Table 2: Preconditions of the *add* primitive reconfiguration operation

$$comp_1, comp_2 \in Components \quad (2) \qquad (comp_2, comp_1) \notin Parent^+ \quad (4)$$

$$comp_1 \neq comp_2 \quad (3) \qquad \forall p \in Parameters, (p, comp_2) \notin Container \quad (5)$$

как указано в таблице 2, следующее: *a*) $comp_1$ и $comp_2$ существуют (2), и они различны (3), *b*) $comp_2$ не является потомком $comp_1$ (4), и *c*) $comp_2$ не имеет параметров (5). Когда эти предварительные условия удовлетворены, выходное условие состоит в добавлении $(comp_1, comp_2)$ к отношению $Parent$, формально $R_{add} = Parent \cup \{(comp_1, comp_2)\}$ (1).

Следуя за основанной на предикатах семантикой базовых конструкций языков программирования [15], рассмотрим операцию реконфигурирования *ope* и две конфигурации c и c', такие что переход между c и c' выполняется с использованием *ope*. Пусть R – условия на конфигурации системы под наблюдением, тогда $wp(ope, R)$ обозначает, как в [8], *самое слабое предварительное условие* на конфигурацию c, такое что активация *ope* может произойти, и тогда она гарантированно ведет к c', удовлетворяющей выходному условию R. Формально, в нашем случае, если $l(c) \Rightarrow wp(ope, R)$ и $c \xrightarrow{ope} c'$, тогда $l(c') \Rightarrow R$. Поэтому для примитивной операции $add(,)$ самым слабым предварительным условием $wp(add, R_{add})$ является конъюнкция предварительных условий (2) – (5).

На базе [8] и используя такие же обозначения, таблица 3 приводит грамматику для *охраняемых реконфигурирований*, имеющую аксиомой <guarded configuration>. Пусть <*ope*> представляет примитивную операцию по реконфигурированию, так-

Таблица 3: Грамматика охраняемых реконфигурирований

Table 3: Guarded reconfigurations grammar

<guarded reconfiguration>	::=	<guard> → <guarded list>
<guard>	::=	<boolean expression>
<guarded list>	::=	<statement>{; <statement> }
<guarded reconfiguration set>	::=	<guarded reconfiguration>{[] <guarded reconfiguration> }
<alternative construct>	::=	**if** <guarded reconfiguration set> **fi**
<repetitive construct>	::=	**do** <guarded reconfiguration set> **od**
<statement>	::=	<alternative construct> | <repetitive construct> | <*ope*>

же называемую *примитивным утверждением*. Расширим множество примитивных операций по реконфигурированию операцией *skip*, которая не вызывает изменения на данной конфигурации. Следовательно, для любого выходного условия R, верно $wp(skip, R) = R$. Дополнительно, как в [8], семантика оператора ";" определена как $wp(S_1; S_2, R) = wp(S_1, wp(S_2, R))$, где S_1 и S_2 – утверждения.

Если множество охраняемых реконфигурирований содержит более одного элемента, они разделены [] [3]. Чтобы представить семантику альтернативной конструкции, пусть IF обозначает **if** $B_1 \to S_1 [] \ldots [] B_n \to S_n$ **fi**, и BB обозначает $(\exists i : 1 \leq i \leq n : B_i)$, тогда $wp(IF, R) = BB \wedge (\forall i : 1 \leq i \leq n : B_i \Rightarrow wp(S_i, R))$. Для периодически повторяющейся конструкции DO обозначает **do** $B_1 \to S_1 [] \ldots [] B_n \to S_n$ **do**. Пусть $H_0(R) = R \wedge \neg BB$ и для $k > 0$, $H_k(R) = wp(IF, H_{k-1}(R)) \vee H_0(R)$, тогда $wp(DO, R) = \exists k : k \geq 0 : H_k(R)$. Интуитивно, $H_k(R)$ является самым слабым предварительным условием, обеспечивающим завершение после не больше чем k выборов из охраняемого списка, оставляя систему в конфигурации, удовлетворяющей R. Пусть $\mathcal{R}_{run} = \mathcal{R} \cup \{run\}$ – множество операций, где $\mathcal{R}$ – конечное множество охраняемых реконфигурирований, инициированное относительно рассматриваемой системы, и run – название универсального действия, представляющего все текущие операции[4] компонентно-ориентированной системы.

Определение 3 (Модель реконфигурирования)**.** *Операционная семантика компонентно-ориентированной системы определяется маркированной системой с переходами* $S = \langle \mathcal{C}, \mathcal{C}^0, \mathcal{R}_{run}, \to, l \rangle$, *где* $\mathcal{C} = \{c, c_1, c_2, \ldots\}$ *– множество конфигураций,* $\mathcal{C}^0 \subseteq \mathcal{C}$ *– множество начальных конфигураций,* $\to \subseteq \mathcal{C} \times \mathcal{R}_{run} \times \mathcal{C}$ *– отношение реконфигурирования, подчиняющееся предикатам* $wp()$, *и* $l : \mathcal{C} \to CP$ *– всюду определенная функция интерпретации конфигураций.*

Обозначим $c \stackrel{ope}{\to} c'$, когда $(c, ope, c') \in \to$. Для модели $S = \langle \mathcal{C}, \mathcal{C}^0, \mathcal{R}_{run}, \to, l \rangle$, определим путь σ в S как последовательность конфигураций $c_0, c_1, c_2, \ldots$, таких что $\forall i \geq 0$. $\exists\, ope_i \in \mathcal{R}_{run}.(c_i \stackrel{ope_i}{\to} c_{i+1})$. Пусть Σ обозначают множество путей, а Σ^f $(\subseteq \Sigma)$ множество конечных путей. Определим выполнение как путь σ в Σ, такой что $\sigma(0) \in \mathcal{C}^0$. Пусть $\sigma(i)$ обозначает i-ю конфигурацию σ. Обозначение σ_i используется для суффикса $\sigma(i), \sigma(i+1), \ldots$, и σ_i^j обозначает отрезок пути $\sigma(i), \sigma(i+1), \ldots, \sigma(j-1), \sigma(j)$. Конфигурация c' достижима из c, когда существует путь $\sigma = c_0, c_1, \ldots, c_n$ в Σ^f, такой что $c = c_0$ и $c' = c_n$ с $n \geq 0$. Пусть c – конфигурация, тогда $reach(c)$ обозначает

[3]Как и в [8], порядок, в котором появляются охраняемые реконфигурирования, семантически не важен.

[4]Нормальное функционирование различных компонентов также изменяет архитектуру, например, изменяя значения параметров или останавливая компоненты.

множество всех конфигураций, достижимых из c. Это понятие может быть поднято с конфигураций на множество конфигураций: $reach(\mathcal{C}) = \{reach(c) \mid c \in \mathcal{C}\}$.

Утверждение 1 (Сохранение непротиворечивости)**.** *Пусть* $\mathcal{C}^0 \subseteq \mathcal{C}$*. Тогда* $\texttt{consistent}(\mathcal{C}^0)$ *влечет* $\texttt{consistent}(reach(\mathcal{C}^0))$*.*

Набросок. Установим сначала, что каждая примитивная операция *ope* сохраняет непротиворечивость конфигурации. Это означает, что для выходного условия R операции *ope* верно $CC \wedge wp(ope, R) = wp(ope, CC \wedge R)$. Докажем этот результат для *add(,)*, доказательство для других примитивных операций является схожим. Пусть $\texttt{consistent}\,(c)$, и предварительные условия для $add(,)$ выполнены на c. Тогда переход $c \xrightarrow{add} c'$ должен вести к непротиворечивой конфигурации c' ($\texttt{consistent}\,(c')$), такой что выходные условия для $add(,)$ также удовлетворяют условиям на непротиворечивость в таблице 1. Формально, $(l(c) \Rightarrow CC \wedge wp(add, R_{add})) \wedge (c \xrightarrow{add}, c') \Rightarrow (l(c') \Rightarrow CC \wedge R_{add})$. Действительно, так как отношение *Parent* из выходного условия (1) не вовлечено в (**CC.1**), (**CC.4**) – (**CC.9**), (**CC.11**), и (**CC.12**), эти условия также выполнены для c'. Для остающихся условий мы имеем:

(**CC.2**): Предварительное условие (5) из таблицы 2 гарантирует, что у родительского компонента $comp_2$ нет параметров, (**CC.2**) выполнено на c' с $(comp_1, comp_2)$, добавленной к *Parent* (cf. (1));

(**CC.3**): Предварительное условие (4) в таблице 2 означает, что $comp_2$ не может быть потомком $comp_1$, таким образом предотвращая цикл в отношении *Parent* для c', когда $comp_2$ становится родителем $comp_1$;

(**CC.10**): Существуют два случая: либо делегированная связь между интерфейсами $comp_1$ и $comp_2$ на c до выполнения операции $add(,)$ уже существовала, либо нет. В последнем случае ограничение (**CC.10**) тривиально выполнено на c'. В первом случае, поскольку $\texttt{consistent}\,(c)$, отношение *Parent* уже содержало $(comp_1, comp_2)$ с интерфейсами правильного типа, и применение $add(,)$ не изменяет типы и отношение, поэтому условие выполнено на c'.

Пусть $c \in reache(\mathcal{C}^0)$. По определению существует $c_0 \in \mathcal{C}^0$ и последовательность операций из $\mathcal{R}_{run}$, чтобы в конечном счете достигнуть c. По определению также существует последовательность примитивных операций $ope_0, ope_1, \ldots, ope_{n-1}$ и ряд промежуточных конфигураций $\mathcal{C}' = \{c_1, c_2, \ldots, c_{n-1}\}$[5], таких что $c_0 \xrightarrow{ope_0} c_1$, $c_1 \xrightarrow{ope_1} c_2$, $\ldots$, $c_{n-1} \xrightarrow{ope_{n-1}} c$, и для $0 \leq i \leq n-1$, c_i (соотв. c_{i+1}), выполняют предварительные условия (соотв. выходные условия) ope_i (с c_n, обозначающей c). Действительно, если бы эта последовательность примитивных операций или $\mathcal{C}'$ не существовали, c не была бы достижима ни из какой конфигурации в $\mathcal{C}^0$.

Теперь докажем, что охраняемое реконфигурирование, имеющее последовательность примитивных утверждений в своем охраняемом списке, сохраняет непротиворечивость. Пусть gl_n – охраняемый список, составленный из $n \geq 0$ примитивных

[5]Заметим, что $\mathcal{C}'$ не обязательно является подмножеством $\mathcal{C}$. Например, если каждая операция в $\mathcal{R}$ является последовательностью двух примитивных операций, тогда промежуточные конфигурации не принадлежат $\mathcal{C}$, и $\mathcal{C}' \not\subset \mathcal{C}$.

операций, т.е., $gl_n = ope_0; ope_1; \ldots; ope_n$, с R_i и R_{i+1}, являющихся соответственно предварительными и выходными условиями ope_i. Обозначим $CC_i = CC \wedge R_i$. Докажем по индукции на n, что $CC_0 = wp(gl_n, CC_{n+1})$. Для $n = 0$ у нас есть $gl_n = ope_0$ и $CC_0 = wp(gl_0, CC_1)$. Рассмотрим теперь $gl_{n+1} = gl_n; ope_{n+1}$; мы имеем $wp(gl_{n+1}, CC_{n+2}) = wp(gl_n, wp(ope_{n+1}, CC_{n+2}))$. Так как $CC_0 = wp(gl_n, CC_{n+1})$ и $CC_{n+1} = wp(ope_{n+1}, CC_{n+2})$, мы имеем, по определению [8], $CC_0 = wp(gl_n, CC_{n+1}) = wp(gl_n, wp(ope_{n+1}, CC_{n+2}))$.

Используя гипотезу на предварительные и выходные условия утверждений, в приложении 6.2. доказано, что охраняемые реконфигурирования с непримитивным утверждением, использующим только примитивные утверждения ($G \to$ **fi** grs **fi** или $G \to$ **do** grs **od**, где grs обозначает $B_0 \to ope_0 [\!] B_1 \to ope_1 [\!] \ldots [\!] B_n \to ope_n$), также предохраняют непротиворечивость.

Таким образом, с тем же рассуждением, рассматривая непримитивные заявления вместо примитивных и используя только гипотезу на предварительные и выходные условия утверждений, мы можем доказать, что непротиворечивость сохранена *a*) для охраняемых реконфигурирований, имеющих охраняемый список, составленный из последовательности непримитивных утверждений ($G \to S_0; S_1; \ldots; S_n$) и *b*) для охраняемых реконфигурирований, имеющих утверждение в качестве охраняемого списка ($G \to$ **fi** grs **fi** или $G \to$ **do** grs **od**, где grs обозначает $B_0 \to S_0 [\!] B_1 \to S_1 [\!] \ldots [\!] B_n \to S_n$). □

4. Реализация на базе *GROOVE*

В этом разделе описывается, как наша модель реализована в *GROOVE*, используя программный инструмент преобразования графов [9]. Это внедрение затем используется для экспериментирования на примере многоуровневого облачно-ориентированного приложения.

4.1. О программном продукте *GROOVE*

GROOVE использует простые графы для моделирования структуры объектно-ориентированных систем во время их проектирования, компиляции и выполнения. Графы имеют вершины и дуги, которые могут быть маркированы. Преобразования графов позволяют выразить изменения моделей или для операционной семантики систем. В нашей работе *GROOVE* используется в типизированном режиме, чтобы гарантировать правильное типизирование графов. Этот режим предоставляет общие типы для переменных и правила над графами, которые позволяют управлять присвоенными приоритетами таким образом, что правило данного приоритета применяется, только если никакое правило с более высоким приоритетом не может быть применено к текущему графу.

Графы преобразовываются по правилам, имеющим *a*) шаблоны, которые должны присутствовать (соотв. отсутствовать) для применения правила, *b*) элементы (вершины и дуги), для добавления или удаления в графе, и *c*) пары вершин, которые будут слиты. Кодировка, использующая цвета и формы, позволяет дать более наглядное представление этих правил. Например, примитивная операция *add* моделируется графовым правилом, представленным на рис. 3. На этом рисунке показаны

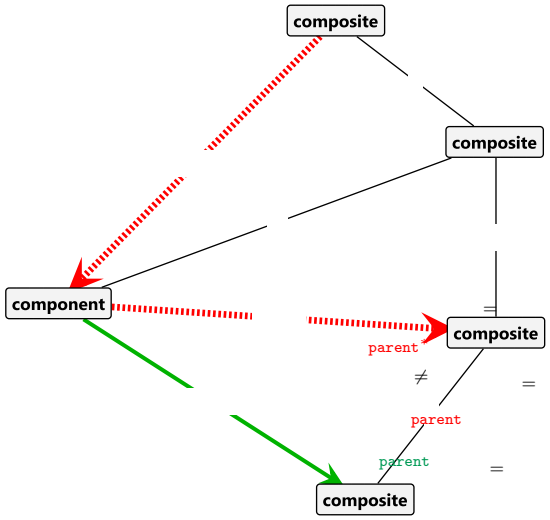
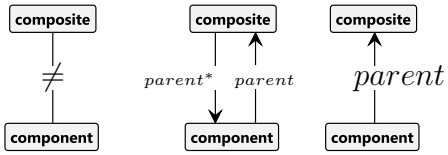


Рис. 3: Примитивное реконфигурирование *add* в *GROOVE*
Fig. 3. *add* primitive operation in *GROOVE*



(a) LHS (b) NAC (c) RHS

Рис. 4: Эквивалент правила *GROOVE* на рис. 3, использующий графы LHS, NAC и RHS
Fig. 4. Equivalent of *GROOVE* rule in Fig. 3 using LHS, NAC, and RHS graphs

a) компонент и составной компонент; дуги с маркировкой "=" гарантируют, что вершины для составных компонентов имеют одинаковый тип *composite*, в то время как дуги с маркировкой "≠" гарантируют, что вершины для обычных компонентов имеют тип *component*, отличающийся от типа для составных компонентов; *b*) запрещающие красные штриховые дуги с маркировкой "*parent**" или "*parent*" гарантируют отсутствие (транзитивного) отношения *parent* между вершинами с маркировкой "composite" и "component". Если вышеупомянутые условия удовлетворены, зеленая дуга с маркировкой "*parent*" создается между вершинами "component" и "composite".

Такое правило преобразований графа может быть представлено с помощью грамматического правила, имеющего *a*) LHS подграф (левая часть правила) для представления предварительных условий правила, *b*) NAC подграф (отрицательное условие для применения), определяющий то, что не должно произойти в соответствии с правилом, и *c*) RHS подграф (правая часть правила) для представления выходных условий. Подграфы LHS, NAC и RHS, выражающие правило *GROOVE* на рис. 3, изображены на рис. 4.

Входными данными для нашего внедрения является граф компонентно-ориентированной системы, представленной с использованием модели из раздела 3. Такой граф показывает конфигурацию, как описано в определении 1, где элементы и отношения представляются соответственно вершинами и дугами.

4.2. Возращаясь к примеру

Возвратимся к VM, представленной на рис. 1. Она смоделирована составным компонентом $virtualMachine$, который может содержать подкомпоненты, представляющие службы приложения $httpServer$, $appServer$ или $dataServer$. Эта VM может также содержать *наблюдателей*, которые являются подкомпонентами для мониторинга служб. Подкомпонент $osObs$ используется для мониторинга операционной системы VM, и он может быть связан с подкомпонентами $httpObs$, $appObs$ или $dataObs$, используемыми соответственно для мониторинга служб подкомпонентов $httpServer$, $appServer$ и $dataServer$.

Таблица 4: Принцип генерации кода инсталляции

Table 4: Install code generation principle

Feature	**data**	**app**	**http**	**managed**
Bit #	3	2	1	0
ic = 0	0	0	0	0
ic = 5	0	$2^2 = 4$	0	$2^0 = 1$
ic = 10	$2^3 = 8$	0	$2^1 = 2$	0
ic = 11	$2^3 = 8$	0	$2^1 = 2$	$2^0 = 1$

Каждая VM имеет свои особенности (опции), определенные бинарным *кодом инсталляции* ic, каждый бит которого действует как флаг для включения или отключения определенной опции. Это представлено в таблице 4, где первая линия показывает опции, а вторая показывает номер соответствующего бита. Последующие

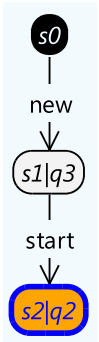


Рис. 5: Пустая ОС
Fig. 5. Bare OS ($ic = 0$)

линии описывают генерацию кода инсталляции для сервера с пустой операционной системой ($ic = 0$), с управляемым сервером приложения ($ic = 5$) и с управляемым (соотв. неуправляемым) LAMP (Linux+Apache+MySQL+PHP) сервером, имеющим код инсталляции 10 (соотв. 11).

Наша программная реализация создает модель компонентно-ориентированной системы, представляющей VM, специфицированную данным кодом инсталляции. Рисунок 5 показывает систему с переходами над графами, полученную с помощью *GROOVE* во время создания VM с пустой ОС ($ic = 0$). Первое состояние (`s0`) представляет пустой граф; `s1` обозначает граф, представляющий составной компонент

VM в остановленном состоянии; s2 представляет граф с тем же самым компонентом VM после запуска. Переходы маркированы выполняемыми примитивными операциями по реконфигурированию.

Точно так же для компонентно-ориентированной системы, представляющей сервер управляемого приложения ($ic = 5$), система с переходами над графами показана на рис. 6. В дополнение к примитивным операциям по реконфигурированию, которые используются для маркировки переходов, вводится этикетка "chk_present_appServerPC" для подтверждения, присутствует ли подкомпонент сервера приложения или нет. Таким образом, используя язык управления *GROOVE*, функция *manage()* добавляет и формирует подкомпоненты с адекватным мониторингом.

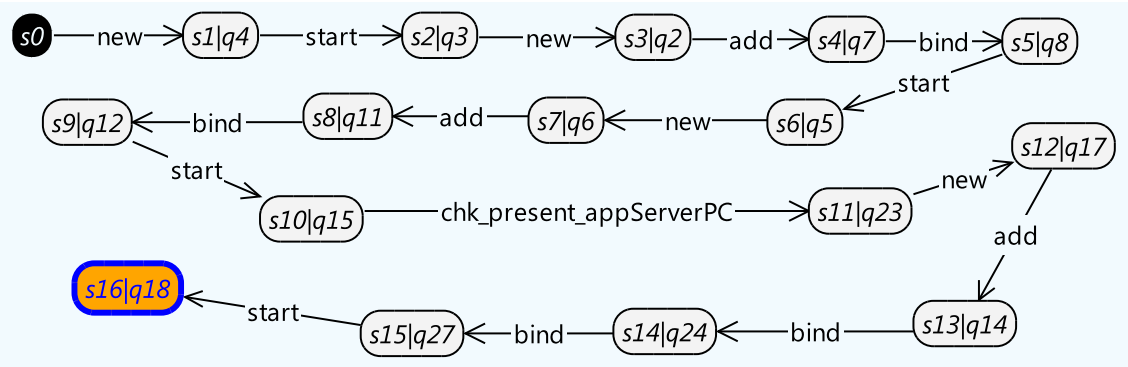


Рис. 6: Сервер управляемого приложения ($ic = 5$)
Fig. 6. Managed application server ($ic = 5$)

Для VM с более чем одним сервисным компонентом, такой как LAMP сервер ($ic = 10$ или $ic = 11$), имеющей http и службу данных, порядок связывания и запуска этих компонентов не является предопределенным, как показано на рис. 7. Изменения сначала происходят детерминированным образом от состояния s0 до s8. Состояние s16, в правом верхнем углу, обозначает граф, соответствующий спецификации для установочного кода 10, т.е. для неуправляемого LAMP сервера. Начиная с этого состояния может быть вызвана *GROOVE* функция *manage()*, чтобы получить управляемый LAMP сервер ($ic = 11$) между s23 и s41. Заметим, что изменения между s8 и s16 недетерминированы. Мы располагаем двумя кратчайшими путями (s8 → s10 → s16 и s8 → s11 → s16), которые могут быть легко обнаружены, используя исследование в ширину.

Для каждого кода инсталляции таблица 5 показывает количество состояний и переходов системы с переходами над графами. У системы с переходами над графами для $ic = 11$, показанной на рис. 7, имеется 26 состояний и 66 переходов. Заметим, что в нашей программной реализации порядок примитивных операций по реконфигурированию полностью определен для $0 \leq ic \leq 5$ и $8 \leq ic \leq 9$.

В таблице 5 для $6 \leq ic \leq 7$ и $10 \leq ic \leq 13$ мы наблюдаем для каждой VM с двумя услугами, что управляемая версия имеет на 16 состояний и переходов больше, чем неуправляемая. Подобный вывод может быть сделан, рассматривая последнюю линию таблицы 5. Это показывает, как изображено на рис. 7, что *GROOVE* функция *manage()* полностью определяет порядок примитивных операций по реконфигурированию. Заметим, что количество состояний и переходов для $ic = 6$ (соотв. $ic = 7$),

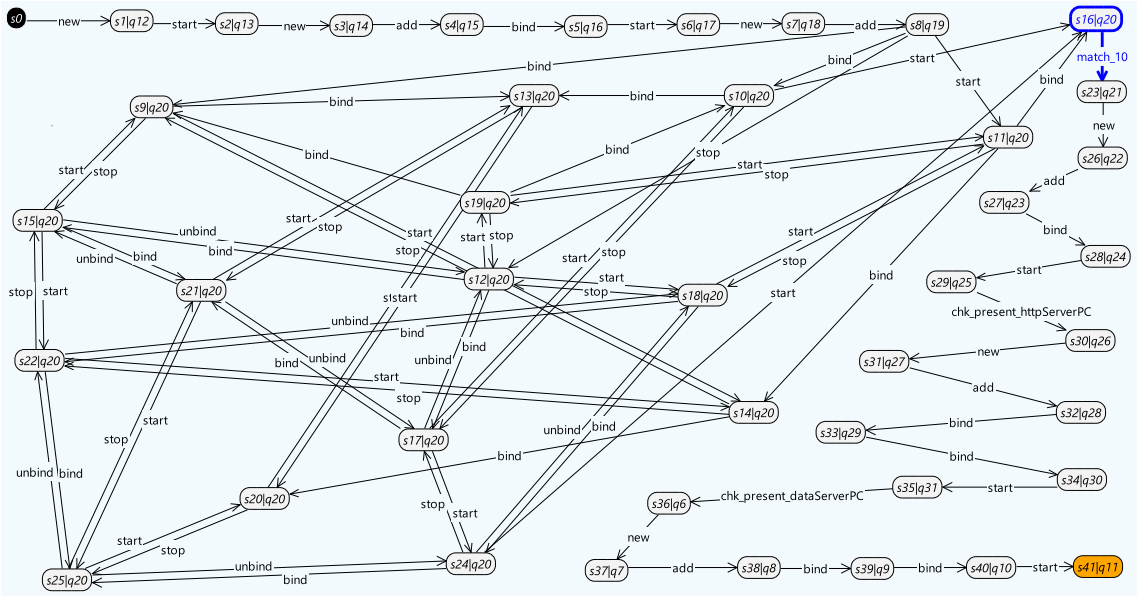


Рис. 7: Управляемый LAMP сервер ($ic = 11$)
Fig. 7. Managed LAMP server ($ic = 11$)

Таблица 5: Количество состояний и переходов для кода инсталляции

Table 5. Number of states and transitions per install code

Unmanaged			**Managed**		
ic	states	transitions	*ic*	states	transitions
0	3	2	1	7	6
2	7	6	3	17	16
4	7	6	5	17	16
6	46	155	7	62	171
8	7	6	9	17	16
10	26	66	11	42	82
12	26	66	13	42	82
14	265	1456	15	288	1479

отличается от их количества для $ic = 10$ или $ic = 12$ (соотв. $ic = 11$ или $ic = 13$). Это объясняется тем фактом, что в отличие от *httpServer* или *appServer*, подкомпонент *dataServer* не имеет требуемого интерфейса (см. рис. 1), что приводит к дополнительному детерминизму, для достижения конфигурации, включающей этот компонент.

5. Имплементация vs. спецификация

В модели спецификации примитивные операции и охраняемые реконфигурирования оставались достаточно абстрактными, и операция *run* не интерпретировась. Формальная семантика компонентно-ориентированной системы с интерпретируемыми

операциями может быть получена, просто обогащая конфигурации более точной памятью состояний и эффектом этих операций на память.

5.1. Интерпретируемые конфигурации и реконфигурирования

Пусть $GM = \{u...\}$ – множество (бесконечное в общем случае) состояний общей глобальной памяти, и $LM = \{v...\}$ – множество (бесконечное в общем случае) состояний локальной памяти данного компонента. Эти состояния памяти читаются и изменяются при выполнении примитивных и непримитивных реконфигурирований, а также операций, конкретизирующих run. Формально все действия $ope \in \mathcal{R}_{run}$ интерпретированы как отображения $\overline{ope}$ из $GM \times LM$ на себя. Кроме того, существуют некоторые действия, свойственные имплементации $\mathcal{R}_{imp}$, такие как $manage$ из раздела 4. Назовем $\mathcal{I} = (GM, LM, (\overline{ope})_{ope \in \mathcal{R}_{run} \cup \mathcal{R}_{imp}})$ интерпретацией базового множества $\mathcal{R}_{run}$. Пусть $\mathcal{I}_{\mathcal{R}_{run}} = \{\mathcal{I}, \mathcal{I}_{GROOVE}...\}$ обозначает класс всех интерпретаций, в частности содержащий $\mathcal{I}_{GROOVE}$ для интерпретации $GROOVE$.

Интерпретируемые конфигурации. В дополнение к уже интерпретируемым параметрам и интерфейсам (см. [7] для дополнительной информации), состояние компонентов может быть описано более точно при помощи состояний локальной памяти. Множество интерпретируемых состояний компонентов есть наименьшее множество $State_{\mathcal{I}}$ такое что, если $s_1, \ldots, s_n$ – элементы в $State$[6], $v_1, \ldots, v_n \in LM$ – состояния локальной памяти, тогда $((s_1, v_1), \ldots, (s_n, v_n))$ являются состоянием в $State_{\mathcal{I}}$. Затем множество интерпретируемых конфигураций $\mathcal{C}_{\mathcal{I}}$ определяется через $GM \times State_{\mathcal{I}}$.

Интерпретируемые переходы. Предположим, что все примитивные действия имеют детерминированный эффект на локальную и глобальную память, всегда заканчиваются (либо нормально, либо с исключением), и дают результат.

Для $\mathcal{I}_{GROOVE}$ из раздела 4, каждый граф представляет интерпретируемую конфигурацию, соответствующую конфигурации, данной в определении 1, тогда как переходы между конфигурациями осуществляются, используя правила над графами.

Для каждой примитивной операции реконфигурирования ope соответствующее правило над графами, обозначенное $\overline{ope}$, имеет эквивалентные или более строгие предварительные условия. Например, для примитивной операции add предварительные условия (3) и (4) из таблицы 2 представлены графами LHS и NAC (рис. 4a и 4b) соответствующего правила над графами, тогда как выходное условие (1) изображено на рис. 4c. Предварительные условия (2) и (5) неявно определены типированием правила, которое содержит вершину типа $component$[7](соотв. $composite$), соответствующие компонентам $comp_1$ (соотв. $comp_2$) в таблице 2. Поскольку обе вершины в правиле над графами наследуют тип $component$, предварительное условие (2) выполнено. Кроме того, тот факт, что вершина, соответствующая $comp_2$, типирована как $composite$, гарантирует, что она не содержит параметров, и таким образом удовлетворяет предварительное условие (5).

[6]Рассматриваемое как отношение.

[7]Так как это абстрактный тип, вершина типа *component* является либо *примитивной*, либо *составной*.

Кроме того, отметим на рис. 4b, в дополнение к дуге с маркировкой $parent^*$, удовлетворяющей предварительное условие (4), другую дугу с маркировкой *parent*, гарантирующей, что вершина типа *composite* не является родителем другой вершины, т.е. $(comp_1, comp_2) \notin Parent$. Это условие не присутствует в таблице 2 из-за спецификации на базе множеств. Однако без этого NAC $(comp_1, comp_2) \notin Parent$ имплементация на базе *GROOVE* может закончить с двумя дугами, маркированными *parent* между вершиной типа *component* и вершиной типа *composite*. Это могло бы произвести граф, не соответствующий спецификации из определения 1.

Наконец, все конструкции теперь ведут себя детерминированно, и недетерминированное глобальное поведение производится произвольным чередованием действий компонентов. Это приводит к следующему определению.

Определение 4 (Семантика имплементации)**.** *Операционная семантика имплементации определяется системой с переходами* $S_{\mathcal{I}} = \langle \mathcal{C}_{\mathcal{I}}, \mathcal{C}^0_{\mathcal{I}}, \mathcal{R}_{run_{\mathcal{I}}}, \rightarrow_{\mathcal{I}}, l_{\mathcal{I}} \rangle$*, где* $\mathcal{C}_{\mathcal{I}}$ *– множество конфигураций вместе с их состояниями памяти,* $\mathcal{C}^0_{\mathcal{I}}$ *– множество начальных конфигураций,* $\mathcal{R}_{run_{\mathcal{I}}} = \{\overline{ope} \mid ope \in \mathcal{R}_{run} \cup \mathcal{R}_{imp}\}$*,* $\rightarrow_{\mathcal{I}} \subseteq \mathcal{C}_{\mathcal{I}} \times \mathcal{R}_{run_{\mathcal{I}}} \times \mathcal{C}_{\mathcal{I}}$ *– отношение реконфигурирования, соблюдающее правила над графами, и* $l_{\mathcal{I}} : \mathcal{C}_{\mathcal{I}} \rightarrow CP$ *– всюду определенная функция интерпретации.*

5.2. Корректные имплементации и сохранение непротиворечивости

Существуют прочные связи между интерпретируемой моделью и моделью спецификации. В этом разделе будет установлено, что имплементация на базе *GROOVE* соответствует спецификации.

Пусть $\mathcal{R}_{run_{\mathcal{I}}} = \{\overline{ope} \mid ope \in \mathcal{R}_{run} \cup \mathcal{R}_{imp}\}$ – множество, в котором $\overline{ope}$ принадлежит конечному множеству охраняемых реконфигурирований, использующих примитивные правила над графами, конкретизированные при имплементации системы. Для имплементации на базе *GROOVE* рассмотрим $\mathcal{R}_{imp} = \{match\}$, где *match* представляет операции для определения значений условий для реконфигурирования, используемых в *GROOVE*. Это не изменяет текущий граф, подобно "chk_present_appServerPC" на рис. 6 в потоке управления.

Чтобы установить связи между интерпретируемой моделью и моделью спецификации, мы предлагаем использовать версию классической τ-симуляции [16], маркируя как τ операции в $\mathcal{R}_{imp}$. Для всех $ope \in \mathcal{R}$ обозначим $c \overset{ope}{\Rightarrow} c'$, когда есть $n, m \geq 0$, такие что $c \overset{\tau^n ope \tau^m}{\longrightarrow} c'$.

Определение 5 (τ-симуляция)**.** *Пусть* S_1 *и* S_2 *– две модели над* $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$*. Бинарное отношение* $\sqsubseteq_\tau \subseteq \mathcal{C}_1 \times \mathcal{C}_2$ *называется* τ*-симуляцией тогда и только тогда, когда для любой операции* ope *в* $\mathcal{R}$*,* $(c_1, c_2) \in \sqsubseteq_\tau$ *влечет: когда* $c_1 \overset{ope}{\Rightarrow}_1 c'_1$*, тогда существует* $c'_2 \in \mathcal{C}_2$*, такое что* $c_2 \overset{ope}{\Rightarrow}_2 c'_2$ *и* $(c'_1, c'_2) \in \sqsubseteq_\tau$*.*

Определим S_1 и S_2 как τ-подобные, что обозначено $S_1 \sqsubseteq_\tau S_2$, если $\forall c^0_1 \in \mathcal{C}^0_1 \exists c^0_2 \in \mathcal{C}^0_2 . (c^0_1, c^0_2) \in \sqsubseteq_\tau$.

Рассмотрим интерпретируемые операции по реконфигурированию в $\mathcal{R}_{run_{\mathcal{I}}}$ и соответствующие им неинтерпретируемые операции. Маркируя операции в $\mathcal{R}_{imp}$ через τ, мы можем утверждать следующее:

Теорема 1 (Моделирование)**.** $S_{\mathcal{I}} \sqsubseteq_{\tau} S$.

Набросок. Установим сначала, как это сделано выше для операции *add*, что в имплементации у любой примитивной операции по реконфигурированию имеются более сильные предварительные условия, чем у соответствующей операции в модели спецификации. Таким образом, мы можем доказать[8], что охраняемые реконфигурирования, состоящие из примитивных операторов $G \to \overline{s}$, с $\overline{s} \in \mathcal{R}_{run_{\mathcal{I}}} \backslash \mathcal{R}_{imp}$, имеют более строгие предварительные условия, чем соответствующие операторы $s \in \mathcal{R}_{run}$.

Рассмотрим последовательность охраняемых реконфигурирований $G_0 \to \overline{s}_0, G_1 \to \overline{s}_1, \ldots, G_n \to \overline{s}_n$, игнорируя закрывающие действия τ в $\mathcal{R}_{imp}$. Для всех i, таких что $0 \leq i \leq n$, $\overline{s}_i$ имеет более строгие предварительные условия, чем s_i; таким образом, существует G'_i, т.ч. $G_i \to G'_i$ и $G'_i \to s_i$, как иллюстрировано ниже.

$$\begin{array}{cccccccccc} S_{\mathcal{I}} & G_0 & \to & \overline{s}_0, & G_1 & \to & \overline{s}_1, & \ldots, & G_n & \to & \overline{s}_n \\ & \Downarrow & & & \Downarrow & & & & \Downarrow & & \\ S & G'_0 & \to & s_0, & G'_1 & \to & s_1, & \ldots, & G'_n & \to & s_n \end{array}$$

Поскольку закрывающие действия τ в $\mathcal{R}_{imp}$ вводятся, чтобы оценить условия последовательностей охраняемых реконфигурирований, они не формируют бесконечных циклов, состоящих только из τ-переходов. Таким образом, всегда существует выход из этих циклов, если таковые имеются, с помощью перехода с этикеткой $\overline{ope}$, и этот переход всегда в конечном счете выполнен, что заканчивает доказательство. □

Этот результат показывает, что модель спецификации является корректным приближением к более реалистичной интерпретируемой модели. Поскольку свойства достижимости совместимы с $\sqsubseteq_{\tau}$, это приводит нас, следовательно, к следующему результату:

Утверждение 2 (Корректность и полнота)**.**

- *Если конфигурация c не достижима в S, то она не достижима ни в какой $S_{\mathcal{I}}$.*
- *С другой стороны, если конфигурация c достижима в S, то существует интерпретация $\mathcal{I}$, такая что c достижима в $S_{\mathcal{I}}$.*

Как следствие теоремы 1 и утверждений 1 и 2, верен следующий результат:

Утверждение 3. *Пусть $S_{\mathcal{I}} = \langle \mathcal{C}_{\mathcal{I}}, \mathcal{C}^0_{\mathcal{I}}, \mathcal{R}_{run_{\mathcal{I}}}, \to_{\mathcal{I}}, l_{\mathcal{I}} \rangle$ – интерпретируемая модель, и $S = \langle \mathcal{C}, \mathcal{C}^0, \mathcal{R}_{run}, \to, l \rangle$ – модель спецификации. При $\mathcal{C}^0_{\mathcal{I}} \subseteq \mathcal{C}_{\mathcal{I}}$, если $S_{\mathcal{I}} \sqsubseteq_{\tau} S$, тогда* `consistent`$(\mathcal{C}^0_{\mathcal{I}})$ *влечет* `conistent`$(reach(\mathcal{C}^0_{\mathcal{I}}))$.

[8]Используя гипотезу о самых слабых предварительных условиях, как определено в [8].

6. Библиографические заметки и заключение

6.1. Библиографические заметки

Самоадаптация – важная и активная область исследования с приложениями в различных областях. В работе [1] подчеркнута важная проблема, состоящая в устранении разрыва между разработкой и имплементацией адаптивных систем. Настоящая статья показывает, что платформа *GROOVE* может помочь устранить этот разрыв. В [5] рекофигурирование компонентно-ориентированных систем осуществлялось во времени выполнения, используя политики адаптации, приводимые в действие с помощью темпоральных шаблонов. Однако рассмотренные в этой работе реконфигурирования были просто последовательностями примитивных операций по реконфигурированию. В настоящей работе, где используются альтернативные и повторяющиеся конструкции для составления реконфигурирований, реконфигурирования могут иметь различные результаты. Это зависит от контекста или может быть вызвано недетерминированными механизмами при имплементации. В данной работе это не только статическая последовательность инструкций реконфигурирования (как это имело место в [5, 10, 12, 17]), но действительно *динамическое* реконфигурирование.

В отличие от [18], мы предполагаем, что реконфигурирования не всегда ведут компоновку компонентов от одной непротиворечивой архитектуры к другой. Таким образом, мы рассматриваем более общий случай реконфигурирований.

Непротиворечивость версий была введена в [17], чтобы минимизировать прерывание службы (*disruption*) и задержки, с которыми компонентно-ориентированные (распределенные) системы обновляются (*timeliness*) посредством реконфигурирований. Она квалифицирует состояние, где транзакции в пределах системы таковы, что данное реконфигурирование может не разрушить систему и произойти в ограниченное время; непротиворечивость версии была связана с *неподвижностью* [19] и *спокойствием* [20] с намерением собрать лучшее из обоих понятий. В отличие от [17, 19, 20], мы рассматриваем архитектурные ограничения как предварительные условия для применения охраняемых реконфигурирований. Таким образом, рассматривая компоненты как черные ящики, принцип разделения проблем принят во внимание. Аппликативная непротиворечивость, связанная с транзакциями в пределах системы или с внешними событиями, может сохраняться во время выполнения (*runtime*), используя механизмы политик адаптации, как описано в [5] для централизованной системы и [6] для децентрализованных или распределенных систем.

Следуя за [21], наше понятие непротиворечивости может быть рассмотрено как специальный архитектурный стиль. Тем не менее, при использовании графовых грамматик мы представляем типы интерфейсов компонентно-ориентированных систем специальными вершинами графов. Таким образом, подобно [22] мы можем осуществлять мониторинг темпоральных свойств на уровне интерфейсов.

6.2. Заключение

Следуя за [8], в данной работе предложена грамматика для охраняемых реконфигурирований. Она позволяет создавать реконфигурирования на базе примитивных операций по реконфигурированию, используя последовательности реконфигурирований, а также альтернативные и повторяющиеся конструкции. Определение самых слабых предварительных условий для применения реконфигурирований позволило нам доказать, что эти охраняемые реконфигурирования сохраняют непротиворечивость конфигураций.

В данной работе также представлена имплементация модели, используя программный инструмент преобразования графов *GROOVE* [9], где компонентно-ориентированные системы представляются как графы, элементы (например, компоненты, интерфейс, параметры, и т.д.) представлены вершинами, а отношения между элементами (например, $Parent$, $Bindings$, и т.д.) представлены дугами. Эта имплементация, используемая для работы с примером управляемой/неуправляемой облачной среды, позволяет моделировать реконфигурирование правилами над графами, используя LHS, RHS и NAC подграфы. Когда различные результаты могут произойти для каждого реконфигурирования, множество возможных выполнений может быть представлено как граф LTS, используя нашу имплементацию под *GROOVE*; возможные состояния, т.е. конфигурации системы под наблюдением, показаны как вершины, а реконфигурирования между ними как переходы.

По отношению к модели реконфигурирования, корректность интерпретируемых систем также была доказана с использованием правил грамматик над графами для выполнения реконфигурирования, что демонстрирует корректность нашей имплементации под *GROOVE*.

Возможное направление дальнейшего исследования включает анализ вышеупомянутых графов LTS, чтобы обнаружить или предотвратить формирование циклов в рамках реконфигурирований. Мы также планируем реализовать динамические реконфигурирования, основанные на грамматиках графа для приложения политик адаптации во время выполнения компонентно-ориентированных систем.

A. Доказательство утверждения 1 для конструкций на базе примитивных утверждений

Рассмотрим множество grs охраняемых реконфигурирований, использующих охраняемые списки, составленные только из примитивных утверждений. Это множество grs содержит $B'_1 \to S_1 [\!] \dots [\!] B'_n \to S_n$ с булевым B'_i и $Si = ope^i_0; ope^i_1; \dots; ope^i_{n_i}$, где n_i представляет количество примитивных утверждений $(ope^i_0, ope^i_1, \dots, ope^i_{n_i})$ охраняемого списка S_i, и R^i_j (соотв. R^i_{j+1}) представляет предварительное условие (соотв. выходное условие) ope^i_j для $0 < i \leq n$ и $0 \leq j \leq n_i$ (соотв. $0 < j \leq n_i + 1$).

Так как R^i_0 – предварительное условие S_i, и конфигурация перед применением S_i рассматривается как непротиворечивая, grs переписывается в виде $B_1 \to S_1 [\!] \dots [\!] B_n \to S_n$, с $B_i = B'_i \wedge CC \wedge R^i_0$. Определим $BB = (\exists i : 1 \leq i \leq n : Bi)$, а также множества $I = \{i \in \mathbb{N}.1 \leq i \leq n\}$ и $I_\top = \{i \in I.Bi\}$.

Альтернативная конструкция

Пусть IF обозначает **if** $B_1 \to S_1 [\!] \ldots [\!] B_n \to S_n$ **fi**. По определению, $wp(IF, R) = BB \wedge \forall i \in I : B_i \Rightarrow wp(S_i, R)$. Ранее было установлено, что для последовательности примитивных утверждений S_i верно $CC \wedge R_0^i \Rightarrow wp(S_i, CC \wedge R_{n_i+1}^i)$; тогда по определению $B_i \Rightarrow wp(S_i, CC \wedge R_{n_i+1}^i) \Rightarrow wp(S_i, CC)$. Это означает, что $wp(IF, CC) = BB \wedge \forall i \in I : B_i \Rightarrow wp(S_i, CC)$, и этого достаточно, чтобы доказать, что непротиворечивость сохраняется альтернативной конструкцией.

Однако для альтернативной конструкции можно установить более строгое выходное условие $CC \wedge \bigwedge\limits_{i \in I_\top} R_{n_i+1}^i$, полагая, что каждый член конъюнкции $\bigwedge\limits_{i \in I_\top} R_{n_i+1}^i$ является частью условия охраняемого списка, имеющего право на выполнение.

Тогда $wp(IF, CC \wedge \bigwedge\limits_{j \in I_\top} R_{n_j+1}^j) = BB \wedge \forall i \in I : B_i \Rightarrow wp(S_i, CC \wedge R_{n_i+1}^i)$, так как по определению $\forall i \in I_\top, B_i = \top$.

Следовательно:

$$\begin{aligned} BB \wedge CC \wedge \bigwedge_{i \in I_\top} R_0^i \;&\Rightarrow\; BB \wedge \bigwedge_{i \in I_\top} wp(S_i, CC \wedge R_{n_i+1}^i) \\ &\Rightarrow\; BB \wedge (\forall i \in I : B_i \Rightarrow wp(S_i, CC \wedge R_{n_i+1}^i)) \\ &\Rightarrow\; wp(IF, CC \wedge \bigwedge_{j \in I_\top} R_{n_j+1}^j) \end{aligned}$$

В качестве примера возьмем частный случай альтернативной конструкции **if** B **then** S **fi**, записанный в виде **if** $B \to S [\!] \neg B \to skip$ **fi**. Его самое слабое предварительное условие – $wp(\textbf{if } B \textbf{ then } S \textbf{ fi}, CC \wedge (B \Rightarrow post_S)) = B \wedge wp(S, CC \wedge R_S)$, где R_S и $post_S$ – соответственно предварительное условие и выходное условие для S.

Повторяющаяся конструкция

Пусть DO обозначает **do** $B_1 \to S_1 [\!] \ldots [\!] B_n \to S_n$ **od**. Пусть $H_0(R) = R \wedge \neg B$, и для $k > 0$, пусть $H_k(R) = wp(IF, H_{k-1}(R)) \vee H_0(R)$, где IF обозначает ту же самую охраняемую конфигурацию, заключенную между "**if fi**". Тогда по определению $wp(DO, R) = \exists k : k > 0 : H_k(R)$.

Это означает, что самое слабое предварительное условие этой конструкции гарантирует надлежащее завершение после как максимум k выборов охраняемого списка, оставляя систему в состоянии, удовлетворяющем R. Рассмотрим $l_0, l_1, \ldots, l_k$, такие что для $0 \le j \le k$, $1 \le l_j \le n$ и $S_{l_0}; S_{l_1}; \ldots; S_{l_k}$ в качестве упорядоченной последовательности утверждений, выбранных при выполнении конструкции до ее завершения. Перед этим мы доказали, что такая последовательность сохраняет непротиворечивость. Поэтому $CC \wedge R_{n_{l_k}+1}^{l_k}$ является истинным выходным условием, и так как с $CC \wedge R_{n_{l_k}+1}^{l_k} \Rightarrow CC \wedge \bigvee\limits_{i \in I} R_{n_i+1}^i$, мы имеем $wp(DO, CC \wedge R_{n_{l_k}+1}^{l_k}) \Rightarrow wp(DO, CC \wedge \bigvee\limits_{i \in I} R_{n_i+1}^i)$.

Для последовательности примитивных заявлений S_i мы установили перед этим, что $CC \wedge R_0^i \Rightarrow wp(S_i, CC \wedge R_{n_i+1}^i)$; тогда $CC \wedge R_0^{l_0} \Rightarrow wp(S_{l_0}; S_{l_1}; \ldots; S_{l_k}, CC \wedge R_{n_{l_k}+1}^{l_k})$.

Следовательно:

$$\begin{aligned} CC \wedge \bigwedge_{i\in I} R_0^i &\Rightarrow CC \wedge R_0^{l_0} \\ &\Rightarrow wp(S_{l_0}; S_{l_1}; \ldots; S_{l_k}, CC \wedge R^{l_k}_{n_{l_k}+1}) \\ &\qquad\qquad \text{for any valid sequence } S_{l_0}; S_{l_1}; \ldots; S_{l_k} \\ &\Rightarrow wp(DO, CC \wedge R^{l_k}_{n_{l_k}+1}) \\ &\Rightarrow wp(DO, CC \wedge \bigvee_{i\in I} R^i_{n_i+1}) \end{aligned}$$

Это доказывает, что повторяющаяся конструкция, примененная к множеству охраняемых реконфигурирований, использующих списки примитивных утверждений, сохраняет непротиворечивость. Это также обеспечивает более строгие предварительные и выходные условия, которые используются для завершения доказательства утверждения 1.

Список литературы

[1] De Lemos R., Giese H., Müller H. A., Shaw M., Andersson J., Litoiu M., Schmerl B., Tamura G., Villegas N. M., Vogel T., et al., “Software engineering for self-adaptive systems: A second research roadmap”, *Software Engineering for Self-Adaptive Systems II*, 2013, 1–32.

[2] Dwyer M. B., Avrunin G. S., Corbett J. C., “Patterns in property specifications for finite-state verification”, Proceedings of the 1999 International Conference on, *Software Engineering*, 1999, 411–420.

[3] Dormoy J., Kouchnarenko O., Lanoix A., “Runtime verification of temporal patterns for dynamic reconfigurations of components”, LNCS, **7253**, eds. Arbab F., Ölveczky P., Springer, Berlin Heidelberg, 2012, 115–132.

[4] Trentelman K., Huisman M., “Extending JML specifications with temporal logic”, *Algebraic Methodology and Software Technology*, 2002, 334–348.

[5] Kouchnarenko O., Weber J. F., “Adapting component-based systems at runtime via policies with temporal patterns”, LNCS, **8348**, eds. Fiadeiro J. L., Liu Z., Xue J., Springer, 2014, 234–253.

[6] Kouchnarenko O., Weber J. F., “Decentralised evaluation of temporal patterns over component-based systems at runtime”, LNCS, **8997**, eds. Lanese I., Madelaine E., Springer, 2015, 108–126.

[7] Lanoix A., Dormoy J., Kouchnarenko O., “Combining proof and model-checking to validate reconfigurable architectures”, *ENTCS*, **279** (2011), 43–57.

[8] Dijkstra E. W., “Guarded commands, nondeterminacy and formal derivation of programs”, *Communications of the ACM*, **18** (1975), 453–457.

[9] Ghamarian A. H., de Mol M., Rensink A., Zambon E., Zimakova M., “Modelling and analysis using GROOVE”, *J. on Software Tools for Technology Transfer*, **14** (2012), 15–40.

[10] Bruneton E., Coupaye T., Leclercq M., Quéma V., Stefani J. B., “The fractal component model and its support in java”, *Softw., Pract. Exper.*, **36** (2006), 1257–1284.

[11] Schneider S., Treharne H., “Csp theorems for communicating b machines”, *Formal Asp. Comput.*, **17** (2005), 390–422.

[12] Seinturier L., Merle P., Rouvoy R., Romero D., Schiavoni V., Stefani J.B., “A component-based middleware platform for reconfigurable service-oriented architectures”, *Software: Practice and Experience*, **42** (2012), 559–583.

[13] Dormoy J., Kouchnarenko O., Lanoix A., "Using temporal logic for dynamic reconfigurations of components", LNCS, **6921**, eds. Barbosa L., Lumpe M., Springer, Berlin Heidelberg, 2012, 200–217.

[14] Hamilton A. G., *Logic for mathematicians*, Cambridge University Press, Cambridge, 1978.

[15] Hoare C. A. R., "An axiomatic basis for computer programming", *Communications of the ACM*, **12** (1969), 576–580.

[16] Milner R., *Communication and concurrency*, Prentice-Hall, Inc., 1989.

[17] Ma X., Baresi L., Ghezzi C., Panzica La Manna V., Lu J., "Version-consistent dynamic reconfiguration of component-based distributed systems", *ACM*, Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, 2011, 245–255.

[18] Boyer F., Gruber O., Pous D., "Robust reconfigurations of component assemblies", Int. Conf. on Software Engineering (ICSE '13, Piscataway), 2013, 13–22.

[19] Kramer J., Magee J., "The evolving philosophers problem: Dynamic change management", *Software Engineering, IEEE Transactions*, **16** (1990), 1293–1306.

[20] Vandewoude Y., Ebraert P., Berbers Y., D'Hondt T., "Tranquility: A low disruptive alternative to quiescence for ensuring safe dynamic updates", *Software Engineering, IEEE Transactions*, **33** (2007), 856–868.

[21] Le Metayer D., "Describing software architecture styles using graph grammars", *IEEE Transactions on Software Engineering*, **24** (1998), 521–533.

[22] Kähkönen K., Lampinen J., Heljanko K., Niemelä I., "The lime interface specification language and runtime monitoring tool", *Int. Wshop on Runtime Verification, RV'09*, LNCS, **5779**, eds. Bensalem S., Peled D.A., Springer, Berlin Heidelberg, 2009, 93–100.

Kouchnarenko O., Weber J.-F., "Component-based Systems Reconfigurations Using Graph Grammars", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 804–825.

DOI: 10.18255/1818-1015-2016-6-804-825

Abstract. Dynamic reconfigurations can modify the architecture of component-based systems without incurring any system downtime. In this context, the main contribution of the present article is the establishment of correctness results proving component-based systems reconfigurations using graph grammars. New guarded reconfigurations allow us to build reconfigurations based on primitive reconfiguration operations using sequences of reconfigurations and the alternative and the repetitive constructs, while preserving configuration consistency. A practical contribution consists of the implementation of a component-based model using the *GROOVE* graph transformation tool. Then, after enriching the model with interpreted configurations and reconfigurations in a consistency compatible manner, a simulation relation is exploited to validate component systems' implementations. This sound implementation is illustrated on a cloud-based multi-tier application hosting environment managed as a component-based system.

Keywords: component-based systems, dynamic reconfigurations, consistency, simulation relation, implementation, *GROOVE*

About the authors:
Olga Kouchnarenko, orcid.org/0000-0003-1482-9015, PhD,
Universite de Bourgogne - Franche-Comte,
16 route de Gray, 25000 Besancon, France, e-mail: olga.kouchnarenko@univ-fcomte.fr

Jean-Francois Weber, graduate student,
Universite de Bourgogne - Franche-Comte,
16 route de Gray, 25000 Besancon,France, e-mail: jfweber@femto-st.fr

Acknowledgments:
This work has been partially funded by the Labex ACTION, ANR-11-LABEX-0001-01.

Моделирование и анализ информационных систем. Т. 23, №6 (2016), с. 826–840
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 826–840

DOI: 10.18255/1818-1015-2016-6-826-840

УДК 004.912

Методические аспекты выделения семантических отношений для автоматической генерации специализированных тезаурусов и их оценки

Лагутина Н. С., Лагутина К. В., Мамедов Э. И., Парамонов И. В.[1]

получена 19 октября 2016

Аннотация. Работа посвящена анализу методов автоматической генерации специализированного тезауруса. Основной алгоритм генерации состоит из трех шагов: отбор и предварительная обработка корпуса текстов, формирование множества терминов для включения в тезаурус и выделение связей между терминами тезауруса. Данное исследование сфокусировано на изучении методов выделения семантических связей, для чего авторами был разработан программный стенд, который позволяет протестировать распространенные алгоритмы выделения гиперонимов и синонимов, использующие в своей работе лексико-синтаксические шаблоны, морфо-синтаксические правила, количество информации терминов, тезаурус общего назначения WordNet и расстояние Левенштейна. Для анализа результирующего тезауруса, созданного на стенде, авторами была разработана комплексная оценка, содержащая следующие характеристики качества: точность выделения терминов, точность и полнота выделения синонимических и гиперонимических связей, а также метрики графа тезауруса (количество выделенных терминов, количество семантических связей различных типов, число компонент связности и число вершин в наибольшей компоненте). Предлагаемый набор метрик позволяет оценить качество тезауруса в целом, выявить отдельные недостатки стандартных методов выделения связей и построить более эффективные гибридные методы, генерирующие тезаурус с лучшими характеристиками по сравнению с тезаурусами, генерируемыми при использовании отдельных методов. Для иллюстрации данного факта в статье рассмотрен один из таких гибридных методов. Он комбинирует лучшие стандартные алгоритмы построения гиперонимических и синонимических связей и строит специализированный тезаурус в области медицины с тем же уровнем качества, что и другие методы, но с большим количеством связей между терминами.

Ключевые слова: тезаурус, семантические отношения, гибридный метод, комплексная оценка, программный стенд

Для цитирования: Лагутина Н. С., Лагутина К. В., Мамедов Э. И., Парамонов И. В., "Методические аспекты выделения семантических отношений для автоматической генерации специализированных тезаурусов и их оценки", *Моделирование и анализ информационных систем*, **23**:6 (2016), 826–840.

Об авторах:
Лагутина Надежда Станиславовна, orcid.org/0000-0002-6137-8643, канд. физ.-мат. наук, доцент,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: lagutinans@rambler.ru

Лагутина Ксения Владимировна, orcid.org/0000-0002-1742-3240, студент,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: lagutinakv@mail.ru

Мамедов Эльдар Интизамович, orcid.org/0000-0003-1997-7084, стажёр-исследователь,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: eldarmamedov90@gmail.com

Парамонов Илья Вячеславович, orcid.org/0000-0003-3984-8423, канд. физ.-мат. наук, доцент,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: Ilya.Paramonov@fruct.org

Благодарности:
[1]Работа выполнена при финансовой поддержке гранта Президента Российской Федерации для государственной поддержки молодых российских ученых (государственный контракт № МК-5456.2016.9).

Введение

Тезаурус можно определить как словарь терминов на естественном языке, в котором все элементы связаны между собой семантическими отношениями, отражающими основные соотношения понятий в описываемой предметной области знаний [1].

Тезаурусы широко применяются в ряде областей. Одной из таких областей является информационный поиск, включающий в себя поиск по запросам пользователей и индексацию документов. Другая область — анализ текстов, в частности рубрикация и классификация документов, моделирование связности текста, автоматический перевод.

Большое практическое значение имеет информационный поиск в специализированных областях, где он существенно отличается от поиска в коллекциях документов из интернета, так как требует лингвистических и онтологических знаний о предметной области. Одним из ресурсов, содержащих такие знания, является специализированный тезаурус [2].

В настоящий момент существует не слишком много информационно-поисковых тезаурусов предметных областей, поскольку процесс построения специализированных тезаурусов является чрезвычайно трудоемким и финансово затратным, кроме того, результат сильно зависит от квалификации экспертов. Автоматизировать процесс построения такого тезауруса также достаточно сложно. Весь процесс целиком и его отдельные части являются предметом многочисленных исследований.

В то же время отмечается ряд проблем, которые мешают эффективному использованию тезаурусов в системах компьютерной обработки текста, в частности нехватка отношений между терминами предметной области [3]. Таким образом, в процессе автоматического построения информационно-поисковых тезаурусов важно добиться качественного моделирования рассматриваемой области, при этом существенными характеристиками результата являются параметры связей между отдельными элементами.

Задачей статьи является анализ и обобщение результатов исследований авторов, представленных в предыдущих работах, посвященных отдельным аспектам алгоритмов автоматического построения тезаурусов. В данной работе значительное внимание уделяется методам выделения связей между терминами. Наличие разных типов связей позволяет максимально эффективно отразить знания о соответствующей предметной области и улучшить в дальнейшем качество применения тезауруса в информационных системах. В связи с этим большое значение приобретают характеристики связности между терминами тезауруса, поэтому авторы рассматривают методические аспекты выделения связей между терминами, основанные на применении гибридных методов. Кроме того, авторы рассматривают вопросы комплексной оценки автоматически сгенерированного тезауруса. Такая оценка может послужить основой для построения полностью автоматических алгоритмов создания тезауруса.

1. Общая схема построения тезауруса

Формальное определение тезауруса может выглядеть следующим образом:

$$T =< D, R_s, R_h, R_a >$$

Здесь D — множество всех терминов тезауруса. $R_s \subset D \times D$ — отношение синонимии, обладающее свойствами симметричности и транзитивности. В каждом множестве синонимов $d_i : \forall i, j = \ldots n(d_i, d_j) \in R_s$ обычно выделяется один дескриптор, соответствующий понятию предметной области, остальные синонимы называются аскрипторами. R_h — отношение иерархии. Иерархическими являются родо-видовые (гипонимо-гиперонимические) связи, а также связи типа часть-целое. Иерархические отношения обладают свойствами несимметричности и транзитивности. R_a — ассоциативное отношение. Чаще всего ассоциативными отношениями называются отношения между дескрипторами предметной области, не являющиеся иерархическими или синонимическими.

Метод автоматизированного построения тезауруса состоит из нескольких этапов:

1. Отбор и предварительная обработка корпуса текстов.
2. Формирование множества терминов для включения в тезаурус.
3. Выделение связей между терминами тезауруса.

Отбор и предварительная обработка корпуса текстов не является предметом обсуждения в данной работе. Корпус представляет собой сформированную по определенным правилам выборку текстов, относящихся к предметной области.

Задача выделения терминов для тезауруса обычно рассматривается как задача выделения ключевых фраз из корпуса текстов [4]. Поэтому для исследования были выбраны соответствующие алгоритмы, описанные в разделе 2.

Для выделения связей между терминами существуют разные типы алгоритмов, зависящие от типов связей и способов их нахождения. Подробное описание исследуемых в данной работе методов построения связей в тезаурусе находится в разделе 3.

Для оценки качества работы методов выделения терминов или отношений между ними практически во всех научных работах используются три статистические характеристики: точность, полнота и F-мера [5]. Точность — это число правильно выделенных ключевых слов, поделенное на число всех выделенных ключевых слов. Полнота — это число правильно выделенных ключевых слов, поделенное на число всех подходящих ключевых слов. F-мера — это среднее гармоническое точности и полноты, которая позволяет оценить баланс между точностью и полнотой, придавая одинаковый вес обеим метрикам. В данной работе авторы вычисляли точность выделенных терминов, иерархических и синонимических отношений; полноту иерархических и синонимических отношений.

Качество тезауруса также зависит от набора его связей: чем больше семантических отношений в тезаурусе и чем выше его связность, тем лучше результаты дает информационный поиск с его участием [3]. Для оценки данной характеристики введем граф тезауруса $G = (D, R)$, где множество вершин D — это множество всех терминов тезауруса, определенное ранее, а множество ребер $R = R_s \cup R_h \cup R_a$ — это множество всех связей между терминами.

Авторами были выбраны следующие характеристики графа тезауруса, позволяющие оценить набор его связей:

- количество выделенных терминов $|D|$;
- количество отношений различных типов $|R_s|, |R_h|, |R_a|$;
- число компонент связности и вершин в наибольшей компоненте, т. е. метрики, описывающие связность графа тезауруса.

Число выделенных терминов и отношений между ними — самые грубые метрики, по которым можно оценить размер тезауруса. Общее число терминов не должно быть маленьким, число вертикальных и горизонтальных связей должно быть больше числа терминов, так как практически все термины имеют хотя бы один гипероним и несколько синонимов или ассоциаций. Выход за рамки данных правил означает низкое качество результирующего тезауруса.

Оценка связности тезауруса выполняется по примеру предыдущей работы авторов [6] и обосновывается тем фактом, что для успешного применения тезауруса в большинстве случаев решающее значение имеет навигация по связям. В частности, авторы предлагают использовать характеристики графа, определенного выше. В том случае, если тезаурус содержит только небольшие компоненты связности, то он не отражает структуру предметной области. Также граф тезауруса не должен содержать изолированных вершин, так как соответствующие им термины не имеют связей и не могут быть использованы на практике.

Полученный набор характеристик является комплексной оценкой качества полученного узкоспециализированного тезауруса. Подобные оценки редко появляются в современных исследованиях, так как чаще всего оцениваются отдельные этапы или методы генерации тезауруса. Нередко предлагаемые методы улучшают одну характеристику тезауруса, но ухудшают другую, например, увеличение полноты сопровождается уменьшением точности [7]. Комплексная оценка тезауруса как единого целого позволяет построить более эффективные гибридные методы, дающие возможность найти компромиссное решение.

2. Методы выделения терминов тезауруса

Для решения задачи автоматического выделения терминов тезауруса из текстов, иначе говоря — ключевых фраз, разработано большое количество подходов. Все существующие алгоритмы состоят из двух шагов: выделение ключевых слов-кандидатов эвристическими методами и выбор релевантных ключевых слов из набора слов-кандидатов при помощи математических методов видов, относящихся к одной из двух категорий: «обучение с учителем» (обучаемые) и «обучение без учителя» (необучаемые).

Для проведения настоящего исследования из этих двух категорий были выбраны два метода: Topical PageRank из категории «обучение без учителя» и Maui — «обучение с учителем».

В работе [6] авторы данной статьи исследуют выбранные стандартные методы для выделения ключевых слов на корпусе текстов о туристических объектах Карелии. Данный корпус из 900 текстов удобен для анализа методов решения задачи построения узкоспециализированного тезауруса в частности тем, что содержит выборку из 200 текстов с выделенными вручную ключевыми словами. Это позволяет

обрабатывать его при помощи алгоритмов как вида «обучение без учителя», так и «обучение с учителем».

В экспериментах оценивалось качество выделенных ключевых слов для каждого алгоритма из четырех стандартных. Для данных экспериментов использовались 200 текстов с выбранными экспертом ключевыми словами, из которых 100 текстов применялись для обучения алгоритмов, а 100 других текстов — для оценки качества. Все алгоритмы запускались на этом наборе, и их результаты сравнивались с ключевыми словами, выделенными экспертом. Релевантность оценивалась по трем метрикам: точности, полноте и F-мере.

Полученные значения характеристик показывают, что алгоритм Maui работает эффективнее, чем остальные алгоритмы: 57.5 % выбранных слов хорошо характеризуют текст, 24.1 % из всех возможных правильных ключевых слов были выбраны, а значение F-меры 34.0 — наибольшее среди всех. Вторым по эффективности является необучаемый алгоритм Topical PageRank: точность — 51.7 %, полнота — 22.0 %.

Несмотря на то, что значения характеристик достаточно хороши по сравнению с результатами других исследований [5], очевидно, что они являются весьма низкими по абсолютным значениям. Можно сделать вывод, что для повышения качества выделения ключевых слов в конкретной предметной области требуется существенная доработка стандартных алгоритмов. Авторами был предложен алгоритм выделения ключевых слов, учитывающий особенности предметной области туризма, а также особенности информационной системы, для которой он разрабатывался [6]. Аналогичный подход может быть использован для повышения качества выделения терминов специализированного тезауруса.

В данной работе авторы не исследуют алгоритмы для выделения терминов, поэтому для выделения ключевых слов во всех экспериментах по выделению семантических отношений был выбран единственный метод TextRank [8] вида «обучение без учителя». Авторы не использовали в данном исследовании обучаемые алгоритмы, так как алгоритмы без учителя не нуждаются в текстах с выбранными вручную ключевыми словами и поэтому представляют разумный компромисс для поставленной задачи максимально автоматизировать построение тезауруса, поскольку показывают результаты лишь немного хуже, чем обучаемые алгоритмы. Хотя одним из лучших необучаемых алгоритмов считается Topical PageRank [9], он может быть успешно заменен на TextRank, так как Topical PageRank выбирает из текстов темы и повторяет для каждой TextRank. В данном исследовании все тексты объединены общей темой, и TextRank достаточно хорошо подходит для поставленной задачи.

3. Методы выделения связей между терминами тезауруса

Последним и крайне важным этапом генерации тезауруса является построение отношений между терминами. В данном разделе авторы анализируют наиболее известные и эффективные подходы к выделению различных типов связей, которые затем используются в реализации программного стенда, описанного в подразделе 4.1. Также данные методы комбинируются для создания гибридных методов (подраздел 4.3.).

Методы для определения семантических связей между терминами можно разделить на несколько групп в зависимости от типа выделяемых связей:

- методы определения ассоциативных связей;
- методы определения гипонимо-гиперонимических связей;
- методы определения синонимических связей;
- методы определения нескольких типов связей.

Рассмотрим данные методы подробно.

3.1. Методы определения ассоциативных связей

Одним из эффективных методов определения ассоциативных связей является метод LSA (Latent Semantic Analysis) [10]. Его основное предназначение заключается в нахождении взаимосвязей между корпусом документов и терминами, в них встречающимися. Также данный метод можно использовать и для нахождения степени связи между терминами корпуса документов.

Метод LSA работает следующим образом: сначала строится матрица термины-на-документы, в которой каждой строке соответствует термин из корпуса документов, каждому столбцу — документ. На пересечении строк и столбцов матрицы записывается число, указывающее, сколько раз соответствующий термин встречается в соответствующем документе. Каждую строку в получившейся матрице можно рассматривать как вектор, характеризующий определенный термин в корпусе и содержащий информацию о встречаемости термина в документах корпуса. После того как матрица построена, для неё вычисляется сингулярное разложение, чтобы получить наилучшее приближение матрицей меньшей размерности. В получившейся матрице каждая строка также характеризует определенный термин корпуса. Для того, чтобы определить степень связи между различными терминами, к соответствующим векторам терминов применяется какая-либо известная мера близости векторов, например, косинусная мера. Чем ближе по выбранной мере оказываются векторы, тем более связанными считаются соответствующие им термины.

3.2. Методы определения гипонимо-гиперонимических связей

Для определения гипонимо-гиперонимических связей часто используются лингвистические методы. Наиболее известными из них являются методы, основанные на морфо-синтаксических правилах и лексико-синтаксических шаблонах.

Метод, основанный на морфо-синтаксических правилах [11], определяет гипонимо-гиперонимические отношения в тех случаях, когда один термин является частью второго. Отношения определяются по следующим правилам:

- если первый термин является строкой-суффиксом для второго термина, то первый является гиперонимом, а второй — гипонимом;
- если первый термин является частью фразы второго многословного термина, то первый является гиперонимом, а второй — гипонимом.

Другой лингвистический метод использует известные лексико-синтаксические шаблоны, в которых часто встречаются термины, состоящие в иерархическом отношении [12]. Каждое предложение в исходном тексте проверяется на наличие опреде-

ленного списка известных лексико-синтаксических шаблонов. В случае нахождения одного из шаблонов термины, в зависимости от их положения в шаблоне, определяются как гипонимы и гиперонимы по отношению друг к другу.

3.3. Методы определения синонимических связей

Одним из вариантов определения синонимичных терминов является поиск различных форм одного и того же слова. В узкоспециализированных тезаурусах формы слова и синонимы часто обозначаются как один и тот же тип связей (пример — медицинский тезаурус MeSH: https://www.nlm.nih.gov/mesh/). Методы нахождения форм слова можно разделить на несколько типов в зависимости от способа их определения: орфографические, морфологические, лексические, структурные, аббревиатурные [4].

Известным морфо-орфографическим методом определения различных вариантов слова является метод, основанный на расстоянии Левенштейна [13]. Он определяет количество изменений, которые требуются для того, чтобы преобразовать одно слово в другое. Учитываются такие изменения, как удаление символа, добавление символа и замена символа. Затем с учетом длины исходного термина и найденного количества требуемых изменений вычисляется коэффициент схожести терминов. Чем больше данный коэффициент, тем более схожими считаются термины, таким образом расстояние Левенштейна может использоваться для поиска однокоренных слов и лексических вариантов терминов тезауруса.

3.4. Методы определения нескольких типов связей

В данном разделе выделены методы, способные определять сразу несколько различных типов связей, в частности синонимические и гипонимо-гиперонимические связи. К таким методам относятся метод, основанный на оценке количества информации терминов, и метод, основанный на использовании тезауруса общего назначения WordNet.

Метод, основанный на оценке количества информации терминов [14], — это статистический метод, определяющий тип связи между терминами, опираясь на такие показатели, как количество информации терминов и схожесть контекстов, в которых встречаются термины. Во-первых, метод вычисляет количество информации для терминов по следующему правилу: чем чаще встречается термин, тем меньше он несет в себе информации. Частота встречаемости слов определяется в некотором большом общем корпусе текстов. Во-вторых, метод определяет термины, которые часто употребляются в схожих контекстах. Далее для каждой пары терминов сравниваются их контекст и количество информации. Если контекст у терминов один и тот же, и количество информации близко по значению, то термины определяются как синонимы. Если контекст одинаковый, а сами термины несут различное количество информации, то они формируют пару гипоним-гипероним. Причем гипонимом в данном случае считается термин, который несет в себе больше информации, а гиперонимом — термин с меньшим количеством информации.

Метод, основанный на использовании WordNet [15], устанавливает связь между терминами создаваемого специализированного тезауруса, если связь такого же типа

была найдена между данными терминами в тезаурусе общего назначения WordNet. Синонимическая связь между терминами устанавливается в том случае, когда данные термины определены как синонимы в WordNet; гипонимо-гиперонимическая связь устанавливается, если между данными терминами в WordNet также определена соответствующая связь.

4. Программный стенд для построения и оценки тезауруса

4.1. Общий алгоритм работы стенда

Для построения и оценки методов автоматической генерации тезаурусов авторы данной работы разработали программный стенд. Он позволяет автоматически сгенерировать узкоспециализированный тезаурус из неструктурированного корпуса текстов с использованием различных методов из предыдущего раздела или их комбинаций. Также стенд оценивает качество терминов и отношений между ними в результирующем тезаурусе. Общий алгоритм стенда включает в себя следующие шаги:

1. Выбор терминов для тезауруса с использованием алгоритма TextRank.
2. Выделение ассоциативных связей алгоритмом LSA.
3. Выделение гиперонимических связей при помощи статистических и семантических методов, а также их комбинаций.
4. Выделение синонимических связей различными статистическими и семантическими алгоритмами и их комбинациями.
5. Фильтрация терминов, которые не имеют связей.
6. Оценка качества терминов и связей в тезаурусе.

Для выделения иерархических связей исследуются четыре метода, описанные в разделе 2: морфо-синтаксические правила; лексико-синтаксические шаблоны; метод, использующий тезаурус общего назначения WordNet; метод на основе измерения количества информации, которое несёт термин. Два последних метода также используются и для выделения синонимов, наряду с методом, использующим расстояние Левенштейна. В разных экспериментах используются разные методы и их комбинации.

После выделения связей фильтруются термины, которые остались без связей, так как они бесполезны в подавляющем большинстве сценариев использования тезаурусов.

Результатом работы алгоритма стенда является узкоспециализированный тезаурус, содержащий термины и синонимические и гипонимо-гиперонимические связи между терминами, описывающий структуру предметной области, заданной конкретным корпусом текстов. Оценка тезауруса выполняется с использованием метрик, описанных в разделе 1.

4.2. Эксперименты на стенде: стандартные методы

Для экспериментов по автоматическому построению тезаурусов использовалась хорошо известная коллекция медицинских текстов MEDLINE (http://ir.dcs.gla.ac.uk/resources/testcollections/medl/). Коллекция MEDLINE содержит 1033 статьи из медицинских журналов и предназначена для оценки методов информационного поиска. В наших экспериментах данная коллекция использовалась как корпус текстов, из которого выделяются термины и связи между ними для автоматического построения тезаурусов в области медицины.

При использовании метода, основанного на оценке количества информации [14], для определения частоты встречаемости слов дополнительно использовался медицинский словарь Стедмана (http://stedmansonline.com/public/LearnMore.aspx?resourceID=Medical). Для определения терминов, встречающихся в схожих контекстах, использовались термины, связанные отношением ассоциации, найденные методом LSA на соответствующем шаге работы стенда.

Для оценки качества автоматически построенных тезаурусов выделенные термины и связи сравнивались с терминами и связями эталонного тезауруса в указанной предметной области. В качестве эталонного тезауруса использовался биомедицинский тезаурус MeSH (https://www.nlm.nih.gov/mesh/). Для оценки качества, в частности, оценивалась полнота и точность выделенных терминов и связей.

Большой интерес в экспериментах по автоматическому построению тезаурусов вызывает качество различных методов для выделения гипонимо-гиперонимических и синонимических связей, и то, насколько их качество отличается друг от друга. Поэтому в первую очередь мы провели эксперименты с тезаурусами, в которых для определения связей использовался какой-либо один из четырёх перечисленных выше стандартных методов. Для определения синонимов во всех указанных экспериментах использовался метод, основанный на расстоянии Левенштейна. Кроме того, в экспериментах с методом, основанном на использовании тезауруса WordNet, и методом, использующим количество информации терминов, эти же самые методы использовались и для выделения синонимов.

В результате проведенных экспериментов на стенде были вычислены некоторые статистические характеристики построенных тезаурусов, позволяющие оценить их общую структуру (см. первые четыре строки в таблице 1). Как видно из результатов, наибольшее количество выделенных терминов, наибольшее количество найденных гипонимо-гиперонимических связей и наибольшая связность графа относится к методу, основанному на использовании WordNet. Наибольшее количество синонимов было найдено с помощью метода, основанного на количестве информации терминов.

Далее мы оценили качество построенных тезаурусов в сравнении с эталонным тезаурусом MeSH. В частности, были вычислены точность и полнота выделения терминов и отношений. Результаты записаны в таблице 2.

Точность выделения терминов примерно одинаковая для всех методов и составляет около 38.5 %. Метод, использующий расстояние Левенштейна для выделения синонимов, показал высокую точность на уровне 76–89 %, но при этом он даёт довольно низкую полноту на уровне 11–12 %. Использование WordNet для нахождения синонимов также дало неплохие результаты. Несмотря на то, что точность уменьшилась до 57.4 %, этот показатель остается на достаточно хорошем уровне. Полно-

Таблица 1: Характеристики автоматически построенных тезаурусов

Table 1: Characteristics of automatically generated thesauri

Методы выделения		Количество выделенных				Количество	
гиперонимов	синонимов	терминов	гиперонимов	синонимов	ассоциаций	компон. связности	вершин в наиб. комп.
Extraction methods for		Quantity of extracted				Quantity of	
hypernyms	synonyms	terms	hypernyms	synonyms	associations	connected comp.	vertices in max. comp.
Лекс	Лев	2167	102	63	4918	86	1936
Морф	Лев	2090	248	46	4731	76	1909
КолИнф	КолИнф, Лев	2105	346	753	3883	84	1881
WordNet	WordNet, Лев	2406	2107	290	4375	35	2318
Гибрид	WordNet, Лев	2411	1570	312	4842	36	2323

Лекс — лексико-синтаксические шаблоны.
Морф — морфо-синтаксические правила.
КолИнф — метод, основанный на оценке количества информации.
WordNet — метод, основанный на использовании WordNet.
Лев — метод, использующий расстояние Левенштейна.
Гибрид — совместное применение методов Морф, КолИнф и WordNet.

та выделения синонимов увеличилась до 13.9 %. Метод, основанный на количестве информации, показал весьма слабые результаты по нахождению синонимов — точность и полнота не превышают уровня 15.5 %.

Качество выделения гипонимо-гиперонимических связей для всех методов оказалось довольно невысоким. Самым точным оказался метод, основанный на морфо-синтаксических правилах. Его точность составляет 23.1 %, что в несколько раз больше, чем точность всех остальных методов. Наибольшую полноту (23.8 %) показал метод, основанный на использовании WordNet, в то время как остальные методы показали значительно меньшее значение этого показателя. Примечательно, что в наших экспериментах метод, основанный на лексико-синтаксических шаблонах, показал наихудшие результаты и не нашел ни одной гипонимо-гиперонимической связи из тезауруса MeSH.

По результатам экспериментов со стандартными методами можно сделать несколько выводов. Метод, использующий WordNet, находит много правильных отношений, но его главный недостаток заключается в том, что тезаурус общего назначения не содержит в себе всех терминов из специализированной предметной области, и многие взаимоотношения не могут быть найдены таким способом. Методы, использующие морфо-синтаксические правила и лексико-синтаксические шаблоны, позволяют найти отношения, явно определенные в тексте, но их использование становится очень ограниченным в неструктурированных тестах и не позволяет найти множество других взамосвязанных терминов. Статистический метод находит большое количество различных отношений, но в то же время этот метод гораздо чаще, чем остальные, находит неверные отношения.

Таблица 2: Точность и полнота выделения терминов и семантических отношений при автоматическом построении тезауруса (в сравнении с тезаурусом MeSH)

Table 2: Precision and recall of extraction of terms and relations for case of automatic thesaurus generation (compared to the MeSH thesaurus)

Методы выдел. гиперонимов	Методы выдел. синонимов	Все термины	Синонимы		Гиперонимы	
		Точность	Точность	Полнота	Точность	Полнота
Extr. meth. for hypernyms	Extr. meth. for synonyms	All terms Precision	Synonyms Precision	Recall	Hypernyms Precision	Recall
Лекс	Лев	39.0	75.8	11.2	0.0	0.0
Морф	Лев	38.2	88.5	11.7	23.1	13.0
КолИнф	КолИнф, Лев	38.5	12.8	15.2	7.1	8.3
WordNet	WordNet, Лев	38.4	57.4	13.9	6.9	23.8
Гибрид	WordNet, Лев	38.4	57.4	15.7	8.3	17.9

Лекс — лексико-синтаксические шаблоны.
Морф — морфо-синтаксические правила.
КолИнф — метод, основанный на оценке количества информации.
WordNet — метод, основанный на использовании WordNet.
Лев — метод, использующий расстояние Левенштейна.
Гибрид — совместное применение методов Морф, КолИнф и WordNet.

4.3. Эксперименты на стенде: гибридный метод

Результаты экспериментов подтвердили, что стандартные методы хороши при выделении определенных типов отношений, а не для всех отношений сразу, что привело нас к идее гибридного метода, заключающегося в использовании комбинации стандартных методов выделения отношений.

В данном исследовании мы реализовали гибридный метод, объединяющий в себе большинство стандартных методов. По результатам экспериментов, практически все алгоритмы достаточно полезны при выделении различных типов связей за исключением лишь нескольких. В частности, мы выяснили, что метод, основанный на лексико-синтаксических шаблонах, показал отрицательные результаты в нахождении гипонимо-гиперонимических связей и не выделил ни одной правильной связи. Также метод, основанный на количестве информации, привел к значительному ухудшению точности в задаче выделения синонимов. Данные обстоятельства привели нас к идее того, чтобы не включать указанные шаги в гибридный метод. В результате работа гибридного метода была построена по следующему алгоритму:

1. Выделение терминов тезауруса.
2. Определение ассоциативных связей методом LSA.
3. Определение гипонимо-гиперонимических связей с помощью метода, основанного на использовании WordNet.
4. Определение гипонимо-гиперонимических связей с помощью метода, основанного на морфо-синтаксических правилах.
5. Определение синонимических связей с помощью метода, основанного на расстоянии Левенштейна.
6. Определение синонимических связей с помощью метода, основанного на использовании WordNet.

7. Определение гипонимо-гиперонимических связей с помощью метода, основанного на количестве информации терминов.
8. Фильтрация терминов, которые не имеют связей.

Следует отметить, что если для пары терминов на некотором этапе была выделена связь, то на следующих шагах алгоритма возможность выделения связи другого типа для этой же пары исключается.

Для построенного гибридного метода мы провели точно такие же эксперименты, как и для стандартных методов. Результаты экспериментов показаны в последней строке таблиц 1 и 2.

Из результатов данных экспериментов видно, что по сравнению со всеми другими методами гибридный метод выделяет наибольшее количество терминов и отношений между ними. Кроме того, тезаурус, построенный с помощью данного метода, имеет высокую связность.

В сравнении с тезаурусом MeSH было замечено, что гибридный метод находит больше совпавших терминов и отношений, чем стандартные методы, но количество несовпавших терминов и отношений также возросло. Точность выделения гипонимо-гиперонимических связей упала до 8.3 %, но полнота в среднем стала выше и оказалась равной 17.9 %. Точность выделения синонимов оказалась на одном уровне с точностью метода, основанного на использовании WordNet, и равняется 57.4 %. Полнота выделения синонимов возросла до 15.7 %, что является лучшим результатом среди всех тестируемых методов. Точность выделения терминов осталась такой же, как и прежде.

Заключение

В данной работе авторы проанализировали несколько методов выделения семантических отношений между терминами тезауруса, которые могут быть использованы в алгоритме полностью автоматической генерации узкоспециализированного тезауруса. Авторами был предложен стенд для подобной генерации и оценки тезауруса, использующий лучшие существующие статистические и семантические алгоритмы выделения терминов и отношений между ними.

На основе анализа результатов экспериментов можно выделить некоторые методические аспекты автоматической генерации тезауруса. Для полностью автоматического выделения терминов тезауруса лучше всего использовать алгоритмы выделения ключевых слов вида «обучение без учителя», так как они работают достаточно эффективно и при этом не требуют работы эксперта. В выделении гиперонимов достаточно хороши два метода, основанные на морфо-синтаксических правилах и тезаурусе WordNet. Первый работает с лучшей точностью, но выделяет мало отношений. Второй работает с лучшей полнотой, но не может выделять связи между специфическими терминами, которые отсутствуют в тезаурусах общего назначения. В выделении синонимов наилучшие результаты показывает алгоритм, считающий расстояние Левенштейна, однако он находит небольшое число отношений и не может выделять связи между неоднокоренными синонимами.

Для более эффективного анализа результатов авторами была предложена комплексная оценка итогового тезауруса, включающая в себя, кроме широко используе-

мых статистических мер точности и полноты, сравнивающих полученный тезаурус с эталонным, метрики связности графа тезауруса. По результатам всех экспериментов лучшую точность выделения иерархических связей показал морфо-синтаксический метод, однако он выделяет существенно меньше связей, чем статистический метод или метод, основанный на использовании WordNet. Лучшей точностью выделения синонимических связей обладает метод, использующий расстояние Левенштейна. Также в задаче выделения синонимов хорошие результаты показал метод, использующий WordNet. Отсюда можно сделать вывод, что существующие методы эффективны в извлечении определенных типов отношений, но недостаточно хороши для построения тезауруса в целом.

Для решения данной проблемы авторами была предложена идея гибридных методов, представляющих собой различные комбинации существующих алгоритмов выделения синонимических и иерархических отношений между терминами. Эксперименты показали: преимущество гибридных методов при выделении обоих типов связей заключается в том, что у них лучшая полнота, т. е. они находят больше правильных отношений, а также данные методы выделяют больше терминов и строят большее количество связей, тем самым повышая связность итогового тезауруса. Таким образом, если для поставленной задачи важна точность выделенных связей, то лучше использовать отдельные стандартные методы. Для обеспечения высокой связности тезауруса и хорошей полноты выделения связей более пригодными являются гибридные методы.

Хотя обнаруженный эффект повышения качества построения тезауруса может зависеть от используемого корпуса текстов или специфических особенностей предметной области, идея гибридных методов выглядит многообещающей в перспективе автоматической генерации узкоспециализированных тезаурусов, что является актуальной задачей в связи с необходимостью минимизировать работу эксперта над тезаурусом. Дальнейшие направления исследований в этой области могут состоять в изучении особенностей гибридных методов, создании подходов для автоматической оценки конкретных методов на конкретных корпусах документов и последующем синтезе гибридных методов для обеспечения надежных результатов.

Список литературы / References

[1] Aitchison J., Gilchrist A. and Bawden D., *Thesaurus construction and use: a practical manual*, Psychology Press, 2000, 230 pp.

[2] Лукашевич Н. В., Добров Б. В., “Проектирование лингвистических онтологий для информационных систем в широких предметных областях”, *Онтология проектирования*, **5**:1(15) (2015), 47–69; [Loukachevitch N. V., Dobrov B. V., “Developing Linguistic Ontologies in Broad Domains”, *Ontology of Designing*, **5**:1(15) (2015), 47–69, (in Russian).]

[3] Лукашевич Н. В., *Тезаурусы в задачах информационного поиска*, Издательство МГУ, М., 2011, 512 с.; [Lukashevich N. V., *Tezaurusy v zadachah informacionnogo poiska*, Izdatelstvo MGU, Moskow, 2011, 512 pp., (in Russian).]

[4] Astrakhantsev N. A., Turdakov D. Yu., “Automatic construction and enrichment of informal ontologies: A survey”, *Programming and computer software*, **39**:1 (2013), 34–42.

[5] Hasan K. S., Ng V., “Automatic Keyphrase Extraction: A Survey of the State of the Art”, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014, 1262–1273.

[6] Paramonov I. et al., "Thesaurus-Based Method of Increasing Text-via-Keyphrase Graph Connectivity During Keyphrase Extraction for e-Tourism Applications", *International Conference on Knowledge Engineering and the Semantic Web*, 2016, 129–141.

[7] Yang D., Powers D.Ṁ., "Automatic thesaurus construction", *Proceedings of the thirty-first Australasian conference on Computer science*, **74** (2008), 147–156.

[8] Mihalcea R., Tarau P., "TextRank: Bringing order into texts", *Proceedings of EMNLP*, 2004, 404–411.

[9] Liu Z. et al., "Automatic keyphrase extraction via topic decomposition", *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 2010, 366–376.

[10] Wiemer-Hastings P., Wiemer-Hastings K., Graesser A., "Latent semantic analysis", *Proceedings of the 16th international joint conference on Artificial intelligence*, 2004, 1–14.

[11] Lefever E., Van de Kauter M., Hoste V., "Evaluation of automatic hypernym extraction from technical corpora in English and Dutch", *9th International Conference on Language Resources and Evaluation (LREC)*, 2014, 490–497.

[12] Oakes M. P., "Using Hearst's Rules for the Automatic Acquisition of Hyponyms for Mining a Pharmaceutical Corpus", *RANLP Text Mining Workshop*, **5** (2005), 63–67.

[13] Noh S., Kim S., Jung C., "A Lightweight Program Similarity Detection Model using XML and Levenshtein Distance", *FECS*, 2006, 3–9.

[14] Мозжерина Е. С., "Автоматическое построение онтологии по коллекции текстовых документов", *Электронные библиотеки: Перспективные Методы и Технологии, Электронные коллекции–RCDL*, 2011, 293–298; [Mozzherina E. S., "Ehlektronnye biblioteki: Perspektivnye Metody i Tekhnologii, Ehlektronnye kollekcii–RCDL", 2011, 293–298, (in Russian).]

[15] Mittelu V. B., "Automatic Extraction of Patterns Displaying Hyponym-Hypernym Co-Occurrence from Corpora", *Proceedings of First Central European Student Conference in Linguistics*, 2006, 21, 8 pp.

Lagutina N. S., Lagutina K. V., Mamedov E. I., Paramonov I. V., "Methodological Aspects of Semantic Relationship Extraction for Automatic Thesaurus Generation", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 826–840.

DOI: 10.18255/1818-1015-2016-6-826-840

Abstract. The paper is devoted to analysis of methods for automatic generation of a specialized thesaurus. The main algorithm of generation consists of three stages: selection and preprocessing of a text corpus, recognition of thesaurus terms, and extraction of relations among terms. Our work is focused on exploring methods for semantic relation extraction. We developed a test bench that allow to test well-known algorithms for extraction of synonyms and hypernyms. These algorithms are based on different relation extraction techniques: lexico-syntactic patterns, morpho-syntactic rules, measurement of term information quantity, general-purpose thesaurus WordNet, and Levenstein distance. For analysis of the result thesaurus we proposed a complex assessment that includes the following metrics: precision of extracted terms, precision and recall of hierarchical and synonym relations, and characteristics of the thesaurus graph (the number of extracted terms and semantic relationships of different types, the number of connected components, and the number of vertices in the largest component). The proposed set of metrics allows to evaluate the quality of the thesaurus as a whole, reveal some drawbacks of standard relation extraction methods, and create more efficient hybrid methods that can generate thesauri with better characteristics than thesauri generated by using separate methods. In order to illustrate this fact, one of such hybrid methods is considered in the paper. It combines the best standard algorithms for hypernym and synonym extraction and generates a specialized medical thesaurus. The hybrid method leaves the thesaurus quality on the same level and finds more relations between terms than well-known algorithms.

Keywords: thesaurus, semantic relations, hybrid method, complex assessment, test bench

About the authors:
Nadezhda S. Lagutina, orcid.org/0000-0002-6137-8643, PhD,
P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl 150000, Russia, e-mail: lagutinans@rambler.ru

Ksenia V. Lagutina, orcid.org/0000-0002-1742-3240, student,
P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl 150000, Russia, e-mail: lagutinakv@mail.ru

Eldar I. Mamedov, orcid.org/0000-0003-1997-7084, intern researcher,
P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl 150000, Russia, e-mail: eldarmamedov90@gmail.com

Ilya V. Paramonov, orcid.org/0000-0003-3984-8423, PhD,
P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl 150000, Russia, e-mail: Ilya.Paramonov@fruct.org

Acknowledgments:
This work was supported by the grant of the President of Russian Federation for state support of young Russian scientists (project MK-5456.2016.9).

Моделирование и анализ информационных систем. Т. 23, № 6 (2016), с. 841–849
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 841–849

DOI: 10.18255/1818-1015-2016-6-841-849

УДК 517.9

Динамика системы из двух простейших автогенераторов с нелинейными финитными обратными связями

Кащенко А. А.

получена 1 сентября 2016

Аннотация. В данной работе рассматривается сингулярно возмущенная система двух дифференциальных уравнений с запаздыванием, которая моделирует два связанных автогенератора с нелинейной обратной связью. Функция обратной связи предполагается финитной, кусочно-непрерывной и сохраняющей знак. В работе доказывается существование релаксационных периодических решений и делается вывод о характере их устойчивости. С помощью специального метода большого параметра строится асимптотика всех решений данной системы с начальными условиями из некоторого класса. По данной асимптотике строится специальное отображение, которое в главном описывает динамику исходной модели. Показано, что при убывании коэффициента связи динамика существенно меняется: при значениях связи порядка единицы имеем устойчивое однородное периодическое решение, а при уменьшении параметра связи возникают более сложные динамические режимы, которые описываются специальным построенным в явном виде одномерным отображением. Удается показать, что при малых значениях связи при некоторых значениях параметров в исходной задаче сосуществуют несколько различных устойчивых релаксационных периодических режимов.

Ключевые слова: асимптотика, устойчивость, большой параметр, релаксационное колебание, периодическое решение

Для цитирования: Кащенко А. А., "Динамика системы из двух простейших автогенераторов с нелинейными финитными обратными связями", *Моделирование и анализ информационных систем*, **23**:6 (2016), 841–849.

Об авторах:
Кащенко Александра Андреевна, orcid.org/0000-0003-3823-9351, канд. физ.-мат. наук,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150003, Россия, e-mail: a.kashchenko@uniyar.ac.ru

Благодарности:
Работа выполнена при финансовой поддержке проекта 1875 госзадания на НИР №2014/258.

Введение

Одной из простейших моделей автогенератора с запаздывающей обратной связью является скалярное уравнение

$$\dot{u} + u = \lambda F(u(t - T)). \tag{1}$$

Параметр запаздывания T и коэффициент λ положительны, а $F(u)$ — некоторая нелинейная функция. Уравнения такого типа возникают во многих прикладных задачах [1–3]. В [1, 4, 5] отмечалось, что важное значение имеет рассмотрение таких нелинейностей $F(u)$, значения которых сосредоточены в некоторой ограниченной области. В связи с этим здесь предполагается, что функция $F(u)$ является финитной, то есть для некоторого $p > 0$ имеет место равенство

$$F(u) = \begin{cases} f(u), & |u| < p, \\ 0, & |u| \geq p. \end{cases} \tag{2}$$

Условие (2) не является ограничением при локальном (например, в малой окрестности какого-либо состояния равновесия) анализе (1), а при нелокальном исследовании уравнения (1) оно играет ключевую роль. Для того, чтобы подчеркнуть важность этого условия, будем предполагать, что параметр λ является достаточно большим:

$$\lambda \gg 1. \tag{3}$$

В [6] было показано, что при некоторых условиях типа невырожденности динамические свойства (1) довольно простые: имеется одно или два устойчивых релаксационных периодических решения и найдена их асимптотика при $\lambda \to \infty$.

В настоящей работе исследуется поведение решений системы из двух связанных уравнений вида (1) при условиях (2) и (3):

$$\begin{cases} \dot{u_1} + u_1 = \lambda F(u_1(t-T)) + \gamma(u_2 - u_1), \\ \dot{u_2} + u_2 = \lambda F(u_2(t-T)) + \gamma(u_1 - u_2). \end{cases} \tag{4}$$

Здесь функция $F(u)$ удовлетворяет условию (2), а кусочно непрерывная функция $f(u)$ отделена от нуля на отрезке $[-p, p]$ и сохраняет знак на интервале $(-p, p)$. Параметр связи γ положителен и мал при $\lambda \to \infty$:

$$\gamma = \gamma_1 (\ln \lambda)^{-1}, \quad (\gamma_1 > 0). \tag{5}$$

Исследование базируется на использовании специальной методики, разработанной в [6, 7]. Суть ее состоит в следующем. В фазовом пространстве $C_{[-T,0]}(R^2)$ системы (4) выделяются некоторые множества $S(x)$, зависящие от параметра x. Затем исследуется асимптотика при $\lambda \to \infty$ всех решений (4) с начальными условиями из $S(x)$. Удается установить, что через некоторый отрезок времени каждое из рассматриваемых решений попадает в множество $S(\bar{x})$, где для $\bar{x}$ получено асимптотическое при $\lambda \to \infty$ представление вида $\bar{x} = \phi(x) + o(1)$. После этого строится оператор последования $\Pi : \Pi S(x) \subset S(\bar{x})$. Итерации этого оператора определяются в главном динамикой одномерного отображения $\bar{x} = \phi(x)$. Используя фундаментальные результаты А.Н. Шарковского [8], приходим к выводу о том, что для системы (4) могут быть характерны сложные нерегулярные колебания.

1. Динамика системы (4) при условии $\gamma = \gamma_1 (\ln \lambda)^{-1}$

Рассмотрим два случая:
a) $f(u) > 0$ при $u \in (-p, p)$;
b) $f(u) < 0$ при $u \in (-p, p)$.

Определим множество $S(x) \subset C_{[-T,0]}(R^2)$ начальных условий для решений (4). Введем несколько обозначений. Пусть x — некоторый параметр, для которого выполнено условие $|x| \geq 1$. Пусть k — параметр, принимающий значения 1 и -1. Еще один параметр m принимает только два значения: 1 и 2, причем считаем, что $m+1$ равно 2, если $m = 1$ и $m+1$ равно 1, если $m = 2$. Фиксируем произвольно значения x, k и m. Пусть при $s \in [-T, 0]$ выполнены условия

$$\begin{cases} |u_m(s)| \geq p, & u_m(0) = kp, \\ |u_{m+1}(s)| \geq p, & u_{m+1}(0) = xp. \end{cases} \tag{6}$$

Множество функций, удовлетворяющих условиям (6), обозначим через $S(x)$. В случае a) мы рассматриваем только положительные начальные условия, а в случае b) только отрицательные.

Отметим одно важное обстоятельство. При увеличении времени t для решений системы (4) последовательно чередуются участки двух типов. В первом из них нелинейность не тождественно нулевая, а во втором — нулевая, и тогда функции u_1 и u_2 являются решением линейной системы обыкновенных дифференциальных уравнений

$$\dot{u_1} + u_1 = \gamma_1(\ln\lambda)^{-1}[u_2 - u_1], \quad \dot{u_2} + u_2 = \gamma_1(\ln\lambda)^{-1}[u_1 - u_2]. \tag{7}$$

Возможность полного асимптотического исследования решений (4) обусловлена тем, что для решений с начальными условиями из выбранного класса длительность участков первого типа имеет порядок $O(1)$ (при $\lambda \to \infty$), а длительность участков второго типа — асимптотически большая — порядка $O(\ln\lambda)$. Отметим еще, что на участках второго типа происходит убывание функций $|u_1(t)|$ и $|u_2(t)|$ от асимптотически больших значений до значений порядка $O(1)$.

Ниже понадобится формула при любых t и τ для решений системы (7):

$$u_1(t) = \frac{1}{2}\Big[u_1(\tau) + u_2(\tau) + (u_1(\tau) - u_2(\tau))\ \exp\Big(\frac{2\gamma_1}{\ln\lambda}(\tau - t)\Big)\Big]\ \exp(\tau - t), \tag{8}$$

$$u_2(t) = \frac{1}{2}\Big[u_1(\tau) + u_2(\tau) + (u_2(\tau) - u_1(\tau))\ \exp\Big(\frac{2\gamma_1}{\ln\lambda}(\tau - t)\Big)\Big]\ \exp(\tau - t). \tag{9}$$

Поскольку при каждом $x = x_0$ функции из множества $S(x_0)$ при $s \in [-T, 0]$ удовлетворяют условиям (6), то на отрезке $t \in [0, T]$ система (4) будет иметь вид (7) и все решения с начальными условиями из $S(x_0)$ совпадут на данном отрезке, а значит, совпадут и при дальнейших значениях t.

Из (8), (9) получаем, что для решений $u_m(t)$ и $u_{m+1}(t)$ с начальными условиями (6) на отрезке $t \in [0, T]$ верны асимптотические равенства

$$u_m(t) = kp(1 + o(1))\exp(-t),$$

$$u_{m+1}(t) = xp(1 + o(1))\exp(-t).$$

Пусть затем $t \in [T, 2T]$. Предположим, что выполнено неравенство

$$|x|\exp(-T) > 1. \tag{10}$$

Тогда приходим к равенствам

$$u_m(t) = \lambda[g_k(t) + o(1)], \tag{11}$$

$$u_{m+1}(t) = \gamma_1 \lambda (\ln \lambda)^{-1} [g_{k,1}(t) + o(1)]. \tag{12}$$

Здесь

$$g_k(t) = \int_T^t \exp(s-t) f(kp\ \exp(T-s)) ds, \tag{13}$$

$$g_{k,1}(t) = \int_T^t \exp(s-t) g_k(s) ds. \tag{14}$$

Пусть теперь $|x| \exp(-T) < 1$. Тогда на отрезке $[0, T]$ существует корень $\tau = \tau(x)$ уравнения

$$|x| \exp(-\tau) = 1.$$

Таким образом, $\tau(x) = \ln |x|$. Для $u_m(t)$ на рассматриваемом отрезке $[T, 2T]$ остается верной формула (11), а формула (12) верна только при $t \in [T, T + \tau(x)]$. Если же $t \in (T + \tau(x), 2T]$, то для $u_{m+1}(t)$ переходим к равенству

$$u_{m+1}(t) = \lambda [g(t, x) + o(1)],$$

где

$$g(t, x) = \int_{T+\tau(x)}^t \exp(s-t) f(xp\ \exp(T-s)) ds.$$

Заметим, что выполнены неравенства

$$g_k(t) \neq 0,\ g_{k,1}(t) \neq 0\ \forall\ t \in (T, 2T], \tag{15}$$

$$g(t, x) \neq 0\ \forall\ t \in (T + \tau(x), 2T] \tag{16}$$

и что функции $u_m(t)$ и $u_{m+1}(t)$ на отрезке $[T, 2T]$ имеют тот же знак, что и на отрезке $[0, T]$.

При $t = 2T$ получаем

$$u_m(2T) = \lambda [g_k(2T) + o(1)], \tag{17}$$

$$u_{m+1}(2T) = \begin{cases} \gamma_1 \lambda (\ln \lambda)^{-1} [g_{k,1}(2T) + o(1)], & \text{при } |x| \exp(-T) > 1, \\ \lambda [g(2T, x) + o(1)], & \text{при } |x| \exp(-T) < 1. \end{cases} \tag{18}$$

Пусть $t \in [2T, 3T]$. Тогда решение системы (4) есть общее решение однородной системы (7) плюс частное решение неоднородной системы (4) (так как $t - T$ принадлежит отрезку $[T, 2T]$, то выражения $\lambda F(u_j(t - T))$ $(j = 1, 2)$ в системе (4) — известные функции). При условиях (15) и (16) порядок частного решения на данном отрезке времени будет не больше $O(\ln \lambda)$. Поскольку общее решение линейной системы будет по порядку не меньше $O(\lambda / \ln \lambda)$, то именно оно будет давать главное приближение решения при $t \in [2T, 3T]$. Следовательно, функции $u_m(t)$ и $u_{m+1}(t)$ на всем отрезке $t \in [2T, 3T]$ будут по модулю больше p. Тогда при $t > 3T$ пока выполнены неравенства $|u_m(t)| > p$ и $|u_{m+1}(t)| > p$, система (4) будет иметь вид (7).

Найдем главное приближение решений при условиях $t > 2T$, $|u_m(t)| > p$ и $|u_{m+1}(t)| > p$. Введем обозначение $\tilde{t} = t - 2T$.

Пусть $|x|\exp(-T) > 1$. Заметим, что из условий на функцию $f(u)$ следует, что

$$g_k(2T) \cdot g_{k,1}(2T) > 0. \tag{19}$$

Тогда будут верны формулы

$$u_m(t) = \frac{\lambda}{2}\exp(-\tilde{t})\ \Big[g_k(2T)\Big(1 + \exp\Big(\frac{-2\gamma_1\tilde{t}}{\ln\lambda}\Big)\Big) + o(1)\Big], \tag{20}$$

$$u_{m+1}(t) = \begin{cases} \frac{\gamma_1\lambda}{\ln\lambda}\exp(-\tilde{t})[g_k(2T)\tilde{t} + g_{k,1}(2T) + o(1)], & \text{при } \tilde{t} = o(\ln\lambda), \\ \frac{\lambda}{2}\exp(-\tilde{t})\big[g_k(2T)\big(1 - \exp\big(\frac{-2\gamma_1\tilde{t}}{\ln\lambda}\big)\big) + o(1)\big], & \text{при } \tilde{t} = O(\ln\lambda). \end{cases} \tag{21}$$

Если же $|x|\exp(-T) < 1$, то функции $u_m(t)$ и $u_{m+1}(t)$ будут иметь вид

$$u_m(t) = \frac{\lambda}{2}\exp(-\tilde{t})\ [g_k(2T)(1 + \exp(-2\gamma_1\tilde{t}(\ln\lambda)^{-1})) + \\ + g(2T,x)(1 - \exp(-2\gamma_1\tilde{t}(\ln\lambda)^{-1})) + o(1)], \tag{22}$$

$$u_{m+1}(t) = \frac{\lambda}{2}\exp(-\tilde{t})\ [g_k(2T)(1 - \exp(-2\gamma_1\tilde{t}(\ln\lambda)^{-1})) + \\ + g(2T,x)(1 + \exp(-2\gamma_1\tilde{t}(\ln\lambda)^{-1})) + o(1)]. \tag{23}$$

Правые части в (20), (21), (22) и (23) стремятся к нулю при $\tilde{t} \to \infty$, поэтому при $t > 2T$ найдутся такие t_m и t_{m+1}, что $|u_m(t_m)| = p$ и $|u_{m+1}(t_{m+1})| = p$. Пусть $t_0(\lambda) = \min(t_m, t_{m+1})$. Отметим, что из неравенств (15), (16) и (19) следует, что $t_0(\lambda) = \ln\lambda \cdot (1 + o(1))$.

Положим

$$R(x) = \begin{cases} (1 + \exp(-2\gamma_1)) \times (1 - \exp(-2\gamma_1))^{-1}, & \text{при } |x|e^{-T} > 1, \\ \dfrac{g_k(2T)(1 + \exp(-2\gamma_1)) + g(2T,x)(1 - \exp(-2\gamma_1))}{g_k(2T)(1 - \exp(-2\gamma_1)) + g(2T,x)(1 + \exp(-2\gamma_1))}, & \text{при } |x|e^{-T} < 1. \end{cases}$$

В качестве значения $\bar{m}$ выберем такое из чисел m и $m+1$, для которого выполнено соотношение $t_{\bar{m}} = t_0(\lambda)$. Тем самым $u_{\bar{m}}(t_0(\lambda)) = \bar{k}p$ и $u_{\overline{m+1}}(t_0(\lambda)) = \bar{x}p$. Заметим, что в силу условий на функцию $f(u)$ и выбора начальных условий имеем $\bar{k} = k$. С помощью функции $R(x)$ величины $\bar{m}$ и $\bar{x}$ определяются равенствами

$$\bar{m} = \begin{cases} m, & \text{при } |R(x)| < 1, \\ m+1, & \text{при } |R(x)| > 1; \end{cases} \tag{24}$$

$$\bar{x} = \begin{cases} kR^{-1}(x), & \text{при } |R(x)| < 1, \\ kR(x), & \text{при } |R(x)| > 1. \end{cases} \tag{25}$$

Таким образом, начиная с $t = t_0(\lambda)$, вернулись к исходной ситуации с заменой m на $\bar{m}$ и x на $\bar{x}$ согласно (24), (25).

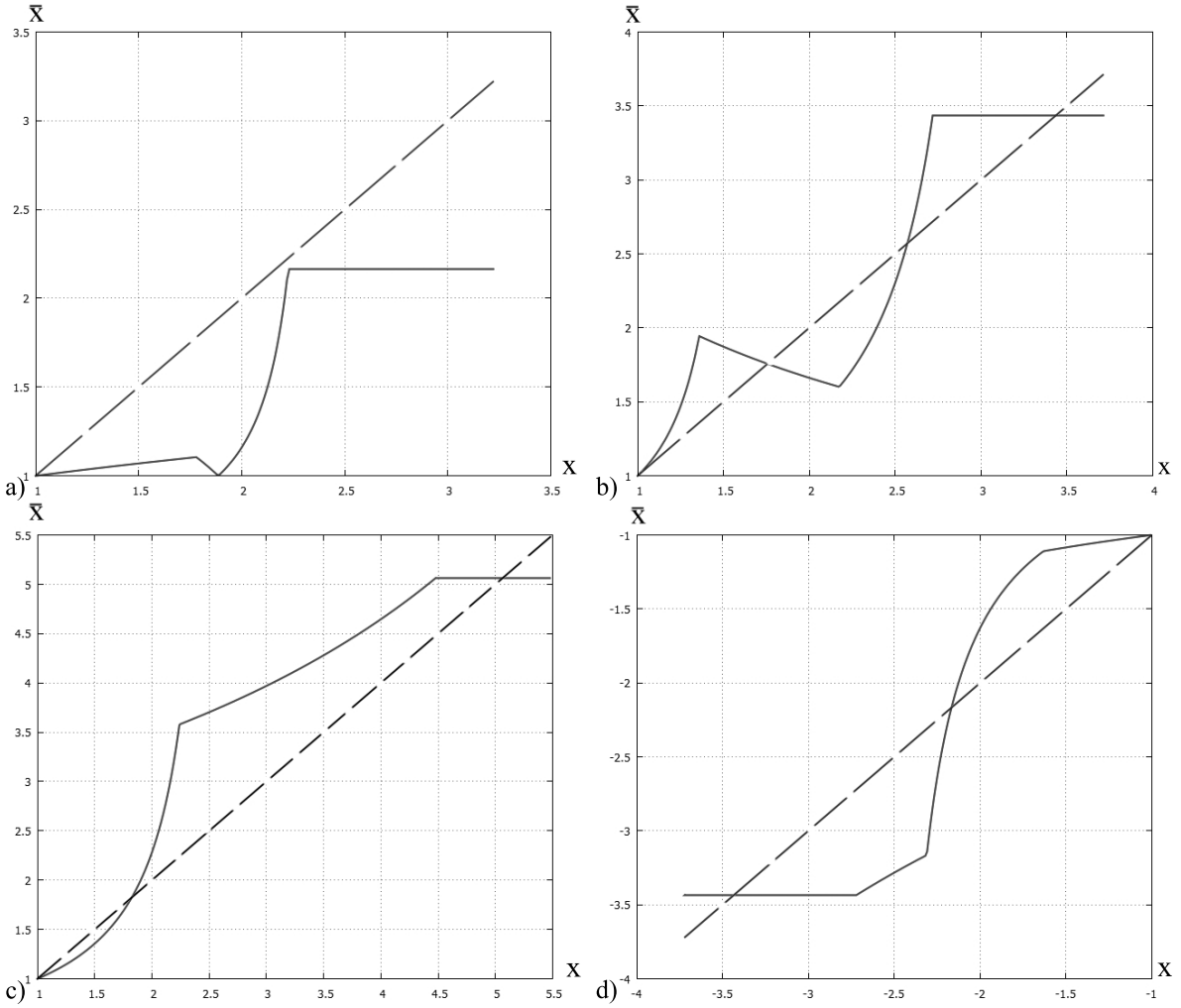


Рис. 1. Примеры графиков отображения (25): a) – c) – случай $f(u) > 0$; d) – случай $f(u) < 0$

Fig. 1. Some plots of mapping (25): a) – c) – case $f(u) > 0$; d) – case $f(u) < 0$

На рисунке 1 изображены некоторые примеры графиков отображения (25) и вспомогательный график $\bar{x} = x$ (пунктирной линией). Несложно доказать, что при любых значениях параметров у отображения (25) найдется хотя бы одно устойчивое состояние равновесия. Как видно из рисунка 1, при некоторых значениях параметров (случаи (b) – (d)) у отображения (25) могут сосуществовать несколько устойчивых неподвижных точек, в том числе могут сосуществовать два устойчивых состояния равновесия, соответствующие неоднородным периодическим режимам исходной задачи (случай (b)). Здесь неоднородными мы называем режимы, для которых выполняется условие $u_1(t) \neq u_2(t)$.

Отдельно остановимся на вопросах устойчивости таких периодических решений (4), о существовании которых говорится выше. Все они имеют ярко выраженную релаксационную структуру: период у них порядка $\ln \lambda$ (при $\lambda \to \infty$); в течение относительно короткого отрезка времени (порядка $O(1)$) соответствующие функции могут совершать скачок от величины порядка 1 до значений порядка λ, а остальное

(на отрезке длины периода) время решения являются экспоненциально убывающими по закону линейной системы (4) при $F \equiv 0$. Для гладких функций F методика оценок мультипликаторов, развитая в [9], позволяет определить их асимптотику. Просто устанавливается, что все, кроме двух, мультипликаторов стремятся к нулю при $\lambda \to \infty$. Один мультипликатор всегда равен 1. Таким образом для ответа на вопрос об устойчивости необходимо исследовать поведение одного оставшегося мультипликатора. Этот мультипликатор совпадает (в главном) с мультипликатором цикла приводимых одномерных отображений. Тем самым устойчивость периодических решений (4) та же, что и устойчивость соответствующих траекторий одномерных отображений. Если же кусочно-непрерывная функция F не является гладкой, то ее можно сколь угодно близко по норме $C_{[-p,p]}$ приблизить некоторой гладкой функцией (обозначим ее F_*). Так как для фиксированного $x_0 \in \mathbb{R}$ все решения системы вида (4) с начальными условиями из множества $S(x_0)$ совпадут на отрезке $t \in [0, T]$ (и, как следствие, при всех больших t), то можно установить взаимнооднозначное соответствие (по значению x) между периодическими решениями системы (4) и системы вида (4), где функция F заменена гладкой функцией F_*. Тогда при достаточно хорошей степени сглаживания получаем, что устойчивость соответствующих решений в этих системах одинакова. Данные построения довольно громоздкие, поэтому ограничимся здесь кратким описанием итоговых выводов.

Сформулируем итоговые результаты.

Теорема 1. *Пусть отображение (24), (25) имеет цикл $(m_i; x_i)$ периода n. Тогда система (4) имеет периодическое решение $(u_m^n(t), u_{m+1}^n(t))$ той же устойчивости, параметры которого определяются значениями $(m_i; x_i)$ соответствующего цикла.*

Напомним, что однородным периодическим решением $(u_1(t), u_2(t))$ (4) называют такое, для которого $u_1(t) \equiv u_2(t) = u_0(t)$. Из теоремы 1 и того, что при достаточно больших значениях параметра γ_1 у отображения (25) есть устойчивая неподвижная точка $\bar{x} = 1$ (в случае положительной функции $f(u)$) или $\bar{x} = -1$ (в случае отрицательной функции $f(u)$), получаем следующий результат.

Теорема 2. *Найдутся такие $\gamma_{1_0} > 0$ и $\lambda_0 > 0$, что при $\gamma_1 \geq \gamma_{1_0}$ и $\lambda \geq \lambda_0$ однородное периодическое решение системы (4) орбитально устойчиво.*

Выводы

Асимптотическими методами исследована нелокальная динамика систем из двух связанных автогенераторов с нелинейной финитной запаздывающей обратной связью. Показано, что при убывании коэффициента связи динамика существенно меняется. Сначала при $\gamma \backsim 1$ имеем устойчивое однородное $(u_1 \equiv u_2)$ периодическое решение. При уменьшении γ в диапазоне $\backsim const(\ln \lambda)^{-1}$ возникают более сложные динамические режимы, которые описываются специальным построенным в явном виде одномерным отображением.

Разработанные аналитические методы позволяют получить асимптотические формулы для всех исследуемых решений.

Список литературы / References

[1] Дмитриев А. С., Кислов В. Я., *Стохастические колебания в радиотехнике*, Наука, Москва, 1989; [Dmitriev A. S., Kislov V. Ya., *Stokhasticheskie kolebaniya v radiotekhnike*, Nauka, Moskva, 1989, (in Russian).]

[2] Рождественский В. В., Стручков И. Н., "Переходный хаос в автогенераторе стохастических колебаний с жестким возбуждением и четной нелинейностью", *ЖТФ*, **62**:10 (1992), 102–110; [Rozhdestvenskiy V. V., Struchkov I. N., "Perekhodnyy khaos v avtogeneratore stokhasticheskikh kolebaniy s zhestkim vozbuzhdeniem i chetnoy nelineynostyu", *Journal of technical physics*, **62**:10 (1992), 102–110, (in Russian).]

[3] Стручков И. Н., "Переходный хаос в апериодическом осцилляторе с запаздыванием", *Радиотехника и электроника*, **38**:3 (1993), 454–463; [Struchkov I. N., "Perekhodnyy khaos v aperiodicheskom ostsillyatore s zapazdyvaniem", *Radiotekhnika i elektronika*, **38**:3 (1993), 454–463, (in Russian).]

[4] Дмитриев А. С, Кащенко С. А., "Динамика генератора с запаздывающей обратной связью и низкодобротным фильтром второго порядка", *Радиотехника и электроника*, **34**:12 (1989), 24–39; [Dmitriev A. S., Kaschenko S. A., "Dinamika generatora s zapazdyvayushchey obratnoy svyazyu i nizkodobrotnym filtrom vtorogo poryadka", *Radiotekhnika i elektronika*, **34**:12 (1989), 24–39, (in Russian).]

[5] Дмитриев А. С, Кащенко С. А., "Асимптотика нерегулярных колебаний в модели автогенератора с запаздывающей обратной связью", *Докл. РАН*, **328**:2 (1993), 174–177; English transl.: Dmitriev A. S., Kaschenko S. A., "Asymptotics of irregular oscillations in a self-sustained oscillator model with delayed feedback", *Doklady Mathematics*, **328** (1993), 157–160.

[6] Кащенко С. А., "Асимптотика релаксационных колебаний в системах дифференциально-разностных уравнений с финитной нелинейностью. I", *Дифференциальные уравнения*, **31**:8 (1995), 1330–1339; [Kaschenko S. A., "Asimptotika relaksatsionnykh kolebaniy v sistemakh differentsialno-raznostnykh uravneniy s finitnoy nelineynostyu. I", *Differentsialnye uravneniya*, **31**:8 (1995), 1330–1339, (in Russian).]

[7] Кащенко С. А., "Асимптотика релаксационных колебаний в системах дифференциально-разностных уравнений с финитной нелинейностью. II", *Дифференциальные уравнения*, **31**:12 (1995), 1968–1976; [Kaschenko S. A., "Asimptotika relaksatsionnykh kolebaniy v sistemakh differentsialno-raznostnykh uravneniy s finitnoy nelineynostyu. II", *Differentsialnye uravneniya*, **31**:12 (1995), 1968–1976, (in Russian).]

[8] Шарковский А. Н., Майстренко Ю. Л., Романенко Е. Ю., *Разностные уравнения и их приложения*, Наукова думка, Киев, 1986; English transl.: Sharkovsky A. N., Maistrenko Yu. L., and Romanenko E. Yu., *Difference equations and their applications*, Springer Science & Business Media, 2012.

[9] Кащенко С. А., "Релаксационные колебания в системе с запаздываниями, моделирующей задачу «хищник–жертва»", *Моделирование и анализ информационных систем*, **20**:1 (2013), 52–98; [Kaschenko S. A., "Relaxation Oscillations in a System with Delays Modeling the Predator-Prey Problem", *Modeling and Analysis of Information Systems*, **20**:1 (2013), 52–98, (in Russian).]

Kashchenko A. A., "Dynamics of a System of Two Simplest Oscillators with Finite Non-linear Feedbacks", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 841–849.

DOI: 10.18255/1818-1015-2016-6-841-849

Abstract. In this paper, we consider a singularly perturbed system of two differential equations with delay which simulates two coupled oscillators with nonlinear feedback. Feedback function is assumed to be finite, piecewise continuous, and with a constant sign. In this paper, we prove the existence of relaxation periodic solutions and make conclusion about their stability. With the help of the special

method of a large parameter we construct asymptotics of the solutions with the initial conditions of a certain class. On this asymptotics we build a special mapping, which in the main describes the dynamics of the original model. It is shown that the dynamics changes significantly with the decreasing of coupling coefficient: we have a stable homogeneous periodic solution if the coupling coefficient is of unity order, and with decreasing the coupling coefficient the dynamics become more complex, and it is described by a special mapping. It was shown that for small values of the coupling under certain values of the parameters several different stable relaxation periodic regimes coexist in the original problem.

Keywords: asymptotics, stability, large parameter, relaxation oscillation, periodic solution

About the authors:
Aleksandra A. Kashchenko, orcid.org/0000-0003-3823-9351, PhD,
P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl 150003, Russia, e-mail: a.kashchenko@uniyar.ac.ru

Acknowledgments:
This work was supported by the Ministry of Education and Science of the Russian Federation (project no. 2014/258-1875).

Моделирование и анализ информационных систем. Т. 23, №6 (2016), с. 850–859
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 850–859

DOI: 10.18255/1818-1015-2016-6-850-859

УДК 517.9

Устойчивые циклы и торы системы из трех и четырех диффузионно связанных осцилляторов

Марушкина Е.А.

получена 15 сентября 2016

Аннотация. Рассматриваются цепочки идентичных диффузионно слабо связанных колебательных систем с различными условиями связи на границе цепочки. Предполагается, что с каждым из взаимодействующих осцилляторов происходит бифуркация Андронова – Хопфа, а коэффициент связи пропорционален величине надкритичности. В этой ситуации на устойчивом интегральном многообразии системы построена нормальная форма, для которой в случае трех взаимодействующих осцилляторов удается проанализировать простейшие состояния равновесия и их фазовые перестройки. При изменении параметра связи для однородного состояния равновесия, соответствующего однородному циклу задачи, возможны два случая, в первом из которых оно теряет устойчивость с появлением двух устойчивых неоднородных состояний, а во втором с ним сливаются два неустойчивых неоднородных состояния и отбирают у него устойчивость. Для состояния равновесия, соответствующего колебаниям осцилляторов в противофазе, также можно выделить два случая. В первом из них это состояние равновесия становится устойчивым в результате стягивания к нему устойчивого предельного цикла системы (бифуркация Андронова – Хопфа), а во втором случае оно становится устойчивым после ответвления от него неустойчивого предельного цикла. В случае, когда число осцилляторов в цепочке равно четырем, проанализирована система разностей фаз осцилляторов, получающаяся при достаточно малом коэффициенте связи.

Ключевые слова: нормальная форма, автоколебания, автогенераторы, бифуркация, инвариантный тор

Для цитирования: Марушкина Е.А., "Устойчивые циклы и торы системы из трех и четырех диффузионно связанных осцилляторов ", *Моделирование и анализ информационных систем*, **23**:6 (2016), 850–859.

Об авторах:
Марушкина Елена Александровна, канд. физ.-мат. наук, научный сотрудник Лаборатории дискретной и вычислительной геометрии им. Б.Н. Делоне,
Ярославский государственный университет им. П.Г. Демидова, ул. Советская, 14, г. Ярославль 150003, Россия, e-mail: marushkina-ea@yandex.ru

Благодарности:
Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 16-31-60039 мол_а_дк.

1. Постановка задачи

Рассматриваются разностные аппроксимации краевой задачи (см. [1–5])

$$\frac{\partial u}{\partial t} = \mu D \Delta u + \Phi(u), \tag{1}$$

$$\left.\frac{\partial u}{\partial \nu}\right|_{\Gamma} = 0, \tag{2}$$

представляющей собой систему типа «реакция-диффузия» и часто встречающейся в физических и биологических приложениях. В системе (1) вектор-функция $u(t,x) \in \mathbb{R}^m$ определена в области Ω с достаточно гладкой границей Γ, D — $m \times m$ диагональная матрица с положительными элементами, Δ — оператор Лапласа, μ — положительный параметр, $\partial u/\partial \nu$ — производная по направлению внешней нормали. Фазовым пространством задачи (1), (2) считаем $\overset{\circ}{W}{}_2^2(\Omega;\mathbb{R}^m)$, где $\overset{\circ}{W}{}_2^2$ — соболевское пространство функций, удовлетворяющих нулевым граничным условиям Неймана. Кроме того, полагаем, что система обыкновенных дифференциальных уравнений в $\mathbb{R}^m$

$$\dot{u} = \Phi(u) \tag{3}$$

имеет единственный аттрактор — экспоненциально орбитально устойчивый цикл.

Предположим далее, что область Ω одномерная и представляет собой отрезок $[0,1]$, а наличие у системы (3) экспоненциально орбитально устойчивого цикла обусловлено происходящей в ней бифуркацией Андронова – Хопфа. Тогда после замены в этой краевой задаче второй частной производной по пространственной переменной на ее разностный аналог приходим к системе диффузионно связанных осцилляторов, связь в которой будем считать слабой

$$\dot{u}_j = \varepsilon D(u_{j-1} - 2u_j + u_{j+1}) + (A_0 + \varepsilon A_1)u_j + F_2(u_j,u_j) + F_3(u_j,u_j,u_j) + O(\| u_j \|^4), \tag{4}$$

где $u_j(t) \in \mathbb{R}^m$, $m \geq 2$, $j = 1,\dots,n$, с условиями на границах цепочки

$$u_0 = u_2, \ u_{n+1} = u_{n-1}. \tag{5}$$

Будем, кроме того, предполагать, что $m \times m$ матрица A_0 имеет пару чисто мнимых собственных чисел $\pm i\omega$, причем остальные ее собственные числа лежат в левой комплексной полуплоскости. Для $m \times m$ матрицы A_1 и квадратичной и кубической нелинейностей $F_2(u,u)$ и $F_3(u,u,u)$ будем считать выполненными условия бифуркации Андронова – Хопфа, то есть при достаточно малых значениях параметра $0 < \varepsilon \ll 1$ у каждого отдельного осциллятора от нулевого состояния равновесия ответвляется орбитально устойчивый цикл. Обозначим a собственный вектор матрицы A_0 такой, что $A_0 a = i\omega a$, соответственно b — удовлетворяет сопряженной системе $A^T b = -i\omega b$, причем $(a,b) = 1$. Величины

$$\varphi_0 + i\psi_0 = (A_1 a, b), \tag{6}$$

$$d_0 + ic_0 = \big(F_3(\bar{a},a,a) + F_3(a,\bar{a},a) + F_3(a,a,\bar{a}) + \\ + F_2(a,w_0) + F_2(w_0,a) + F_2(w_1,\bar{a}) + F_2(\bar{a},w_1), b\big), \tag{7}$$

где $w_0 = -A_0^{-1}\left(F_2(a,\bar{a}) + F_2(\bar{a},a)\right)$, $w_1 = (2i\omega E - A_0)^{-1}F_2(a,a)$, позволяют сформулировать условие возникновения указанной фазовой перестройки (см., например, [1,6]). Для рождения цикла необходимо, чтобы для величин φ_0 и d_0 в формулах (6), (7) выполнялись неравенства

$$\varphi_0 > 0, \quad d_0 < 0. \tag{8}$$

Рассмотрим задачу об устойчивых режимах, возникающих у системы (4) при изменении матрицы связи D.

Выполним в (4) стандартную замену асимптотического метода Крылова–Боголюбова–Митропольского (нами использовался алгоритм из [6])

$$u_j = \sqrt{\varepsilon}u_{0,j}(t,s) + \varepsilon u_{1,j}(t,s) + \varepsilon^{3/2}u_{2,j}(t,s) + \dots, \quad j = 1,\dots,n, \tag{9}$$

где $s = \varepsilon t$ — медленное время,

$$u_{0,j}(t,s) = z_j(s)\exp(i\omega t)a + \bar{z}_j(s)\exp(-i\omega\tau_j)\bar{a}, \tag{10}$$

функции $u_{k,j}(t,s)$ являются векторными тригонометрическими полиномами аргумента ωt. В результате этой замены на третьем шаге выполнения алгоритма после отбрасывания слагаемых высшего порядка малости система (4) преобразуется к виду

$$z_j' = (\varphi_0 + i\psi_0)z_j + (d_0 + ic_0)|z_j|^2 z_j + \varkappa\exp(i\delta)(z_{j-1} - 2z_j + z_{j+1}), \quad j = 1,\dots,n, \tag{11}$$

где $z_0 = z_2$, $z_{n-1} = z_{n+1}$, $(Da, b) = \varkappa\exp(i\delta)$, штрихом обозначена производная по s.

Полученную систему комплексных амплитуд будем рассматривать при $n = 3$ и $n = 4$.

2. Синхронные режимы амплитудной системы при $n = 3$

Будем называть (см. [1, 6]) *автомодельным циклом* системы (11) ее периодическое решение вида

$$z_j(s) = z_j^0\exp(i\sigma s)\,, \quad j = 1,\dots,n\,, \tag{12}$$

где $z_j^0 \in \mathbb{C}$, $j = 1, 2$ и $\sigma \in \mathbb{R}$ – некоторые постоянные. В свою очередь *двумерным автомодельным тором* системы (11) будем называть ее квазипериодическое решение вида

$$z_j(s) = z_j^0(s)\exp(i\sigma s)\,, \quad j = 1,\dots,n\,, \tag{13}$$

где σ — вещественная постоянная, а комплекснозначные функции $z_j^0(s)$, $j = 1,\dots,n$ периодичны по s с некоторым периодом $T_0 > 0$. Предполагаем, что решение (13) не приводится к виду (12).

Для автомодельных циклов и торов системы (11) выполнена следующая стандартная теорема о соответствии (см., например, [6, 8]).

Теорема 1. *Предположим, что система (11) имеет некоторый автомодельный цикл (тор), экспоненциально орбитально устойчивый или дихотомичный. Тогда найдется такое достаточно малое $\varepsilon_0 > 0$, что при всех $0 < \varepsilon \le \varepsilon_0$ исходная система (4) имеет цикл (двумерный инвариантный тор) с теми же свойствами устойчивости, асимптотика которого задается формулами (9), (10) с учетом равенств (12) или (13) соответственно.*

Впервые система (11) была выписана и исследована для двух уравнений Хатчинсона в [9], затем в работах [10–13] для этой системы были выписаны условия

существования и устойчивости синхронного цикла и цикла, соответствующего колебаниям в противофазе. Общие результаты для автомодельных циклов и торов двумерной системы были получены в статье [1].

Рассмотрим далее случай трех осцилляторов $n = 3$, тогда для упрощения анализа системы (11) выполним в ней полярную замену (см. [4,5,7])

$$z_j(s) = \sqrt{-\frac{\mathrm{Re}(A_1 a, b)}{d_0}}\xi_j(s)\exp(i\tau_j(s)), \quad j = 1,2,3 \tag{14}$$

и нормирующую замену времени $\mathrm{Re}(A_1 a, b)s \to s$, в результате имеем следующие уравнения:

$$\begin{aligned}
\xi_1' &= 2\varkappa\xi_2\cos(\tau_2-\tau_1+\delta) + (1-2\varkappa\cos\delta-\xi_1^2)\xi_1\,,\\
\xi_2' &= \varkappa\big(\xi_1\cos(\tau_2-\tau_1-\delta)+\xi_3\cos(\tau_2-\tau_3-\delta)\big) + (1-2\varkappa\cos\delta-\xi_2^2)\xi_2\,,\\
\xi_3' &= 2\varkappa\xi_2\cos(\tau_2-\tau_3+\delta) + (1-2\varkappa\cos\delta-\xi_3^2)\xi_3\,,\\
\tau_1' &= \psi_0 - 2\varkappa\sin\delta + 2\varkappa\frac{\xi_2}{\xi_1}\sin(\tau_2-\tau_1+\delta) - b_0\xi_1^2\,,\\
\tau_2' &= \psi_0 - 2\varkappa\sin\delta + \varkappa\Big(\frac{\xi_1}{\xi_2}\sin(\tau_1-\tau_2+\delta)+\frac{\xi_3}{\xi_2}\sin(\tau_3-\tau_2+\delta)\Big) - b_0\xi_2^2\,,\\
\tau_3' &= \psi_0 - 2\varkappa\sin\delta + 2\varkappa\frac{\xi_2}{\xi_3}\sin(\tau_2-\tau_3+\delta) - b_0\xi_3^2\,,
\end{aligned} \tag{15}$$

где $\psi_0 = \dfrac{\mathrm{Im}(A_1 a, b)}{\mathrm{Re}(A_1 a, b)}$, $\varkappa = \dfrac{|(Da, b)|}{\mathrm{Re}(A_1 a, b)}$, $\delta = \arg((Da, b))$, $b_0 = \dfrac{c_0}{d_0}$.

От полученной шестимерной системы может быть отщеплена пятимерная система относительно переменных ξ_1, ξ_2, ξ_3 и $\alpha_1 = \tau_2 - \tau_1, \alpha_2 = \tau_2 - \tau_3$,

$$\begin{aligned}
\xi_1' &= 2\varkappa\xi_2\cos(\alpha_1+\delta) + (1-2\varkappa\cos\delta-\xi_1^2)\xi_1\,,\\
\xi_2' &= \varkappa(\xi_1\cos(\alpha_1-\delta)+\xi_3\cos(\alpha_2-\delta)) + (1-2\varkappa\cos\delta-\xi_1^2)\xi_1\,,\\
\xi_3' &= 2\varkappa\xi_2\cos(\alpha_2+\delta) + (1-2\varkappa\cos\delta-\xi_2^2)\xi_2\,,\\
\alpha_1' &= -\varkappa\left[\Big(\frac{\xi_1}{\xi_2}+2\frac{\xi_2}{\xi_1}\Big)\sin(\alpha_1-\delta)+\frac{\xi_3}{\xi_2}\sin(\alpha_2-\delta)\right] + b_0(\xi_1^2-\xi_2^2)\,,\\
\alpha_2' &= -\varkappa\left[\frac{\xi_1}{\xi_2}\sin(\alpha_1-\delta)+\Big(\frac{\xi_3}{\xi_2}+2\frac{\xi_2}{\xi_3}\Big)\sin(\alpha_2-\delta)\right] + b_0(\xi_3^2-\xi_2^2)\,.
\end{aligned} \tag{16}$$

Система (16) имеет два простейших состояния равновесия

$$\xi_1 = \xi_2 = \xi_3 = 1, \quad \alpha_1 = \alpha_2 = 0, \tag{17}$$

$$\xi_1 = \xi_2 = \xi_3 = \sqrt{1-2\varkappa\cos\delta}, \quad \alpha_1 = \alpha_2 = \pi, \tag{18}$$

Первое из них соответствует синфазным колебаниям в исходной системе, а второе — колебаниям в противофазе.

Следующее утверждение позволяет выяснить характер потери устойчивости состояния равновесия (17).

Теорема 2. *Пусть* $(\varkappa - \varkappa_{cr})c > 0$, *где*

$$c = c(\delta, b_0) \equiv \frac{1}{\varkappa_{cr}}\Big(2b_0\sqrt{1-\cos^2\delta}(b_0^2\cos\delta - 5\cos\delta + 2) + \\ + 2(3\cos^2\delta - 1)(b_0^2+1)\Big), \quad \varkappa_{cr} = -\cos\delta - b_0\sin\delta, \quad (19)$$

тогда в достаточно малой окрестности неподвижной точки $(1,1,1,0,0)^T$ *имеются два состояния равновесия* $(\xi_1^*, \xi_2^*, \xi_3^*, \alpha_1^*, \alpha_2^*)^T$ *и* $(\xi_3^*, \xi_2^*, \xi_1^*, -\alpha_1^*, -\alpha_2^*)^T$, *которые устойчивы при* $\varkappa - \varkappa_{cr} < 0$ *и* $c < 0$ *и неустойчивы при* $\varkappa - \varkappa_{cr} > 0$ *и* $c > 0$. *Величины* $\xi_1^*, \xi_2^*, \xi_3^*, \alpha_1^*, \alpha_2^*$ *допускают при* $|\varkappa - \varkappa_{cr}| \ll 1$ *асимптотическое представление*

$$\begin{aligned} \xi_1^* &= 1 - \sqrt{(\varkappa - \varkappa_{cr})/c} + O(\varkappa - \varkappa_{cr}), \\ \xi_2^* &= 1 + 2\sqrt{(\varkappa - \varkappa_{cr})/c} + O(\varkappa - \varkappa_{cr}), \\ \xi_2^* &= 1 - \sqrt{(\varkappa - \varkappa_{cr})/c} + O(\varkappa - \varkappa_{cr}), \\ \alpha_1^* &= -\alpha_2^* = -2(1 + \varkappa\cos\delta)\sqrt{\frac{\varkappa - \varkappa_{cr}}{c(1-\cos^2\delta)}} + O(\varkappa - \varkappa_{cr}). \end{aligned} \quad (20)$$

Утверждение теоремы получается на основе разложения правых частей системы (16) в ряд по степеням $\sqrt{|\varkappa - \varkappa_{cr}|}$. Коэффициент разложения при $\sqrt{|\varkappa - \varkappa_{cr}|}$ и, соответственно, асимптотика (20) определяется из условий разрешимости алгебраической системы при $|\varkappa - \varkappa_{cr}|^{3/2}$.

Из результатов теорем 1 и 2 можно заключить, что при изменении параметра связи для однородного состояния равновесия, соответствующего однородному циклу задачи, возможны два случая, в первом из которых оно теряет устойчивость с появлением двух устойчивых неоднородных состояний, а во втором — с ним сливаются два неустойчивых неоднородных состояния и отбирают у него устойчивость.

Условия устойчивости точки (18) позволят определить критическое значение

$$\varkappa_\pi = \frac{1}{4\cos\delta} \quad (21)$$

так, что при $\varkappa > \varkappa_\pi$ данная неподвижная точка неустойчива, а при $\varkappa < \varkappa_\pi$ — устойчива.

Обозначим $\sigma = \varkappa - \varkappa_\pi$ и рассмотрим характер потери устойчивости данного состояния равновесия, в частности, докажем следующее утверждение.

Теорема 3. *Пусть* $b_0 > -\operatorname{ctg} 2\delta$ *и* $\sigma d_\pi < 0$, *где*

$$d_\pi = -\frac{3(-3 + 3b^2 + (1+3b^2)\cos 2\delta + 2b\sin 2\delta)}{8(-2b\cos\delta + \sin\delta)^2}, \quad (22)$$

тогда при $d_\pi < 0$ $(d_\pi > 0)$ *найдется такое* $\sigma_0 > 0$, *что для любого* $0 < \sigma \le \sigma_0$ $(-\sigma_0 \le \sigma < 0)$ *в достаточно малой окрестности неподвижной точки* (18) *имеется орбитально асимптотически устойчивый (неустойчивый) цикл, асимптотика которого задается формулой*

$$(\xi_1(\tau), \xi_2(\tau), \xi_3(\tau), \alpha_1(\tau), \alpha_2(\tau))^T = \sqrt{-\frac{\sigma\phi_\pi}{d_\pi}} \times \\ \times \left(\exp\left(\left(i\omega_* + \sigma\left(\psi_\pi - \frac{\phi_\pi c_\pi}{d_\pi}\right)\right)\tau\right)(a_1, -2a_1, a_1, 1, -1)^T + к.с.\right) + O(\sigma), \quad (23)$$

где

$$\omega_* = \frac{1}{2}\sqrt{-1 - 2b\,\mathrm{tg}\,\delta + \mathrm{tg}^2\,\delta}, \quad a_1 = -\frac{2i\omega_* + \sec^2\delta - 2b\,\mathrm{tg}\,\delta}{4\sqrt{2}(1+i\omega_*)(2b - \mathrm{tg}\,\delta)}, \tag{24}$$

$$\phi_\pi + i\psi_\pi = 2\cos\delta\left(2 - \frac{2i\omega_*}{\cos 2\delta + b\sin 2\delta}\right), \tag{25}$$

$$c_\pi = -\frac{3\omega_*\cos\delta\big(\big(3-13b^2\big)\cos\delta + \big(1+b^2\big)\cos 3\delta + 8b\sin\delta\big)}{8(-2b\cos\delta + \sin\delta)^2(\cos 2\delta + b\sin 2\delta)}. \tag{26}$$

Для доказательства данного утверждения при $|\sigma| \ll 1$ в системе (16) выполняется стандартная замена метода нормальных форм

$$(\xi_1(\tau), \xi_2(\tau), \xi_3(\tau), \alpha_1(\tau), \alpha_2(\tau))^T = (\xi^*, \xi^*, \pi)^T + \\ + \sqrt{|\sigma|}\big(\eta(\tau)e^{i\omega_* s}(a_1, -2a_1, a_1, 1, 1)^T + \text{к.с.}\big) + \sigma w_1(s,\tau) + |\sigma|^{3/2} w_2(s,\tau) + \ldots, \ \tau = |\sigma| s.$$

Из условий разрешимости задачи, возникающей при $|\sigma|^{3/2}$ для $w_2(s,\tau)$, в классе $2\pi/\omega_*$-периодических по s функций получается уравнение

$$\frac{d\eta}{d\tau} = \mathrm{sign}(\sigma)(\phi_\pi + i\psi_\pi)\eta + (d_\pi + ic_\pi)|\eta|^2\eta \tag{27}$$

относительно комплекснозначной функции $\eta(\tau)$. Утверждение теоремы получается из требования существования и устойчивости (неустойчивости) автомодельных циклов уравнения (27).

Применение теорем 1 и 3 позволяет заключить, что для состояния равновесия, соответствующего колебаниям осцилляторов в противофазе, можно выделить два случая. В первом из них это состояние равновесия становится устойчивым в результате стягивания к нему устойчивого предельного цикла системы (бифуркация Андронова – Хопфа), а во втором случае оно становится устойчивым после ответвления от него неустойчивого предельного цикла.

Система (16) имеет и другие состояния равновесия, однако их определение оказывается технически трудной задачей.

3. Фазовая модель системы из четырех диффузионно слабо связанных осцилляторов

Рассмотрим теперь систему (11) при $n = 4$. В этом случае удается лишь при $\varkappa = \mu\nu$, где μ — положительный малый параметр, построить систему фазовых уравнений В [3, 4] показано, что при $\mu = 0$ система (11) имеет глобально экспоненциально устойчивое интегральное многообразие $\xi_j = (-\varphi_0/d_0)^{1/2}$, $j = 1, 2, 3, 4$, поведение траекторий на котором описывается уравнениями

$$\dot{\alpha}_j = 0, \quad j = 1, \ldots, n-1.$$

Можно показать, что при всех малых μ она имеет аналогичное глобально экспоненциально устойчивое интегральное многообразие, уравнения на котором имеют в первом приближении вид (см. [4])

$$\begin{aligned}\dot\alpha_1 &= 2\sin\alpha_1 - \sin\alpha_2 + \nu(1-\cos\alpha_2),\\ \dot\alpha_2 &= 2\sin\alpha_2 - \sin\alpha_1 - \sin\alpha_3 + \nu(\cos\alpha_1 - \cos\alpha_3),\\ \dot\alpha_3 &= 2\sin\alpha_3 - \sin\alpha_2 + \nu(\cos\alpha_2 - 1).\end{aligned} \tag{28}$$

При определении состояний равновесия системы (28) центральное значение имеет уравнение

$$\sin\big((\alpha_1+\alpha_3)/2\big)\cdot\cos\big(\gamma_3-(\alpha_1-\alpha_3)/2\big)=0, \tag{29}$$

где $\gamma_3 = \arcsin\left(\nu(\nu^2+1)^{-1/2}\right)$, получающееся после сложения всех правых частей системы (28). Учитывая 2π-периодичность переменных $\alpha_1, \alpha_2, \alpha_3$ системы (28), из условия $\sin\big((\alpha_1+\alpha_3)/2\big)=0$ получаем следующие состояния равновесия:

$$(0,0,0), \tag{30}$$

$$(\pi,0,\pi), \tag{31}$$

$$(-\arcsin(\nu),\pi,\arcsin(\nu)), \tag{32}$$

$$(\pi+\arcsin(\nu),\pi,\pi-\arcsin(\nu)), \tag{33}$$

последние два из которых существуют лишь при $|\nu|\le 1$. Анализ матрицы устойчивости позволяет заключить, что тривиальное решение системы при любых значениях ν – неустойчивый узел, состояние равновесия (31), соответствующее колебаниям в противофазе, – седло-узел, состояние равновесия (32) – также седло-узел и, наконец, точка (33) является устойчивым узлом при $0<\nu<\sqrt{(\sqrt5-1)/2}$ и седло-узлом при $\sqrt{(\sqrt5-1)/2}<\nu<1$, напомним, что при $\nu>1$ этого состояния равновесия не существует.

Все состояния равновесия, получающиеся в результате обнуления второго сомножителя уравнения (29), оказываются неустойчивыми, и их вид здесь приводиться не будет.

Система (28) имеет инвариантные множества, позволяющие прояснить характер расположения траекторий системы. В частности, выполнено следующее простое утверждение:

Теорема 4. *Прямая $\alpha_1+\alpha_3=0,\ \alpha_2=\pi$ является инвариантным множеством системы (28).*

Подстановка $\alpha_1+\alpha_3=0,\ \alpha_2=\pi$, превращает все уравнения системы (28) в одно соотношение:

$$\dot\alpha_1 = 2\sin\alpha_1 + 2\nu\,,$$

которое имеет два состояния равновесия: неустойчивое $\alpha_1=-\arcsin(\nu)$ и устойчивое $\alpha_1=\pi-\arcsin(\nu)$.

Суммируя сказанное, можно утверждать, что состояние равновесия (33) является единственным устойчивым режимом исследуемой динамической системы при $0<\nu<\sqrt{(\sqrt5-1)/2}$.

Увеличение ν приводит к усложнению динамики. Численный анализ системы (28), выполненный в [4], показал, что при $\nu>\sqrt{(\sqrt5-1)/2}$ возникают устойчивые

однообходные циклы, которые, по-видимому, остаются единственными устойчивыми режимами системы (28) при всех ν за исключением промежутка (ν_1, ν_2), где $\nu_1 \approx 1.13, \nu_2 \approx 1.20$. В этом промежутке поведение решений системы носит неупорядоченный характер.

Представляет интерес сравнение полученных результатов со случаем, когда в системе (11) $n = 4$ и заданы периодические условия $z_0 = z_4,\ z_5 = z_1$.

Уравнения на устойчивом интегральном многообразии имеют в этом случае следующее первое приближение:

$$\begin{aligned}
\dot\alpha_1 &= 2\sin\alpha_1 + \sin(\alpha_1+\alpha_2+\alpha_3) - \sin\alpha_2 + \nu(\cos(\alpha_1+\alpha_2+\alpha_3) - \cos\alpha_2),\\
\dot\alpha_2 &= 2\sin\alpha_2 - \sin\alpha_1 - \sin\alpha_3 + \nu(\cos\alpha_1 - \cos\alpha_3),\\
\dot\alpha_3 &= 2\sin\alpha_3 + \sin(\alpha_1+\alpha_2+\alpha_3) - \sin\alpha_2 + \nu(\cos\alpha_2 - \cos(\alpha_1+\alpha_2+\alpha_3)).
\end{aligned} \tag{34}$$

Для системы (34) в [4] показано, что в ее фазовом пространстве имеются области заполненная предельными циклами. Негрубый характер полученных режимов не позволяет сделать каких-либо выводов относительно существования устойчивых режимов в системе (4), поэтому построим для системы (11) асимптотические формулы устойчивого интегрального многообразия и системы фазовых уравнений на нем более высокой степени точности. Уточненная система, выписанная для переменных

$$x_1 = \frac{\alpha_1+\alpha_2}{2} + \frac{\pi}{2}\,,\ x_2 = \frac{\alpha_2+\alpha_3}{2} + \frac{\pi}{2}\,,\ x_3 = \frac{\alpha_1+\alpha_3}{2} + \frac{\pi}{2}\,, \tag{35}$$

приобретает в данном случае вид

$$\begin{aligned}
\dot x_1 &= \varkappa^*\cos x_1\sin x_2\sin(x_3+\gamma) - \frac{\mu}{2}\Big(\cos x_3\Big(\cos x_3 + x_0(\sin x_3 - 2\sin x_3\cos 2x_2)\Big) +\\
&\quad + \frac{\nu}{2}\Big(-\frac{1}{2}\sin 2x_3 + x_0(\cos^2 x_3 - \cos 2x_3\cos 2x_2)\Big)\Big)\sin 2x_1\,,\\
\dot x_2 &= \varkappa^*\sin x_1\cos x_2\sin(x_3-\gamma) - \frac{\mu}{2}\Big(\cos x_3\Big(\cos x_3 + x_0(\sin x_3 - 2\sin x_3\cos 2x_1)\Big) +\\
&\quad + \frac{\nu}{2}\Big(\frac{1}{2}\sin 2x_3 + x_0(\cos^2 x_3 - \cos 2x_3\cos 2x_1)\Big)\Big)\sin 2x_2\,,\\
\dot x_3 &= 4\sin x_1\sin x_2\cos x_3 + \frac{\mu}{2}\nu\Big(\cos x_3(\cos 2x_2 - \cos 2x_1) +\\
&\quad + x_0\sin x_3(2\cos 2x_1\cos 2x_2 - \cos 2x_1 - \cos 2x_2)\Big)\cos x_3\,.
\end{aligned}$$

где $\varkappa^* = (4+\nu^2)^{1/2}$, $\gamma = \arccos\big(2(4+\nu^2)^{-1/2}\big)$.

При $\mu \neq 0$ представляет интерес найти все негрубые решения полученной системы.

В заключение отметим, что анализ диффузионного взаимодействия осцилляторов для $n = 3$ и $n = 4$ позволил проанализировать простейшие состояния равновесия нормализованных систем и их фазовые перестройки. Оказалось, что при изменении параметра связи для однородного состояния равновесия, соответствующего однородному циклу задачи, возможны два случая, в первом из которых оно теряет устойчивость с появлением двух устойчивых неоднородных состояний, а во втором — с ним сливаются два неустойчивых неоднородных состояния и отбирают у

него устойчивость. Для состояния равновесия, соответствующего колебаниям осцилляторов в противофазе, также выделяются два случая. В первом из них это состояние равновесия становится устойчивым в результате стягивания к нему устойчивого предельного цикла системы (бифуркация Андронова – Хопфа), а во втором случае оно становится устойчивым после ответвления от него неустойчивого предельного цикла. Для числа осцилляторов, равного четырем, удается проанализировать лишь систему для разностей фаз осцилляторов, которая получается при достаточно малом коэффициенте связи и может иметь довольно сложную динамику.

Список литературы / References

[1] Глызин С. Д., "Динамические свойства простейших конечноразностных аппроксимаций краевой задачи "реакция-диффузия"", *Дифференц. уравнения*, **33**:6 (1997), 805–811; [English transl.: Glyzin S. D., "Dynamic properties of the simplest finite-difference approximations of the "reaction-diffusion" boundary value problem", *Differential Equations*, **33**:6 (1997), 808–814].

[2] Глызин С. Д., Колесов А.Ю., Розов Н.Х., "Хаотическая буферность в цепочках связанных осцилляторов", *Дифференц. уравнения*, **41**:1 (2005), 41–49; [English transl.: Glyzin S. D., Kolesov A. Yu., and Rozov N. Kh., "Chaotic buffering property in chains of coupled oscillators", *Differential Equations*, **41**:1 (2005), 41–49].

[3] Колесов А. Ю., "Описание фазовой неустойчивости системы гармонических осцилляторов, слабо связанных через диффузию", *Докл. АН СССР*, **300**:1 (1988), 831 – 835; [English transl.: Kolesov A. Yu., "Description of Phase Instability of a System of Harmonic Oscillators Weakly Coupled by Diffusion", *Dokl. Akad. Nauk SSSR*, **300** (1988), 831 – 835].

[4] Глызин С. Д., "Разностные аппроксимации уравнения «реакция-диффузия» на отрезке", *Моделирование и анализ информационных систем*, **16**:3 (2009), 96 – 116; [Glyzin S.D., "Difference approximations of "reaction – diffusion" equation on a segment", *Modeling and Analysis of Information Systems*, **16**:3 (2009), 96 – 116 (in Russian)].

[5] Глызин С. Д., Колесов А.Ю., Розов Н.Х., "Конечномерные модели диффузионного хаоса", *Ж. вычисл. матем. и матем. физ.*, **50**:5 (2010), 860–875; [English transl.: Glyzin S.D., Kolesov A.Yu., and Rozov N.Kh., "Finite-dimensional models of diffusion chaos", *Comput. Math. Math. Phys.*, **50**:5 (2010), 816–830].

[6] Глызин С.Д., Колесов А.Ю., *Локальные методы анализа динамических систем: учебное пособие*, ЯрГУ, Ярославль, 2006, 92 с.; [Glyzin S.D., Kolesov A.Yu., *Lokalnye metody analiza dinamicheskikh sistem: uchebnoe posobie*, YrGU, Yaroslavl, 2006, 92 pp., (in Russian)].

[7] Глызин С. Д., "Размерностные характеристики диффузионного хаоса", *Моделирование и анализ информационных систем*, **20**:1 (2013), 30 – 51; [Glyzin S. D., "Dimensional Characteristics of Diffusion Chaos", *Modeling and Analysis of Information Systems*, **20**:1 (2013), 30 – 51 (in Russian)].

[8] Колесов А.Ю., Розов Н.Х., *Инвариантные торы нелинейных волновых уравнений*, Физматлит, М., 2004; [Kolesov A. Yu., and Rozov N. Kh., *Invariant Tori of Nonlinear Wave Equations*, FIZMATLIT, Moscow, 2004 (in Russian)].

[9] Глызин С. Д., "Стационарные режимы одной конечноразностной аппроксимации уравнения Хатчинсона с диффузией", *Качественные и приближенные методы исследования операторных уравнений: Межвуз. сб.*, ЯрГУ, Ярославль, 1986, 112–127; [Glyzin S.D., "Statsionarnyye rezhimy odnoy konechnoraznostnoy approksimatsii uravneniya Hatchinsona s diffuziyey", *Kachestvennyye i priblizhennyye metody issledovaniya operatornykh uravneniy: Mezhvuz. sb.*, YarGU, Yaroslavl, 1986, 112–127 (in Russian)].

[10] Aronson D. G., Ermentrout G. B., Kopell N., “Amplitude Response of Coupled Oscillators”, *Physica D*, **41** (1990), 403–449.

[11] Somers D., Kopell N., “Rapid synchronization through fast threshold modulation”, *Biol. Cybern.*, **68** (1993), 393–407.

[12] Somers D., Kopell N., “Anti-phase solutions in relaxation oscillators coupled through excitatory interactions”, *J. Math. Biol.*, **33** (1995), 261–280.

[13] Terman D., “An Introduction to Dynamical Systems and Neuronal Dynamics”, *Tutorials in Mathematical Biosciences I, Lecture Notes in Mathematics*, **1860** (2005), 21–68.

Marushkina E.A., "Stable Cycles and Tori of a System of Three and Four Diffusive Coupled Oscillators", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 850–859.

DOI: 10.18255/1818-1015-2016-6-850-859

Abstract. We consider chains of identical diffusive weakly coupled oscillation systems with different conditions on coupling on a chain bound. We suppose that every interacting oscillator undergoes Andronov–Hopf bifurcation, and the coefficient of coupling is proportional to the supercriticality value. For this case on a stable integral manifold of the system a normal form is constructed for which, in case of three oscillators interact, we can study the simplest stationary states and their phase transformations. As the coupling parameter changes, for the uniform stationary state corresponding to the uniform cycle of the problem there can be two cases, in the former case it loses stability with the emergence of two stable nonuniform states, and in the latter one it merges with two unstable nonuniform states and transfers to them its stability. For the stationary state corresponding to oscillations in antiphase two cases can be also distinguished. In the first one, this stationary state becomes stable due to the contraction of a stable limit cycle of the system into it (Andronov–Hopf bifurcation), and in the second case it becomes stable after branching from it an unstable limit cycle. If there are four oscillators in the chain, the system of phase difference for a small coupling coefficient was studied.

Keywords: normal form, self-oscillations, oscillators, bifurcation, invariant torus

About the authors:
Elena A. Marushkina, PhD, Researcher
P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl 150003, Russia, e-mail: marushkina-ea@yandex.ru

Acknowledgments:
The reported study was funded by RFBR, according to the research project No. 16-31-60039 mol_a_dk.