ISSN 1818-1015
eISSN 2313-5417

2024 Volume 31 No. 3

MODELING AND ANALYSIS OF INFORMATION SYSTEMS

SCIENTIFIC JOURNAL

Start date of publication — 1999
Published quarterly

FOUNDER
P.G. Demidov Yaroslavl State University

EDITORIAL OFFICE
14 Sovetskaya str., Yaroslavl 150003, Russian Federation

Website: http://mais-journal.ru
E-mail: mais@uniyar.ac.ru
Phone: +7 (4852) 79-77-73

2024 Том 31 № 3

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ

НАУЧНЫЙ ЖУРНАЛ

Издается с 1999 года
Выходит 4 раза в год

УЧРЕДИТЕЛЬ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Ярославский государственный университет им. П. Г. Демидова»

РЕДАКЦИЯ
ул. Советская, 14, Ярославль, 150003, Российская Федерация
Website: http://mais-journal.ru
E-mail: mais@uniyar.ac.ru
Телефон: +7 (4852) 79-77-73

Свидетельство о регистрации СМИ ПИ № ФС77-66186 от 20.06.2016 выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций. Подписной индекс в каталоге «Урал-Пресс» — 31907. Технический редактор, компьютерная вёрстка — К. В. Лагутина. Дата выхода в свет 30.09.2024. Формат 200×265 мм. Объ-ем 136 с. Тираж 28 экз. Свободная цена. Заказ 24060. Издатель и его адрес: Ярославский государственный университет им. П. Г. Демидова; ул. Советская, 14, Ярославль, 150003, Россия. Типография и ее адрес: ООО «Филигрань»; ул. Свободы, 91, Ярославль, 150049, Россия.
Содержание предназначено для детей старше 12 лет.

Contents

Содержание

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 31, NO. 3, 2024
journal homepage: www.mais-journal.ru

ARTIFICIAL INTELLIGENCE

Methods of Implicit Aspect Detection in Russian Publicism Sentences

A. Y. Poletaev[1], I. V. Paramonov[1], E. M. Kolupaev[1]

DOI: 10.18255/1818-1015-2024-3-226-239

[1]P.G. Demidov Yaroslavl State University, Yaroslavl, Russia

MSC2020: 68T50
Research article
Full text in Russian

Received July 1, 2024
Revised July 25, 2024
Accepted July 31, 2024

The paper compares performance of various methods of automatic implicit aspect detection in publicism sentences in Russian. The task of implicit aspect detection is an auxiliary task in the aspect-oriented sentiment analysis. The experiments were conducted on a corpus of sentences extracted from political campaign materials. The best results, with F1-measure reaching 0.84, were obtained using the Navec embeddings and classifiers based on the support vector machine method. Fairly high results, with F1-measure reaching 0.77, were obtained using the bag-of-words model and the naive Bayesian classifier. Other methods showed lower performance. It was also revealed during the experiments that the detection quality can differ significantly between the aspects. The detection quality is the highest for the aspects associated with characteristic marker words, for example, “health car” and “holding elections”. More general aspects, such as “quality of governance”, are detected with the worst quality.

Keywords: aspect detection; implicit aspects; sentiment analysis; publicism

INFORMATION ABOUT THE AUTHORS

Poletaev, Anatoliy Y. (corresponding author)	ORCID iD: 0000-0003-0116-4739. E-mail: anatoliy-poletaev@mail.ru Post-graduate student
Paramonov, Ilya V.	ORCID iD: 0000-0003-3984-8423. E-mail: ilya.paramonov@fruct.org PhD, Associate professor
Kolupaev, Egor M.	ORCID iD: 0009-0006-4312-2413. E-mail: kolupaew.eg@yandex.ru Student

Funding: Russian Science Foundation (Project no. 23-21-00495).

For citation: A. Y. Poletaev, I. V. Paramonov, and E. M. Kolupaev, “Methods of implicit aspect detection in Russian publicism sentences”, *Modeling and Analysis of Information Systems*, vol. 31, no. 3, pp. 226–239, 2024. DOI: 10.18255/1818-1015-2024-3-226-239.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 31, № 3, 2024
сайт журнала: www.mais-journal.ru

ARTIFICIAL INTELLIGENCE

Методы определения неявно упоминаемых аспектов в публицистических предложениях на русском языке

А. Ю. Полетаев[1], И. В. Парамонов[1], Е. М. Колупаев[1]

DOI: 10.18255/1818-1015-2024-3-226-239

[1]Ярославский государственный университет им. П.Г. Демидова, Ярославль, Россия

УДК 004.912
Научная статья
Полный текст на русском языке

Получена 1 июля 2024 г.
После доработки 25 июля 2024 г.
Принята к публикации 31 июля 2024 г.

В работе сравнивается качество работы различных методов определения неявно упоминаемых аспектов социально-экономической жизни в публицистических предложениях на русском языке. Задача определения неявно упоминаемых аспектов является вспомогательной для задач аспектно-ориентированного анализа тональности. Эксперименты проводились на корпусе предложений, извлечённых из политической агитации. Лучшие результаты, с F1-мерой, достигающей 0.84, были получены с использованием эмбеддингов Navec и классификаторов, основанных на методе опорных векторов. Достаточно высокие результаты, с F1-мерой до 0.77, были получены при использовании модели «мешок слов» и наивного байесовского классификатора. Остальные методы показали более низкие результаты. Также в ходе экспериментов было выявлено, что качество определения различных аспектов может достаточно сильно отличаться. Лучше всего определяются аспекты, с которыми в речи связаны характерные слова-маркеры, например, «здравоохранение» и «проведение выборов» Хуже всего определяются упоминания достаточно общих аспектов, таких как «качество управления».

Ключевые слова: определение аспектов; неявные аспекты; анализ тональности; публицистический стиль

ИНФОРМАЦИЯ ОБ АВТОРАХ

Полетаев, Анатолий Юрьевич (автор для корреспонденции)	ORCID iD: 0000-0003-0116-4739. E-mail: anatoliy-poletaev@mail.ru Аспирант
Парамонов, Илья Вячеславович	ORCID iD: 0000-0003-3984-8423. E-mail: ilya.paramonov@fruct.org Канд. физ.-мат. наук, доцент
Колупаев, Егор Михайлович	ORCID iD: 0009-0006-4312-2413. E-mail: kolupaew.eg@yandex.ru Студент

Финансирование: Российский научный фонд (проект № 23-21-00495).

Для цитирования: A. Y. Poletaev, I. V. Paramonov, and E. M. Kolupaev, “Methods of implicit aspect detection in Russian publicism sentences”, *Modeling and Analysis of Information Systems*, vol. 31, no. 3, pp. 226–239, 2024. DOI: 10.18255/1818-1015-2024-3-226-239.

Введение

Анализ тональности — направление компьютерной лингвистики, изучающее автоматическое определение выраженного в тексте авторского отношения [1]. В зависимости от поставленной задачи анализ может производиться на разных уровнях, например, на уровне текста, абзаца или отдельного предложения. Кроме того, может выполняться как анализ общей тональности (определение отношения к теме текста в целом), так и аспектно-ориентированный анализ (определение отношения к конкретным аспектам темы текста) [2].

Поскольку в отдельно взятом предложении обычно затрагиваются не все аспекты обсуждаемой темы, а только часть из них, при аспектно-ориентированном анализе тональности возникает вспомогательная задача: по данному предложению определить, какие из аспектов в нём упоминаются [3]. Эта задача может решаться как для явных упоминаний аспектов (explicit aspects), так и для всех упоминаний, включая неявные (implicit aspects). Если в предложении можно выделить конкретную группу слов, называющих аспект, то аспект упоминается явно. Иначе речь идёт о неявном упоминании. Например, в предложении «Кандидат Михаил Петров пообещал жителям Приволжска, что к их городу будет построено современное шоссе» явно упоминаются аспекты социально-экономической жизни «Михаил Петров» и «Приволжск» (следуя [4], авторы данной статьи рассматривают все именованные сущности как аспекты) и неявно упоминаются аспекты «проведение выборов» и «состояние дорожной сети».

Данная работа посвящена сравнению методов определения неявно упоминаемых аспектов в публицистических предложениях на русском языке и выявлению наилучшего качества, которого можно достичь с их помощью. Более формально: исследовались методы решения следующей задачи: дано предложение S и набор аспектов $A_1, \ldots, A_n$; требуется для каждого аспекта A_i определить, упоминается ли он в предложении S. Эта задача может быть переформулирована как n задач бинарной классификации: дано предложение S и аспект A_i; требуется определить, упоминается ли A_i в S или нет. В рамках исследования рассматривались публицистические тексты, посвящённые социально-экономической жизни некоторого региона, при этом список рассматриваемых аспектов был составлен вручную на основе собранного корпуса. Необходимо отметить, что задача выделения аспектов (aspect extraction), т. е. автоматического составления списка всех аспектов, упоминаемых в тексте, выходит за рамки данной работы.

Важная особенность данной работы — проведение экспериментов на корпусе предложений, извлечённых из материалов предвыборной агитации кандидатов на различные должности государственного и муниципального управления. Поскольку в подобных материалах упоминаются многие аспекты социально-экономической жизни [5], это даёт возможность сравнить сложность определения упоминаний различных аспектов.

1. Обзор существующих методов

Задача определения неявно упоминаемых аспектов в тексте (иногда называемая задачей определения категорий аспектов — aspect category extraction, в противовес задаче определения терминов аспектов — aspect term extraction, которые упоминаются в тексте явно [2]) является не самой распространённой решаемой задачей в области анализа тональности, однако для английского языка существует множество устоявшихся методов её решения. Основными из них являются: методы, основанные на правилах; методы тематического моделирования; методы обучения без учителя или с частичным обучением, в т. ч. основанные на кластеризации и совместной частоте встречаемости; методы машинного обучения и нейросетевые классификаторы. Рассмотрим их кратко, а более полную информацию можно найти в обзорах [3, 6, 7].

Идея методов, основанных на правилах, заключается в определении некоторого набора шаблонов для фрагментов текста, позволяющих фиксировать появление аспектов. Лучше всего такой под-

ход работает для явно упоминаемых аспектов либо в предметных областях со стандартизованной структурой предложений. В целом для неявно упоминаемых аспектов данный подход не является популярным ввиду сложности реализации.

Также для выделения неявно упоминаемых аспектов могут использоваться методы кластеризации, однако их применение может быть затруднено ввиду сложности интерпретации аспектов, соответствующих построенным кластерам. Когда выделяемые аспекты известны заранее, может быть полезным подход с частичным обучением, когда используется некоторое количество слов, описывающих аспекты, в качестве «затравочных», а затем используются техники, основанные на использовании совместной частоты встречаемости слов в корпусе.

Методы тематического моделирования — это вероятностные методы, нацеленные на идентификацию слов, характерных предложениям с конкретными неявно упоминаемыми аспектами. Их эффективность для рассматриваемой задачи обусловлена близостью последней к задаче тематического моделирования.

Наиболее обширный класс методов включает в себя методы машинного обучения (в т. ч. глубокого), а также нейросетевые классификаторы. Как правило, данные методы дают неплохие результаты для типичных предметных областей аспектно-ориентированного анализа тональности, таких как твиты и отзывы на различные продукты. Типичные значения F-меры в таких задачах обычно варьируются в диапазоне 70–85 %.

Некоторым исследователям удаётся добиться очень высоких результатов за счёт построения гибридных классификаторов, использующих различные методы. Например, в работе [8] авторам удалось достичь F-меры около 96 % на открытом наборе данных соревнования SemEval 2015 ABSA Dataset.

К сожалению, для русского языка существуют лишь единичные работы, посвящённые задаче определения неявно упоминаемых аспектов для анализа тональности.

В частности, в работе [9] предложен алгоритм определения категорий аспектов (по существу неявно упоминаемых аспектов) на основе построения семантического графа для заданной предметной области с последующим использованием этого графа при кластеризации анализируемых текстов. На корпусе отзывов о ресторанах соревнования SemEval-2016 (Task 5) точность определения категорий «ресторан», «обслуживание», «интерьер», «еда» с применением разработанного алгоритма составила 65–72 %.

В работе [10] рассматривается нейросетевая модель, предназначенная для одновременного определения аспектов в предложениях, фрагментов, к ним относящимся, и информации о тональности. Используются абстрактные конечные автоматы для извлечения сущностей и рекуррентные нейронные сети для их классификации. Для русского языка модель тестировалась на отзывах на товары AliExpress и позволила получить F-меру, равную 72 %.

Ввиду крайней недостаточности исследований, посвящённых русскому языку, представляется актуальным проведение исследований практически с любыми моделями и методами для решения задачи выделения неявно упоминаемых аспектов. Авторы настоящей статьи остановились на применении наиболее популярных для англоязычных текстов традиционных классификаторов в совокупности с доступными для русского языка эмбеддингами.

2. Корпуса текстов

Перед началом экспериментов был собран корпус предложений из материалов предвыборной агитации на русском языке, и для каждого предложения этого корпуса были указаны упоминаемые аспекты социально-экономической жизни. Список аспектов был составлен исходя из двух принципов. Во-первых, аспект должен упоминаться в достаточно большом числе предложений корпуса, иначе будет сложно оценить качество определения упоминаний этого аспекта. Во-вторых, включаемые в список аспекты не должны быть слишком общими, т. е. должны характеризовать какую-то

Table 1. The number of sentences containing a particular aspect

Таблица 1. Количество предложений, в которых упоминаются аспекты

Аспект	Первоначальный корпус	Расширенный корпус
развитие промышленности	317	618
качество и стоимость услуг ЖКХ	401	611
проведение выборов	279	581
качество управления	308	459
качество здравоохранения	215	365
состояние жилья	198	297
развитие сельского хозяйства	204	294
состояние дорожной сети	214	286
газификация	167	275
строительство атомной электростанции	141	222

конкретную сторону социально-экономической жизни. Например, в предвыборной агитации часто говорится об обществе в целом («работаю на пользу общества», «общество меня понимает»), но если выделить «общество» как аспект, то придётся считать, что он неявно упоминается практически во всех предложениях политической агитации.

В результате в список попали следующие аспекты: развитие промышленности, качество и стоимость услуг ЖКХ, проведение выборов, качество управления, качество здравоохранения, состояние жилья, развитие сельского хозяйства, состояние дорожной сети, газификация, строительство атомной электростанции.

Для первого этапа экспериментов был собран и размечен первоначальный корпус из 2050 случайно выбранных из предвыборной агитации предложений. Среди них оказались как предложения, в которых упоминается более чем один аспект, так и предложения, в которых не упоминается ни одного аспекта из списка. Подробная информация о том, в каком числе предложений упоминался тот или иной аспект, приведена в таблице 1. Как можно видеть, в первоначальном корпусе каждый аспект упоминается как минимум в 141 предложении, а большая часть аспектов — как минимум в 200 предложениях.

Для второго этапа экспериментов и изучения того, как размер корпуса влияет на качество определения упоминаний аспектов, первоначальный корпус был расширен до 3400 предложений. Новые предложения подбирались таким образом, чтобы каждый аспект упоминался хотя бы в 200 предложениях. Подробная информация о том, в каком количестве предложений расширенного корпуса упоминается тот или иной аспект, также приведена в таблице 1.

3. Постановка экспериментов

В этом разделе описаны методы, применённые авторами на различных этапах решения задачи определения неявно упоминаемых в предложении аспектов социально-экономической жизни.

3.1. Предобработка предложений

На этапе предобработки предложения преобразуются так, чтобы метод классификации мог более эффективно их обработать. В данной работе использовались следующие техники предобработки:

- токенизация — преобразование строки предложения к списку токенов — его слов;
- лемматизация — замена каждого токена на начальную форму соответствующего слова;
- стемминг — замена каждого токена на грамматическую основу соответствующего слова;
- удаление из предложения всех токенов короче трёх символов.

Токенизация необходима для работы практически всех методов автоматической обработки текста, поэтому она проводилась всегда.

Цель лемматизации и стемминга — абстрагироваться от конкретных словоформ и облегчить для классификатора выделение связей терминов с аспектами [11]. Лемматизация позволяет сохранять больше информации о связях между словами, чем стемминг, что может повлиять на качество классификации. При проведении экспериментов могла применяться либо лемматизация, либо стемминг, либо ни одна из этих техник. Для лемматизации использовался морфологический анализатор из состава библиотеки Natasha, а для стемминга — Snowball Stemmer из состава библиотеки NLTK [12].

Цель удаления из предложений слов короче трёх символов — избавиться от служебных слов и коротких местоимений, а также чисел, которые могут создавать «шум» во входных данных и затруднять классификацию [13]. Эта техника применялась или не применялась вне зависимости от того, использовались ли лемматизация и стемминг.

3.2. Векторизация предложений

На этапе векторизации для каждого предложения строилось его векторное представление. Для этого использовались два метода с использованием эмбеддингов.

Первый метод — получение векторов-эмбеддингов для каждого слова предложения с последующим усреднением этих векторов [14]. Использовались эмбеддинги: Word2Vec [15, 16] длины 100, предобученный на русскоязычной Википедии и дообученный на предложениях из политической агитации; FastText [17] длины 512, обученный на предложениях из политической агитации; Navec (основанный на GloVe) [18, 19] длины 300 из состава проекта Natasha, причём использовались два варианта: обученный на текстах художественной литературы и обученный на новостных текстах.

В рамках второго метода для получения векторов предложений использовались эмбеддинги Doc2Vec [20], обученные на предложениях из политической агитации и 2750 случайно выбранных предложениях из СМИ.

3.3. Классификация предложений

На этом этапе обучались различные классификаторы для решения задачи бинарной классификации: дано предложение в виде списка токенов; требуется определить, упоминается ли в предложении определённый аспект из списка аспектов социально-экономической жизни. При этом использовалась пятикратная кросс-валидация.

Эксперименты проводились со следующими классификаторами:

- наивный байесовский классификатор (Naive Bayes Classifier);
- классификатор гребневой регрессии (Ridge Classifier);
- классификатор на основе логистической регрессии (Logistic Regression Classifier);
- классификатор на основе стохастического градиентного спуска (SGD Classifier);
- классификатор на основе метода опорных векторов с линейным ядром (Linear SVC);
- классификатор на основе метода опорных векторов с ядром на основе радиально-базисной функции (RBF SVC);
- дерево решений (Decision Tree Classifier);
- случайный лес (Random Forest Classifier).

Наивный байесовский классификатор использовал модель «мешка слов», а все остальные классификаторы — векторизацию предложений с помощью эмбеддингов.

Для случайного леса эксперименты проводились с максимальной глубиной от 3 до 7, а также без ограничений на максимальную глубину. Реализации классификаторов были взяты из библиотеки scikit-learn [21].

Table 2. Top 5 results with Naive Bayes with extended corpus

Таблица 2. 5 лучших результатов с Naive Bayes с расширенным корпусом

Классификатор	Число слов в «мешке слов»	F_1
Bernoulli	200	0.77
Multinomial	200	0.76
Bernoulli	300	0.75
Multinomial	300	0.74
Multinomial	400	0.73

Table 3. Best result with Naive Bayes with extended corpus

Таблица 3. Лучший результат с байесовским классификатором с расширенным корпусом

Аспект	Точность	Полнота	F_1
проведение выборов	0.77	0.83	0.80
качество управления	0.74	0.71	0.73
качество здравоохранения	0.83	0.71	0.77
строительство атомной электростанции	0.88	0.89	0.88
развитие промышленности	0.85	0.91	0.88
развитие сельского хозяйства	0.71	0.76	0.73
состояние жилья	0.64	0.75	0.69
качество и стоимость услуг ЖКХ	0.86	0.79	0.82
состояние дорожной сети	0.76	0.68	0.72
газификация	0.71	0.66	0.68

Для сравнения качества классификаторов использовалась усреднённая F_1-мера классификации по всем подвыборкам кросс-валидации и по всем аспектам. Также для каждого аспекта вычислялись усреднённые точность, полнота и F_1-мера. Это делалось для того, чтобы можно было оценить, упоминания каких аспектов выделяются лучше, а каких — хуже.

4. Результаты

4.1. Модель «мешка слов»

При экспериментах с наивным байесовским классификатором были опробованы несколько вариаций классификатора: классический байесовский классификатор (Gaussian Naive Bayes), классификатор для полиномиальных моделей (Multinomial Naive Bayes), классификатор с алгоритмом дополнения (Complement Naive Bayes) и классификатор для многомерных моделей (Bernoulli Naive Bayes). Также варьировалось число слов, формирующих «мешок слов» — от 200 до 2000 с шагом 100. Пять лучших результатов приведены в таблице 2. Все они были получены с использованием стемминга и без удаления служебных слов.

Метрики обнаружения упоминаний конкретных аспектов для наилучшего результата приведены в таблице 3. Как можно видеть, упоминания всех аспектов, кроме «состояния жилья» и «газификации», обнаруживаются с F_1-мерой не ниже 0.7, а для многих она даже превышает 0.8.

4.2. Эмбеддинги Word2Vec

При всех экспериментах с Word2Vec использовались эмбеддинги, предобученные на русскоязычной Википедии и дообученные на предложениях из политической агитации в течение 5, 10, 20 или 30 эпох. Пять лучших результатов приведены в таблице 4. Все они были получены при дообучении эмбеддингов в течение 10 эпох и без использования стемминга. Наилучший результат был получен без использования лемматизации, однако 4 других среди 5 лучших — с её использованием. Лучшие результаты показала модель с использованием лемматизации и 10 эпохами обучения.

Table 4. Top 5 results with Word2Vec embedding with extended corpus

Таблица 4. 5 лучших результатов с эмбеддингами Word2Vec с расширенным корпусом

Лемматизация	Удаление коротких токенов	Классификатор	F_1
Нет	Нет	Logistic Regression	0.66
Да	Да	RBF SVC	0.65
Да	Да	Linear SVC	0.61
Да	Да	Logistic Regression	0.61
Да	Да	Ridge Regression	0.60

Table 5. Best result with Word2Vec embedding with extended corpus

Таблица 5. Лучший результат с эмбеддингами Word2Vec с расширенным корпусом

Аспект	Точность	Полнота	F_1
проведение выборов	0.78	0.93	0.84
качество управления	0.47	0.8	0.59
качество здравоохранения	0.70	0.85	0.77
строительство атомной электростанции	0.41	0.78	0.54
развитие промышленности	0.66	0.88	0.76
развитие сельского хозяйства	0.56	0.80	0.66
состояние жилья	0.44	0.81	0.57
качество и стоимость услуг ЖКХ	0.51	0.80	0.62
состояние дорожной сети	0.59	0.83	0.69
газификация	0.47	0.83	0.60

С данной моделью наилучший результат дал классификатор, основанный на логистической регрессии. Подробные данные о метриках выделения отдельных аспектов представлены в таблице 5. В среднем результат оказался хуже, чем при использовании наивного байесовского классификатора. Тем не менее, стали гораздо лучше определяться упоминания аспекта «проведение выборов»: F_1-мера возросла на 0.20 и достигла 0.84.

4.3. Эмбеддинги Doc2Vec

В ходе экспериментов эмбеддинги Doc2Vec обучались на предложениях политической агитации и случайно выбранных 2750 предложениях из СМИ. В ходе экспериментов варьировались следующие параметры:

- длина эмбеддинга: 64, 128, 256 или 512;
- алгоритм обучения: распределённая память (distributed memory, DM) или распределённый мешок слов (distributed bag of words, DBOW);
- минимальное число вхождений слова в обучающий набор: 0, 5 или 15;
- число эпох обучения: 10, 30 или 50.

Пять лучших результатов, полученных с использованием Doc2Vec, приведены в таблице 6. Во всех случаях проводилось обучение по алгоритму распределённого мешка слов, с использованием лемматизации и без использования стемминга, только на словах, которые встречались в обучающем наборе хотя бы 5 раз, в течение 50 эпох. Средние результаты получились немного хуже, чем при использовании Word2Vec, при этом с теми же классификаторами использовались вектора эмбеддингов гораздо большей длины.

Подробные метрики для выделения различных аспектов у лучшего результата приведены в таблице 7. Как можно видеть, практически все они также немного хуже, чем при использовании Word2Vec, однако, Doc2Vec дал гораздо лучшее качество для определения упоминаний аспекта «строительство атомной электростанции».

Table 6. Top 5 results with Doc2Vec embedding with extended corpus

Таблица 6. 5 лучших результатов с эмбеддингами Doc2Vec с расширенным корпусом

Размерность вектора	Удаление коротких токенов	Классификатор	F_1
512	Нет	Linear SVC	0.65
256	Да	Linear SVC	0.63
256	Да	RBF SVC	0.61
512	Нет	RBF SVC	0.59
256	Нет	Logistic Regression	0.59

Table 7. Best result with Doc2Vec embedding with extended corpus

Таблица 7. Лучший результат с эмбеддингами Doc2Vec с расширенным корпусом

Аспект	Точность	Полнота	F_1
проведение выборов	0.76	0.86	0.81
качество управления	0.53	0.73	0.61
качество здравоохранения	0.51	0.77	0.61
строительство атомной электростанции	0.61	0.80	0.69
развитие промышленности	0.83	0.83	0.83
развитие сельского хозяйства	0.49	0.71	0.58
состояние жилья	0.46	0.73	0.56
качество и стоимость услуг ЖКХ	0.78	0.79	0.78
состояние дорожной сети	0.43	0.73	0.54
газификация	0.43	0.77	0.55

4.4. Эмбеддинги FastText

При проведении экспериментов была использована реализация алгоритма обучения эмбеддингов FastText из библиотеки gensim. Обучение производилось на корпусе предложений из политической агитации. Эксперименты проводились как на предложениях из первоначального корпуса, так и из расширенного. По пять лучших результатов для каждого из корпусов приведены в таблице 8. Все они были получены с использованием лемматизации, без использования стемминга и без удаления коротких токенов. Лучший результат на расширенном корпусе был получен с использованием классификатора на основе метода опорных векторов. Метрики качества определения упоминаний каждого из аспектов представлены в таблице 9. Как можно видеть, с увеличением объёма корпуса качество определения хорошо выделявшихся аспектов (например, «проведение выборов», «развитие промышленности») увеличилось достаточно слабо, в пределах 5 %, а качество определения упоминаний аспекта «строительство АЭС» даже снизилось. Для плохо определявшихся аспектов, например, «газификация» или «состояние жилья» рост был гораздо более существенным, вплоть до 16 %.

4.5. Эмбеддинги Navec

При проведении экспериментов были использованы две модели Navec. Первая модель hudlit-12B-500K-300d-100q, обученная на предложениях художественной литературы, содержит векторы для 500 тысяч слов. Вторая модель news-1B-250K-300d-100q обучена на текстах новостей и содержит векторы для 250 тысяч слов. Эксперименты проводились как на предложениях из первоначального корпуса, так и из расширенного. По пять лучших результатов на каждом из корпусов приведены в таблице 10. Все они были получены с использованием лемматизации, без использования стемминга и при удалении коротких токенов. Как можно видеть, увеличение количества предложений приводит к существенному росту качества определения упоминаний аспектов. При этом необхо-

Table 8. Top 5 results with FastText embedding with initial and extended corpora

Таблица 8. 5 лучших результатов с эмбеддингами FastText с изначальным и расширенным корпусами

Первоначальный корпус			Расширенный корпус		
Модель	Классификатор	F_1	Модель	Классификатор	F_1
FastText	RBF SVC	0.75	FastText	RBF SVC	0.79
	Linear SVC	0.74		Linear SVC	0.78
	Logistic Regression	0.72		Logistic Regression	0.76
	Ridge Regression	0.72		Ridge Regression	0.74
	SGD	0.65		SGD	0.71

Table 9. Best results with FastText embedding with initial and extended corpus

Таблица 9. Лучшие результаты с эмбеддингами FastText с изначальным и расширенным корпусами

Аспект	Первоначальный корпус			Расширенный корпус		
	Точность	Полнота	F_1	Точность	Полнота	F_1
проведение выборов	0.84	0.87	0.86	0.91	0.92	0.91
качество управления	0.66	0.80	0.73	0.69	0.83	0.75
качество здравоохранения	0.72	0.85	0.78	0.78	0.85	0.81
строительство атомной электростанции	0.89	0.84	0.87	0.85	0.82	0.83
развитие промышленности	0.77	0.86	0.82	0.83	0.89	0.86
развитие сельского хозяйства	0.63	0.80	0.71	0.62	0.81	0.71
состояние жилья	0.52	0.74	0.61	0.61	0.77	0.68
качество и стоимость услуг ЖКХ	0.72	0.84	0.78	0.77	0.86	0.81
состояние дорожной сети	0.76	0.76	0.76	0.70	0.80	0.75
газификация	0.60	0.75	0.66	0.79	0.85	0.82

димо отметить, что в обоих случаях лучший результат был получен с помощью использования метода опорных векторов с ядром RBF и эмбеддингов, обученных на новостях. Результаты обнаружения упоминаний отдельных аспектов представлены в таблице 11. Лучше всего определяются упоминания аспектов «проведение выборов», «качество здравоохранения», «строительство атомной электростанции» и «развитие промышленности», а хуже всего — аспектов «качество управления» и «состояние жилья».

Нужно особо отметить, что при использовании эмбеддингов, обученных на текстах художественной литературы, существенно снижается качество определений аспекта «газификация», так как в них отсутствует вектор для слова «газификация» и его производных.

5. Обсуждение результатов

В таблице 12 представлены 5 лучших результатов, полученных после экспериментов со всеми 4 описанными моделями.

Полученный с помощью наивного байесовского классификатора уровень качества со средней F_1-мерой, равной 0.77 оказался достаточно высоким. Результат существенно выше, с F_1-мерой, достигающей 0.84, был получен только при использовании эмбеддингов Navec и классификатора на основе метода опорных векторов. Из этого следует, что в задаче обнаружения упоминаний аспектов достаточно хорошие результаты могут достигаться даже с помощью достаточно примитивных методов при условии достаточно большого корпуса.

На этапе предобработки к повышению качества стабильно приводила лемматизация слов; стемминг и удаление служебных слов иногда приводили к повышению качества, а иногда — наоборот.

Table 10. Top 5 results with Navec embedding with initial and extended corpora

Таблица 10. 5 лучших результатов с эмбеддингами Navec с изначальным и расширенным корпусами

Первоначальный корпус			Расширенный корпус		
Модель	Классификатор	F_1	Модель	Классификатор	F_1
news	RBF SVC	0.69	news	RBF SVC	0.84
news	Ridge Regression	0.69	hudlit	RBF SVC	0.81
news	Linear SVC	0.69	news	Logistic Regression	0.78
news	Logistic Regression	0.68	news	Linear SVC	0.77
hudlit	RBF SVC	0.66	hudlit	Logistic Regression	0.75

Table 11. Best Navec results with initial and extended corpora for every aspect

Таблица 11. Лучшие результаты Navec с первоначальным и расширенным корпусами для каждого аспекта

Аспект	Первоначальный корпус			Расширенный корпус		
	Точность	Полнота	F_1	Точность	Полнота	F_1
проведение выборов	0.76	0.81	0.78	0.92	0.95	0.94
качество управления	0.56	0.65	0.60	0.73	0.84	0.78
качество здравоохранения	0.82	0.72	0.76	0.87	0.90	0.88
строительство атомной электростанции	0.88	0.67	0.76	0.95	0.91	0.93
развитие промышленности	0.73	0.73	0.73	0.87	0.93	0.90
развитие сельского хозяйства	0.68	0.62	0.65	0.72	0.84	0.78
состояние жилья	0.62	0.58	0.60	0.66	0.84	0.74
качество и стоимость услуг ЖКХ	0.73	0.71	0.72	0.79	0.88	0.83
состояние дорожной сети	0.86	0.69	0.77	0.79	0.81	0.80
газификация	0.70	0.52	0.60	0.78	0.82	0.80

Table 12. Best results

Таблица 12. Лучшие результаты

Модель	Классификатор	F_1
Navec news	RBF SVC	0.84
Navec hudlit	RBF SVC	0.81
FastText	RBF SVC	0.79
FastText	Linear SVC	0.78
Navec news	Logistic Regression	0.78

При использовании эмбеддингов во-первых, важно, чтобы они были обучены на достаточно большом корпусе — эмбеддинги нужно обучать на корпусе существенно большего объёма, чем тот, на котором обучается классификатор. Во-вторых, качество классификации повышается, если эмбеддинги обучены на корпусе меньшего объёма, но более близкой предметной области: новостные тексты ближе к политической агитации, чем художественная литература, и поэтому при использовании Navec, обученного на новостных текстах, качество было существенно выше, несмотря на меньший объём обучающего корпуса.

На качество классификации влияют ещё несколько факторов. Во-первых, это выбор самого классификатора. Метод опорных векторов и логистическая регрессия дают стабильно более высокие результаты, чем гребневая регрессия и классификаторы на основе решающих деревьев. Другой фактор — число предложений с упоминанием аспекта, на которых обучается классификатор. Аспекты «развитие промышленности» и «развитие сельского хозяйства» схожи, они оба обладают достаточно устойчивым набором слов-маркеров, но из-за того, что предложений с упоминанием

развития промышленности в два раза больше, чем с упоминанием развития сельского хозяйства, его упоминания обнаруживаются как минимум на 10 % лучше. Также очень серьёзный прирост качества был получен при расширении корпуса, причём этот прирост в большинстве случаев произошёл за счёт роста полноты, а не точности, т. е. классификаторы смогли лучше обучиться тому, в каких именно предложениях может упоминаться тот или иной аспект.

Сложность обнаружения упоминаний различных аспектов может очень сильно отличаться. Проще всего обнаруживаются упоминания аспектов, у которых есть свои характерные слова-маркеры, например, «строительство АЭС», «здравоохранение», «проведение выборов». Сложнее всего обнаруживаются упоминания аспектов «качество управления» и «состояние жилья», потому что они достаточно общие, у них нет своей характерной лексики, и их легко спутать с другими аспектами.

Нужно отметить, что удачная комбинация классификатора и типа эмбеддингов показывает примерно одинаково высокое качество обнаружения упоминаний всех аспектов, а неудачная — высокое качество для отдельных легко обнаруживаемых аспектов, и значительно более низкое — для всех остальных.

Нужно обратить внимание, что всех экспериментах, кроме экспериментов с FastText, полнота превышает точность, то есть классификаторы обнаруживают упоминания аспекта чаще, чем эксперт-разметчик. Это может быть вызвано тем, что почти все аспекты социально-экономической жизни тесно связаны между собой, в отличие, например, от аспектов товаров или услуг.

Заключение

В работе было проведено сравнение методов определения неявно упоминаемых аспектов в публицистических предложениях на русском языке. Эксперименты проводились на корпусе предложений из текстов предвыборной агитации. Они показали, что наилучшие результаты с F_1-мерой, равной 0.84, получаются при использовании эмбеддингов Navec, обученных на новостных текстах, и классификатора на основе опорных векторов с ядром на основе радиально-базисной функции. При этом достаточно хорошие результаты, с F_1-мерой, равной 0.77, могут быть получены при использовании наивного байесовского классификатора и представления предложений в виде «мешка слов».

Также в ходе экспериментов было выявлено, что сложность определения различных аспектов достаточно сильно отличается. Лучше всего определяются упоминания аспектов, для которых характерна определённая лексика. Такими аспектами являются, например, «строительство АЭС» или «проведение выборов». Хуже всего определяются упоминания общих аспектов, таких, как «качество управления». При этом лучшие методы позволяют достаточно хорошо определять упоминания как легкообнаружимых, так и труднообнаружимых аспектов, тогда как остальные показывают хорошие результаты для легкообнаружимых и плохие — для труднообнаружимых.

References

[1] B. Liu, *Sentiment Analysis and Opinion Mining*. Springer, 2022, 167 pp.

[2] W. Zhang, X. Li, Y. Deng, L. Bing, and W. Lam, “A survey on aspect-based sentiment analysis: Tasks, methods, and challenges”, *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 11, pp. 11 019–11 038, 2022. DOI: 10.1109/TKDE.2022.3230975.

[3] M. M. Truşcă and F. Frasincar, “Survey on aspect detection for aspect-based sentiment analysis”, *Artificial Intelligence Review*, vol. 56, no. 5, pp. 3797–3846, 2023.

[4] A. Naumov, R. Rybka, A. Sboev, A. Selivanov, and A. Gryaznov, “Neural-network method for determining text author’s sentiment to an aspect specified by the named entity”, in *CEUR Workshop Proceedings*, vol. 2648, 2020, pp. 134–143.

[5] E. V. Sergeeva, "Features of speech exposure in the preelection media discourse", in *Aktual'nye problemy gumanitarnogo znaniya v tekhnicheskom vuze*, in Russian, 2021, pp. 237–239.

[6] A. Nazir, Y. Rao, L. Wu, and L. Sun, "Issues and challenges of aspect-based sentiment analysis: A comprehensive survey", *IEEE Transactions on Affective Computing*, vol. 13, no. 2, pp. 845–863, 2020. DOI: 10.1109/TAFFC.2020.2970399.

[7] P. K. Soni and R. Rambola, "A survey on implicit aspect detection for sentiment analysis: Terminology, issues, and scope", *IEEE Access*, vol. 10, pp. 63 932–63 957, 2022. DOI: 10.1109/ACCESS.2022.3183205.

[8] B. Mohammed *et al.*, "Hybrid approach to extract adjectives for implicit aspect identification in opinion mining", in *11th International Conference on Intelligent Systems: Theories and Applications (SITA)*, IEEE, 2016, pp. 1–5. DOI: 10.1109/SITA.2016.7772284.

[9] A. O. Kornej and E. N. Kryuchkova, "Semantiko-statisticheskij algoritm opredeleniya kategorij aspektov v zadachah sentiment-analiza", *Izvestiya Yuzhnogo federal'nogo universiteta. Tekhnicheskie nauki*, no. 6 (216), pp. 66–74, 2020, in Russian. DOI: 10.18522/2311-3103-2020-6-66-74.

[10] E. I. Gribkov and Y. P. Ekhlakov, "Nejrosetevaya model' na osnove sistemy perekhodov dlya izvlecheniya sostavnyh ob'ektov i ih atributov iz tekstov na estestvennom yazyke", *Doklady Tomskogo gosudarstvennogo universiteta sistem upravleniya i radioelektroniki*, vol. 23, no. 1, pp. 47–52, 2020, in Russian. DOI: 10.21293/1818-0442-2020-23-1-47-52.

[11] L. Hickman, S. Thapa, L. Tay, M. Cao, and P. Srinivasan, "Text preprocessing for text mining in organizational research: Review and recommendations", *Organizational Research Methods*, vol. 25, no. 1, pp. 114–146, 2022. DOI: 10.1177/1094428120971683.

[12] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[13] U. Naseem, I. Razzak, and P. W. Eklund, "A survey of pre-processing techniques to improve short-text quality: A case study on hate speech detection on Twitter", *Multimedia Tools and Applications*, vol. 80, pp. 35 239–35 266, 2021. DOI: 10.1007/s11042-020-10082-6.

[14] J. Coates and D. Bollegala, "Frustratingly easy meta-embedding – computing meta-embeddings by averaging source word embeddings", in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, Association for Computational Linguistics, 2018, pp. 194–198. DOI: 10.18653/v1/N18-2031.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: 1301.3781v3 [cs.CL].

[16] I. Yamada *et al.*, "Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia", in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Association for Computational Linguistics, 2020, pp. 23–30. DOI: 10.18653/v1/2020.emnlp-demos.4.

[17] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification", in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Association for Computational Linguistics, 2017, pp. 427–431. DOI: 10.48550/arXiv.1607.01759.

[18] A. Kukushkin. "Navec – kompaktnye embeddingi dlya russkogo yazyka". in Russian. (2020), [Online]. Available: https://natasha.github.io/navec/ (visited on 08/11/2024).

[19] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation", in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

[20] Q. Le and T. Mikolov, "Distributed representations of sentences and documents", in *International conference on machine learning*, PMLR, 2014, pp. 1188–1196.

[21] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 31, NO. 3, 2024
journal homepage: www.mais-journal.ru

THEORY OF SOFTWARE

LTL-Specification for Development and Verification of Logical Control Programs in Feedback Systems

M. V. Neyzov[1], E. V. Kuzmin[2]

DOI: 10.18255/1818-1015-2024-3-240-279

[1]Institute of Automation and Electrometry SB RAS, Novosibirsk, Russia
[2]P.G. Demidov Yaroslavl State University, Yaroslavl, Russia

MSC2020: 68N30
Research article
Full text in Russian

Received August 6, 2024
Revised August 22, 2024
Accepted August 28, 2024

The article continues the series of publications on the development and verification of control programs based on LTL-specifications of a special type. Earlier, a declarative LTL-specification was proposed to describe the strictly deterministic behavior of programs, ways of its verification and translation were worked out: for verification, the model checking tool nuXmv is used, and the translation is carried out into an imperative programming language ST for programmable logic controllers. When verifying the declarative LTL-specification of the behavior of programs, there may be a need to simulate the behavior of its environment. In general, it is required to ensure the possibility of constructing closed-loop systems "program-environment". In this work, an LTL-specification of constraintly nondeterministic behavior of a Boolean variable is proposed to describe the behavior of the environment of logical control programs. This specification allows defining the behavior of Boolean feedback signals, as well as fairness conditions to exclude unrealistic scenarios of behavior. The article proposes an approach to the development and verification of logical control programs, within which the behavior model of the program environment is described in the form of constraints on the behavior of its input signals, what allows avoiding a separate detailed representation of the processes of the environment operation. As a result, the obtained behavior model of the closed-loop system "program-environment" provides a number of advantages: simplification of the modeling process, reduction of the state space of the verified model, and reduction of verification time. If it is impossible to reduce the behavior of the environment to the behavior of existing input signals, this approach suggests using "imaginary" sensors — additional Boolean variables that are used as an auxiliary means for describing the behavior of input signals. The purpose of introducing imaginary sensors is to compensate for missing sensors to track the specific behavior of some elements of the environment that needs to be taken into account when defining realistic behavior of the inputs of a logical control program. The proposed approach to the development and verification of programs taking into account the behavior of the environment (a control object) is demonstrated by the example of an industrial plastic molding plant.

Keywords: control software; PLC program; declarative LTL-specification; LTL-specification of constraintly nondeterministic behavior; fairness conditions; closed-loop systems verification; plant behavior model; imaginary sensor; temporal properties; model checking; nuXmv verifier; SMV-specification

INFORMATION ABOUT THE AUTHORS

Neyzov, Maxim V. (corresponding author) | ORCID iD: 0009-0000-6893-6137. E-mail: neyzov.max@gmail.com
Researcher

Kuzmin, Egor V. | ORCID iD: 0000-0003-0500-306X. E-mail: kuzmin@uniyar.ac.ru
Head of the Chair of Theoretical Informatics, Dr. Sc.

Funding: State task IAaE SB RAS, project No. 122031600173-8; Yaroslavl State University (project VIP-016).

For citation: M. V. Neyzov and E. V. Kuzmin, "LTL-specification for development and verification of logical control programs in feedback systems", *Modeling and Analysis of Information Systems*, vol. 31, no. 3, pp. 240–279, 2024. DOI: 10.18255/1818-1015-2024-3-240-279.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 31, № 3, 2024
сайт журнала: www.mais-journal.ru

LTL-спецификация для разработки и верификации программ логического управления в системах с обратной связью

М. В. Нейзов[1], Е. В. Кузьмин[2]

DOI: 10.18255/1818-1015-2024-3-240-279

[1]Институт автоматики и электрометрии СО РАН, Новосибирск, Россия
[2]Ярославский государственный университет им. П.Г. Демидова, Ярославль, Россия

УДК 004.424+519.683.8
Научная статья
Полный текст на русском языке

Получена 6 августа 2024 г.
После доработки 22 августа 2024 г.
Принята к публикации 28 августа 2024 г.

Статья продолжает цикл публикаций по разработке и верификации управляющих программ на основе LTL-спецификаций специального вида. Ранее для описания строго детерминированного поведения программ была предложена декларативная LTL-спецификация, проработаны способы её верификации и трансляции: для верификации используется инструмент проверки модели nuXmv, трансляция осуществляется в императивный язык программирования ST для программируемых логических контроллеров. При верификации декларативной LTL-спецификации поведения программ может возникнуть необходимость в моделировании поведения её окружения. В общем случае требуется обеспечить возможность построения замкнутых систем «программа-окружение». В настоящей работе для описания поведения окружения программ логического управления предложена LTL-спецификация ограниченно недетерминированного поведения булевой переменной. Данная спецификация позволяет задавать поведение булевых сигналов обратной связи, а также условия справедливости для исключения нереалистичных сценариев поведения. В статье предлагается подход к разработке и верификации программ логического управления, в рамках которого модель поведения окружения программы описывается в виде ограничений на поведение её входных сигналов, что позволяет избежать отдельного детального представления процессов функционирования окружения. В результате полученная модель поведения замкнутой системы «программа-окружение» даёт ряд преимуществ: упрощение процесса моделирования, сокращение пространства состояний проверяемой модели, снижение времени верификации. При невозможности сведения поведения окружения к поведению имеющихся входных сигналов данный подход предполагает применение «мнимых» датчиков — дополнительных булевых переменных, использующихся как вспомогательное средство для описания поведения входных сигналов. Цель введения мнимых датчиков состоит в компенсации недостающих датчиков для отслеживания специфического поведения отдельных элементов окружения, которое необходимо учесть при задании реалистичного поведения входов программы логического управления. Предложенный подход к разработке и верификации программ с учётом поведения окружения (объекта управления) демонстрируется на примере промышленной установки для литья пластмасс.

Ключевые слова: управляющее программное обеспечение; ПЛК-программа; декларативная LTL-спецификация; LTL-спецификация ограниченно недетерминированного поведения; условия справедливости; верификация замкнутых систем; модель поведения установки; мнимый датчик; темпоральные свойства; проверка модели; верификатор nuXmv; SMV-спецификация

ИНФОРМАЦИЯ ОБ АВТОРАХ

Нейзов, Максим Вячеславович (автор для корреспонденции)	ORCID iD: 0009-0000-6893-6137. E-mail: neyzov.max@gmail.com Исследователь
Кузьмин, Егор Владимирович	ORCID iD: 0000-0003-0500-306X. E-mail: kuzmin@uniyar.ac.ru Заведующий кафедрой теоретической информатики, доктор физ.-мат. наук

Финансирование: Госзадание ИАиЭ СО РАН, проект № 122031600173-8; ЯрГУ (проект VIP-016).

Для цитирования: M. V. Neyzov and E. V. Kuzmin, “LTL-specification for development and verification of logical control programs in feedback systems”, *Modeling and Analysis of Information Systems*, vol. 31, no. 3, pp. 240–279, 2024. DOI: 10.18255/1818-1015-2024-3-240-279.

Введение

Промышленные *киберфизические системы* (КФС) [1—3] используются для автоматизации производства [4]. *Управляющее программное обеспечение* (УПО) является неотъемлемой частью КФС и зависит от её назначения [5]. КФС условно можно разделить на УПО и его окружение. К окружению могут относиться:

- программные компоненты, не входящие в УПО (служба времени, сторонние сервисы);
- аппаратные компоненты системы управления (датчики, исполнительные устройства);
- компоненты технологического оборудования (механизмы, резервуары, камеры нагрева).

Окружение (или его часть) подвергается управлению со стороны УПО. Управляемая часть окружения является *объектом управления* (ОУ). Технологическое оборудование промышленной установки представляет собой *технологический объект управления* (ТОУ). *Программируемый логический контроллер* (ПЛК) [6] является наиболее распространённой вычислительной платформой для УПО промышленных КФС [4, 7]. К ПЛК подключаются датчики и исполнительные устройства, посредством которых он взаимодействует с ТОУ.

УПО работает циклически: считывает входные данные из окружения, выполняет вычисления, обновляет выходные данные для окружения. Таким образом, УПО КФС представляет собой *реагирующую систему* [8], так как постоянно взаимодействует со своим окружением. Совокупность всех входных и выходных сигналов УПО образует его *интерфейс*. Любой выходной сигнал УПО является сигналом *прямой связи* (feedforward) с ОУ. Входной сигнал УПО, который информирует о реакции ОУ на некоторое управляющее воздействие, является сигналом *обратной связи* (feedback). Система с прямой и обратной связями образует *замкнутый контур* (closed-loop), поэтому такие системы называются *замкнутыми*.

К промышленным КФС предъявляются высокие требования надежности и безопасности [9, 10]. Функционирование КФС напрямую зависит от УПО, поэтому его *корректность* является критическим показателем [11, 12]. Для повышения уровня доверия к УПО и гарантий его корректности используют *формальные методы верификации* [13, 14]. *Проверка модели* (model checking) [15—17] является наиболее распространённым формальным методом верификации реагирующих систем. Проверке подлежат *темпоральные свойства* [18—20], которые чаще всего выражаются в виде формул *линейной темпоральной логики* LTL (Linear Temporal Logic) или *логики деревьев вычислений* CTL (Computation Tree Logic) [15]. Фактическое поведение реагирующей системы (УПО) задаётся в виде конечной системы переходов — *структуры Крипке* [15—17]. Верификация осуществляется с помощью инструментальных средств, которые проверяют соответствие модели фактического поведения заданной *спецификации* — набору требуемых свойств. Для проверки некоторых свойств недостаточно модели поведения УПО, может потребоваться модель поведения замкнутой системы «УПО-ОУ» [21].

Согласно [22] можно выделить три подхода к формальной верификации УПО ПЛК:

1. Подход, *основанный на модели* (model based) поведения ОУ. Верификации подлежит модель поведения замкнутой системы, состоящей из УПО и ОУ.
2. Подход, *не основанный на модели* (non model based) поведения ОУ. Проводится верификация только модели поведения УПО. Входы УПО ведут себя абсолютно недетерминировано.
3. Подход, *основанный на ограничениях* (constrained based) поведения входов УПО. На входы УПО накладываются ограничения, основанные на знаниях о невозможном поведении окружения.

Настоящая работа имеет отношение к первому и третьему подходам: к первому — так как даёт возможность описания поведения элементов ОУ, к третьему — поскольку нацелена на описание ограничений, накладываемых на поведение входных сигналов УПО.

Окружение воздействует на входы УПО и определяет их поведение. Таким образом, поведение входов УПО полностью или частично отражает поведение окружения. Этот факт даёт возможность задания модели поведения окружения в виде модели поведения входов УПО с некоторым дополне-

нием в случае необходимости. При описании поведения входных сигналов УПО трудность обычно вызывают сигналы обратной связи, так как они зависят от поведения ОУ, которое, в свою очередь, зависит от выходных сигналов УПО — все эти особенности влияют на поведение сигналов обратной связи. Данная модель поведения по сути является некоторым недетерминированным дискретным аналогом *передаточной функции* (transfer function) ОУ из теории автоматического управления.

В случае трудности или невозможности сведения модели поведения окружения к модели поведения входов УПО предлагается применять подход, основанный на модели: использовать дополнительные переменные, отражающие состояние компонентов из окружения. В этом случае можно считать, что дополнительная переменная представляет собой *мнимый* датчик, показания которого предназначены для описания поведения входов УПО. При этом получается, что в общем случае модель поведения окружения представляет собой модель поведения входов УПО и мнимых датчиков.

В итоге при верификации модели поведения замкнутого контура «УПО-ОУ» нет необходимости в детальном описании поведения ОУ, которое приводит к повышению трудозатрат и требует учёта всех его возможных сценариев поведения. При описании ограничений входов необходимо лишь учесть все запрещённые сценарии, что по нашему мнению сделать проще. Также наличие модели поведения ОУ в контуре увеличивает пространство состояний результирующей модели «УПО-ОУ» и, как следствие, время верификации, а введение ограничений на входы УПО, наоборот, сокращает пространство достижимых состояний [23].

Таким образом, при построении модели замкнутого контура «УПО-ОУ» помимо описания модели поведения УПО основной целью данной статьи является задание ограничений на поведение сигналов обратной связи без определения деталей в поведении ОУ. К детализации в поведении ОУ следует прибегать в крайнем случае, когда поведение сигналов обратной связи сложно или невозможно описать без некоторых характеристик ОУ.

УПО может решать большой спектр задач. Настоящая работа сфокусирована на задачах *логического управления* (ЛУ) [24]. Данный вид управления использует только булевы входные и выходные сигналы. УПО или его часть с таким интерфейсом назовём программой ЛУ (ПЛУ). Отметим, что в общем случае окружение ПЛУ содержит в себе окружение УПО, а также ту часть УПО, которая не входит в ПЛУ. Таким образом, при выделении ПЛУ из УПО происходит расширение окружения/ОУ. Пусть, например, УПО содержит счётчик и компаратор. Счётчик фиксирует количество нажатий кнопки и выдаёт сигнал, когда их число достигнет заданного значения. Компаратор срабатывает, когда фактическое значение температуры выше уставки. Данные компоненты входят в окружение ПЛУ. Более того, если счётчик имеет вход для сброса, то он является и ОУ для ПЛУ.

В настоящей работе предлагается описывать поведение окружения ПЛУ с помощью LTL-спецификации специального вида, которая основана на идее из публикаций по согласованному поведению датчиков [25, 26]. Данная LTL-спецификация в общем случае задаёт недетерминированное, но ограниченное поведение булевых переменных. При этом поведение ПЛУ является абсолютно детерминированным и описывается с помощью *декларативной* [27] LTL-спецификации. В [28] для ускорения верификации с помощью инструмента символьной проверки модели nuXmv [29] декларативная LTL-спецификация преобразуется в SMV-спецификацию — код на входном языке инструмента nuXmv. Настоящая работа также для ускорения верификации предлагает преобразование LTL-спецификации ограниченно недетерминированного поведения окружения ПЛУ в SMV-спецификацию. Далее SMV-спецификации ПЛУ и её окружения объединяются и образуют SMV-спецификацию КФС, которая подлежит верификации.

Настоящее исследование является продолжением цикла статей по разработке и верификации УПО на основе LTL-спецификаций специального вида. В качестве языка спецификации поведения был выбран формализм LTL, а не входной язык инструмента проверки модели. Это сделано по двум причинам:

1. Отсутствует привязка к конкретному инструменту проверки модели. LTL-спецификацию можно использовать как инструментально независимый формализм, который всегда можно перевести во входной язык используемого в данный момент инструмента. Это обусловлено тем, что LTL-спецификация не навязывает применение конкретной модели, в отличие от инструментов верификации [30].
2. LTL-спецификация является математическим объектом, что позволяет работать с ней, используя соответствующий математический аппарат и любые инструментальные средства, поддерживающие язык LTL. Например, можно сравнить две LTL-формулы с помощью программного средства Spot [31]. Это позволяет легко выполнять эквивалентные преобразования LTL-спецификаций, определять их избыточность или противоречивость. Сравнение выражений (например определение эквивалентности) на входном языке инструмента проверки модели может быть затруднительно.

Содержание работы. Раздел 1 представляет обзор связанных работ. Раздел 2 содержит предварительные сведения об LTL-спецификации поведения программ. Частный случай декларативной LTL-спецификации для булевой переменной представлен в разделе 3. В разделе 4 определена LTL-спецификация ограниченно недетерминированного поведения булевой переменной для описания окружения программ. В разделе 5 приведён пример LTL-спецификации поведения установки для литья пластмасс, а в разделе 6 — её LTL-спецификация свойств. В разделе 7 осуществляется перевод LTL-спецификации поведения установки в SMV-спецификацию для ускорения верификации. Выполнена стандартизация предложенных шаблонов условий справедливости. В разделе 8 на основе LTL-спецификации поведения ПЛУ построена ST-программа для ПЛК. В заключительном разделе подводятся итоги работы. Приложения 1–6 содержат листинги спецификаций и программного кода, а также доказательства утверждений.

1. Обзор связанных работ

Относительно предлагаемого в настоящей статье подхода к верификации рассмотрим краткий обзор работ, связанных с построением моделей поведения окружения УПО для верификации замкнутого контура «УПО-окружение» методом проверки модели. В обозреваемых работах ТОУ чаще всего называется установкой (plant), поэтому в данном разделе будем использовать этот термин.

В работах [32—44] для описания поведения установки используют формализмы на основе *сетей Петри* и *автоматов* специального вида.

Системы сетевых условий/событий (Net Condition/Event Systems, NCES) [32] — расширение сетей Петри, которое получило достаточно широкое распространение в области моделирования поведения замкнутых систем. В [37—39] разработка и верификация NCES-модели осуществляется с помощью инструмента ViVe/SESA. В работах [40, 41] используют дальнейшее расширение формализма NCES — Timed NCES (TNCES).

Также применяются формализмы, которые являются расширениями автоматов. В работе [42] *временные автоматы* (Timed Automata, TA) переводятся во входной язык верификатора NuSMV. В [43] TA-модель проверяется с помощью верификатора UPPAAL. Работа [44] нацелена на построение моделей, описывающих реалистичное поведение установок. Для этого используется формализм TA *с дискретными данными* (TA with Discrete Data, TADD). Авторы не выбирают конкретный верификатор для проверки TADD-моделей.

В работах [33—36] предлагается строить дискретную модель поведения установки не с нуля, а на основе уже имеющихся артефактов. В [33] TNCES-модель строится на основе *гибридного автомата* (hybrid automaton), заданного в виде модели Stateflow. В [34, 35] TNCES-модель установки транслируется из UML-диаграмм. Для верификации модели замкнутой системы применяется инструмент проверки модели SESA. В [36] предлагается на основе CAD-модели установки строить

имитационную модель, далее по ней формируется TNCES-модель. Процедура преобразования частично автоматизирована.

Отметим, что при использовании вышеуказанных формализмов из работ [32—44] получается достаточно громоздкое описание модели поведения установки. В работе [45] вообще утверждается, что нет удобного систематического способа описания таких моделей. Кроме того, требуются специализированные инструменты для работы с их графическим представлением. Напротив, LTL-спецификация представляет собой более компактное текстовое описание.

В работах [46—51] применяются специализированные нотации [52—55], которые используются и для разработки УПО, и для описания моделей поведения установки/окружения.

В [46] для описания модели поведения замкнутой системы, и установки в частности, предлагается использовать функциональные блоки стандарта МЭК 61499 [52]. В [47—49] данные блоки автоматически переводятся во входной язык инструмента NuSMV. Недетерминизм поведения блоков обеспечивается с помощью специальных событий NDT, инициирующих недетерминированные переходы в модели поведения установки.

В [50] для описания УПО и его модели окружения используется язык poST [53] — процесс-ориентированное расширение языка ST стандарта МЭК 61131-3. Недетерминизм окружения обеспечивается за счёт переменных с абсолютно недетерминированным поведением. Разработан транслятор с языка poST на Promela — входной язык верификатора Spin.

Синхронный язык [56] программирования Lustre [54] для реактивных систем можно рассматривать как подмножество темпоральной логики [57]. В работе [51] предлагается использовать Lustre для разработки и верификации критичных систем реального времени: программа, её окружение и проверяемые свойства задаются на одном языке. Поведение окружения описывается с помощью механизма *утверждений* (assertion). Согласно [57] любое свойство *безопасности* может быть выражено на Lustre. Для верификации используется инструмент Lesar [58].

Программы на расширении языка Lustre могут быть верифицированы с помощью инструмента проверки модели Kind 2 [59]. Поведение окружения и проверяемые свойства для компонентов Lustre-программы задаются с помощью контрактов в стиле *«предположение-гарантия»* (assume-guarantee). Контракты записываются на этом же языке. Язык CoCoSpec [55] является расширением языка Lustre и предназначен для спецификации контрактов реактивных систем с учетом режимов.

Отметим, что использование специализированных и стандартизированных нотаций [52—55] делает спецификации удобочитаемыми, однако их выразительные возможности ограничены: могут возникнуть трудности (в том числе непреодолимые) при описании *мгновенных циклических зависимостей* (instantaneous cyclic dependency) [60], а также ограничений *справедливости* [16, 57].

В работе [61] подчеркивается, что при верификации КФС необходимо моделирование её физических аспектов. Авторы предлагают разделение КФС на кибер- и физическую части с фиксацией интерфейса между ними. Модель поведения определяется только для киберчасти, поведение физической части сводится к описанию поведения интерфейса между ними. Также отмечается, что при наличии определенных знаний о поведении среды требуется добавить дополнительный компонент для её моделирования. Таким образом, данная работа предлагает гибридный подход к верификации, который основан на модели поведения окружения и на ограничениях входов киберчасти. В этом заключается сходство с нашей работой.

Для описания киберчасти используется Lingua Franca (LF) — язык программирования КФС, основанный на *реакторах*. LF-программа переводится на язык реактивных объектов Rebeca — входной язык инструмента проверки модели Afra. Внешние недетерминированные воздействия моделируются через серверы сообщений.

В [62] выполняется автоматический перевод LF-программ в формальную аксиоматическую модель, которая подлежит верификации методом *ограниченной проверки модели* (bounded model

checking) с помощью инструмента Uclid5. Требуемые свойства модели задаются в виде формул Safety Metric Temporal Logic (Safety MTL).

В работе [61] представлен простой пример контроллера дверей поезда, в [62] — системы помощи водителю автомобиля. Данные примеры демонстрируют незамкнутые КФС. В [61, 62] авторы не раскрывают особенностей разработки и верификации систем с замкнутым контуром, поэтому трудно оценить достоинства и недостатки их работ в этом отношении.

Lutin [63] является специализированным языком спецификации недетерминированного поведения. Язык описывает ограничения, накладываемые на хаотическое поведение. Основная цель языка — моделирование (выполнение) сценариев: по Lutin-спецификации генерируются тестовые последовательности для инструмента Lurette. Также язык может использоваться для прототипирования — описания поведения незавершенных компонентов реактивной системы и её окружения. Lutin не ориентирован на верификацию методом проверки модели, однако имеет такой же принцип описания поведения, как и в настоящей работе, который заключается в наложении ограничений на абсолютный недетерминизм.

В синтезе реактивных систем [15, 64] GR(1) [15, 65] является широко используемым фрагментом логики LTL, который ограничивает вид LTL-формул, описывающих предположения (assumptions) относительно окружения синтезируемой системы. В GR(1) для описания эволюции системы используется формула вида $\mathbf{G}(\varphi \Rightarrow \mathbf{X}\psi)$, а для описания целей, которые необходимо достигать бесконечно часто, — формула вида $\mathbf{GF}\gamma$, где φ, ψ и γ — формулы состояния. В настоящей работе также предлагается описывать поведение окружения УПО в виде LTL-формул специального вида, однако есть некоторые отличия. В GR(1) описание допустимых переходов выглядит следующим образом: для некоторого множества исходных состояний (удовлетворяющих φ) задаётся ограничение (в виде ψ) на состояния, которые являются их непосредственными потомками. В настоящей работе на заданное множество переходов для конкретной булевой переменной накладываются ограничения на состояния, образующие эти переходы. Другое отличие: в спецификации GR(1) допускается только *безусловная справедливость* [16], в LTL-спецификации ограниченного недетерминизма — более сложные формы справедливости.

Ряд работ [66—71] направлен на синтез моделей поведения. Работа [66] освещает вопрос идентификации модели поведения замкнутой системы «контроллер-установка» на основе сигналов их интерфейса. Результатом идентификации является недетерминированный автономный автомат с выходом (non deterministic autonomous automaton with output).

В работе [67] представлен автоматический вывод модели поведения установки на основе трассировок выполнения и их LTL-свойств. Трассировки могут быть получены в результате симуляции или реальной работы установки. Вывод модели в виде недетерминированного *автомата Мура* сводится к задаче *выполнимости* (satisfiability, SAT). Проверка CTL-свойств модели замкнутой системы осуществляется с помощью инструмента NuSMV. В подобной работе [68] предлагается вывод модели только на основе трассировок, полученных в результате симуляции. Более масштабируемые методы генерации для моделей большой размерности были представлены в [69].

В [70] на основе трассировок выполнения, полученных от цифрового двойника установки, генерируется её SMV-спецификация. УПО в виде функциональных блоков стандарта МЭК 61499 также транслируется в SMV-код. Далее полученные SMV-спецификации объединяются и результирующая модель замкнутой системы проверяется с помощью инструмента NuSMV.

В [71] при симуляции работы установки формируется журнал событий, на основании которого алгоритм альфа-обнаружения процессов формирует модель поведения в виде сети Петри. Далее сеть Петри преобразуется в функциональный блок стандарта МЭК 61499. В результате его трансляции получается SMV-код для дальнейшей верификации в NuSMV.

Для [66—71] отметим, что в результате синтеза на основе трассировок выполнения может получиться неточная модель поведения установки, что недопустимо при формальной верификации, так как это влияет на достоверность полученных результатов. Синтезированные модели поведения установок могут применяться для поиска ошибок в УПО, но не для доказательства их отсутствия.

2. Предварительные сведения об LTL-спецификации поведения программ

В работе [27] для описания поведения управляющих программ представлена LTL-спецификация, которая содержит *основные* $V = \{v_1, \ldots, v_n\}$ и *вспомогательные* $\_V = \{\_v_1, \ldots, \_v_n\}$ переменные, принимающие значения из соответствующих множеств $D_1, \ldots, D_n$. Основные переменные $v_1, \ldots, v_n$ хранят входные и выходные данные, а также внутреннее состояние программы. Вспомогательные переменные $\_v_1, \ldots, \_v_n$ предназначены для возможности обращения к значениям соответствующих основных переменных в предыдущий момент времени. Множество $S = (D_1 \times \ldots \times D_n)^2$ представляет собой пространство возможных состояний программы. Каждый цикл управления приводит к переходу из состояния программы $s \in S$ в состояние $s' \in S$.

В качестве модели поведения программы используется *система переходов* (transition system) $TS = \langle S, S_0, R, P, L\rangle$, где S — множество состояний, $S_0 \subseteq S$ — множество начальных состояний, $R \subseteq S \times S$ — *тотальное* отношение переходов, $P = \{p_1, \ldots, p_m\}$ — множество атомарных утверждений относительно значений переменных из $V \cup \_V$, $L\colon S \to 2^P$ — функция разметки состояний атомарными утверждениями, истинными в этих состояниях. *Полная* система переходов $cTS = \langle S, S_0, R_c, P, L\rangle$, где отношение переходов $R_c = S \times S$, является моделью поведения программы с абсолютно недетерминированным поведением всех её переменных из $V \cup \_V$. Отношение R_c допускает переходы между любыми двумя состояниями $s, s' \in S$.

Обозначим S^ω множество всех бесконечных слов в алфавите S. *Путь* — бесконечная последовательность $\pi \in S^\omega$. Система переходов TS задаёт множество всех возможных в ней путей Π_{TS} — путей, начинающихся в начальных состояниях:

$$\Pi_{TS} = \{\, \pi \in S^\omega \mid \big(\pi(0) \in S_0\big) \wedge (\forall i \ \in \mathbb{N}_0)\ \big(\pi(i), \pi(i+1)\big) \in R \,\},$$

где $\pi(i)$ — i-ое состояние пути π, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

Модель поведения программы TS получается в результате ограничений, накладываемых на полную систему переходов cTS. Ограничения задаются с помощью линейной темпоральной логики LTL (Linear Temporal Logic). Пусть $p \in P$, тогда синтаксис LTL-формул определяется грамматикой:

$$\varphi, \psi ::= \ \mathit{true} \mid \mathit{false} \mid p \mid \neg\,\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \Rightarrow \psi \mid \varphi \Leftrightarrow \psi \mid \mathbf{X}\varphi \mid \psi\mathbf{U}\varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\varphi.$$

Индуктивно определим отношение выполнимости $\models$ формулы φ логики LTL для произвольного состояния s_i, где $i \in \mathbb{N}_0$, некоторого пути $\pi = s_0\, s_1\, s_2 \ldots$ системы переходов:

$$\begin{array}{ll}
s_i \models \mathit{true}; & s_i \not\models \mathit{false};\\
s_i \models p & \Longleftrightarrow p \in L(s_i);\\
s_i \models \neg\,\varphi & \Longleftrightarrow s_i \not\models \varphi;\\
s_i \models \varphi \wedge \psi & \Longleftrightarrow s_i \models \varphi \ \text{ и } \ s_i \models \psi;\\
s_i \models \varphi \vee \psi & \Longleftrightarrow s_i \models \varphi \ \text{ или } \ s_i \models \psi;\\
s_i \models \varphi \Rightarrow \psi & \Longleftrightarrow s_i \models \neg\varphi \ \text{ или } \ s_i \models \psi;\\
s_i \models \varphi \Leftrightarrow \psi & \Longleftrightarrow s_i \models \varphi \Rightarrow \psi \ \text{ и } \ s_i \models \psi \Rightarrow \varphi;\\
s_i \models \mathbf{X}\varphi & \Longleftrightarrow s_{i+1} \models \varphi;\\
s_i \models \psi\mathbf{U}\varphi & \Longleftrightarrow (\exists k \geq i)\ s_k \models \varphi \ \text{ и } \ (\forall j,\ i\ \leq j < k)\ s_j \models \psi;\\
s_i \models \mathbf{F}\varphi & \Longleftrightarrow (\exists k \geq i)\ s_k \models \varphi;\\
s_i \models \mathbf{G}\varphi & \Longleftrightarrow (\forall j \geq i)\ s_j \models \varphi.
\end{array}$$

Также определим семантику отношения $\models$ на путях и системах переходов:

$$\begin{aligned}
\pi &\models \varphi \iff \pi(0) \models \varphi;\\
\Pi &\models \varphi \iff (\forall \pi \in \Pi)\ \pi \models \varphi;\\
TS, s &\models \varphi \iff (\forall \pi \in \Pi_{TS})\ [(\pi(0) = s) \Rightarrow (\pi \models \varphi)];\\
TS &\models \varphi \iff (\forall s \in S_0)\ TS, s \models \varphi.
\end{aligned}$$

Декларативная LTL-спецификация поведения переменной $v \in V$ имеет следующий вид [27]:

$$\begin{aligned}
&\mathbf{GX}\big(\neg(v = _v) \Rightarrow cond_1 \wedge (v = expr_1) \vee \ldots \vee cond_k \wedge (v = expr_k)\big) \wedge\\
&\mathbf{GX}\big(\ \ (v = _v) \Rightarrow \neg\,(cond_1 \vee \ldots \vee cond_k)\ \ \big).
\end{aligned} \tag{1}$$

Данная формула описывает, каким образом происходит изменение значения переменной $v \in V$. Логическое выражение $cond_i$ является необходимым и достаточным условием для изменения значения переменной v согласно выражению $expr_i$, где $i = 1, \ldots, k$. В выражениях $cond_i$ и $expr_i$ могут использоваться любые переменные из $(V \cup \_V) \setminus \{v\}$, константы, логические и арифметические операторы, а также операторы сравнения. Выражения вида $v = \_v$ и $v = expr_i$ являются элементарными высказываниями из множества P, а выражения вида $cond_i$ — пропозициональными формулами.

Декларативная LTL-спецификация (1) поведения переменной $v \in V$ должна удовлетворять условию *изменчивости* значения переменной [27]:

$$\mathbf{GX}\big(cond_1 \Rightarrow \neg\,(\_v = expr_1)\big) \wedge \ldots \wedge \mathbf{GX}\big(cond_k \Rightarrow \neg\,(\_v = expr_k)\big), \tag{2}$$

т. е. при истинном условии $cond_i$ выражение $expr_i$ должно возвращать значение, отличное от предыдущего значения переменной v. Также для (1) должна выполняться *ортогональность* условий изменения значения переменной для любых $i, j = 1, \ldots, k$ при $i \neq j$ [27]:

$$\mathbf{GX}\big(cond_i \Rightarrow \neg cond_j\big), \tag{3}$$

т. е. одновременно истинным может быть не более одного условия $cond_i$.

При задании выражений $expr_1, \ldots, expr_k$ LTL-спецификация поведения переменной $v_i \in V$ должна удовлетворять условию *ограниченности*:

$$\mathbf{GX}\big(cond_1 \Rightarrow (val(expr_1) \in D_i)\big) \wedge \ldots \wedge \mathbf{GX}\big(cond_k \Rightarrow (val(expr_k) \in D_i)\big), \tag{4}$$

т. е. если условие $cond_j$ истинно ($j = 1, \ldots, k$), то выражение $expr_j$ возвращает значение $val(expr_j)$, которое принадлежит области значений данной переменной.

Декларативная LTL-спецификация (1) задаёт строго детерминированное поведение переменной $v \in V$. Декларативная LTL-спецификация поведения программы — формула $\varphi = \varphi_{var} \wedge \varphi_0$, где φ_{var} — конъюнкция формул вида (1) для всех переменных из V (кроме входных) при соблюдении условий (2), (3) и (4), φ_0 — формула инициализации. LTL-спецификация φ должна удовлетворять условию *запрета мгновенных циклических зависимостей* [28]:

$$\forall v \in V\ [(v, v) \notin Dep^T], \tag{5}$$

где $Dep \subseteq V \times V$ — отношение непосредственной зависимости переменных, его транзитивное замыкание Dep^T задаёт отношение непосредственной и опосредованной зависимости переменных. Переменная v непосредственно зависит от переменной v', т. е. $(v, v') \in Dep$, если в декларативной LTL-спецификации (1) $cond_i$ или $expr_i$, где $i = 1, \ldots, k$, содержит переменную $v' \in V$.

Императивная LTL-спецификация поведения переменной $v \in V$ [27]:

$$\mathbf{GX}\big(cond_1 \Rightarrow (v = expr_1)\big) \wedge \ldots \wedge \mathbf{GX}\big(cond_k \Rightarrow (v = expr_k)\big) \wedge \\ \mathbf{GX}\big(\neg\, cond_1 \wedge \ldots \wedge \neg\, cond_k \Rightarrow (v = \_v)\big). \tag{6}$$

Данная спецификация описывает поведение переменной в императивном стиле «если…, то…» и является *конструктивной*, т. е. по ней непосредственно может быть построен код программы на императивном языке программирования. Декларативная и императивная LTL-спецификации эквивалентны [27], т. е. задают одно и то же программное поведение.

3. LTL-спецификация поведения ПЛУ

ПЛУ имеют булевы входные и выходные переменные, принимающие значения из множества $\{0, 1\}$. Рассмотрим частный случай декларативной LTL-спецификации (1). Пусть v — булева переменная, $expr_1 \equiv 1$ и $expr_2 \equiv 0$, тогда формула

$$\mathbf{GX}\big(\neg(v = \_v) \Rightarrow cond_1 \wedge (v = 1) \vee cond_2 \wedge (v = 0)\big) \wedge \\ \mathbf{GX}\big(\ \ (v = \_v) \Rightarrow \neg\,(cond_1 \vee cond_2)\ \ \big)$$

может быть переписана следующим образом:

$$\mathbf{GX}\big(\neg(\_v \wedge v \vee \neg\_v \wedge \neg v) \Rightarrow cond_1 \wedge v \vee cond_2 \wedge \neg v\big) \wedge \\ \mathbf{GX}\big(\ \ (\_v \wedge v \vee \neg\_v \wedge \neg v) \Rightarrow \neg\,(cond_1 \vee cond_2)\ \ \big).$$

Поскольку для логических выражений $cond_1$ и $cond_2$ должны выполняться условия изменчивости $cond_1 \Rightarrow \neg(\_v = 1)$ и $cond_2 \Rightarrow \neg(\_v = 0)$, рассмотрим эти выражения в виде $\neg\_v \wedge cnd_1$ и $\_v \wedge cnd_2$ соответственно, где cnd_1 и cnd_2 — логические выражения, выполнимость которых необходима для изменения переменной v без наложения на них условий изменчивости. Также выражения $\neg\_v \wedge cnd_1$ и $\_v \wedge cnd_2$ обеспечивают выполнение ортогональности (3), так как имеют вхождения переменной $\_v$ со знаком отрицания и без него соответственно. Получим LTL-формулу

$$\mathbf{GX}\big(\neg(\_v \wedge v \vee \neg\_v \wedge \neg v) \Rightarrow \neg\_v \wedge cnd_1 \wedge v \vee \_v \wedge cnd_2 \wedge \neg v\big) \wedge \\ \mathbf{GX}\big(\ \ (\_v \wedge v \vee \neg\_v \wedge \neg v) \Rightarrow \neg\,(\neg\_v \wedge cnd_1 \vee \_v \wedge cnd_2)\ \ \big).$$

Вынесем **GX** за скобки и избавимся от знака импликации:

$$\mathbf{GX}\Big(\big((\_v \wedge v) \vee (\neg\_v \wedge \neg v) \vee (\neg\_v \wedge v \wedge cnd_1) \vee (\_v \wedge \neg v \wedge cnd_2)\big) \wedge \\ \big((\_v \wedge \neg v) \vee (\neg\_v \wedge v) \vee (\neg\_v \wedge \neg cnd_1) \vee (\_v \wedge \neg cnd_2)\big)\Big).$$

Раскрыв скобки, получим

$$\mathbf{GX}\big((\neg\_v \wedge v \wedge cnd_1) \vee (\_v \wedge \neg v \wedge cnd_2) \vee (\neg\_v \wedge \neg v \wedge \neg cnd_1) \vee (\_v \wedge v \wedge \neg cnd_2)\big),$$

что эквивалентно формуле

$$\mathbf{GX}\big((\neg\_v \wedge v \Rightarrow cnd_1) \wedge (\_v \wedge \neg v \Rightarrow cnd_2) \wedge (\neg\_v \wedge \neg v \Rightarrow \neg cnd_1) \wedge (\_v \wedge v \Rightarrow \neg cnd_2)\big).$$

Внесём **GX** внутрь скобок, получим:

$$\mathbf{GX}\big(\neg\_v \wedge \ \ v \Rightarrow \ \ cnd_1\big) \wedge \\ \mathbf{GX}\big(\neg\_v \wedge \neg v \Rightarrow \neg cnd_1\big) \wedge \\ \mathbf{GX}\big(\ \ \_v \wedge \neg v \Rightarrow \ \ cnd_2\big) \wedge \\ \mathbf{GX}\big(\ \ \_v \wedge \ \ v \Rightarrow \neg cnd_2\big).$$

Внесём $\mathbf{X}$ внутрь скобок, учитывая, что $\mathbf{X}(\_v) = v$, получим декларативную LTL-спецификацию поведения булевой переменной v:

$$\begin{aligned}
&\mathbf{G}\big(\neg v \wedge \ \ \mathbf{X}(v) \Rightarrow \ \ \mathbf{X}(cnd_1)\big) \wedge \\
&\mathbf{G}\big(\neg v \wedge \neg\mathbf{X}(v) \Rightarrow \neg\mathbf{X}(cnd_1)\big) \wedge \\
&\mathbf{G}\big(\ \ v \wedge \neg\mathbf{X}(v) \Rightarrow \ \ \mathbf{X}(cnd_2)\big) \wedge \\
&\mathbf{G}\big(\ \ v \wedge \ \ \mathbf{X}(v) \Rightarrow \neg\mathbf{X}(cnd_2)\big).
\end{aligned} \tag{7}$$

Из (7) видно, что cnd_1 является *необходимым* условием для изменения значения переменной v с *False* на *True* (строка 1). Также данное условие является *достаточным*, что подтверждается формулой $\mathbf{G}\big(\neg v \wedge \mathbf{X}(cnd_1) \Rightarrow \mathbf{X}(v)\big)$, которая получена в результате контрапозиции во второй строке. Аналогично cnd_2 является необходимым и достаточным условием для обратного изменения значения переменной v с *True* на *False*. Таким образом, набор ограничений в (7) допускает только строго детерминированное поведение.

Отметим, что если $cnd_2 = \neg cnd_1$, то LTL-спецификацию (7) можно сократить. Получим *сокращённую* декларативную LTL-спецификацию поведения булевой переменной v:

$$\mathbf{G}\big(\ \mathbf{X}(v) \Leftrightarrow \mathbf{X}(cnd_1)\ \big). \tag{8}$$

Поведение всех булевых переменных ПЛУ (кроме входных) будем описывать с помощью декларативной LTL-спецификации — формулы (7) или (8).

Выполним контрапозицию в (7), получим:

$$\begin{aligned}
&\mathbf{G}\big(\neg v \wedge \neg\mathbf{X}(cnd_1) \Rightarrow \neg\mathbf{X}(v)\big) \wedge \\
&\mathbf{G}\big(\neg v \wedge \ \ \mathbf{X}(cnd_1) \Rightarrow \ \ \mathbf{X}(v)\big) \wedge \\
&\mathbf{G}\big(\ \ v \wedge \neg\mathbf{X}(cnd_2) \Rightarrow \ \ \mathbf{X}(v)\big) \wedge \\
&\mathbf{G}\big(\ \ v \wedge \ \ \mathbf{X}(cnd_2) \Rightarrow \neg\mathbf{X}(v)\big).
\end{aligned}$$

Сгруппируем отдельно подформулы в первой и третьей строках:

$$\begin{aligned}
&\mathbf{G}\big(\neg v \wedge \mathbf{X}(cnd_1) \Rightarrow \ \ \mathbf{X}(v)\big) \wedge \\
&\mathbf{G}\big(\ \ v \wedge \mathbf{X}(cnd_2) \Rightarrow \neg\mathbf{X}(v)\big) \wedge \\
&\mathbf{G}\big(\neg v \wedge \neg\mathbf{X}(cnd_1) \Rightarrow \neg\mathbf{X}(v)\big) \wedge \mathbf{G}\big(v \wedge \neg\mathbf{X}(cnd_2) \Rightarrow \mathbf{X}(v)\big).
\end{aligned}$$

После преобразования получим императивную LTL-спецификацию поведения булевой переменной:

$$\begin{aligned}
&\mathbf{G}\big(\neg v \wedge \mathbf{X}(cnd_1) \Rightarrow \ \ \mathbf{X}(v)\big) \wedge \\
&\mathbf{G}\big(\ \ v \wedge \mathbf{X}(cnd_2) \Rightarrow \neg\mathbf{X}(v)\big) \wedge \\
&\mathbf{G}\big(\neg[\neg v \wedge \mathbf{X}(cnd_1)] \wedge \neg[v \wedge \mathbf{X}(cnd_2)] \Rightarrow [v \Leftrightarrow \mathbf{X}(v)]\big).
\end{aligned} \tag{9}$$

Далее формулу (9) будем использовать для построения императивного кода программы.

Отметим, что внутренние переменные ПЛУ могут иметь любой тип — не только булев. В этом случае их поведение описывается с помощью декларативной LTL-спецификации общего вида (1).

4. LTL-спецификация поведения окружения ПЛУ

Согласно предлагаемому в данной работе подходу поведение окружения представляет собой поведение входных переменных ПЛУ и мнимых датчиков. Под мнимыми датчиками понимаются дополнительные булевы переменные, описывающие состояния компонентов из окружения. Цель

введения мнимых датчиков состоит в компенсации недостающих датчиков для отслеживания специфического поведения, которое необходимо учесть при задании реалистичного поведения входов ПЛУ. Как уже было отмечено, в настоящей работе мы ограничиваемся рассмотрением только ПЛУ — программ с булевыми входными и выходными переменными. Поведение входных переменных ПЛУ и мнимых датчиков является абсолютно недетерминированным, так как не описывается в декларативной LTL-спецификации. Их поведение будем задавать путем наложения ограничений на абсолютный недетерминизм.

Автоматная модель *абсолютного недетерминизма* булевой переменной представлена на рис. 1a. Из графа переходов видно, что переменная может принимать любые значения независимо от внешних условий. Автоматная модель *ограниченного недетерминизма* булевой переменной представлена на рис. 1b. Каждый из четырёх возможных переходов имеет ограничение: переход возможен только при выполнении указанного на нём условия $cond_{ij}$, выраженного в виде пропозициональной формулы, где i и j — исходное и новое значения булевой переменной соответственно. Например, если переменная изменила своё значение с *False* на *True*, то обязательно выполняется условие $cond_{01}$ в тот момент, когда значение переменной равно *True*. Таким образом, $cond_{ij}$ является только *необходимым* (не является *достаточным*) условием для совершения перехода.

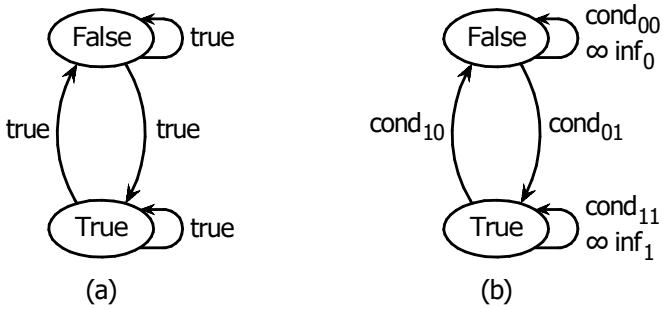


Fig. 1. Nondeterminism of a boolean variable: absolute (a), constrained (b)

Рис. 1. Недетерминизм булевой переменной: абсолютный (a), ограниченный (b)

На рис. 1b каждая петля имеет дополнительное условие inf_x, выраженное в виде LTL-формулы, где x — значение булевой переменной при совершении данного перехода. «Залипание» в состоянии x (бесконечное совершение перехода в это состояние по петле) разрешено только при выполнении условия inf_x. В связи с этим в графической нотации перед условием inf_x стоит знак бесконечности «∞». Например, если переменная изменила своё значение с *False* на *True* и начиная с этого момента истинна LTL-формула inf_1, то «залипание» в состоянии *True* разрешено. Таким образом, inf_x является *необходимым* условием для «залипания».

Формализуем вышеприведённое описание — LTL-спецификация *ограниченно недетерминированного* поведения булевой переменной имеет вид:

$$\begin{aligned}
&\mathbf{G}\big(\neg v \wedge \ \ \mathbf{X}(v) \Rightarrow \ \mathbf{X}(cond_{01})\big) \wedge \\
&\mathbf{G}\big(\neg v \wedge \neg\mathbf{X}(v) \Rightarrow \ \mathbf{X}(cond_{00})\big) \wedge \\
&\mathbf{G}\big(\ \ v \wedge \neg\mathbf{X}(v) \Rightarrow \ \mathbf{X}(cond_{10})\big) \wedge \\
&\mathbf{G}\big(\ \ v \wedge \ \ \mathbf{X}(v) \Rightarrow \ \mathbf{X}(cond_{11})\big) \wedge \\
&\quad \mathbf{G}\big(\mathbf{G}(\neg v) \Rightarrow \ inf_0\big) \wedge \\
&\quad \mathbf{G}\big(\mathbf{G}(\ \ v) \Rightarrow \ inf_1\big).
\end{aligned} \tag{10}$$

Первые четыре строки формулы описывают необходимые условия при совершении каждого из четырёх переходов. Отметим, что если, например, $cond_{01} \equiv 1$, то переход из *False* в *True* всегда разрешён,

если $cond_{01} \equiv 0$, то — всегда запрещён. Последние две строки формулы описывают необходимые условия при «залипании» в каждом из состояний. Например, последняя строка формулы утверждает, что всегда, если произошло «залипание» в состоянии *True* (всегда $v = True$), то LTL-формула inf_1 выполняется в начальный момент этого «залипания».

С помощью формулы (10) может быть описано как недетерминированное, так и детерминированное поведение булевой переменной. Если установить следующие ограничения: $cond_{00} \equiv \neg cond_{01}$, $cond_{11} \equiv \neg cond_{10}$, $inf_0 \equiv 1$, $inf_1 \equiv 1$, то получим декларативную LTL-спецификацию поведения булевой переменной (7), где $cnd_1 = cond_{01}$ и $cnd_2 = cond_{10}$. Таким образом, декларативная LTL-спецификацию поведения булевой переменной (7) является частным случаем LTL-спецификации ограниченно недетерминированного поведения (10).

Далее формулу (10) будем использовать для описания поведения окружения ПЛУ. При этом рекомендуется условия $cond_{00}$, $cond_{01}$, $cond_{10}$, $cond_{11}$, inf_0, inf_1 задавать максимально *слабыми*, чтобы LTL-спецификация допускала как можно больше возможных путей, пусть даже не существующих. В этом случае поведение КФС будет иметь больше путей, чем в действительности. Если LTL-спецификация КФС задаёт множество путей Π^+, а фактическое множество путей $\Pi \subset \Pi^+$, то из $\Pi^+ \models \psi$ следует $\Pi \models \psi$, где ψ — проверяемое свойство. Таким образом, не нужно стремиться описывать максимально реалистичную модель, нужно вводить только самые очевидные ограничения, чтобы захватить как можно больше путей и отбросить как можно меньше. А при попытке описания максимально реалистичной модели поведения возникает риск отбросить пути, входящие в Π — получим множество $\Pi^- \subset \Pi$. В этом случае верификация будет проводиться на некорректной модели, что ставит под сомнения полученный результат, так как из $\Pi^- \models \psi$ не следует $\Pi \models \psi$.

5. LTL-спецификация поведения установки для литья пластмасс

5.1. Описание установки для литья пластмасс

В качестве примера рассмотрим LTL-спецификацию поведения замкнутой системы «ПЛУ-ОУ» — установки для литья пластмасс, схема которой представлена на рис. 2. Гранулированный пластик хранится в бункере (Storage Bunker for Granular Plastic) и с помощью шнекового механизма подачи (Feed Mechanism) транспортируется в дозатор (Dosing Unit) при открытой крышке (Lid). Дозатор имеет электрический нагреватель (*Heater*) для плавления гранул пластика и весоизмерительную платформу (Weighing Platform) для определения собственного веса. При открытом клапане (*Valve*) расплавленный пластик из дозатора по сливному трубопроводу подаётся в литейную форму (Form). С помощью конвейера (Conveyor) осуществляется транспортировка формы справа налево.

Система управления установкой построена на базе ПЛК — его УПО координирует работу оборудования. Оператор взаимодействует с установкой посредством кнопок и ламп, расположенных на панели. В составе УПО выделим ПЛУ, интерфейс которой представлен на рис. 3.

Интерфейс взаимодействия с панелью оператора. Вход *PBStart* принимает команду запуска установки (Plant Start Command) от кнопки (Button), расположенной на панели (Panel), вход *PBStop* — команду немедленного останова установки (Plant Stop Command), вход *PBCompl* — команду плавного завершения работы установки (Plant Complete Work Command), вход *PBConvr* — команду включения конвейера (Conveyor Turn On Command) в ручном режиме.

Также панель оператора содержит ряд ламп (Lamps) для индикации текущих режимов работы и состояний. Выход *SysOn* сигнализирует о включении системы (System is On Mode) в автоматическом режиме, выход *Compl* — о процессе плавного завершения работы (Completion Mode), выход *FErr* — об ошибке подачи пластика в дозатор (Feed Error), выход *CErr* — об ошибке работы конвейера (Conveyor Error), выход *HErr* — об ошибке работы нагревателя (Heater Error), выход *Disch* — о режиме разгрузки пластика из дозатора (Plastic Discharge Mode), выход *Mltng* — о режиме плавлении пластика (Plastic Melting Mode), выход *Mlted* — о завершении плавления пластика (Plastic is Melted).

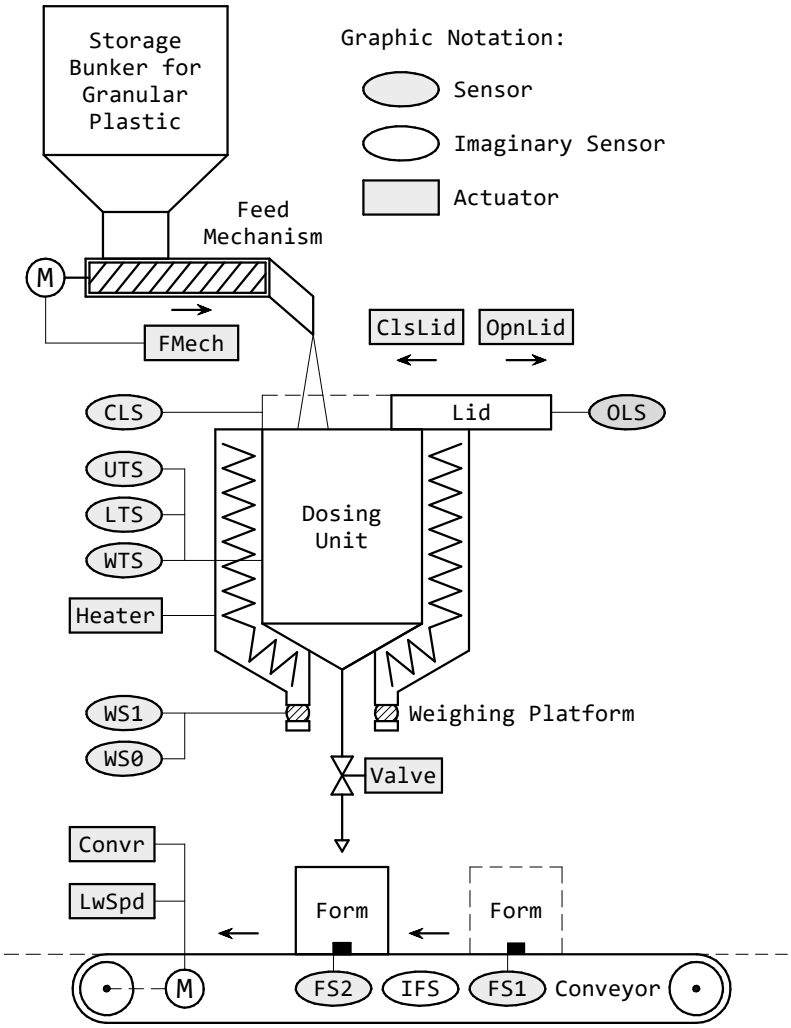


Fig. 2. Plastic molding plant diagram

Рис. 2. Схема установки для литья пластмасс

Интерфейс взаимодействия с установкой. Установка (Plant) оснащена датчиками (Sensors) и исполнительными устройствами (Actuators). Их расположение изображено на рис. 2, а названия совпадают с соответствующими входными/выходными сигналами ПЛУ. Датчики изображены в виде серых овалов, исполнительные устройства — в виде серых прямоугольников. Вход ПЛУ *FS1* принимает сигнал от датчика наличия формы на первой позиции (Form on Position 1) конвейера, вход *FS2* — от датчика наличия формы на второй позиции (Form on Position 2) конвейера, вход *OLS* — от датчика открытой крышки (Lid is Opened), вход *CLS* — от датчика закрытой крышки (Lid is Closed), вход *WS0* — от датчика веса о том, что дозатор не пуст (Dosing Unit is not Empty), вход *WS1* — от датчика веса о том, что дозатор заполнен (Dosing Unit is Full), вход *UTS* — от датчика высокой температуры дозатора (Dosing Unit Temperature at the Up Level), вход *LTS* — от датчика низкой температуры дозатора (Dosing Unit Temperature at the Low Level), вход *WTS* — от датчика рабочей температуры дозатора (Dosing Unit Temperature at the Working Level). Пластик плавится при рабочей температуре. Низкая и высокая температуры выше рабочей и предназначены для управления нагревателем.

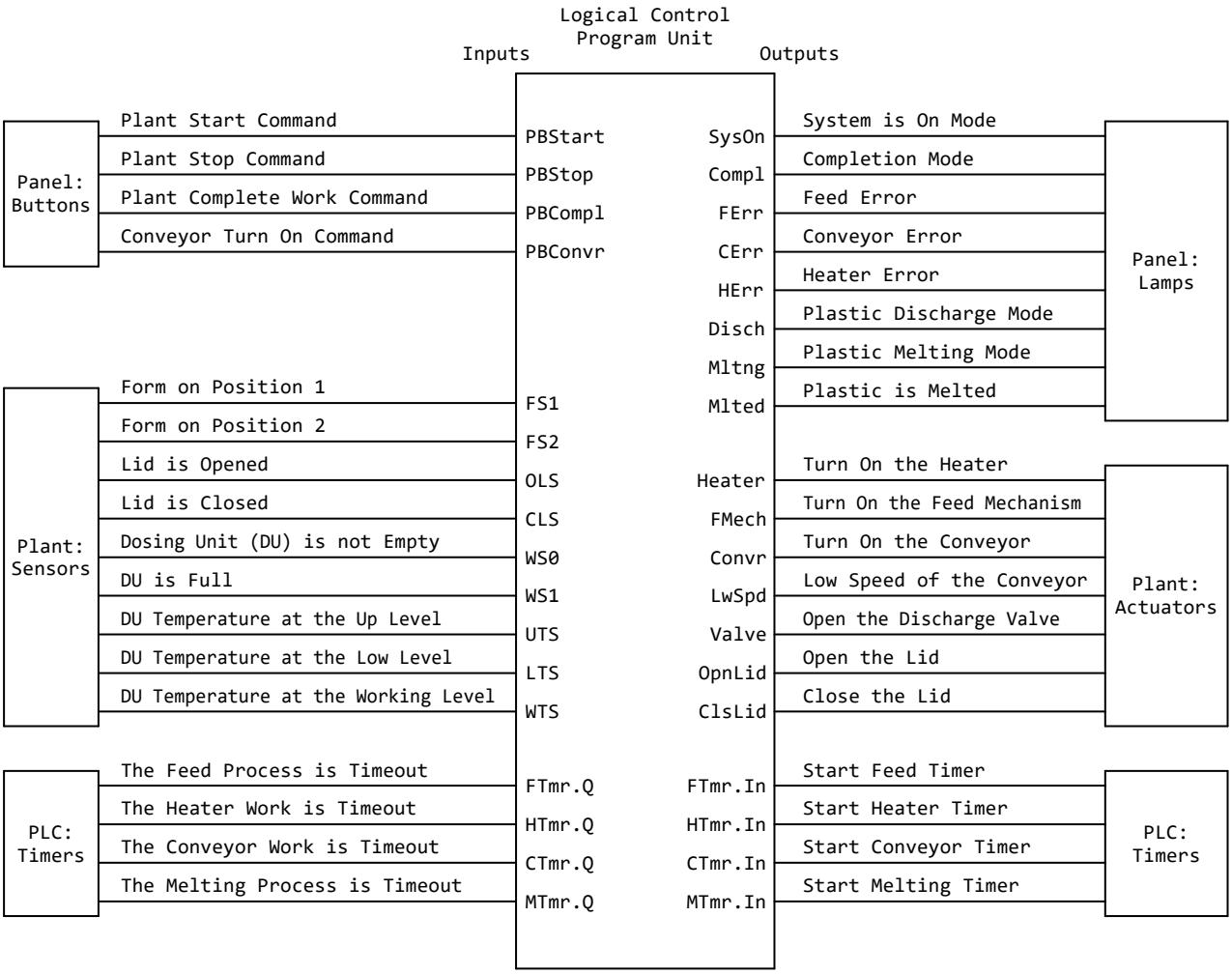


Fig. 3. Logical control program interface

Рис. 3. Интерфейс программы логического управления

Выход ПЛУ *Heater* подаёт сигнал включения нагревателя (Turn On the Heater), выход *FMech* — сигнал включения механизма подачи (Turn On the Feed Mechanism) пластика, выход *Convr* — сигнал включения конвейера (Turn On the Conveyor), выход *LwSpd* — сигнал установки пониженной скорости конвейера (Low Speed of the Conveyor), выход *Valve* — сигнал открытия выпускного клапана (Open the Discharge Valve), выход *OpnLid* — сигнал открытия крышки (Open the Lid) дозатора, выход *ClsLid* — сигнал закрытия крышки (Close the Lid) дозатора.

Интерфейс взаимодействия с таймерами. Для работы с временными интервалами УПО ПЛК (PLC) содержит таймеры (Timers). Вход ПЛУ *FTmr.Q* принимает сигнал от таймера об истечении времени процесса подачи (The Feed Process is Timeout), вход *HTmr.Q* — сигнал об истечении времени прогрева нагревателя (The Heater Work is Timeout), вход *CTmr.Q* — сигнал об истечении времени ожидания подачи формы конвейером (The Conveyor Work is Timeout), вход *MTmr.Q* — сигнал об истечении времени процесса плавления (The Melting Process is Timeout). Выходы ПЛУ *FTmr.In*, *HTmr.In*, *CTmr.In* и *MTmr.In* предназначены для запуска вышеуказанных таймеров. Таким образом, ПЛУ получает сигналы обратной связи от датчиков установки и таймеров ПЛК.

Работа установки. При получении команды запуска установки (*PBStart*) включается регулирование температуры и начинается работа оборудования по циклу. Регулятор температуры работает вне цикла и поддерживает рабочую температуру (*WTS*): включает нагреватель (*Heater*) при понижении температуры ($\neg LTS$), выключает ($\neg Heater$) — при повышении (*UTS*), где $\neg LTS$ означает $LTS = False$. Если работа нагревателя не позволяет достичь рабочей температуры (*WTS*) в течение заданного времени или произошло снижение температуры ($\neg WTS$) при включенном нагревателе, то устанавливается ошибка его работы (*HErr*).

Рассмотрим цикл работы оборудования установки. Изначально включается механизм подачи (*FMech*) и поддерживается его работа до тех пор, пока не сработает датчик веса (*WS1*), сигнализирующий о заполнении дозатора. Если время процесса подачи вышло ($FTmr.Q$) и дозатор не заполнен, то выдается сигнал ошибки (*FErr*) и работа останавливается. Данная ошибка может возникнуть из-за пустого бункера или поломки механизма подачи.

Параллельно с механизмом подачи включается конвейер (*Convr*), который транспортирует форму в направлении справа налево к месту для литья. Каждая форма оснащена меткой (чёрный прямоугольник на рис. 2), на которую реагируют датчики *FS1* и *FS2*. При достижении формой (выделена штриховой линией) первой позиции (*FS1*) конвейер снижает скорость (*LwSpd*). Далее транспортировка продолжается на пониженной скорости для повышения точности координаты остановки. Конвейер останавливается ($\neg Convr$), когда форма (выделена основной линией) достигнет второй позиции (*FS2*). Если время работы конвейера вышло ($CTmr.Q$) и форма не на второй позиции ($\neg FS2$), то выдается сигнал ошибки (*CErr*) и работа установки останавливается. Предполагается, что конвейер всегда работает исправно и причиной ошибки *CErr* может быть только отсутствие формы на ленте конвейера.

После заполнения дозатора его крышка закрывается (*ClsLid*) и активируется разгрузочный режим (*Disch*). Если крышка закрыта (*CLS*) и дозатор имеет рабочую температуру (*WTS*), то запускается таймер отсчёта времени плавления ($MTmr.In$). По истечении данного времени ($MTmr.Q$) если форма находится на рабочей позиции (*FS2*), то открывается клапан (*Valve*) для её заполнения. После опустошения дозатора ($\neg WS0$) клапан закрывается ($\neg Valve$), на нормальной скорости ($\neg LwSpd$) включается конвейер (*Convr*) и освобождает место для новой формы. Также закрытие клапана ($\neg Valve$) влечёт открытие крышки (*OpnLid*) дозатора. После её открытия (*OLS*) цикл работы повторяется.

О работе установки в автоматическом режиме свидетельствует сигнал *SysOn*. При наличии любой ошибки (*FErr*, *CErr*, *HErr*) или нажатии кнопки останова (*PBStop*) работа установки немедленно прекращается. Повторное нажатие кнопки *PBStop* приводит к сбросу сигналов об ошибках. Нажатие кнопки включения конвейера (*PBConvr*) запускает его (*Convr*) в ручном режиме и отключает автоматический режим работы установки ($\neg SysOn$). Кнопка плавного завершения работы (*PBCompl*) устанавливает специальный режим (*Compl*), который завершает текущий цикл и не начинает новый.

5.2. LTL-спецификация поведения ПЛУ

Рассмотрим декларативную LTL-спецификацию поведения переменной *Heater*, которая управляет работой нагревателя:

$$\begin{aligned}
&\mathbf{G}\big(\neg Heater \wedge \ \ \mathbf{X}(Heater) \Rightarrow \ \ (\mathbf{X}(SysOn) \wedge \neg\mathbf{X}(LTS))\big) \wedge \\
&\mathbf{G}\big(\neg Heater \wedge \neg\mathbf{X}(Heater) \Rightarrow \ \neg(\mathbf{X}(SysOn) \wedge \neg\mathbf{X}(LTS))\big) \wedge \\
&\mathbf{G}\big(\ \ Heater \wedge \neg\mathbf{X}(Heater) \Rightarrow \ \ (\neg\mathbf{X}(SysOn) \vee \mathbf{X}(UTS))\big) \wedge \\
&\mathbf{G}\big(\ \ Heater \wedge \ \ \mathbf{X}(Heater) \Rightarrow \ \neg(\neg\mathbf{X}(SysOn) \vee \mathbf{X}(UTS))\big).
\end{aligned} \tag{11}$$

Включение нагревателя ($\neg Heater \wedge \mathbf{X}(Heater)$) происходит при работе системы (*SysOn*) и относительно низком уровне температуры (отсутствует сигнал *LTS*). Нагреватель выключается ($Heater \wedge$

$\neg\mathbf{X}(Heater)$) при отключении системы (*SysOn*) или при достижении высокого уровня температуры (присутствует сигнал *UTS*).

Аналогичным образом зададим поведение всех переменных программы в виде декларативной LTL-спецификации в синтаксисе инструмента nuXmv (см. приложение 1). Отметим, что для описания поведения программы была введена одна дополнительная внутренняя булева переменная *Fin*, которая сигнализирует о необходимости завершения процесса работы. Она получает истинное значение, если либо был активен режим плавного завершения (*Compl*) при открытой заслонке (*OLS*) и опустошённом дозаторе ($\neg$*WS0*), либо выставлена хотя бы одна из ошибок *FErr*, *HErr*, *CErr*, либо нажата кнопка *PBStop* или кнопка *PBConvr*. В противном случае сигнал *Fin* сбрасывается.

5.3. LTL-спецификация поведения окружения ПЛУ

Рассмотрим построение LTL-спецификации поведения для датчиков конвейера. Предполагается, что поведение датчиков положения формы *FS1* и *FS2* взаимосвязано: датчик *FS2* не может сработать, если до этого не сработал датчик *FS1*. Из рис. 2 видно, что форма может находиться в промежуточном положении между двумя датчиками, когда ни один из них не выдает сигнала о наличии формы. Таким образом, для описания поведения датчика *FS2* невозможно использовать только текущие показания датчика *FS1* — этого недостаточно, нужно запоминать, что датчик *FS1* сработал в прошлом до момента срабатывания датчика *FS2*. Для этой цели введём мнимый датчик наличия формы *IFS* (Imaginary Form Sensor) между датчиками *FS1* и *FS2*. Мнимый датчик *IFS* изображён на рис. 2 в виде овала с незакрашенным фоном. Данный датчик будет использоваться при описании поведения датчика *FS2*: датчик *FS2* не может сработать, если в этот момент не исчезает сигнал датчика *IFS*. Формально опишем поведение датчика *FS2* согласно (10), получим:

$$
\begin{aligned}
1:&\quad \mathbf{G}\big(\neg FS2 \wedge \ \mathbf{X}(FS2) \Rightarrow\ IFS \wedge \neg\mathbf{X}(IFS) \wedge \neg FS1 \wedge \neg\mathbf{X}(FS1) \wedge Convr\big) \wedge\\
2:&\quad \mathbf{G}\big(\neg FS2 \wedge \neg\mathbf{X}(FS2) \Rightarrow\ true\big) \wedge\\
3:&\quad \mathbf{G}\big(\ FS2 \wedge \neg\mathbf{X}(FS2) \Rightarrow\ \neg IFS \wedge \neg\mathbf{X}(IFS) \wedge \neg FS1 \wedge \neg\mathbf{X}(FS1) \wedge Convr\big) \wedge\\
4:&\quad \mathbf{G}\big(\ FS2 \wedge \ \mathbf{X}(FS2) \Rightarrow\ \neg IFS \wedge \neg\mathbf{X}(IFS) \wedge \neg FS1 \wedge \neg\mathbf{X}(FS1)\big) \wedge\\
5:&\quad \mathbf{G}\big(\mathbf{G}(\neg FS2) \Rightarrow\ true\big) \wedge\\
6:&\quad \mathbf{G}\big(\mathbf{G}(\ FS2) \Rightarrow\ \mathbf{F}(\mathbf{G}(\neg Convr))\big).
\end{aligned}
\tag{12}
$$

Первая строка формулы (12) описывает необходимые условия для включения датчика *FS2*: датчик *IFS* был включен, а сейчас отключён, датчик *FS1* был отключён и сейчас отключён, конвейер (*Convr*) был включен в предыдущий момент времени. При выполнении данных условий датчик *FS2* может включиться, а может и остаться в выключенном состоянии, так как непосредственно на этот переход не накладывается никаких ограничений (строка 2). При отключении датчика *FS2* также обязательно должен работать конвейер в предыдущий момент времени (строка 3). Это отражает тот факт, что работа конвейера является причиной, а смена состояния датчика следствием. Причина и следствие не могут быть одновременны — следствие проявляется после причины.

«Залипание» датчика *FS2* в состоянии *False* допустимо при любых условиях (строка 5). Оно может произойти из-за отсутствия формы. Таким образом, форма никогда не достигнет положения для срабатывания датчика *FS2*. «Залипание» датчика *FS2* в состоянии *True* разрешено, только если конвейер в какой-то момент остановится и больше никогда не включится (строка 6). Таким образом, форма никогда не сможет покинуть положение срабатывания датчика *FS2*. В итоге формула (12) описывает возможное поведение сигнала обратной связи *FS2* при управляющем воздействии на конвейер.

Поведение мнимого датчика *IFS* имеет вид:

$$
\begin{aligned}
1:&\quad \mathbf{G}\big(\neg IFS \wedge \ \mathbf{X}(IFS) \Rightarrow\ \neg FS2 \wedge \neg\mathbf{X}(FS2) \wedge FS1 \wedge \neg\mathbf{X}(FS1) \wedge Convr\big) \wedge\\
2:&\quad \mathbf{G}\big(\neg IFS \wedge \neg\mathbf{X}(IFS) \Rightarrow\ true\big) \wedge\\
3:&\quad \mathbf{G}\big(\ \ IFS \wedge \neg\mathbf{X}(IFS) \Rightarrow\ \neg FS2 \wedge\ \ \mathbf{X}(FS2) \wedge \neg FS1 \wedge \neg\mathbf{X}(FS1) \wedge Convr\big) \wedge\\
4:&\quad \mathbf{G}\big(\ \ IFS \wedge\ \ \mathbf{X}(IFS) \Rightarrow\ \neg FS2 \wedge \neg\mathbf{X}(FS2) \wedge \neg FS1 \wedge \neg\mathbf{X}(FS1)\big) \wedge\\
5:&\quad \mathbf{G}\big(\mathbf{G}(\neg IFS) \Rightarrow\ true\big) \wedge\\
6:&\quad \mathbf{G}\big(\mathbf{G}(\ \ IFS) \Rightarrow\ \mathbf{F}(\mathbf{G}(\neg Convr))\big).
\end{aligned}
\tag{13}
$$

При включении датчика *IFS* происходит отключение датчика *FS1*, датчик *FS2* не меняет своего состояния — остается отключённым (строка 1). Отключение датчика *IFS* сопровождается включением датчика *FS2*, датчик *FS1* остается отключённым (строка 3). Условия для «залипаний» датчиков *IFS* и *FS2* совпадают (строки 5 и 6).

Поведение оставшегося датчика конвейера *FS1* имеет вид:

$$
\begin{aligned}
1:&\quad \mathbf{G}\big(\neg FS1 \wedge \ \mathbf{X}(FS1) \Rightarrow\ \neg FS2 \wedge \neg\mathbf{X}(FS2) \wedge \neg IFS \wedge \neg\mathbf{X}(IFS) \wedge Convr\big) \wedge\\
2:&\quad \mathbf{G}\big(\neg FS1 \wedge \neg\mathbf{X}(FS1) \Rightarrow\ true\big) \wedge\\
3:&\quad \mathbf{G}\big(\ \ FS1 \wedge \neg\mathbf{X}(FS1) \Rightarrow\ \neg FS2 \wedge \neg\mathbf{X}(FS2) \wedge \neg IFS \wedge\ \ \mathbf{X}(IFS) \wedge Convr\big) \wedge\\
4:&\quad \mathbf{G}\big(\ \ FS1 \wedge\ \ \mathbf{X}(FS1) \Rightarrow\ \neg FS2 \wedge \neg\mathbf{X}(FS2) \wedge \neg IFS \wedge \neg\mathbf{X}(IFS)\big) \wedge\\
5:&\quad \mathbf{G}\big(\mathbf{G}(\neg FS1) \Rightarrow\ true\big) \wedge\\
6:&\quad \mathbf{G}\big(\mathbf{G}(\ \ FS1) \Rightarrow\ \mathbf{F}(\mathbf{G}(\neg Convr))\big).
\end{aligned}
\tag{14}
$$

При включении датчика *FS1* конвейер работал в предыдущий момент времени, датчики *FS2* и *IFS* остаются в выключенном состоянии (строка 1). При отключении датчика *FS1* также работал конвейер, датчик *FS2* остаётся в выключенном состоянии, датчик *IFS* включается (строка 3).

Отметим, что заданная LTL-спецификация поведения группы датчиков *FS1*, *FS2*, *IFS* задаёт мгновенные циклические зависимости между ними, что запрещено в декларативной LTL-спецификации (1) и её частном случае (7). Например, из (14) в строке 1 видно, что включение датчика *FS1* зависит от текущего состояния датчика *FS2*, который должен быть выключен ($\neg\mathbf{X}(FS2)$). А из (12) в строке 1 видно, что включение датчика *FS2* также зависит от текущего состояния датчика *FS1*, который тоже должен быть выключен ($\neg\mathbf{X}(FS1)$). Таким образом, формулы (14) и (12) задают мгновенную циклическую зависимость между датчиками *FS1* и *FS2*. LTL-спецификация ограниченно недетерминированного поведения (10) допускает такой способ описания зависимостей — это обеспечивает дополнительные выразительные возможности при задании поведения окружения. Запрет на мгновенные циклические зависимости (5) в декларативной LTL-спецификации введён для возможности её трансляции в последовательно исполняемый код на императивном языке программирования.

Поведение трёх датчиков конвейера было описано согласно шаблону (10), который детально отражает все условия для переходов и «залипаний». Однако при задании поведения взаимосвязанной группы переменных может возникнуть противоречие или избыточность. Для исключения данных проблем предлагается в процессе разработки LTL-спецификации выполнять анализ связанной группы переменных с помощью инструмента для работы с LTL-формулами Spot. Все LTL-формулы в (10), которые имеют значение *True* в правой части импликации, избыточны, так как являются тождественно истинными и не накладывают никаких ограничений на поведение, поэтому могут быть исключены из LTL-спецификации — это строки 2 и 5 формул (12), (13), (14). После их удаления получим эквивалентную LTL-спецификацию поведения φ_{sens} группы датчиков *FS1*, *FS2*, *IFS*. Далее с помощью инструмента Spot было выявлено, что строка 4 в (12) является логическим следствием

φ_{sens} при заданных начальных условиях: $\neg FS1 \wedge \neg FS2 \wedge \neg IFS$. Таким образом, строку 4 можно исключить из LTL-спецификации φ_{sens}. Аналогичным образом строка 4 формул (13) и (14) может быть исключена из φ_{sens}.

В итоге формулы (12), (13), (14) содержат полную (избыточную) информацию по каждому датчику *FS2*, *IFS* и *FS1* соответственно, а сокращённая спецификация φ_{sens} — только необходимую информацию для всей группы датчиков. Модификация спецификации поведения отдельно взятого датчика может привести к кардинальному изменению сокращённой спецификации φ_{sens}. Поэтому рекомендуется разрабатывать и хранить полную спецификацию, а сокращённую использовать при верификации.

Зададим LTL-спецификацию ограниченно недетерминированного поведения окружения программы в синтаксисе инструмента nuXmv (см. приложение 2). Отметим, что на поведение кнопок не накладывается никаких ограничений, поэтому выполняется только их инициализация. Поведение всех датчиков и таймеров описано вместе с их инициализацией.

Рассмотрим поведение таймера $MTmr.Q$ из данной спецификации:

$$
\begin{aligned}
0:&\quad \neg MTmr.Q\\
1:&\quad \mathbf{G}\big(\neg MTmr.Q \wedge \mathbf{X}(MTmr.Q) \Rightarrow MTmr.In\big) \wedge\\
2:&\quad \mathbf{G}\big(\neg MTmr.Q \wedge \neg\mathbf{X}(MTmr.Q) \Rightarrow true\big) \wedge\\
3:&\quad \mathbf{G}\big(MTmr.Q \wedge \neg\mathbf{X}(MTmr.Q) \Rightarrow \neg MTmr.In\big) \wedge\\
4:&\quad \mathbf{G}\big(MTmr.Q \wedge \mathbf{X}(MTmr.Q) \Rightarrow MTmr.In\big) \wedge\\
5:&\quad \mathbf{G}\big(\mathbf{G}(\neg MTmr.Q) \Rightarrow \mathbf{F}(\mathbf{G}(\neg MTmr.In)) \vee \mathbf{G}(MTmr.In \Rightarrow \mathbf{F}(\neg MTmr.In))\big) \wedge\\
6:&\quad \mathbf{G}\big(\mathbf{G}(MTmr.Q) \Rightarrow \mathbf{G}(MTmr.In)\big).
\end{aligned}
\tag{15}
$$

Изначально выход таймера выключен (строка 0). При включении таймера его вход должен быть установлен в предыдущий момент времени (строка 1). Сигнал $MTmr.In$ является управляющим воздействием на таймер, а $MTmr.Q$ — сигналом обратной связи. Таким образом, таймер — такой же объект управления для ПЛУ, как и технологическое оборудование. При наличии входного сигнала таймер может и не включиться (строка 2). Вместе строки 1 и 2 задают недетерминированное включение таймера. При отключении таймера его вход должен быть сброшен в предыдущий момент времени (строка 3), в противном случае таймер продолжает оставаться включённым (строка 4). Строки 3 и 4 задают строго детерминированное отключение таймера.

«Залипание» таймера в отключенном состоянии возможно, если начиная с какого-то момента времени в будущем вход таймера больше никогда не устанавливается или всегда после его установки в будущем следует сброс (строка 5). Таким образом, таймер либо больше не работает, либо не успевает выдержать заданный интервал времени, поэтому может никогда не установить свой выход. Разрешим «залипание» таймера во включенном состоянии при «залипании» его входного сигнала (строка 6). На самом деле данное условие является лишним — с помощью инструмента Spot было проверено, что строка 6 является логическим следствие строки 4. В итоге строки 2 и 6 исключаются при построении сокращённой LTL-спецификации поведения таймера.

6. LTL-спецификация свойств

Зададим требуемый набор свойств установки для литья пластмасс. Свойства $P1, \ldots, P15$ и их LTL-формализация представлены в таблице 1, свойства $P16, \ldots, P28$ — в таблице 2. В целом спецификация свойств $\mathbb{P} = \{P1, \ldots, P28\}$ построена на основе шаблонов: (1) устанавливающих безопасные состояния группы устройств, (2) нацеленных на проверку возможности изменения значений переменных, (3) накладывающих ограничения на порядок работы устройств в группе, (4) определяющих поведение устройств в некотором режиме. Модель поведения ПЛУ установки должна удовлетворять спецификации свойств $\mathbb{P}$.

Table 1. Properties $P1 \ldots P15$ **Таблица 1.** Свойства $P1 \ldots P15$

Св-во	Описание и формализация
$P1$	$\mathbf{G}(SysOn \Rightarrow \neg(PBStop \vee PBConvr \vee FErr \vee CErr \vee HErr))$
	Всегда, если система включена, то неверно, что нажата кнопка экстренного останова, кнопка запуска конвейера или есть ошибки (подачи, конвейера или нагревателя).
$P2$	$\mathbf{G}(\neg SysOn \Rightarrow \neg(Valve \vee FMech \vee Heater \vee OpnLid \vee ClsLid \vee Compl) \wedge (Convr \Rightarrow PBConvr))$
	Всегда при выключенной системе не работает ни одно из устройств (клапан, механизм подачи, нагреватель, привод крышки дозатора), кроме конвейера, а также не активен режим плавного завершения; работа конвейера допускается только по команде оператора.
$P3$	$\mathbf{G}(FMech \Rightarrow \neg Valve \wedge \neg OpnLid \wedge \neg ClsLid) \wedge \mathbf{G}(Valve \Rightarrow \neg FMech \wedge \neg OpnLid \wedge \neg ClsLid) \wedge \mathbf{G}(OpnLid \Rightarrow \neg FMech \wedge \neg Valve \wedge \neg ClsLid) \wedge \mathbf{G}(ClsLid \Rightarrow \neg FMech \wedge \neg Valve \wedge \neg OpnLid)$
	Взаимное исключение работы устройств: всегда, если работает указанное устройство, то все остальные устройства группы не работают.
$P4$	$\boxed{\mathbf{G}\neg(Convr \wedge Valve)}$ Никогда форма не заливается во время её перемещении: конвейер и разливной клапан никогда не работают одновременно.
$P5$	$\mathbf{G}(\neg(OpnLid \vee ClsLid) \Rightarrow (CLS \vee OLS \vee \neg SysOn))$
	Всегда, если крышка дозатора не перемещается (открывается или закрывается), то она находится в крайнем положении (открыта или закрыта) или система отключена.
$P6$	$\boxed{\mathbf{G}(WS1 \Rightarrow \neg FMech)}$ Дозатор никогда не переполняется: всегда, если дозатор заполнен, то подача пластика отключена.
$P7$	$\boxed{\mathbf{G}(Valve \Rightarrow (FS2 \wedge \neg Convr))}$ Пластик никогда не разливается на конвейер: всегда, если разливной клапан открыт, то форма на позиции и конвейер остановлен.
$P8$	$\mathbf{G}([FErr \wedge \neg\mathbf{X}(FErr) \vee HErr \wedge \neg\mathbf{X}(HErr) \vee CErr \wedge \neg\mathbf{X}(CErr)] \Rightarrow \mathbf{X}(PBStop) \wedge \neg\mathbf{X}(SysOn))$
	Организация сброса ошибок в программе: всегда, если ошибка сброшена, то была нажата кнопка останова и система отключена.
$P9$	$\mathbf{G}(\neg Valve \wedge FS2 \wedge WS0 \Rightarrow \neg[(FS2 \wedge WS0)\ \mathbf{U}\ (Convr \wedge \neg PBConvr \wedge FS2 \wedge WS0)])$
	Весь пластик дозатора заливается в одну форму: всегда, если заливка формы была прервана (дозатор ещё не пуст), то до включения конвейера дозатор будет опустошён.
$P10$	$\mathbf{G}(Valve \wedge \mathbf{X}(\neg Valve) \wedge \mathbf{X}(FS2) \wedge \neg\mathbf{X}(WS0) \Rightarrow \neg\mathbf{X}(FS2\ \mathbf{U}\ (Valve \wedge FS2)))$
	Никогда не происходит переполнения формы: всегда, если клапан закрывается, разгрузив пластик из дозатора в форму, то он не откроется повторно при наличии данной формы.
$P11$	$\mathbf{G}(Valve \wedge \mathbf{X}(\neg Valve) \wedge \mathbf{X}(FS2) \wedge \neg\mathbf{X}(WS0) \Rightarrow \neg\mathbf{X}(\neg Convr\ \mathbf{U}\ Valve))$
	Транспортировка формы конвейером обязательна между двумя заливками формы: всегда, если клапан закрывается, разгрузив пластик из дозатора в форму, то неверно, что конвейер будет выключен до следующего открытия клапана.
$P12$	$\mathbf{G}(\neg Valve \wedge \neg WS1 \wedge OLS \Rightarrow \neg[\neg WS1\ \mathbf{U}\ (Valve \wedge \neg WS1)])$
	Всегда, если дозатор не заполнен, то разливной клапан не откроется до его заполнения.
$P13$	$\boxed{\mathbf{G}(\neg FS2 \wedge \mathbf{X}(FS2) \Rightarrow LwSpd)}$ Всегда, если сработал датчик наличия формы на второй позиции, то конвейер транспортировал форму на пониженной скорости.
$P14$	$\boxed{\mathbf{G}(\neg FS1 \wedge \mathbf{X}(FS1) \Rightarrow \neg LwSpd)}$ Всегда, если сработал датчик наличия формы на первой позиции, то конвейер транспортировал форму на нормальной (не пониженной) скорости.
$P15$	$\mathbf{G}(OpnLid \wedge \neg\mathbf{X}(OpnLid) \wedge \mathbf{X}(SysOn) \Rightarrow \neg FMech \wedge \mathbf{X}(FMech)) \wedge \mathbf{G}(FMech \wedge \neg\mathbf{X}(FMech) \wedge \mathbf{X}(SysOn) \Rightarrow \neg ClsLid \wedge \mathbf{X}(ClsLid)) \wedge \mathbf{G}(Valve \wedge \neg\mathbf{X}(Valve) \wedge \mathbf{X}(SysOn) \Rightarrow \neg Convr \wedge \mathbf{X}(Convr))$
	Последовательность работы устройств: всегда при включённой системе после открытия заслонки включается механизм подачи, после механизма подачи заслонка закрывается, после закрытия клапана включается конвейер.

Table 2. Properties $P16 \ldots P28$ **Таблица 2.** Свойства $P16 \ldots P28$

Св-во	Описание и формализация
$P16$	$\mathbf{G}(FTmr.Q \Rightarrow \mathbf{X}(\neg FTmr.Q)) \wedge \mathbf{G}(HTmr.Q \Rightarrow \mathbf{X}(\neg HTmr.Q)) \wedge \mathbf{G}(CTmr.Q \Rightarrow \mathbf{X}(\neg CTmr.Q))$
	Организация работы с таймерами в программе: всегда, если указанный таймер сработал, то в следующий момент времени он отключен.
$P17$	$\mathbf{G}(Convr \wedge \neg FS2 \wedge \mathbf{X}(FS2) \Rightarrow \mathbf{X}(\neg PBConvr \Rightarrow \neg Convr))$
	Всегда, если при работе конвейера срабатывает датчик наличия формы на второй позиции, то в этот момент конвейер отключается, если не нажата кнопка запуска конвейера.
$P18$	$\mathbf{G}(\neg FS2 \wedge \mathbf{X}(FS2) \wedge \mathbf{X}(SysOn) \Rightarrow \neg\mathbf{X}((SysOn \wedge \neg Valve)\ \mathbf{U}\ (SysOn \wedge Convr)))$
	В каждую остановленную форму подаётся пластик: при постоянно работающей системе всегда, если форма появилась под дозатором, то будет открыт разливной клапан до включения конвейера (не верно, что клапан будет закрыт до включения конвейера).
$P19$	$\mathbf{G}(Valve \wedge \neg\mathbf{X}(Valve) \wedge \mathbf{X}(SysOn) \Rightarrow \neg\mathbf{X}((SysOn \wedge \neg FMech)\ \mathbf{U}\ (SysOn \wedge Valve)))$
	Между литьём форм происходит заполнение дозатора: при постоянно работающей системе всегда, если разливной клапан закрылся, то неверно, что механизм подачи пластика в дозатор будет выключен до следующего открытия клапана.
$P20$	$\mathbf{G}(ClsLid \wedge \neg\mathbf{X}(ClsLid) \wedge \mathbf{X}(SysOn) \Rightarrow \mathbf{X}(\neg OpnLid\ \mathbf{U}\ (Valve \vee \neg SysOn))) \wedge$ $\mathbf{G}(Valve \wedge \neg\mathbf{X}(Valve) \wedge \mathbf{X}(SysOn) \Rightarrow \mathbf{X}(\neg Valve\ \mathbf{U}\ (OpnLid \vee \neg SysOn)))$
	Ограничения на повторную работу устройств: всегда при включённой системе после закрытия крышки она не открывается до тех пор, пока не откроется разливной клапан, после закрытия клапана он повторно не открывается до тех пор, пока не откроется крышка.
$P21$	$\mathbf{G}(\neg OpnLid \Rightarrow \mathbf{F}(OpnLid \vee \neg SysOn)) \wedge \mathbf{G}(\neg ClsLid \Rightarrow \mathbf{F}(ClsLid \vee \neg SysOn))$
	Бесконечная работа устройств по циклу: всегда, если работа заслонки прекратилась, то в будущем она обязательно возобновится. Исключение — когда система отключена.
$P22$	$\mathbf{G}(\neg FMech \Rightarrow \mathbf{F}(FMech \vee \neg SysOn)) \wedge \mathbf{G}(\neg Valve \Rightarrow \mathbf{F}(Valve \vee \neg SysOn)) \wedge$ $\mathbf{G}(\neg Convr \Rightarrow \mathbf{F}(Convr \vee \neg SysOn)) \wedge \mathbf{G}(\neg Heater \Rightarrow \mathbf{F}(Heater \vee \neg SysOn))$
	Бесконечная работа устройств по циклу: всегда, если работа устройства прекратилась, то в будущем она обязательно возобновится. Исключение — когда система отключена.
$P23$	$\mathbf{G}(FMech \Rightarrow \mathbf{F}(\neg FMech)) \wedge \mathbf{G}(Valve \Rightarrow \mathbf{F}(\neg Valve)) \wedge \mathbf{G}(OpnLid \Rightarrow \mathbf{F}(\neg OpnLid)) \wedge \mathbf{G}(ClsLid \Rightarrow \mathbf{F}(\neg ClsLid)) \wedge \mathbf{G}(Convr \Rightarrow \mathbf{F}(\neg Convr \vee PBConvr)) \wedge \mathbf{G}(Heater \Rightarrow (\mathbf{F}(\neg Heater) \vee \mathbf{G}(\neg UTS)))$
	Конечное время работы устройств: всегда, если указанное устройство включилось, то в будущем оно обязательно отключится. Исключения: для конвейера — удержание кнопки его работы, для нагревателя — невозможность достигнуть температуры его отключения.
$P24$	$\boxed{\mathbf{G}(Compl \Rightarrow \mathbf{F}(\neg Compl))}$ Результативность режима плавного завершения работы: всегда, если началось плавное завершение работы, то оно обязательно закончится.
$P25$	$\boxed{\mathbf{G}(MTmr.Q \Rightarrow \mathbf{F}(\neg MTmr.Q))}$ Организация работы с таймерами в программе: всегда, если таймер плавления сработал, то в будущем он будет отключен.
$P26$	$\boxed{\mathbf{G}(Convr \Rightarrow (Convr\ \mathbf{U}\ (FS2 \vee \neg SysOn)))}$ Непрерывность работы конвейера: всегда, если конвейер включился, то он работает до появления формы или отключения системы.
$P27$	$\mathbf{G}[FMech \Rightarrow (FMech\ \mathbf{U}\ (WS1 \vee \neg SysOn))]$
	Непрерывность загрузки дозатора: всегда, если включился механизм подачи, то он будет работать до заполнения дозатора или отключения системы.
$P28$	$\mathbf{G}[Valve \Rightarrow (Valve\ \mathbf{U}\ (\neg WS0 \vee \neg SysOn))]$
	Непрерывность литья: всегда, если открылся разливной клапан, то он будет открыт до опустошения дозатора или отключения системы.

7. Построение и верификация SMV-спецификации

В LTL-спецификации поведения булевых переменных не используются вспомогательные переменные из $\_V$, поэтому задача непосредственной верификации имеет вид $cTS \models (\varphi \Rightarrow \psi)$, где φ — LTL-спецификация поведения ПЛУ и её окружения, а ψ — LTL-спецификация свойств. Непосредственная верификация декларативной LTL-спецификации поведения УПО с помощью инструмента nuXmv занимает длительное время [28]. Для ускорения процесса верификации в [28] было предложено преобразование декларативной LTL-спецификации (dcl_LTL) в декларативную SMV-спецификацию (dclSMV). В настоящей работе воспользуемся этим преобразованием для LTL-спецификаций поведения ПЛУ (приложение 1) и её окружения (приложение 2).

При преобразовании LTL-спецификации поведения окружения ПЛУ в SMV-спецификацию есть некоторые особенности:

1. Преобразованию подлежит не полная (приложение 2), а сокращённая LTL-спецификация (см. параграф 5.3).
2. В (10) последние две формулы представляют собой условия *справедливости* и задаются в SMV-спецификации специальным стандартным образом.
3. Не все условия справедливости могут потребоваться для проверки свойств.

Стандартизация условий справедливости. Инструмент проверки модели nuXmv поддерживает две формы справедливости [72]:

1. *Безусловная справедливость* [16]: p выполняется бесконечно часто — $\mathbf{GF}(p)$. В SMV-спецификации это записывается **FAIRNESS**(p).
2. *Сильная справедливость* [16]: p выполняется бесконечно часто, если q также выполняется бесконечно часто — $\mathbf{GF}(q) \Rightarrow \mathbf{GF}(p)$. В SMV-спецификации это записывается **COMPASSION**(q, p).

Приведём каждое условие справедливости из LTL-спецификации поведения окружения ПЛУ (приложение 2) к одной из двух вышеуказанных форм. Для этого определим набор шаблонов, который будет иметь три булевы переменные: act_1 — действие, приводящее к реакции *rea*, act_0 — противодействие действию act_1, т. е. act_0 приводит к отсутствию реакции *rea*.

Общий шаблон (шаблон № 1) имеет вид

$$\mathbf{G}[\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1) \vee \mathbf{G}(act_1 \Rightarrow \mathbf{F}(act_0))\,] \tag{16}$$

и утверждает, что всегда, если реакция постоянно отсутствует, то с некоторого момента времени всегда нет действия act_1 или любое действие act_1 в будущем сопровождается противодействием act_0. Данному шаблону соответствуют четыре условия справедливости из LTL-спецификации поведения окружения ПЛУ (см. таблицу 3). Для первого условия справедливости $act_1 = OpnLid$, $act_0 = ClsLid$, $rea = \neg CLS$. Действие *OpnLid* (открытие) приводит к реакции $\neg CLS$ (крышка не закрыта), действие *ClsLid* (закрытие) имеет обратный эффект.

Для (16) при постоянном отсутствии противодействия ($act_0 = False$) получим частный случай:

$$\mathbf{G}(\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1)\,). \tag{17}$$

Доказательство см. в приложении 3. Полученному шаблону № 2 соответствуют три условия справедливости (см. таблицу 3). Первое условие утверждает, что всегда, если датчик *FS1* «залип» в состоянии *True*, то с некоторого момента времени конвейер больше никогда не включается. У действия включения конвейера ($act_1 = Convr$) нет противодействия, так как привод конвейера не может вращаться в обратную сторону.

Если отсутствие действия является противодействием ($act_0 = \neg act_1$), то для (16) получим другой частный случай (шаблон № 3):

$$\mathbf{G}[\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1) \vee \mathbf{G}(act_1 \Rightarrow \mathbf{F}(\neg act_1))\,]. \tag{18}$$

Table 3. Fairness patterns

Таблица 3. Шаблоны справедливости

№	Шаблон справедливости	Условие справедливости
1	$\mathbf{G}[\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1) \vee \mathbf{G}(act_1 \Rightarrow \mathbf{F}(act_0))\,]$	$\mathbf{G}[\,\mathbf{G}(CLS) \Rightarrow \mathbf{FG}(\neg OpnLid) \vee \mathbf{G}(OpnLid \Rightarrow \mathbf{F}(ClsLid))\,]$
		$\mathbf{G}[\,\mathbf{G}(OLS) \Rightarrow \mathbf{FG}(\neg ClsLid) \vee \mathbf{G}(ClsLid \Rightarrow \mathbf{F}(OpnLid))\,]$
		$\mathbf{G}[\,\mathbf{G}(WS0) \Rightarrow \mathbf{FG}(\neg(Valve \wedge WTS)) \vee \mathbf{G}(Valve \wedge WTS \Rightarrow \mathbf{F}(FMech \wedge \neg CLS))\,]$
		$\mathbf{G}[\,\mathbf{G}(WS1) \Rightarrow \mathbf{FG}(\neg(Valve \wedge WTS)) \vee \mathbf{G}(Valve \wedge WTS \Rightarrow \mathbf{F}(FMech \wedge \neg CLS))\,]$
2	$\mathbf{G}(\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1)\,)$	$\mathbf{G}(\,\mathbf{G}(FS1) \Rightarrow \mathbf{FG}(\neg Convr)\,)$
		$\mathbf{G}(\,\mathbf{G}(IFS) \Rightarrow \mathbf{FG}(\neg Convr)\,)$
		$\mathbf{G}(\,\mathbf{G}(FS2) \Rightarrow \mathbf{FG}(\neg Convr)\,)$
3	$\mathbf{G}[\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1) \vee \mathbf{G}(act_1 \Rightarrow \mathbf{F}(\neg act_1))\,]$	$\mathbf{G}[\,\mathbf{G}(UTS) \Rightarrow \mathbf{FG}(Heater) \vee \mathbf{G}(\neg Heater \Rightarrow \mathbf{F}(Heater))\,]$
		$\mathbf{G}[\,\mathbf{G}(LTS) \Rightarrow \mathbf{FG}(Heater) \vee \mathbf{G}(\neg Heater \Rightarrow \mathbf{F}(Heater))\,]$
		$\mathbf{G}[\,\mathbf{G}(WTS) \Rightarrow \mathbf{FG}(Heater) \vee \mathbf{G}(\neg Heater \Rightarrow \mathbf{F}(Heater))\,]$
		$\mathbf{G}[\,\mathbf{G}(\neg MTmr.Q) \Rightarrow \mathbf{FG}(\neg MTmr.In) \vee \mathbf{G}(MTmr.In \Rightarrow \mathbf{F}(\neg MTmr.In))\,]$

Шаблону (18) соответствуют оставшиеся условия справедливости из LTL-спецификации поведения окружения ПЛУ (см. таблицу 3). Первое условие утверждает, что всегда, если датчик высокой температуры *UTS* «залип» в состоянии *True*, то с некоторого момента нагреватель больше никогда не выключается или после выключения обязательно следует его включение. Здесь $act_1 = \neg Heater$, $act_0 = Heater$, $rea = \neg UTS$. Чтобы снизить температуру ($\neg UTS$), необходимо выключить нагреватель ($\neg Heater$) — его включение ($Heater$) является противодействием. Четвертое условие накладывает ограничение на справедливое включение таймера $MTmr.Q$. Формулы для таймеров $HTmr.Q$, $FTmr.Q$, $CTmr.Q$ аналогичны формуле для таймера $MTmr.Q$.

Общий шаблон (16) — шаблон № 1 — представляет собой сильную справедливость:

$$\mathbf{GF}(act_1) \Rightarrow \mathbf{GF}(act_0 \vee rea). \tag{19}$$

Доказательство представлено в приложении 3. В (19) подставим $act_0 = False$, получим стандартную форму для шаблона № 2:

$$\mathbf{GF}(act_1) \Rightarrow \mathbf{GF}(rea). \tag{20}$$

Стандартную форму для шаблона № 3 получим путём подстановки $act_0 = \neg act_1$ в (19) и дальнейшего преобразования (см. приложение 3). В результате получим:

$$\mathbf{GF}(act_1 \Rightarrow rea). \tag{21}$$

Соответствие шаблонов условий справедливости (16), (17), (18) и их стандартных форм (в том числе в SMV-формате) представлено в таблице 4. В итоге, шаблоны № 1 и № 2 выражают сильную справедливость (COMPASSION), а шаблон № 3 — безусловную (FAIRNESS).

Table 4. Patterns standard form **Таблица 4.** Стандартная форма шаблонов

№	Шаблон справедливости	Стандартная форма	SMV-формат
1	$\mathbf{G}[\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1) \vee \mathbf{G}(act_1 \Rightarrow \mathbf{F}(act_0))\,]$	$\mathbf{GF}(act_1) \Rightarrow \mathbf{GF}(act_0 \vee rea)$	COMPASSION(act_1, $act_0 \mid rea$)
2	$\mathbf{G}(\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1)\,)$	$\mathbf{GF}(act_1) \Rightarrow \mathbf{GF}(rea)$	COMPASSION(act_1, rea)
3	$\mathbf{G}[\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1) \vee \mathbf{G}(act_1 \Rightarrow \mathbf{F}(\neg act_1))\,]$	$\mathbf{GF}(act_1 \Rightarrow rea)$	FAIRNESS(act_1 -> rea)

Результат преобразования LTL-спецификаций поведения ПЛУ (приложение 1) и её окружения (приложение 2) представляет собой SMV-спецификацию поведения КФС, которая находится в приложении 4. SMV-спецификация описывает поведение всех переменных ПЛУ и её окружения. Так как поведение всех таймеров однотипно, оно было описано в виде модуля *Timer* (см. листинг 1). Данный листинг является фрагментом SMV-спецификации поведения КФС.

Листинг 1 (SMV-спецификация модуля Timer)**:**

```
MODULE Timer
VAR
    In : boolean; -- Input  (Logical Control Program Unit Output)
    Q  : boolean; -- Output (Logical Control Program Unit Input)
ASSIGN
    INIT ( !Q & !In )
    TRANS( !Q &  next(Q) ->  In )
    TRANS(  Q & !next(Q) -> !In )
    TRANS(  Q &  next(Q) ->  In )
    FAIRNESS (In -> Q) -- Timer Fair Turn On.
```

Модуль содержит объявление и инициализацию входной (In) и выходной (Q) переменных таймера, поведение переменной Q, условие честного срабатывания. Все таймеры УПО являются экземплярами данного модуля.

При верификации условия справедливости добавляются в SMV-спецификацию поведения КФС только в случае необходимости. Мы используем подход уточнения абстракции (модели поведения КФС) на основе контрпримера (CounterExample-Guided Abstraction Refinement, CEGAR) [15]. Если обнаружен ложный контрпример, демонстрирующий нарушение свойства из-за несоблюдения некоторого условия справедливости, то данное условие добавляется в SMV-спецификацию. Для проверки всей совокупности свойств помимо условия справедливости для модуля таймера из листинга 1 потребовалось добавить ещё пять условий справедливости (см. листинг 2).

Листинг 2 (SMV-спецификация добавленных условий справедливости)**:**

```
COMPASSION (Convr, !FS2)              -- Fair sticking of FS2 sensor in True  state.
COMPASSION (OpnLid, ClsLid | OLS)                -- ... OLS sensor in False state.
COMPASSION (ClsLid, OpnLid | CLS)                -- ... CLS sensor in False state.
COMPASSION (Valve & WTS, (FMech & !CLS) | !WS0)  -- ... WS0 sensor in True  state.
FAIRNESS   (!Heater -> !WTS)                     -- ... WTS sensor in True  state.
```

Результаты верификации. Потенциально возможное число состояний модели поведения замкнутой системы (установки для литья пластмасс) составляет $2.74878 \cdot 10^{11}$ (2^{38}), так как SMV-спецификация установки содержит 38 булевых переменных. Верификация проводилась на персональном компьютере с процессором Intel Core i5-3570 3.4 ГГц и 8 ГБ оперативной памяти. Результаты представлены в таблице 5.

Table 5. Verification results **Таблица 5.** Результаты верификации

Свойство	Тип свойства: безопасность (Saf), живость (Liv)	Недетерминизм окружения		
		Абсолютный	Ограниченный	
			Без справедлив.	Со справедлив.
Столбец 1	Столбец 2	Столбец 3	Столбец 4	Столбец 5
$P1$	Saf	**1**	**1**	**1**
$P2$	Saf	**1**	**1**	**1**
$P3$	Saf	**1**	**1**	**1**
$P4$	Saf	**1**	**1**	**1**
$P5$	Saf	**1**	**1**	**1**
$P6$	Saf	**1**	**1**	**1**
$P7$	Saf	**1**	**1**	**1**
$P8$	Saf	**1**	**1**	**1**
$P9$	Saf	**1**	**1**	**1**
$P10$	Saf	0	**1**	**1**
$P11$	Saf	0	**1**	**1**
$P12$	Saf	0	**1**	**1**
$P13$	Saf	0	**1**	**1**
$P14$	Saf	0	**1**	**1**
$P15$	Saf	0	**1**	**1**
$P16$	Saf	0	**1**	**1**
$P17$	Saf	0	**1**	**1**
$P18$	Liv	0	**1**	**1**
$P19$	Liv	0	**1**	**1**
$P20$	Liv	0	0	**1**
$P21$	Liv	0	0	**1**
$P22$	Liv	0	0	**1**
$P23$	Liv	0	0	**1**
$P24$	Liv	0	0	**1**
$P25$	Liv	0	0	**1**
$P26$	Liv	0	0	**1**
$P27$	Liv	0	0	**1**
$P28$	Liv	0	0	**1**
Количество состояний:		1381496 ($2^{20.3978}$)	16150 ($2^{13.9792}$)	

Столбец 1 содержит идентификаторы проверяемых свойств $P1, \ldots, P28$, столбец 2 — тип свойства: $P1, \ldots, P17$ являются свойствами *безопасности*, $P18, \ldots, P28$ — свойствами *живости*. Столбцы 3, 4 и 5 отражают результаты верификации свойств при различных условиях. Наличие «1» в ячейке таблицы означает, что соответствующее свойство выполняется, а «0» — нарушается. Столбец 3 содержит результаты верификации при абсолютном недетерминизме окружения ПЛУ, столбец 4 — при ограниченном недетерминизме без условий справедливости, столбец 5 — при ограниченном недетерминизме с условиями справедливости.

Из таблицы 5 видно, что свойства $P1, \ldots, P9$ могут быть проверены при любых условиях (столбцы 3, 4 и 5 содержат «1»). При абсолютно недетерминированном поведении окружения ПЛУ система переходов имеет $1.38146 \cdot 10^6$ ($2^{20.3978}$) достижимых состояний (нижняя часть столбца 3), а при огра-

ниченно недетерминированном — 16150 ($2^{13.9792}$) состояний (нижняя часть столбцов 4 и 5). Таким образом, ограничение недетерминизма привело к сокращению пространства состояний примерно в 85 раз. Это ускоряет верификацию: время проверки свойств $P1, \ldots, P9$ при абсолютном недетерминизме составляет 6,3 секунды, а ограниченном (без условий справедливости) — 1,6 секунды. Таким образом, время верификации сократилось примерно в четыре раза. Также данное ограничение недетерминизма позволяет проверять дополнительные свойства $P10, \ldots, P19$ (см. столбец 4), которые требуют наличия реалистичных ограничений в поведении окружения.

Для проверки свойств $P20, \ldots, P28$ в модель поведения окружения необходимо добавить условия справедливости (см. столбец 5). Однако добавление условий справедливости увеличивает время верификации: время проверки свойств $P1, \ldots, P9$ при ограниченном недетерминизме с условиями справедливости составляет 74 секунды, что многократно превышает время верификации, равное 6,3 секунды и 1,6 секунды в двух предыдущих случаях. Проверка с помощью инструмента nuXmv всех свойств $P1, \ldots, P28$ с учётом ограничений и условий справедливости заняла около 4 минут.

В итоге описание поведения сигналов обратной связи (без условий справедливости) позволило проверить свойства безопасности $P10, \ldots, P17$, а также свойства живости $P18, P19$. Использование условий справедливости добавило возможность проверки свойств живости $P20, \ldots, P28$.

8. Построение ST-программы для ПЛК

Согласно [27] на основе декларативной (приложение 1) была построена императивная LTL-спецификация (приложение 5), которая далее переводится в программу на языке ST (приложение 6). ST-программа (PLC_PRG) имеет разделы для объявления входных (VAR_INPUT), выходных (VAR_OUTPUT) и внутренних (VAR) переменных. Таймеры являются внутренними объектами УПО и внешними по отношению к ПЛУ.

Заключение

В работе предложен специальный вид LTL-спецификации поведения программ логического управления для систем с обратной связью. Детерминированное поведение внутренних и выходных булевых переменных программы описывается с помощью частного случая декларативной LTL-спецификации, а недетерминированное поведение входных булевых переменных — с помощью LTL-спецификации ограниченно недетерминированного поведения. Последняя LTL-спецификация позволяет описывать поведение сигналов обратной связи и мнимых датчиков (дополнительных булевых переменных для описания состояния компонентов из окружения программы). В ней задаются условия для переходов между состояниями (*False* и *True*) каждой булевой переменной, а также условия, необходимые при «залипании» в этих состояниях, — условия справедливости. Предложенный способ LTL-спецификации позволяет строить достаточно простые модели поведения при верификации замкнутых систем, что уменьшает трудозатраты на разработку программ. Результаты работы продемонстрированы на примере промышленной установки для литья пластмасс.

Для ускорения процесса верификации с помощью инструмента проверки модели nuXmv LTL-спецификация переводится в SMV-спецификацию — входной язык инструмента nuXmv. Для представленного в работе примера установки определён набор шаблонов условий справедливости и связей между ними. Доказано, что каждый из шаблонов является условием сильной или безусловной справедливости, которые могут быть выражены с помощью стандартных средств инструмента nuXmv. Выполнена проверка SMV-спецификации поведения установки для литья пластмасс на предмет соответствия заданному набору LTL-свойств. Продемонстрировано, что только часть свойств может быть проверена без моделирования поведения окружения программы. Для проверки другой части свойств требуется такая модель. Отсутствие условий справедливости в данной модели также не позволяет проверить ряд свойств. Таким образом, LTL-спецификация ограниченно недетерминированного поведения окружения программы позволяет задавать всю необходимую

информацию для проверки всего набора свойств. Кроме того, задание ограничений в поведении сигналов обратной связи сокращает пространство состояний модели при верификации.

Декларативной LTL-спецификации поведения программы соответствует эквивалентная императивная LTL-спецификация, по которой построена ST-программа для ПЛК. Поведение данной программы гарантированно соответствует поведению исходной декларативной LTL-спецификации.

Будущие исследования могут быть направлены на автоматизацию синтеза ST-программ и SMV-спецификаций по исходной LTL-спецификации КФС.

References

[1] S. Oks and et al., "Cyber-Physical Systems in the Context of Industry 4.0: A Review, Categorization and Outlook", *Information Systems Frontiers*, pp. 1–42, 2022. DOI: 10.1007/s10796-022-10252-x.

[2] K. Zhang, Y. Shi, S. Karnouskos, T. Sauter, H. Fang, and A. W. Colombo, "Advancements in Industrial Cyber-Physical Systems: An Overview and Perspectives", *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 716–729, 2023. DOI: 10.1109/TII.2022.3199481.

[3] S. J. Oks, *Industrial Cyber-Physical Systems: Advancing Industry 4.0 from Vision to Application* (MAU), 1st ed. Springer, 2024, ISBN: 978-3-658-44416-7. DOI: 10.1007/978-3-658-44417-4.

[4] C. Dey and S. K. Sen, *Industrial Automation Technologies*, 1st ed. CRC Press, 2020, ISBN: 9780429299346. DOI: 10.1201/9780429299346.

[5] K. Thramboulidis, "A Cyber–Physical System-Based Approach for Industrial Automation Systems", *Computers in Industry*, vol. 72, pp. 92–102, 2015. DOI: 10.1016/j.compind.2015.04.006.

[6] *IEC 61131-1:2003 Programmable controllers – Part 1: General information.* [Online]. Available: https://webstore.iec.ch/publication/4550.

[7] R. Langmann and L. F. Rojas-Peña, "A PLC as an Industry 4.0 Component", in *Remote Engineering and Virtual Instrumentation*, 2016, pp. 10–15. DOI: 10.1109/REV.2016.7444433.

[8] D. Harel and A. Pnueli, "On the Development of Reactive Systems", in *Logics and Models of Concurrent Systems*, vol. 13, 1985, pp. 477–498. DOI: 10.1007/978-3-642-82453-1_17.

[9] A. Maurya and D. Kumar, "Reliability of safety-critical systems: A state-of-the-art review", *Quality and Reliability Engineering International*, vol. 36, Aug. 2020. DOI: 10.1002/qre.2715.

[10] D. J. Smith and K. G. Simpson, *The Safety Critical Systems Handbook*, 5th. Butterworth-Heinemann, 2020, ISBN: 978-0-12-820700-0.

[11] V. Vyatkin, "Software Engineering in Industrial Automation: State-of-the-Art Review", *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1234–1249, 2013. DOI: 10.1109/TII.2013.2258165.

[12] S. Mitra, *Verifying Cyber-Physical Systems: A Path to Safe Autonomy* (Cyber Physical Systems Series). MIT Press, 2021, ISBN: 978-0262044806.

[13] V. D'Silva, D. Kroening, and G. Weissenbacher, "A Survey of Automated Techniques for Formal Software Verification", in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, 2008, pp. 1165–1178. DOI: 10.1109/TCAD.2008.923410.

[14] R. Sinha, S. Patil, L. Gomes, and V. Vyatkin, "A Survey of Static Formal Methods for Building Dependable Industrial Automation Systems", *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 3772–3783, 2019. DOI: 10.1109/TII.2019.2908665.

[15] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, *Handbook of Model Checking*, 1st. Springer, 2018, vol. 10, ISBN: 3319105744. DOI: 10.1007/978-3-319-10575-8.

[16] Y. G. Karpov, *MODEL CHECKING. Verification of Parallel and Distributed Program Systems.* BHV-Peterburg, 2010, p. 560, in Russian, ISBN: 978-5-9775-0404-1.

[17] E. M. Clarke, O. Grumberg, and D. Peled, *Verification of Program Models: Model Checking.* MCNMO, 2002, p. 416, Transl. from English to Russian, ISBN: 5-94057-054-2.

[18] A. Pnueli, "Applications of Temporal Logic to the Specification and Verification of Reactive Systems: A Survey of Current Trends", in *Current Trends in Concurrency*, vol. 224, Springer, 1986, pp. 510–584. DOI: 10.1007/BFb0027047.

[19] K. Schneider, J. Shabolt, and J. G. Taylor, *Verification of reactive systems: formal methods and algorithms* (EATCS), 1st ed. Springer, 2004, ISBN: 978-3-540-00296-3. DOI: 10.1007/978-3-662-10778-2.

[20] Z. Manna and A. Pnueli, *Temporal Verification of Reactive Systems: Safety*, 1st ed. Springer, 2012, ISBN: 978-0-387-94459-3. DOI: 10.1007/978-1-4612-4222-2.

[21] J. Galvão, C. Oliveira, and et al., "Formal Verification: Focused on the Verification Using a Plant Model", in *Innovation, Engineering and Entrepreneurship*, Springer, 2019, pp. 124–131. DOI: 10.1007/978-3-319-91334-6_18.

[22] G. Frey and L. Litz, "Formal Methods in PLC Programming", in *IEEE International Conference On Systems, Man and Cybernetics*, vol. 4, 2000, pp. 2431–2436. DOI: 10.1109/ICSMC.2000.884356.

[23] J. M. Machado, B. Denis, and et al., "Logic Controllers Dependability Verification Using a Plant Model", *IFAC Proceedings Volumes*, vol. 39, no. 17, pp. 37–42, 2006. DOI: 10.3182/20060926-3-PL-4904.00007.

[24] A. A. Shalyto, "Logic Control and "Reactive" Systems: Algorithmization and Programming", *Automation and Remote Control*, vol. 62, no. 1, pp. 1–29, 2001. DOI: 10.1023/A:1002837232103.

[25] E. Kuzmin, D. Ryabukhin, and V. Sokolov, "Modeling a Consistent Behavior of PLC-Sensors", *Modeling and Analysis of Information Systems*, vol. 21, no. 4, pp. 75–90, 2014, in Russian. DOI: 10.18255/1818-1015-2014-4-75-90.

[26] E. Kuzmin, D. Ryabukhin, and V. Sokolov, "Modeling a Consistent Behavior of PLC-Sensors", *Automatic Control and Computer Sciences*, vol. 48, no. 7, pp. 602–614, 2014. DOI: 10.3103/S0146411614070256.

[27] M. Neyzov and E. Kuzmin, "LTL-specification for Development and Verification of Control Programs", *Modeling and Analysis of Information Systems*, vol. 30, no. 4, pp. 308–339, 2023, in Russian. DOI: 10.18255/1818-1015-2023-4-308-339.

[28] M. Neyzov and E. Kuzmin, "Verification of Declarative LTL-specification of Control Programs Behavior", *Modeling and Analysis of Information Systems*, vol. 31, no. 2, pp. 120–141, 2024, in Russian. DOI: 10.18255/1818-1015-2024-2-120-141.

[29] *nuXmv home*. [Online]. Available: https://nuxmv.fbk.eu/.

[30] M. Frappier, B. Fraikin, R. Chossart, R. Chane-Yack-Fa, and M. Ouenzar, "Comparison of Model Checking Tools for Information Systems", in *Formal Methods and Software Engineering*, ser. LNCS, vol. 6447, Springer, 2010, pp. 581–596, ISBN: 978-3-642-16901-4. DOI: 10.1007/978-3-642-16901-4_38.

[31] *Spot home*. [Online]. Available: https://spot.lre.epita.fr/.

[32] M. Xavier, S. Patil, V. Dubinin, and V. Vyatkin, "Formal Modelling, Analysis, and Synthesis of Modular Industrial Systems Inspired by Net Condition/Event Systems", in *Applications and Theory of Petri Nets and Concurrency*, 2023, pp. 16–33. DOI: 10.1007/978-3-031-33620-1_2.

[33] V. Vyatkin, H.-M. Hanisch, C. Pang, and C.-H. Yang, "Closed-Loop Modeling in Future Automation System Engineering and Validation", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 39, no. 1, pp. 17–28, 2009. DOI: 10.1109/TSMCC.2008.2005785.

[34] A. Lobov, J. Lastra, and R. Tuokko, "Application of UML in Plant Modeling for Model-Based Verification: UML Translation to TNCES", in *IEEE Industrial Informatics*, 2005, pp. 495–501. DOI: 10.1109/INDIN.2005.1560426.

[35] A. Lobov, J. Lastra, and R. Tuokko, "On Controller and Plant Modeling for Model-Based Formal Verification", in *IEEE Emerging Technologies and Factory Automation*, vol. 1, 2005, pp. 121–128. DOI: 10.1109/ETFA.2005.1612510.

[36] S. Preuße, *Technologies for Engineering Manufacturing Systems Control in Closed Loop*, 10th ed. Logos Verlag Berlin GmbH, 2013, ISBN: 978-3-8325-3600-8.

[37] S. Patil, S. Bhadra, and V. Vyatkin, "Closed-Loop Formal Verification Framework with Non-Determinism, Configurable by Meta-Modelling", in *IEEE Industrial Electronics Society*, 2011, pp. 3770–3775. DOI: 10.1109/IECON.2011.6119923.

[38] S. Patil, V. Vyatkin, and M. Sorouri, "Formal Verification of Intelligent Mechatronic Systems with Decentralized Control Logic", in *IEEE Emerging Technologies & Factory Automation*, 2012, pp. 1–7. DOI: 10.1109/ETFA.2012.6489678.

[39] S. Patil, V. Vyatkin, and C. Pang, "Counterexample-Guided Simulation Framework for Formal Verification of Flexible Automation Systems", in *IEEE Industrial Informatics*, 2015, pp. 1192–1197. DOI: 10.1109/INDIN.2015.7281905.

[40] C. Gerber, S. Preuße, and H.-M. Hanisch, "A Complete Framework for Controller Verification in Manufacturing", in *IEEE Emerging Technologies & Factory Automation*, 2010, pp. 1–9. DOI: 10.1109/ETFA.2010.5641220.

[41] S. Preuße, H.-C. Lapp, and H.-M. Hanisch, "Closed-Loop System Modeling, Validation, and Verification", in *IEEE Emerging Technologies & Factory Automation*, 2012, pp. 1–8. DOI: 10.1109/ETFA.2012.6489679.

[42] J. Machado, B. Denis, and J.-J. Lesage, "A Generic Approach to Build Plant Models for DES Verification Purposes", in *International Workshop on Discrete Event Systems*, 2006, pp. 407–412. DOI: 10.1109/WODES.2006.382508.

[43] J. Machado, E. Seabra, and et al., "Safe Controllers Design for Industrial Automation Systems", *Computers & Industrial Engineering*, vol. 60, no. 4, pp. 635–653, 2011. DOI: 10.1016/j.cie.2010.12.020.

[44] M. Perin and J.-M. Faure, "Building Meaningful Timed Models of Closed-Loop DES for Verification Purposes", *Control Engineering Practice*, vol. 21, no. 11, pp. 1620–1639, 2013. DOI: 10.1016/j.conengprac.2012.05.002.

[45] H.-M. Hanisch, "Closed-Loop Modeling and Related Problems of Embedded Control Systems in Engineering", in *Abstract State Machines 2004. Advances in Theory and Practice*, Springer, 2004, pp. 6–19. DOI: 10.1007/978-3-540-24773-9_2.

[46] C. Pang and V. Vyatkin, "Systematic Closed-Loop Modelling in IEC 61499 Function Blocks: A Case Study", *IFAC Proceedings Volumes*, vol. 42, pp. 199–204, 2009. DOI: 10.3182/20090603-3-RU-2001.0264.

[47] D. Drozdov, S. Patil, V. Dubinin, and V. Vyatkin, "Formal Verification of Cyber-Physical Automation Systems Modelled with Timed Block Diagrams", in *IEEE Industrial Electronics*, 2016, pp. 316–321. DOI: 10.1109/ISIE.2016.7744910.

[48] M. Xavier, S. Patil, and V. Vyatkin, "Cyber-Physical Automation Systems Modelling with IEC 61499 for their Formal Verification", in *IEEE Industrial Informatics*, 2021, pp. 1–6. DOI: 10.1109/INDIN45523.2021.9557416.

[49] G. Lilli *et al.*, "Formal Verification of the Control Software of a Radioactive Material Remote Handling System, Based on IEC 61499", *IEEE Open Journal of the Industrial Electronics Society*, vol. 4, pp. 417–431, 2023. DOI: 10.1109/OJIES.2023.3321084.

[50] N. Garanina, S. Staroletov, V. Zyubin, and I. Anureev, "Model Checking Programs in Process-Oriented IEC 61131-3 Structured Text", *Modeling and Analysis of Information Systems*, vol. 31, no. 1, pp. 32–53, 2024, in Russian. DOI: 10.18255/1818-1015-2024-1-32-53.

[51] N. Halbwachs, F. Lagnier, and C. Ratel, "Programming and Verifying Real-Time Systems by Means of the Synchronous Data-Flow Language LUSTRE", *IEEE Transactions on Software Engineering*, vol. 18, no. 9, pp. 785–793, 1992. DOI: 10.1109/32.159839.

[52] *IEC 61499-1:2012 Function blocks – Part 1: Architecture*. [Online]. Available: https://webstore.iec.ch/publication/5506.

[53] V. E. Zyubin, A. S. Rozov, I. S. Anureev, N. O. Garanina, and V. Vyatkin, "poST: A Process-Oriented Extension of the IEC 61131-3 Structured Text Language", *IEEE Access*, vol. 10, pp. 35 238–35 250, 2022. DOI: 10.1109/ACCESS.2022.3157601.

[54] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud, "The Synchronous Data Flow Programming Language LUSTRE", *Proceedings of the IEEE*, vol. 79, no. 9, pp. 1305–1320, 1991. DOI: 10.1109/5.97300.

[55] A. Champion, A. Gurfinkel, and et al., "CoCoSpec: A Mode-Aware Contract Language for Reactive Systems", in *Software Engineering and Formal Methods*, ser. LNTCS, vol. 9763, Springer, 2016, pp. 347–366. DOI: 10.1007/978-3-319-41591-8_24.

[56] A. Benveniste, P. Caspi, and et al., "The Synchronous Languages 12 Years Later", *Proceedings of the IEEE*, vol. 91, no. 1, pp. 64–83, 2003. DOI: 10.1109/JPROC.2002.805826.

[57] A. Bouajjani, J. Fernandez, and N. Halbwachs, "On the Verification of Safety Properties", *Rapport Technique SPECTRE L12*, 1990.

[58] P. Raymond, "Synchronous Program Verification with Lustre/Lesar", in *Modeling and Verification of Real-Time Systems*, John Wiley & Sons, 2008, ch. 6, pp. 171–206, ISBN: 9780470611012. DOI: 10.1002/9780470611012.ch6.

[59] A. Champion, A. Mebsout, C. Sticksel, and C. Tinelli, "The Kind 2 Model Checker", in *Computer Aided Verification*, ser. LNTCS, vol. 9780, Springer, 2016, pp. 510–517. DOI: 10.1007/978-3-319-41540-6_29.

[60] N. Halbwachs, *Synchronous Programming of Reactive Systems*. Springer, 1993, ISBN: 978-0792393115.

[61] M. Sirjani, E. A. Lee, and E. Khamespanah, "Verification of Cyberphysical Systems", *Mathematics*, vol. 8, no. 7, 2020. DOI: 10.3390/math8071068.

[62] S. Lin *et al.*, "Towards Building Verifiable CPS using Lingua Franca", *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 5s, pp. 1–24, 2023. DOI: 10.1145/3609134.

[63] P. Raymond, Y. Roux, and E. Jahier, "Lutin: A Language for Specifying and Executing Reactive Scenarios", *EURASIP Journal on Embedded Systems*, vol. 2008, pp. 1–11, 2008. DOI: 10.1155/2008/753821.

[64] B. Finkbeiner, "Synthesis of Reactive Systems", *Dependable Software Systems Engineering*, vol. 45, pp. 72–98, 2016. DOI: 10.3233/978-1-61499-627-9-72.

[65] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of Reactive (1) Designs", in *Verification, Model Checking, and Abstract Interpretation*, vol. 3855, Springer, 2006, pp. 364–380. DOI: 10.1007/11609773_24.

[66] M. Roth, L. Litz, and J.-J. Lesage, "Identification of Discrete Event Systems: Implementation Issues and Model Completeness", in *Informatics in Control, Automation and Robotics*, vol. 3, 2010, pp. 73–80.

[67] I. Buzhinsky and V. Vyatkin, "Automatic Inference of Finite-State Plant Models From Traces and Temporal Properties", *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1521–1530, 2017. DOI: 10.1109/TII.2017.2670146.

[68] P. Ovsiannikova, D. Chivilikhin, V. Ulyantsev, and A. Shalyto, "Closed-Loop Verification of a Compensating Group Drive Model Using Synthesized Formal Plant Model", in *IEEE Emerging Technologies and Factory Automation*, 2017, pp. 1–4. DOI: 10.1109/ETFA.2017.8247714.

[69] I. Buzhinsky, A. Pakonen, and V. Vyatkin, "Scalable Methods of Discrete Plant Model Generation for Closed-Loop Model Checking", in *IEEE Industrial Electronics Society*, 2017, pp. 5483–5488. DOI: 10.1109/IECON.2017.8216949.

[70] M. Xavier, J. Håkansson, S. Patil, and V. Vyatkin, "Plant Model Generator from Digital Twin for Purpose of Formal Verification", in *IEEE Emerging Technologies and Factory Automation*, 2021, pp. 1–4. DOI: 10.1109/ETFA45728.2021.9613704.

[71] M. Xavier, V. Dubinin, S. Patil, and V. Vyatkin, "Plant Model Generation From Event Log Using ProM for Formal Verification of CPS", *arXiv preprint arXiv:2211.03681*, 2022. DOI: 10.48550/arXiv.2211.03681.

[72] *nuXmv User Manual*. [Online]. Available: https://nuxmv.fbk.eu/downloads/nuxmv-user-manual.pdf.

Приложение 1. Декларативная LTL-спецификация поведения ПЛУ

```
!FErr &
G( !FErr &  X(FErr) ->  (X(FTmr.Q) & !X(WS1)) ) &
G( !FErr & !X(FErr) -> !(X(FTmr.Q) & !X(WS1)) ) &
G(  FErr & !X(FErr) ->  X(PBStop) ) &
G(  FErr &  X(FErr) -> !X(PBStop) ) &
!CErr &
G( !CErr &  X(CErr) ->  (X(CTmr.Q) & !X(FS2)) ) &
G( !CErr & !X(CErr) -> !(X(CTmr.Q) & !X(FS2)) ) &
G(  CErr & !X(CErr) ->  X(PBStop) ) &
G(  CErr &  X(CErr) -> !X(PBStop) ) &
!HErr &
G( !HErr &  X(HErr) ->  (!X(WTS) & (X(HTmr.Q) | Heater & WTS)) ) &
G( !HErr & !X(HErr) -> !(!X(WTS) & (X(HTmr.Q) | Heater & WTS)) ) &
G(  HErr & !X(HErr) ->  X(PBStop) ) &
G(  HErr &  X(HErr) -> !X(PBStop) ) &
!Fin &
G( X(Fin) <-> (Compl & X(OLS) & !X(WS0) | X(FErr) | X(HErr) | X(CErr) |
               X(PBStop) | X(PBConvr)) ) &
!SysOn &
G( !SysOn &  X(SysOn) ->  (X(PBStart) & !X(Fin)) ) &
G( !SysOn & !X(SysOn) -> !(X(PBStart) & !X(Fin)) ) &
G(  SysOn & !X(SysOn) ->  X(Fin) ) &
G(  SysOn &  X(SysOn) -> !X(Fin) ) &
!Compl &
G( !Compl &  X(Compl) ->  (X(SysOn) & X(PBCompl)) ) &
G( !Compl & !X(Compl) -> !(X(SysOn) & X(PBCompl)) ) &
G(  Compl & !X(Compl) -> !X(SysOn) ) &
G(  Compl &  X(Compl) ->  X(SysOn) ) &
!Mltng &
G( X(Mltng) <-> X(SysOn) & X(CLS) & X(Disch) & X(WS0) ) &
!Mlted &
G( !Mlted &  X(Mlted) ->  (X(MTmr.Q) & X(WTS) & X(WS0) & X(CLS)) ) &
G( !Mlted & !X(Mlted) -> !(X(MTmr.Q) & X(WTS) & X(WS0) & X(CLS)) ) &
G(  Mlted & !X(Mlted) ->  (!X(WTS) | !X(CLS) | !X(WS0)) ) &
G(  Mlted &  X(Mlted) -> !(!X(WTS) | !X(CLS) | !X(WS0)) ) &
!Disch &
G( !Disch &  X(Disch) ->  (X(OLS) & X(WS1)) ) &
G( !Disch & !X(Disch) -> !(X(OLS) & X(WS1)) ) &
G(  Disch & !X(Disch) ->  (!X(FS2) & !X(WS1)) ) &
G(  Disch &  X(Disch) -> !(!X(FS2) & !X(WS1)) ) &
-- Actuators:
!Heater &
G( !Heater &  X(Heater) ->  (X(SysOn) & !X(LTS)) ) &
G( !Heater & !X(Heater) -> !(X(SysOn) & !X(LTS)) ) &
G(  Heater & !X(Heater) ->  (!X(SysOn) | X(UTS)) ) &
G(  Heater &  X(Heater) -> !(!X(SysOn) | X(UTS)) ) &
!ClsLid &
G( X(ClsLid) <-> X(SysOn) & !X(CLS) & X(Disch) ) &
!OpnLid &
G( X(OpnLid) <-> X(SysOn) & !X(OLS) & !X(Disch) ) &
!Convr &
G( X(Convr) <-> (X(SysOn) & (X(FS2) & X(Disch) & !X(WS0) | !X(FS2)) | X(PBConvr)) ) &
```

```
!LwSpd &
G( !LwSpd &  X(LwSpd) ->  X(FS1) ) &
G( !LwSpd & !X(LwSpd) -> !X(FS1) ) &
G(  LwSpd & !X(LwSpd) ->  X(FS2) ) &
G(  LwSpd &  X(LwSpd) -> !X(FS2) ) &
!FMech &
G( X(FMech) <-> X(SysOn) & !X(Disch) & X(OLS) & !X(WS1) ) &
!Valve &
G( X(Valve) <-> X(SysOn) & X(Disch) & X(Mlted) & X(FS2) ) &
-- Impact_On_Timers:
G( X(FTmr.In) <-> X(FMech)  & !X(WS1) )                    & !FTmr.In &
G( X(HTmr.In) <-> X(Heater) & !X(WTS) )                    & !HTmr.In &
G( X(CTmr.In) <-> X(Convr)  & !X(FS2) & X(SysOn) )         & !CTmr.In &
G( X(MTmr.In) <-> X(Mltng)  &  X(WTS) & X(WS0) & X(CLS) ) & !MTmr.In
```

Приложение 2. LTL-спецификация поведения окружения ПЛУ

```
-- Buttons:
!PBStart & !PBStop & !PBCompl & !PBConvr &
-- Sensors:
!FS1 &
G( !FS1 &  X(FS1) ->  !FS2 & !X(FS2) & !IFS & !X(IFS) & Convr ) &
G( !FS1 & !X(FS1) ->  true ) &
G(  FS1 & !X(FS1) ->  !FS2 & !X(FS2) & !IFS &  X(IFS) & Convr ) &
G(  FS1 &  X(FS1) ->  !FS2 & !X(FS2) & !IFS & !X(IFS)         ) &
G( G(!FS1) ->  true         ) &
G( G( FS1) ->  F(G(!Convr)) ) &
!IFS &
G( !IFS &  X(IFS) ->  !FS2 & !X(FS2) &  FS1 & !X(FS1) & Convr ) &
G( !IFS & !X(IFS) ->  true ) &
G(  IFS & !X(IFS) ->  !FS2 &  X(FS2) & !FS1 & !X(FS1) & Convr ) &
G(  IFS &  X(IFS) ->  !FS2 & !X(FS2) & !FS1 & !X(FS1)         ) &
G( G(!IFS) ->  true         ) &
G( G( IFS) ->  F(G(!Convr)) ) &
!FS2 &
G( !FS2 &  X(FS2) ->   IFS & !X(IFS) & !FS1 & !X(FS1) & Convr ) &
G( !FS2 & !X(FS2) ->  true ) &
G(  FS2 & !X(FS2) ->  !IFS & !X(IFS) & !FS1 & !X(FS1) & Convr ) &
G(  FS2 &  X(FS2) ->  !IFS & !X(IFS) & !FS1 & !X(FS1)         ) &
G( G(!FS2) ->  true         ) &
G( G( FS2) ->  F(G(!Convr)) ) &
!CLS &
G( !CLS &  X(CLS) ->  !OLS & !X(OLS) & ClsLid ) &
G( !CLS & !X(CLS) ->  true ) &
G(  CLS & !X(CLS) ->  !OLS & !X(OLS) & OpnLid ) &
G(  CLS &  X(CLS) ->  !OLS & !X(OLS) ) &
G( G(!CLS) ->  F(G(!ClsLid)) | G(ClsLid -> F(OpnLid)) ) &
G( G( CLS) ->  F(G(!OpnLid)) | G(OpnLid -> F(ClsLid)) ) &
!OLS &
G( !OLS &  X(OLS) ->  !CLS & !X(CLS) & OpnLid ) &
G( !OLS & !X(OLS) ->  true ) &
G(  OLS & !X(OLS) ->  !CLS & !X(CLS) & ClsLid ) &
G(  OLS &  X(OLS) ->  !CLS & !X(CLS) ) &
G( G(!OLS) ->  F(G(!OpnLid)) | G(OpnLid -> F(ClsLid)) ) &
```

```
G( G( OLS) ->  F(G(!ClsLid)) | G(ClsLid -> F(OpnLid)) ) &
!WS0 &
G( !WS0 &  X(WS0) ->  !WS1 & !X(WS1) & FMech & !CLS ) &
G( !WS0 & !X(WS0) ->  !WS1 & !X(WS1) ) &
G(  WS0 & !X(WS0) ->  !WS1 & !X(WS1) & Valve &  WTS ) &
G(  WS0 &  X(WS0) ->  true ) &
G( G(!WS0) ->  true ) &
G( G( WS0) ->  F(G(!Valve | !WTS)) | G(Valve & WTS -> F(FMech & !CLS)) ) &
!WS1 &
G( !WS1 &  X(WS1) ->  WS0 & X(WS0) & FMech & !CLS ) &
G( !WS1 & !X(WS1) ->  true ) &
G(  WS1 & !X(WS1) ->  WS0 & X(WS0) & Valve &  WTS ) &
G(  WS1 &  X(WS1) ->  WS0 & X(WS0) ) &
G( G(!WS1) ->  true ) &
G( G( WS1) ->  F(G(!Valve | !WTS)) | G(Valve & WTS -> F(FMech & !CLS)) ) &
!UTS &
G( !UTS &  X(UTS) ->  WTS & X(WTS) & LTS & X(LTS) & Heater ) &
G( !UTS & !X(UTS) ->  true ) &
G(  UTS & !X(UTS) ->  WTS & X(WTS) & LTS & X(LTS) ) &
G(  UTS &  X(UTS) ->  WTS & X(WTS) & LTS & X(LTS) ) &
G( G(!UTS) ->  true ) &
G( G( UTS) ->  F(G(Heater)) | G(!Heater -> F(Heater)) ) &
!LTS &
G( !LTS &  X(LTS) ->  WTS &  X(WTS) & !UTS & !X(UTS) & Heater ) &
G( !LTS & !X(LTS) -> !UTS & !X(UTS) ) &
G(  LTS & !X(LTS) ->  WTS &  X(WTS) & !UTS & !X(UTS) ) &
G(  LTS &  X(LTS) ->  WTS &  X(WTS) ) &
G( G(!LTS) ->  true ) &
G( G( LTS) ->  F(G(Heater)) | G(!Heater -> F(Heater)) ) &
!WTS &
G( !WTS &  X(WTS) ->  !LTS & !X(LTS) & !UTS & !X(UTS) & Heater ) &
G( !WTS & !X(WTS) ->  !LTS & !X(LTS) & !UTS & !X(UTS) ) &
G(  WTS & !X(WTS) ->  !LTS & !X(LTS) & !UTS & !X(UTS) ) &
G(  WTS &  X(WTS) ->  true ) &
G( G(!WTS) ->  true ) &
G( G( WTS) ->  F(G(Heater)) | G(!Heater -> F(Heater)) ) &
-- Timers:
!MTmr.Q &
G( !MTmr.Q &  X(MTmr.Q) ->  MTmr.In ) &
G( !MTmr.Q & !X(MTmr.Q) ->  true    ) &
G(  MTmr.Q & !X(MTmr.Q) -> !MTmr.In ) &
G(  MTmr.Q &  X(MTmr.Q) ->  MTmr.In ) &
G( G(!MTmr.Q) ->  F(G(!MTmr.In)) | G(MTmr.In -> F(!MTmr.In)) ) &
G( G( MTmr.Q) ->  G(MTmr.In) ) &
!HTmr.Q &
G( !HTmr.Q &  X(HTmr.Q) ->  HTmr.In ) &
G( !HTmr.Q & !X(HTmr.Q) ->  true    ) &
G(  HTmr.Q & !X(HTmr.Q) -> !HTmr.In ) &
G(  HTmr.Q &  X(HTmr.Q) ->  HTmr.In ) &
G( G(!HTmr.Q) ->  F(G(!HTmr.In)) | G(HTmr.In -> F(!HTmr.In)) ) &
G( G( HTmr.Q) ->  G(HTmr.In) ) &
!FTmr.Q &
G( !FTmr.Q &  X(FTmr.Q) ->  FTmr.In ) &
G( !FTmr.Q & !X(FTmr.Q) ->  true    ) &
```

```
G(  FTmr.Q & !X(FTmr.Q) -> !FTmr.In ) &
G(  FTmr.Q &  X(FTmr.Q) ->  FTmr.In ) &
G( G(!FTmr.Q) ->  F(G(!FTmr.In)) | G(FTmr.In -> F(!FTmr.In)) ) &
G( G( FTmr.Q) ->  G(FTmr.In) ) &
!CTmr.Q &
G( !CTmr.Q &  X(CTmr.Q) ->  CTmr.In ) &
G( !CTmr.Q & !X(CTmr.Q) ->  true    ) &
G(  CTmr.Q & !X(CTmr.Q) -> !CTmr.In ) &
G(  CTmr.Q &  X(CTmr.Q) ->  CTmr.In ) &
G( G(!CTmr.Q) ->  F(G(!CTmr.In)) | G(CTmr.In -> F(!CTmr.In)) ) &
G( G( CTmr.Q) ->  G(CTmr.In) )
```

Приложение 3. Доказательства

Утверждение 1 (Об эквивалентности). $\mathbf{GF}(a) \wedge \mathbf{G}(a \Rightarrow \mathbf{F}(b)) = \mathbf{GF}(a) \wedge \mathbf{GF}(b)$.

Доказательство. Представим доказательство в виде двух шагов.

Шаг 1. $\mathbf{GF}(a) \wedge \mathbf{GF}(b) \vdash \mathbf{GF}(a) \wedge \mathbf{G}(a \Rightarrow \mathbf{F}(b))$ очевидно, так как $\mathbf{GF}(b) \vdash \mathbf{G}(\neg a \vee \mathbf{F}(b))$.

Шаг 2. Докажем $\mathbf{GF}(a) \wedge \mathbf{G}(a \Rightarrow \mathbf{F}(b)) \vdash \mathbf{GF}(a) \wedge \mathbf{GF}(b)$ методом от противного. Пусть существует такой путь, что на нём имеем справедливость формул $\mathbf{GF}(a) \wedge \mathbf{G}(a \Rightarrow \mathbf{F}(b))$ и $\neg(\mathbf{GF}(a) \wedge \mathbf{GF}(b)) = \mathbf{FG}(\neg a) \vee \mathbf{FG}(\neg b)$. Тогда достаточно рассмотреть следующие два варианта: 1) истинность $\mathbf{FG}(\neg a) = \neg\mathbf{GF}(a)$ противоречит подформуле $\mathbf{GF}(a)$ из посылки, 2) если верна $\mathbf{FG}(\neg b)$, то $\mathbf{G}(a \Rightarrow \mathbf{F}(b)) = \mathbf{G}(\neg a \vee \mathbf{F}(b))$ будет выполняться на этом пути только в виде $\mathbf{FG}(\neg a)$, что соответствует предыдущему варианту 1. Пришли к противоречию. Следовательно, верно обратное, т. е. утверждение на шаге 2 доказано. □

Обозначим доказанную эквивалентность (утверждение 1) как правило $R1$. Перечислим ещё ряд правил, которые будут использоваться для доказательства следующих утверждений:

$R1:$	$\mathbf{GF}(a) \wedge \mathbf{G}(a \Rightarrow \mathbf{F}(b)) = \mathbf{GF}(a) \wedge \mathbf{GF}(b)$	$R10:$	$\mathbf{GFG}(p) = \mathbf{FG}(p)$
$R2:$	$p \Rightarrow q = \neg p \vee q$	$R11:$	$\mathbf{FGF}(p) = \mathbf{GF}(p)$
$R3:$	$p \vee q = p \vee q \wedge \neg p$	$R12:$	$\mathbf{GF}(p \vee q) = \mathbf{GF}(p) \vee \mathbf{GF}(q)$
$R4:$	$\neg\neg p = p$	$R13:$	$\mathbf{F}(\mathit{False}) = \mathit{False}$
$R5:$	$\neg\mathbf{G}(p) = \mathbf{F}(\neg p)$	$R14:$	$p \vee \mathit{False} = p$
$R6:$	$\neg\mathbf{F}(p) = \mathbf{G}(\neg p)$	$R15:$	$p \vee q = p$, если $q \vdash p$
$R7:$	$\neg\mathbf{FG}(p) = \mathbf{GF}(\neg p)$	$R16:$	$p \wedge q = p$, если $p \vdash q$
$R8:$	$\neg\mathbf{GF}(p) = \mathbf{FG}(\neg p)$	$R17:$	$\neg(p \wedge q) = (\neg p \vee \neg q)$
$R9:$	$\mathbf{F}(p \vee q) = \mathbf{F}(p) \vee \mathbf{F}(q)$	$R18:$	$\neg\mathbf{X}(p) = \mathbf{X}(\neg p)$

Правила $R2, \ldots, R18$ — хорошо известные эквивалентные преобразования или следствия из семантики языка LTL, которые легко могут быть доказаны, как, например, это было сделано для $R1$.

Утверждение 2 (Об эквивалентности). *Шаблон справедливости № 1 представляет собой сильную справедливость:* $\mathbf{G}[\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1) \vee \mathbf{G}(act_1 \Rightarrow \mathbf{F}(act_0))\,] = \mathbf{GF}(act_1) \Rightarrow \mathbf{GF}(act_0 \vee rea)$.

Доказательство.

$$\mathbf{G}(\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1) \vee \mathbf{G}(act_1 \Rightarrow \mathbf{F}(act_0))\,) = \qquad [R2, R5, R4]$$

$$\mathbf{G}(\,\mathbf{F}(rea) \vee \mathbf{FG}(\neg act_1) \vee \mathbf{G}(act_1 \Rightarrow \mathbf{F}(act_0))\,) = \qquad [R3]$$

$$\mathbf{G}(\,\mathbf{F}(rea) \vee \mathbf{FG}(\neg act_1) \vee \mathbf{G}(act_1 \Rightarrow \mathbf{F}(act_0)) \wedge \neg\mathbf{FG}(\neg act_1)\,) = \qquad [R7, R4]$$

$$\mathbf{G}(\,\mathbf{F}(rea) \vee \mathbf{FG}(\neg act_1) \vee \mathbf{G}(act_1 \Rightarrow \mathbf{F}(act_0)) \wedge \ \mathbf{GF}(\ act_1)\,) = \qquad [R1]$$

$$\mathbf{G}(\,\mathbf{F}(rea) \vee \mathbf{FG}(\neg act_1) \vee \mathbf{GF}(act_0) \wedge \mathbf{GF}(act_1)\,) = \qquad [R7, R4]$$

$$\mathbf{G}(\,\mathbf{F}(rea) \vee \mathbf{FG}(\neg act_1) \vee \mathbf{GF}(act_0) \wedge \neg\mathbf{FG}(\neg act_1)\,) = \quad [R3]$$
$$\mathbf{G}(\,\mathbf{F}(rea) \vee \mathbf{FG}(\neg act_1) \vee \mathbf{GF}(act_0)\,) = \quad [R11]$$
$$\mathbf{G}(\,\mathbf{F}(rea) \vee \mathbf{FG}(\neg act_1) \vee \mathbf{FGF}(act_0)\,) = \quad [R9]$$
$$\mathbf{GF}(\,rea \vee \mathbf{G}(\neg act_1) \vee \mathbf{GF}(act_0)\,) = \quad [R12]$$
$$\mathbf{GF}(rea) \vee \mathbf{GFG}(\neg act_1) \vee \mathbf{GFGF}(act_0) = \quad [R10, R11]$$
$$\mathbf{GF}(rea) \vee \mathbf{FG}(\neg act_1) \vee \mathbf{GF}(act_0) = \quad [R12]$$
$$\mathbf{FG}(\neg act_1) \vee \mathbf{GF}(act_0 \vee rea) = \quad [R2, R7, R4]$$
$$\mathbf{GF}(act_1) \Rightarrow \mathbf{GF}(act_0 \vee rea). \quad \square$$

Утверждение 3 (О частном случае). *Формула* $\mathbf{G}(\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1)\,)$ *является частным случаем формулы* (16) *при* $act_0 = False$.

Доказательство.

$$\mathbf{G}[\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1) \vee \mathbf{G}(act_1 \Rightarrow \mathbf{F}(False))\,] = \quad [R13]$$
$$\mathbf{G}[\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1) \vee \mathbf{G}(act_1 \Rightarrow False)\,] = \quad [R2]$$
$$\mathbf{G}[\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1) \vee \mathbf{G}(\neg act_1 \vee False)\,] = \quad [R14]$$
$$\mathbf{G}[\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1) \vee \mathbf{G}(\neg act_1)\,] = \quad [R15]$$
$$\mathbf{G}[\,\mathbf{G}(\neg rea) \Rightarrow \mathbf{FG}(\neg act_1)\,]. \quad \square$$

Утверждение 4 (О частном случае). *Формула* $\mathbf{GF}(act_1 \Rightarrow rea)$ *является частным случаем формулы* (19) *при* $act_0 = \neg act_1$.

Доказательство.

$$\mathbf{GF}(act_1) \Rightarrow \mathbf{GF}(\neg act_1 \vee rea) = \quad [R2]$$
$$\neg\mathbf{GF}(act_1) \vee \mathbf{GF}(\neg act_1 \vee rea) = \quad [R12]$$
$$\neg\mathbf{GF}(act_1) \vee \mathbf{GF}(\neg act_1) \vee \mathbf{GF}(rea) = \quad [R7]$$
$$\neg\mathbf{GF}(act_1) \vee \neg\mathbf{FG}(act_1) \vee \mathbf{GF}(rea) = \quad [R17]$$
$$\neg(\mathbf{GF}(act_1) \wedge \mathbf{FG}(act_1)) \vee \mathbf{GF}(rea) = \quad [R16]$$
$$\neg\mathbf{FG}(\ \ act_1) \vee \mathbf{GF}(rea) = \quad [R7]$$
$$\mathbf{GF}(\neg act_1) \vee \mathbf{GF}(rea) = \quad [R12]$$
$$\mathbf{GF}(\neg act_1 \vee rea) = \quad [R2]$$
$$\mathbf{GF}(act_1 \Rightarrow rea). \quad \square$$

Приложение 4. SMV-спецификация поведения КФС: ПЛУ и её окружения

```
MODULE Timer
VAR
    In : boolean; -- Input  (Logical Control Program Unit Output)
    Q  : boolean; -- Output (Logical Control Program Unit Input)
ASSIGN
    INIT ( !Q & !In )
    TRANS( !Q &  next(Q) ->  In )
    TRANS(  Q & !next(Q) -> !In )
    TRANS(  Q &  next(Q) ->  In )
    FAIRNESS (In -> Q) -- Timer Fair Turn On.
```

```
MODULE main
VAR
-- Buttons:
PBStart : boolean;  PBStop : boolean;  PBCompl : boolean;  PBConvr : boolean;
-- Imaginary Sensors:
IFS : boolean;
-- Sensors:
FS1 : boolean;  FS2 : boolean;  OLS : boolean;  CLS : boolean;  WS0 : boolean;
WS1 : boolean;  UTS : boolean;  LTS : boolean;  WTS : boolean;
-- Timers:
FTmr : Timer;   HTmr : Timer;   CTmr : Timer;   MTmr : Timer;
-- Variables:
SysOn : boolean;  Compl : boolean;  FErr : boolean;  CErr : boolean;  HErr : boolean;
Disch : boolean;  Mlted : boolean;  Mltng : boolean;  Fin : boolean;
-- Actuators:
Heater : boolean;   FMech : boolean;   Convr : boolean;  LwSpd : boolean;
Valve  : boolean;  OpnLid : boolean;  ClsLid : boolean;
-- Behavior
-- Buttons:
INIT ( !PBStart & !PBStop & !PBCompl & !PBConvr )
-- Sensors:
INIT ( !FS1 )
TRANS( !FS1 &  next(FS1) ->  !FS2 & !next(FS2) & !IFS & !next(IFS) & Convr )
TRANS(  FS1 & !next(FS1) ->  !FS2 & !next(FS2) & !IFS &  next(IFS) & Convr )
INIT ( !IFS )
TRANS( !IFS &  next(IFS) ->  !FS2 & !next(FS2) &  FS1 & !next(FS1) & Convr )
TRANS(  IFS & !next(IFS) ->  !FS2 &  next(FS2) & !FS1 & !next(FS1) & Convr )
INIT ( !FS2 )
TRANS( !FS2 &  next(FS2) ->   IFS & !next(IFS) & !FS1 & !next(FS1) & Convr )
TRANS(  FS2 & !next(FS2) ->  !IFS & !next(IFS) & !FS1 & !next(FS1) & Convr )
INIT ( !OLS )
TRANS( !OLS &  next(OLS) ->  !CLS & !next(CLS) & OpnLid )
TRANS(  OLS & !next(OLS) ->  !CLS & !next(CLS) & ClsLid )
INIT ( !CLS )
TRANS( !CLS &  next(CLS) ->  !OLS & !next(OLS) & ClsLid )
TRANS(  CLS & !next(CLS) ->  !OLS & !next(OLS) & OpnLid )
INIT ( !WS0 )
TRANS( !WS0 &  next(WS0) ->  !WS1 & !next(WS1) & FMech & !CLS )
TRANS(  WS0 & !next(WS0) ->  !WS1 & !next(WS1) & Valve &  WTS )
INIT ( !WS1 )
TRANS( !WS1 &  next(WS1) ->  WS0 & next(WS0) & FMech & !CLS )
TRANS(  WS1 & !next(WS1) ->  WS0 & next(WS0) & Valve &  WTS )
INIT ( !UTS )
TRANS( !UTS &  next(UTS) ->  WTS & next(WTS) & LTS & next(LTS) & Heater )
TRANS(  UTS & !next(UTS) ->  WTS & next(WTS) & LTS & next(LTS)          )
INIT ( !LTS )
TRANS( !LTS &  next(LTS) ->  WTS & next(WTS) & !UTS & !next(UTS) & Heater )
TRANS(  LTS & !next(LTS) ->  WTS & next(WTS) & !UTS & !next(UTS)          )
INIT ( !WTS )
TRANS( !WTS &  next(WTS) ->  !LTS & !next(LTS) & !UTS & !next(UTS) & Heater )
TRANS(  WTS & !next(WTS) ->  !LTS & !next(LTS) & !UTS & !next(UTS)          )
-- Control Software
-- Variables:
INIT ( !FErr )
```

```
TRANS( !FErr &  next(FErr) ->  (next(FTmr.Q) & !next(WS1)) )
TRANS( !FErr & !next(FErr) -> !(next(FTmr.Q) & !next(WS1)) )
TRANS(  FErr & !next(FErr) ->  next(PBStop) )
TRANS(  FErr &  next(FErr) -> !next(PBStop) )
INIT ( !CErr )
TRANS( !CErr &  next(CErr) ->  (next(CTmr.Q) & !next(FS2)) )
TRANS( !CErr & !next(CErr) -> !(next(CTmr.Q) & !next(FS2)) )
TRANS(  CErr & !next(CErr) ->  next(PBStop) )
TRANS(  CErr &  next(CErr) -> !next(PBStop) )
INIT ( !HErr )
TRANS( !HErr &  next(HErr) ->  (!next(WTS) & (next(HTmr.Q) | Heater & WTS)) )
TRANS( !HErr & !next(HErr) -> !(!next(WTS) & (next(HTmr.Q) | Heater & WTS)) )
TRANS(  HErr & !next(HErr) ->  next(PBStop) )
TRANS(  HErr &  next(HErr) -> !next(PBStop) )
INIT ( !Fin )
TRANS( next(Fin) <-> (Compl & next(OLS) & !next(WS0) | next(FErr) | next(HErr) |
                      next(CErr) | next(PBStop) | next(PBConvr)) )
INIT ( !SysOn )
TRANS( !SysOn &  next(SysOn) ->  (next(PBStart) & !next(Fin)) )
TRANS( !SysOn & !next(SysOn) -> !(next(PBStart) & !next(Fin)) )
TRANS(  SysOn & !next(SysOn) ->  next(Fin) )
TRANS(  SysOn &  next(SysOn) -> !next(Fin) )
INIT ( !Compl )
TRANS( !Compl &  next(Compl) ->  (next(SysOn) & next(PBCompl)) )
TRANS( !Compl & !next(Compl) -> !(next(SysOn) & next(PBCompl)) )
TRANS(  Compl & !next(Compl) -> !next(SysOn) )
TRANS(  Compl &  next(Compl) ->  next(SysOn) )
INIT ( !Mltng )
TRANS( next(Mltng) <-> next(SysOn) & next(CLS) & next(Disch) & next(WS0) )
INIT ( !Mlted )
TRANS( !Mlted &  next(Mlted) ->  (next(MTmr.Q) & next(WTS) & next(WS0) & next(CLS)) )
TRANS( !Mlted & !next(Mlted) -> !(next(MTmr.Q) & next(WTS) & next(WS0) & next(CLS)) )
TRANS(  Mlted & !next(Mlted) ->  (!next(WTS) | !next(CLS) | !next(WS0)) )
TRANS(  Mlted &  next(Mlted) -> !(!next(WTS) | !next(CLS) | !next(WS0)) )
INIT ( !Disch )
TRANS( !Disch &  next(Disch) ->  (next(OLS) & next(WS1)) )
TRANS( !Disch & !next(Disch) -> !(next(OLS) & next(WS1)) )
TRANS(  Disch & !next(Disch) ->  (!next(FS2) & !next(WS1)) )
TRANS(  Disch &  next(Disch) -> !(!next(FS2) & !next(WS1)) )
-- Actuators:
INIT ( !Heater )
TRANS( !Heater &  next(Heater) ->  (next(SysOn) & !next(LTS)) )
TRANS( !Heater & !next(Heater) -> !(next(SysOn) & !next(LTS)) )
TRANS(  Heater & !next(Heater) ->  (!next(SysOn) | next(UTS)) )
TRANS(  Heater &  next(Heater) -> !(!next(SysOn) | next(UTS)) )
INIT ( !ClsLid )
TRANS( next(ClsLid) <-> next(SysOn) & !next(CLS) & next(Disch) )
INIT ( !OpnLid )
TRANS( next(OpnLid) <-> next(SysOn) & !next(OLS) & !next(Disch) )
INIT ( !Convr )
TRANS( next(Convr) <-> (next(SysOn) & (next(FS2) & next(Disch) & !next(WS0) |
                        !next(FS2)) | next(PBConvr)) )
INIT ( !LwSpd )
TRANS( !LwSpd &  next(LwSpd) ->  next(FS1) )
```

```
TRANS( !LwSpd & !next(LwSpd) -> !next(FS1) )
TRANS(  LwSpd & !next(LwSpd) ->  next(FS2) )
TRANS(  LwSpd &  next(LwSpd) -> !next(FS2) )
INIT ( !FMech )
TRANS( next(FMech) <-> next(SysOn) & !next(Disch) & next(OLS) & !next(WS1) )
INIT ( !Valve )
TRANS( next(Valve) <-> next(SysOn) & next(Disch) & next(Mlted) & next(FS2) )
-- Impact_On_Timers:
TRANS( next(FTmr.In) <-> next(FMech)  & !next(WS1) )
TRANS( next(HTmr.In) <-> next(Heater) & !next(WTS) )
TRANS( next(CTmr.In) <-> next(Convr)  & !next(FS2) & next(SysOn) )
TRANS( next(MTmr.In) <-> next(Mltng)  &  next(WTS) & next(WS0) & next(CLS) )
-- Fairness Conditions:
COMPASSION (Convr, !FS2)
COMPASSION (OpnLid, ClsLid | OLS)
COMPASSION (ClsLid, OpnLid | CLS)
COMPASSION (Valve & WTS, (FMech & !CLS) | !WS0)
FAIRNESS   (!Heater -> !WTS)
-- [no use] COMPASSION (ClsLid, OpnLid | !OLS)
-- [no use] COMPASSION (OpnLid, ClsLid | !CLS)
-- [no use] COMPASSION (Valve & WTS, (FMech & !CLS) | !WS1)
-- [no use] FAIRNESS (!Heater -> !UTS)
-- [no use] FAIRNESS (!Heater -> !LTS)
```

Приложение 5. Императивная LTL-спецификация поведения ПЛУ

```
!FErr &
G( !FErr & X(FTmr.Q) & !X(WS1) ->  X(FErr) ) &
G(  FErr & X(PBStop)           -> !X(FErr) ) &
G( !(!FErr & X(FTmr.Q) & !X(WS1)) & !(FErr & X(PBStop)) -> (X(FErr) <-> FErr) ) &
!CErr &
G( !CErr & X(CTmr.Q) & !X(FS2) ->  X(CErr) ) &
G(  CErr & X(PBStop)           -> !X(CErr) ) &
G( !(!CErr & X(CTmr.Q) & !X(FS2)) & !(CErr & X(PBStop)) -> (X(CErr) <-> CErr) ) &
!HErr &
G( !HErr & (!X(WTS) & (X(HTmr.Q) | Heater & WTS)) ->  X(HErr) ) &
G(  HErr & X(PBStop)                              -> !X(HErr) ) &
G( !(!HErr & (!X(WTS) & (X(HTmr.Q) | Heater & WTS)) & !(HErr & X(PBStop)) ->
                                                     (X(HErr) <-> HErr) ) &
!Fin &
G( X(Fin) <-> (Compl & X(OLS) & !X(WS0) | X(fErr) | X(HErr) | X(CErr) |
                                          X(PBStop) | X(PBConvr)) ) &
!SysOn &
G( !SysOn & X(PBStart) & !X(Fin) ->  X(SysOn) ) &
G(  SysOn & X(Fin)               -> !X(SysOn) ) &
G( !(!SysOn & X(PBStart) & !X(Fin)) & !(SysOn & X(Fin)) -> (X(SysOn) <-> SysOn) ) &
!Compl &
G( !Compl &  X(SysOn) & X(PBCompl) ->  X(Compl) ) &
G(  Compl & !X(SysOn)              -> !X(Compl) ) &
G( !(!Compl & X(SysOn) & X(PBCompl)) & !(Compl & !X(SysOn)) -> (X(Compl)<->Compl) ) &
!Mltng &
G( X(Mltng) <-> X(SysOn) & X(CLS) & X(Disch) & X(WS0) ) &
!Mlted &
G( !Mlted & X(MTmr.Q) & X(WTS) &  X(WS0) & X(CLS) ->  X(Mlted) ) &
```

```
    G(  Mlted & (!X(WTS) | !X(CLS) | !X(WS0))         -> !X(Mlted) ) &
    G( !(!Mlted & X(MTmr.Q) & X(WTS) & X(WS0) & X(CLS)) &
       !(Mlted & (!X(WTS) | !X(CLS) | !X(WS0))) -> (X(Mlted) <-> Mlted) ) &
    !Disch &
    G( !Disch &  X(OLS) &  X(WS1) ->  X(Disch) ) &
    G(  Disch & !X(FS2) & !X(WS1) -> !X(Disch) ) &
    G( !(!Disch & X(OLS) & X(WS1)) & !(Disch & !X(FS2) & !X(WS1)) ->
                                          (X(Disch) <-> Disch) ) &
    -- Actuators:
    !Heater &
    G( !Heater & ( X(SysOn) & !X(LTS)) ->  X(Heater) ) &
    G(  Heater & (!X(SysOn) |  X(UTS)) -> !X(Heater) ) &
    G( !(!Heater & (X(SysOn) & !X(LTS))) & !(Heater & (!X(SysOn) |  X(UTS))) ->
                                                  (X(Heater) <-> Heater) ) &
    !ClsLid &
    G( X(ClsLid) <-> X(SysOn) & !X(CLS) & X(Disch) ) &
    !OpnLid &
    G( X(OpnLid) <-> X(SysOn) & !X(OLS) & !X(Disch) ) &
    !Convr &
    G( X(Convr) <-> (X(SysOn) & (X(FS2) & X(Disch) & !X(WS0) | !X(FS2)) | X(PBConvr)) ) &
    !LwSpd &
    G( !LwSpd & X(FS1) ->  X(LwSpd) ) &
    G(  LwSpd & X(FS2) -> !X(LwSpd) ) &
    G( !(!LwSpd & X(FS1)) & !(LwSpd & X(FS2)) -> (X(LwSpd) <-> LwSpd))
    G( X(FMech) <-> X(SysOn) & !X(Disch) & X(OLS) & !X(WS1) ) & !FMech &
    G( X(Valve) <-> X(SysOn) & X(Disch) & X(Mlted) & X(FS2) ) & !Valve &
    -- Impact_On_Timers:
    G( X(FTmr.In) <-> X(FMech)  & !X(WS1) )                  & !FTmr.In &
    G( X(HTmr.In) <-> X(Heater) & !X(WTS) )                  & !HTmr.In &
    G( X(CTmr.In) <-> X(Convr)  & !X(FS2) & X(SysOn) )       & !CTmr.In &
    G( X(MTmr.In) <-> X(Mltng)  &  X(WTS) & X(WS0) & X(CLS) ) & !MTmr.In
```

Приложение 6. ST-программа управления установкой для литья пластмасс

```
PROGRAM PLC_PRG
VAR_INPUT
    PBStart, PBStop, PBCompl, PBConvr: BOOL := FALSE;           (* Buttons *)
    FS1, FS2, OLS, CLS, WS0, WS1, UTS, LTS, WTS: BOOL := FALSE; (* Sensors *)
END_VAR
VAR_OUTPUT
    SysOn, Compl, FErr, CErr, HErr, Disch, Mltng, Mlted: BOOL := FALSE; (* Lamps     *)
    Heater, FMech, Convr, LwSpd, Valve, OpnLid, ClsLid: BOOL := FALSE;  (* Actuators *)
END_VAR
VAR
    (* Timers *)
    MTmr: TON := (In := FALSE, Q := FALSE, PT := T#6s );
    HTmr: TON := (In := FALSE, Q := FALSE, PT := T#12s);
    FTmr: TON := (In := FALSE, Q := FALSE, PT := T#10s);
    CTmr: TON := (In := FALSE, Q := FALSE, PT := T#18s);
    Fin: BOOL := FALSE;     (* Internal variable *)
    (* Auxiliary variables *)
    _SysOn, _Compl, _Mlted, _Disch, _Heater : BOOL := FALSE;
    _LwSpd, _CErr, _FErr, _HErr, _WTS : BOOL := FALSE;
END_VAR
```

```
(* Calling timers *)
MTmr(); HTmr(); FTmr(); CTmr();
(* Variables *)
IF NOT _FErr AND FTmr.Q AND NOT WS1 THEN FErr:=1;  (* FErr *)
ELSIF  _FErr AND PBStop             THEN FErr:=0;
END_IF;
IF NOT _CErr AND CTmr.Q AND NOT FS2 THEN CErr:=1;  (* CErr *)
ELSIF  _CErr AND PBStop             THEN CErr:=0;
END_IF;
IF NOT _HErr AND NOT WTS AND
   (HTmr.Q OR _Heater AND _WTS) THEN HErr:=1;  (* HErr *)
ELSIF  _HErr AND PBStop         THEN HErr:=0;
END_IF;
Fin:= _Compl AND OLS AND NOT WS0 OR FErr OR HErr OR CErr OR
       PBStop OR PBConvr;  (* Fin *)
IF NOT _SysOn AND PBStart AND NOT Fin THEN SysOn:=1;  (* SysOn *)
ELSIF  _SysOn AND                 Fin THEN SysOn:=0;
END_IF;
IF NOT _Compl AND     SysOn AND PBCompl THEN Compl:=1;  (* Compl *)
ELSIF  _Compl AND NOT SysOn             THEN Compl:=0;
END_IF;
Mltng := SysOn AND CLS AND Disch AND WS0;  (* Mltng *)
IF NOT _Mlted AND MTmr.Q AND WTS AND WS0 AND CLS  THEN Mlted:=1;  (* Mlted *)
ELSIF  _Mlted AND (NOT WTS OR NOT CLS OR NOT WS0) THEN Mlted:=0;
END_IF;
IF NOT _Disch AND OLS     AND     WS1 THEN Disch:=1;  (* Disch *)
ELSIF  _Disch AND NOT FS2 AND NOT WS1 THEN Disch:=0;
END_IF;
(* Actuators *)
IF NOT _Heater AND      SysOn AND NOT LTS  THEN Heater:=1;  (* Heater *)
ELSIF  _Heater AND (NOT SysOn OR      UTS) THEN Heater:=0;
END_IF;
ClsLid:= SysOn AND NOT CLS AND     Disch;                              (* ClsLid *)
OpnLid:= SysOn AND NOT OLS AND NOT Disch;                              (* OpnLid *)
Convr := SysOn AND (FS2 AND Disch AND NOT WS0 OR NOT FS2) OR PBConvr;  (* Convr  *)
IF NOT _LwSpd AND FS1 THEN LwSpd:=1;                                   (* LwSpd  *)
ELSIF  _LwSpd AND FS2 THEN LwSpd:=0;
END_IF;
FMech := SysOn AND NOT Disch AND OLS AND NOT WS1;                      (* FMech  *)
Valve := SysOn AND     Disch AND Mlted AND FS2;                        (* Valve  *)
(* Impact_On_Timers *)
FTmr.In:= FMech  AND NOT WS1;              (* FTmr.In *)
HTmr.In:= Heater AND NOT WTS;              (* HTmr.In *)
CTmr.In:= Convr  AND SysOn AND NOT FS2;    (* CTmr.In *)
MTmr.In:= Mltng  AND WTS AND WS0 AND CLS;  (* MTmr.In *)
(* Pseudo operator section: saving previous values of variables *)
_SysOn:=SysOn; _Compl:=Compl; _Disch:=Disch; _Mlted:=Mlted; _Heater:=Heater;
_LwSpd:=LwSpd; _FErr:=FErr; _CErr:=CErr; _HErr:=HErr; _WTS:=WTS;
```

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 31, NO. 3, 2024
journal homepage: www.mais-journal.ru

COMPUTING METHODOLOGIES AND APPLICATIONS

Matrix-qubit Algorithm for Semantic Analysis of Probabilistic Data

I. A. Surov[1]

DOI: 10.18255/1818-1015-2024-3-280-293

[1]ITMO University, Saint-Petersburg, Russia

MSC2020: 68Q09, 15A23
Research article
Full text in Russian

Received July 22, 2024
Revised August 19, 2024
Accepted August 28, 2024

The paper presents a method for semantic data analysis by means of complex-valued matrix decomposition. The method is based on the quantum model of contextual decision-making, according to which observable probabilities are generated by qubit states, representing subjective meaning of the contexts relative to the basis decision. In the simplest three-context case, one of these qubits is decomposed to superposition of the remaining two, mathematically encoding semantic relations between the three contexts. For use in data analysis this model is translated to the matrix form, in which rows and columns correspond to the contexts and instances of experiment. The observable real-valued data then emerge from a complex-valued amplitude matrix, decomposed to a product of a real basis matrix and complex-valued matrix of superposition coefficients. This decomposition reveals stable process-semantic relations between the considered contexts, not captured by other methods of analysis. As a result, the data are approximated with higher precision and fewer parameters than singular and non-negative matrix decompositions, truncated to the same dimension. The model is experimentally approved in descriptive and prognostic regimes. The result opens prospects for development of nature-like computational architectures on novel logical grounds.

Keywords: semantic analysis; behavioral modeling; matrix decomposition; context; quantum probability; quantum logic; qubit

INFORMATION ABOUT THE AUTHORS

Surov, Ilya A. (corresponding author) | ORCID iD: 0000-0001-5690-7507. E-mail: ilya.a.surov@itmo.ru
PhD, Senior researcher

Funding: Russian Science Foundation, project No. 23-71-01046.

For citation: I. A. Surov, "Matrix-qubit algorithm for semantic analysis of probabilistic data", *Modeling and Analysis of Information Systems*, vol. 31, no. 3, pp. 280–293, 2024. DOI: 10.18255/1818-1015-2024-3-280-293.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 31, № 3, 2024
сайт журнала: www.mais-journal.ru

Матрично-кубитный алгоритм семантического анализа вероятностных данных

И. А. Суров[1]

DOI: 10.18255/1818-1015-2024-3-280-293

[1]Университет ИТМО, Санкт-Петербург, Россия

УДК 51-77
Научная статья
Полный текст на русском языке

Получена 22 июля 2024 г.
После доработки 19 августа 2024 г.
Принята к публикации 28 августа 2024 г.

В статье представлен метод семантического анализа данных посредством комплекснозначного матричного разложения. Метод основан на квантовой модели контекстно-чувствительных решений, согласно которой наблюдаемые вероятности порождаются кубитными состояниями, представляющими субъективный смысл контекстов для базисного решения. В простейшем трёхконтекстом случае один из кубитов раскладывается в суперпозицию оставшихся двух, математически представляющую смысловые отношения между контекстами. Для использования в задаче анализа данных эта модель представлена в матричной форме так, что строки и столбцы соответствуют контекстам и постановкам эксперимента. При этом наблюдаемые действительные данные порождаются матрицей комплекснозначных амплитуд, раскладываемой на произведение действительной матрицы базисных векторов и комплекснозначной матрицы коэффициентов суперпозиции. Это разложение выявляет устойчивые процессно-смысловые соотношения контекстов, не обнаруживаемые другими методами. В результате данные воспроизводятся более точно и с меньшим числом параметров, чем при использовании сингулярного и неотрицательного матричных разложений той же размерности. Модель успешно испытана в описательном и предсказательном режимах. Результат открывает возможности для разработки природоподобных вычислительных архитектур на новых логических принципах.

Ключевые слова: семантический анализ; поведенческое моделирование; матричное разложение; контекст; квантовая вероятность; квантовая логика; кубит

ИНФОРМАЦИЯ ОБ АВТОРАХ

Суров, Илья Алексеевич (автор для корреспонденции) | ORCID iD: 0000-0001-5690-7507. E-mail: ilya.a.surov@itmo.ru
Канд. физ.-мат. наук, старший научный сотрудник

Финансирование: РНФ, проект № 23-71-01046.

Для цитирования: I. A. Surov, “Matrix-qubit algorithm for semantic analysis of probabilistic data”, *Modeling and Analysis of Information Systems*, vol. 31, no. 3, pp. 280–293, 2024. DOI: 10.18255/1818-1015-2024-3-280-293.

Введение

Характерным свойством современных вычислительных систем является большая ресурсоёмкость, для многих практических задач требующая суперкомпьютерных мощностей. В шахматах и го, например, естественное мышление показывает сходные результаты с несравненно меньшим энергопотреблением и объёмом обучающих данных. С другой стороны, современная вычислительная парадигма не позволяет воспроизвести поведение нематоды Caernohabditis elegans, нервная система которого из 302 нейронов полностью картирована 38 лет назад [1]. Эти результаты указывают на принципиальное несовершенство современных имитаций естественного интеллекта, эффективность которого во многих случаях остаётся недостижимой.

Данная трудность мотивирует разработку методов анализа данных на новых логических принципах, более соответствующих работе естественного мышления. В качестве таковых рассматриваются законы оптическо-голографических и квантово-физических процессов, позволяющих использовать новые форматы кодирования и алгоритмы обработки информации [2—4]. В отличие от квантовых вычислений [5], этот подход не всегда требует использования новых типов материальных носителей. Такие системы не ведут к ускорению, характерному для квантовых компьютеров, однако позволяют добиваться прогресса в когнитивно-поведенческом моделировании благодаря лучшему соответствию с принципами естественного мышления [4, 6].

Это преимущество новых вычислительных принципов обусловлено переходом от Булевской «логики множеств» к «логике (когнитивных) волн», представляющих смысловое содержание нейронных возбуждений в естественном мышлении [7]. При этом информация кодируется не только амплитудными, но и фазовыми параметрами квантово-волновых состояний, что соответствует переходу от действительных к комплексным числам. Благодаря фазовым степеням свободы линейное наложение комплекснозначных амплитуд позволяет моделировать нелинейные («интерференционные») закономерности поведенческих данных, выходящие за рамки классической рациональности и теории вероятности Колмогорова [6, 7]. Как и в квантовой физике, такое моделирование поведенческих данных идёт посредством недоступных для прямого наблюдения квантово-волновых объектов, представляющих когнитивно-психические состояния рассматриваемой системы.

Эти преимущества волнового и квантово-подобного подходов указывают на целесообразность их сопряжения с матричной алгеброй современных нейросетевых архитектур. Для этого наиболее удобна матричная формулировка квантовой теории, согласно которой квантовые состояния и наблюдаемые представляются векторами и операторами в многомерном пространстве [8]. В отличие от обычной линейной алгебры нейросетевых тензоров, эти вектора и операторы являются комплекснозначными как того требует волновая природа нейронных возбуждений. При этом фазовые параметры необходимы для «внутренних» вычислений, тогда как последующий переход к действительным наблюдаемым осуществляется путём взятия квадратного модуля от итоговых комплекснозначных величин. Эта операция, известная в квантовой физике как правило Борна и ключевая для отмеченной линеаризации, отличает данный подход от чисто комплекснозначных нейронных сетей [9] и матричных разложений [10], ограниченных работой с комплекснозначными либо двумерно-векторными данными.

Примером сопряжения квантовой логики и матричной алгебры является квантово-подобная модификация сингулярного матричного разложения [11], являющегося прообразом тензорной алгебры современных нейросетей. Как показано в этой работе, структура матриц сингулярного разложения позволяет соотнести их с элементами квантово-подобных поведенческих моделей. Последующее обобщение этих матриц с действительных на комплексные числа повышает точность воспроизведения данных за счёт более эффективного использования того же числа свободных параметров модели (там же). Это свойство, отмеченное ранее для квантово-подобного моделирования семантики естественного языка [12], подтвердило эффективность квантово-волновой алгебры

для имитации мышления человека. Ещё раз подчеркнём, что данный подход реализуется средствами классических компьютеров и не имеет прямого отношения к построению квантовых вычислительных систем, хотя сопряжение этих направлений также представляет интерес.

Недостатком работы [11] является отсутствие интерпретации элементов полученного разложения, аналогичной, например, интерпретации элементов неотрицательного разложения [13]. Возможность такой интерпретации, однако, следует из работ [14, 15] где показано, что простейшее квантовое состояние — кубит — можно использовать для математической формализации субъективно-эмоционального смысла. При этом декартовы оси пространства кубитных состояний соответствуют семантическим факторам Ч. Осгуда оценка — сила — активность, тогда как угловые сектора соответствуют главным эмоциональными состояниям. Поскольку в мышлении человека последние выполняют роль естественных смысловых категорий (там же), это соответствие открывает возможности разработки природоподобных вычислительных методов.

По этой причине в данной статье (в отличие от работы [11]) искомое сопряжение матричной алгебры и квантово-подобного моделирования строится со стороны последнего. Используемая для этого интерпретируемая модель вероятностных поведенческих данных представлена в разделе 1. В разделе 2 эта модель приводится к матричной форме, аналогичной неотрицательному разложению. Результаты описательного и прогнозного моделирования представлены в последующих разделах 3 и 4.

1. Кубитная модель контекстно-чувствительных решений

Объектом исходной поведенческой модели являются вероятности осуществления целевого события X в трёх связанных между собой ситуациях — контекстах A, B и C. Классическим примером такой ситуации является так называемая двухэтапная игра, в которой событием X является участие в следующем коне игры в орлянку в ситуациях, когда предыдущий кон выигран (A), проигран (B), или исход этого кона неизвестен (C) [16]. Полученные в ряде экспериментов вероятности этих решений $0 \leqslant p^a, p^b, p^c \leqslant 1$ систематически отклоняются от ожиданий классической рациональности, что позволяет использовать их для испытания методов анализа данных на новых логико-математических принципах [7].

Согласно модели [17, 18], принимая целевое решение X субъект порождает направленное на него двумерное комплекснозначное векторное (гильбертово) пространство, используемое для кодирования эмоционально-смысловых состояний принимающего решение лица. Контексты A, B и C представляются в этом пространстве тройкой векторов, в квантовых обозначениях записываемых как

$$
\begin{aligned}
|\Psi_a\rangle &= \begin{bmatrix} a_0 \\ e^{i\alpha} a_1 \end{bmatrix} = a_0|0\rangle + e^{i\alpha} a_1|1\rangle, \\
|\Psi_b\rangle &= \begin{bmatrix} b_0 \\ e^{i\beta} b_1 \end{bmatrix} = b_0|0\rangle + e^{i\beta} b_1|1\rangle, \\
|\Psi_c\rangle &= \begin{bmatrix} e^{i\gamma_0} c_0 \\ e^{i\gamma_1} c_1 \end{bmatrix} = e^{i\gamma_0} c_0|0\rangle + e^{i\gamma_1} c_1|1\rangle, \quad \gamma_1 - \gamma_0 = \gamma.
\end{aligned}
\tag{1}
$$

При этом $0 \leqslant a_k, b_k, c_k \leqslant 1$ есть действительные амплитуды, а $0 \leqslant \alpha, \beta, \gamma_k < 2\pi$ — фазовые параметры каждого из векторов (1), известных в квантовой информатике как кубитные состояния [8][1]. Этим термином обусловлено название «кубитная модель», используемое далее для обозначения модели, представленной в этом разделе.

[1] Специфическая квантовая нотация векторов (1), без потери смысла заменяющаяся на обычные векторные обозначения, использована для соответствия с работами [15, 17, 18]. В последующих разделах статьи эти обозначения практически не используются.

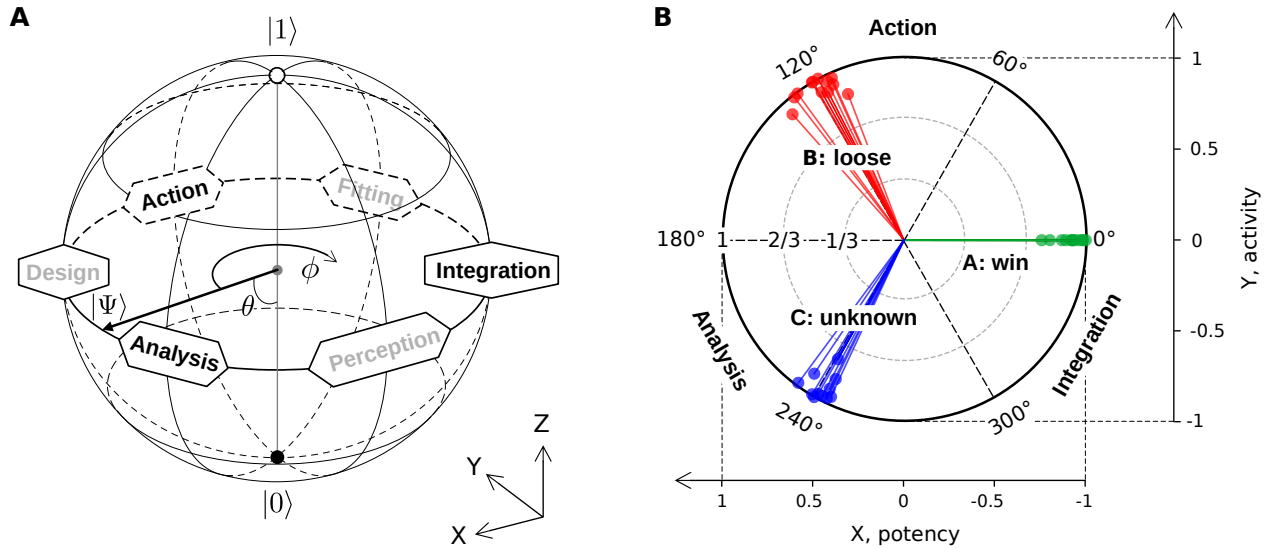


Fig. 1. A: Qubit state space in the Bloch sphere representation. Polar angle θ and azimuthal angle ϕ (α, β, γ) encode favorability and process function of the context in relation to the basis 1/0 decision in subject's cognition. B: 12 triples of states (1) in qubit model of two-stage gamble. Green, red and blue correspond to the contexts A, B and C, respectively.

Рис. 1. A: Пространство кубитных состояний в представлении сферы Блоха. Полярный угол θ и азимутальный угол ϕ (α, β, γ) кодируют благоприятность и процессную функцию контекста относительно базисного решения 1/0 в психике субъекта. B: 12 троек состояний (1) кубитной модели двухэтапной игры. Зелёным, красным и синим показаны контексты A, B и C.

Базисные состояния $|1\rangle$ и $|0\rangle$ соответствуют принятию и не-принятию целевого решения X, а комплекснозначные амплитуды перед ними определяют вероятности осуществления соответствующих альтернатив посредством операции квадратного модуля (правило Борна):

$$\begin{aligned} p^a &= |e^{i\alpha_1} a_1|^2 = a_1^2, & 1 - p^a &= a_0^2, \\ p^b &= |e^{i\beta_1} b_1|^2 = b_1^2, & 1 - p^b &= b_0^2, \\ p^c &= |e^{i\gamma_1} c_1|^2 = c_1^2, & 1 - p^c &= |e^{i\gamma_0} c_0|^2 = c_0^2, \end{aligned} \tag{2}$$

что определяет нормировку векторов (1).

Нормировка (2) позволяет представить двумерные комплекснозначные вектора (1) трёхмерными векторами на единичной сфере в действительном трёхмерном пространстве. При этом $\theta_{a,b,c} = 2\arcsin(a_1, b_1, c_1)$ являются полярными углами, а фазы α, β и $\gamma = \gamma_1 - \gamma_0$ являются азимутальным углами. Согласно модели [14, 18], полярные углы кодирует субъективную благоприятность контекста для базисного решения «1», определяющую вероятности (2). Азимутальные углы α, β, γ кодируют функцию контекста в жизненном цикле деятельности по осуществлению базисного решения X. Этот цикл состоит их последовательности этапов «восприятие», «анализ», «проектирование», «действие», «согласование» и «внедрение» (там же) как показано на рисунке 1A.

В рассматриваемой модели вектор $|\Psi_c\rangle$ «неопределённого» контекста C задаётся суперпозицией оставшихся векторов (1) как

$$|\Psi_c\rangle = N\left(|\Psi_a\rangle + e^{ix}|\Psi_b\rangle\right), \tag{3}$$

где $0 \leqslant x < 2\pi$ есть суперпозиционная фаза, а $N > 0$ — нормировочный множитель[2].

[2]Необходимость фазы γ_0 у нулевой компоненты вектора $|\Psi_c\rangle$, без потери общности положенной нулём в кубитах $|\Psi_a\rangle$ и $|\Psi_b\rangle$, следует из первой строки этого уравнения.

При фиксированном N система уравнений (1), (2), (3) решается в аналитическом виде и позволяет найти тройку векторов (1), представляющих контексты A, B и C в психике лица, принимающего решение X [17]. При этом вероятностные значения чувствительны лишь к относительным углам между векторами (1) на сфере Блоха, в силу чего фазу одного из них можно задать произвольно.

Этот выбор делается на основе процессной функции моделируемых контекстов. Для двухэтапной игры, например, «неопределённый» контекст C соответствует результату первого этапа жизненного цикла «восприятие», на котором из фона вниманием выделяется некоторый объект. Далее следует анализ и объяснение этого объекта в некотором языке, а затем — постановка целей в этом отношении и разработка планов по их осуществлению («проектирование», «design») с учётом состояния среды. Проигрыш предыдущего кона в контексте B соответствует неблагоприятному обстоятельству, преодоление которого происходит на следующих этапах «действие» и «согласование». Их результатом является контекст A — выигрыш. Фаза α соответствующего ему вектора $|\Psi_a\rangle$ фиксировалась на значении 0°. Процессный цикл завершается внедрением («integration») полученного результата в жизнь, например, путём использования выигранной суммы.

С этим условием модель с нормировкой $N = 1$ построена для $n = 12$ постановок двухэтапной игры, результаты которых приведены в таблице 1. Полученные 12 троек кубитных состояний (1) показаны на рисунке 1Б в проекции на экваториальную плоскость сферы Блоха. Согласно положенному значению $\alpha = 0^\circ$ вектора $|\Psi_a\rangle$ контекста A расположены на границе этапов «согласование» и «внедрение». В соответствии с вышеописанным ожиданием состояния $|\Psi_b\rangle$ и $|\Psi_c\rangle$ лежат на границах этапов «проектирование» — «действие» и «восприятие» — «анализ» соответственно. Фазы этих состояний β и γ, а также интерференционная фаза x (3), полученные решением уравнений (1)-(3), лежат в диапазонах

$$x = 111.3 \pm 8.8^\circ, \qquad \beta = 119.5 \pm 5.8^\circ, \qquad \gamma = 241.1 \pm 3.3^\circ. \tag{4}$$

Эта группировка характеризует устойчивость субъективных смысловых соотношений между контекстами принятия базисного решения в анализируемой выборке данных [17].

2. Матричная форма

Для n экспериментов компоненты $|0\rangle$ и $|1\rangle$ векторов (1) характеризуется матрицами комплекснозначных амплитуд размером $3 \times n$, строки которых соответствуют контекстам A, B и C:

$$\mathbf{Q_1} = \begin{bmatrix} a_{11}e^{i\alpha_1} & a_{12}e^{i\alpha_2} & \dots & a_{1n}e^{i\alpha_n} \\ b_{11}e^{i\beta_1} & b_{12}e^{i\beta_2} & \dots & b_{1n}e^{i\beta_n} \\ c_{11}e^{i\gamma_{11}} & c_{12}e^{i\gamma_{12}} & \dots & c_{1n}e^{i\gamma_{1n}} \end{bmatrix}, \qquad \mathbf{Q_0} = \begin{bmatrix} a_{01} & a_{02} & \dots & a_{0n} \\ b_{01} & b_{12} & \dots & b_{0n} \\ c_{01}e^{i\gamma_{01}} & c_{02}e^{i\gamma_{02}} & \dots & c_{0n}e^{i\gamma_{0n}} \end{bmatrix}, \tag{5}$$

где второй индекс у амплитуд a_k, b_k, c_k и фаз γ_k есть номер эксперимента. Согласно уравнениям (2) первая из этих матриц связана с матрицей исходных данных операцией поэлементного квадратного модуля:

$$\mathbf{P}_1 = \begin{bmatrix} \mathbf{p}_1^a \\ \mathbf{p}_1^b \\ \mathbf{p}_1^c \end{bmatrix} = \begin{bmatrix} p_1^a & p_2^a & \dots & p_n^a \\ p_1^b & p_2^b & \dots & p_n^b \\ p_1^c & p_2^c & \dots & p_n^c \end{bmatrix} = |\mathbf{Q}_1|^2, \qquad \mathbf{P}_0 = 1 - \mathbf{P}_1 = \begin{bmatrix} \mathbf{p}_0^a \\ \mathbf{p}_0^b \\ \mathbf{p}_0^c \end{bmatrix} = |\mathbf{Q}_0|^2. \tag{6}$$

Table 1. Results of 12 experiments «two-stage gamble» [7]

Таблица 1. Результаты 12 экспериментов «двухэтапная игра» [7]

p^a	0.69	0.75	0.69	0.60	0.83	0.80	0.68	0.64	0.53	0.73	0.59	0.30
p^b	0.57	0.69	0.59	0.47	0.70	0.37	0.32	0.47	0.38	0.49	0.71	0.24
p^c	0.38	0.73	0.35	0.47	0.62	0.43	0.38	0.38	0.24	0.60	0.70	0.17

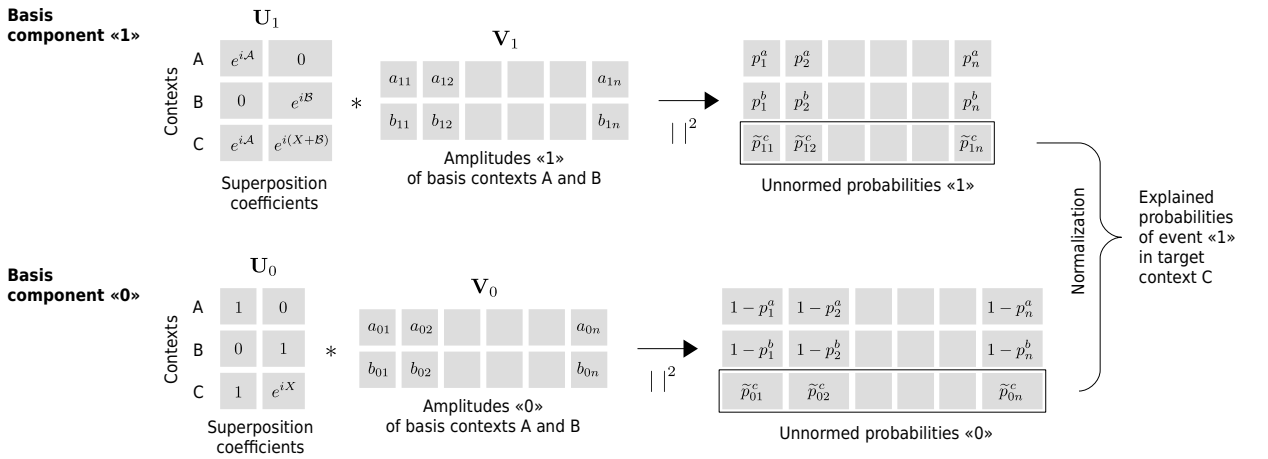


Fig. 2. Schematic of the matrix-qubit model

Рис. 2. Общая схема матрично-кубитной модели

2.1. Матричное разложение

Соотношение (3) выражает последние строки матриц (5) через суперпозицию пар их первых строк с коэффициентами e^{ix_k} и 1 соответственно. В матричной записи это соотношение принимает вид

$$\begin{aligned}
\mathbf{Q_1} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} a_{11}e^{i\alpha_1} & a_{12}e^{i\alpha_2} & \dots & a_{1n}e^{i\alpha_n} \\ b_{11}e^{i(\beta_1+x_1)} & b_{12}e^{i(\beta_2+x_2)} & \dots & b_{1n}e^{i(\beta_n+x_n)} \end{bmatrix}, \\
\mathbf{Q_0} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} a_{01} & a_{02} & \dots & a_{0n} \\ b_{01}e^{ix_1} & b_{02}e^{ix_2} & \dots & b_{0n}e^{ix_n} \end{bmatrix}.
\end{aligned} \tag{7}$$

Кучность значений x_k и β_k в рассматриваемой выборке (4) позволяет заменить их на общие значения X и $\mathcal{B}$. Обозначая произвольные $\alpha_k = \text{const}$ буквой $\mathcal{A}$, соответствующие фазовые множители могут быть перенесены из вторых матриц уравнений (7) в первые как показано на рисунке 2:

$$\begin{aligned}
\widetilde{\mathbf{Q}}_\mathbf{1} &= \begin{bmatrix} e^{i\mathcal{A}} & 0 \\ 0 & e^{i\mathcal{B}} \\ e^{i\mathcal{A}} & e^{i(X+\mathcal{B})} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ b_{11} & b_{12} & \dots & b_{1n} \end{bmatrix} = \mathbf{U}_1\mathbf{V}_1 \approx \mathbf{Q_1}, \\
\widetilde{\mathbf{Q}}_\mathbf{0} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & e^{iX} \end{bmatrix} \begin{bmatrix} a_{01} & a_{02} & \dots & a_{0n} \\ b_{01} & b_{02} & \dots & b_{0n} \end{bmatrix} = \mathbf{U}_0\mathbf{V}_0 \approx \mathbf{Q_0},
\end{aligned} \tag{8}$$

где элементы действительных матриц $\mathbf{V}_1$ и $\mathbf{V}_0$ определяются из данных таблицы 1 посредством первых двух строк (2). Таким образом параметрами матричной кубитной модели являются общие фазы $\mathcal{A}$, $\mathcal{B}$ и X, первая из которых фиксируется из тех же соображений, что и при моделировании отдельных экспериментов. В исходной кубитной модели, напротив, пара свободных фаз находится для каждого из n этих экспериментов отдельно. Таким образом для данных по двухэтапной игре матричная форма сокращает количество параметров модели в $n = 12$ раз.

2.2. Нахождение параметров

Из-за такого сокращения числа параметров, разложение (8) точно воспроизводит амплитуды (5) только в «базисных» контекстах A и B. Амплитуды компонент $|0\rangle$ и $|1\rangle$ контексте C, напротив, воспроизводятся лишь приближённо в силу нестрогости последних равенств в уравнениях (8).

Мерами соответствующих отклонения могут служить квадратичные нормы

$$L_1(X,\mathcal{A},\mathcal{B}) = \frac{\|\mathbf{p}_1^c - \widetilde{\mathbf{p}}_1^c\|}{\|\mathbf{p}_1^c\|}, \qquad \widetilde{\mathbf{p}}_1^c = \left|\left[e^{i\mathcal{A}} \quad e^{i(X+\mathcal{B})}\right]\mathbf{V}_1\right| \tag{9a}$$

$$L_0(X) = \frac{\|\mathbf{p}_0^c - \widetilde{\mathbf{p}}_0^c\|}{\|\mathbf{p}_0^c\|}, \qquad \widetilde{\mathbf{p}}_0^c = \left|\left[1 \quad e^{iX}\right]\mathbf{V}_0\right|, \tag{9b}$$

которые могут использовалась в качестве функций ошибки для нахождения фаз X и $\mathcal{B}$ методами многомерной оптимизации. При этом величина X может быть найдена напрямую из (9b), после чего $\mathcal{B}$ находится из (9a) с учётом заданного вручную $\mathcal{A}$ как отмечено выше.

Полученные таким образом строки $\widetilde{\mathbf{p}}_1^c$ и $\widetilde{\mathbf{p}}_0^c$ дают в сумме единицу лишь в среднем. Строгая нормировка (2) для каждого эксперимента достигается поэлементным делением каждой из этих строк на их сумму:

$$\mathbf{p}_1^c[\mathrm{mod}] = \frac{\widetilde{\mathbf{p}}_1^c}{\widetilde{\mathbf{p}}_0^c + \widetilde{\mathbf{p}}_1^c}, \qquad \mathbf{p}_0^c[\mathrm{mod}] = \frac{\widetilde{\mathbf{p}}_0^c}{\widetilde{\mathbf{p}}_0^c + \widetilde{\mathbf{p}}_1^c}. \tag{10}$$

Эти вектора являются моделью третьих строк матриц (6) на основе первых двух её строк. Общая точность модели характеризуется нормой

$$L = \frac{\|\mathbf{P}_1 - \mathbf{P}_1[\mathrm{mod}]\|}{\|\mathbf{P}_1\|}. \tag{11}$$

3. Испытания в описательном режиме

Представленная модель построена для данных в таблице 1 для трёх вариантов выбора пары базисных контекстов: AB, BC и CA. В первом случае общая для всех экспериментов фаза $\mathcal{A}$ контекста A задаётся равной 0, общая фаза $\mathcal{B}$ контекста B определяется в ходе оптимизации как описано в разделе 2.2, тогда как фазы γ контекста C (1) определяются аргументом комплекснозначных амплитуд в третьих строках матриц $\widetilde{\mathbf{Q}}_0$ и $\widetilde{\mathbf{Q}}_1$ (8). В частности, для первого варианта базисной пары (AB) получено значение $\mathcal{B} = 120.6°$, тогда как среднее значение и среднеквадратическое отклонение фаз γ есть $C = 240.5 \pm 3.3°$.

Для других вариантов базисных пар обозначения фаз сохраняются. В частности, при базисе BC фаза $\mathcal{B}$ контекста B фиксируется на значении $\mathcal{B} = 120°$ согласно схеме на графике 1B. Общая фаза C находится посредством оптимизации, тогда как диапазон фаз $\mathcal{A}$ целевого контекста A вычисляется из полученных матриц $\widetilde{\mathbf{Q}}_0$ и $\widetilde{\mathbf{Q}}_1$. Для каждого из вариантов базиса полученные значения фаз контекстов и интерференционных фаз X приведены в столбцах таблицы 2.

Точность полученных моделей приведена в последних строках таблицы. В предпоследней строке указана ошибка воспроизведения целевого контекста (9a), вычисленная после итоговой нормировки (10). В последней строке приведена относительная норма отклонения полной матрицы (11).

Table 2. Parameters and precision of descriptive modeling

Таблица 2. Параметры и точность описательного моделирования

Базисная пара Целевой контекст	AB C	BC A	CA B
X	111.9°	132.5°	118.1°
$\mathcal{A}$	0°	358.3 ± 2.9°	357.7°
$\mathcal{B}$	120.6°	120°	118.7 ± 5.6°
C	240.5 ± 3.3°	236.7°	240°
Ошибка L_1 (9a)	0.21	0.17	0.18
Общая ошибка (11)	0.10	0.12	0.10

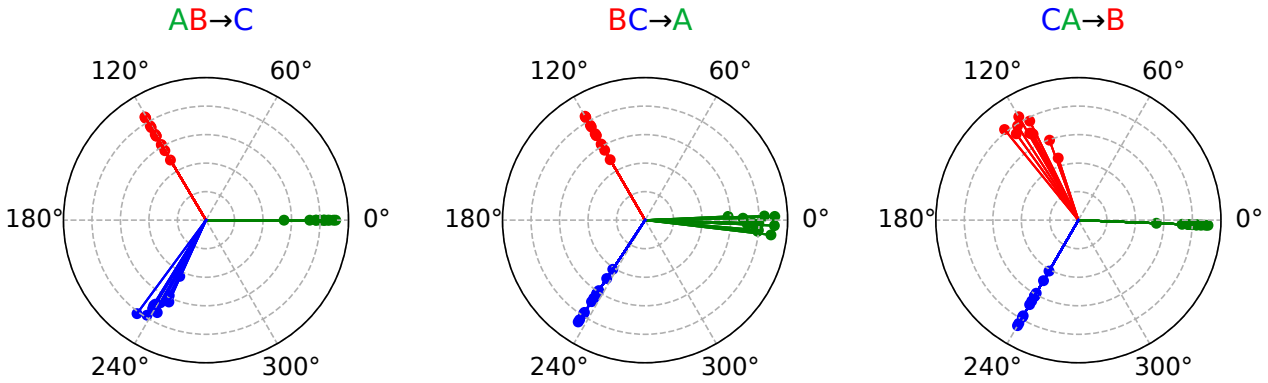


Fig. 3. Qubit states (1) in matrix model, projected onto equatorial plane as in Figure 1. Graphics correspond to three choices for the pair of basis contexts, indicated on top.

Рис. 3. Кубитные состояния (1) матричной модели в проекции на экваториальную плоскость сферы Блоха аналогично рисунку 1. Графики соответствуют трём вариантам базисной пары контекстов, указанной сверху.

Эти величины целесообразно сравнить с аналогичными ошибками моделирования тех же данных с помощью сингулярного (SVD) и неотрицательного (NMF) разложений [19], усечённых до размерности 2. Для обоих разложений функция (11) составила 0.255, что превышает ошибки в последней строке таблицы 2 более чем вдвое. Достигнутое преимущество тем более существенно, что представленная матричная модель имеет всего два свободных параметра (X и одна из фаз $\mathcal{A}$, $\mathcal{B}$, C). Для сравнения, у сингулярного разложения таких параметров 26, а у неотрицательного — 30.

Каждая из трёх построенных моделей определяет матрицы комплекснозначных амплитуд $\widetilde{\mathbf{Q}}_0$ и $\widetilde{\mathbf{Q}}_1$ (8), порождающие наблюдаемые вероятности согласно формулам (6). После нормировки (10) матрица $\widetilde{\mathbf{Q}}_0$ теряет информативность, тогда как $\widetilde{\mathbf{Q}}_1$ полностью характеризует построенную модель для каждого из целевых экспериментов. Как и в исходной кубитной модели (раздел 1), каждому из экспериментов в таблице 1 соответствует тройка векторов (1), соответствующих контекстам A, B и C. 12 таких троек для каждой из трёх моделей показаны на рисунке 3

Функциональное расположение контекстов в азимутальной плоскости сферы Блоха совпадает с представленным на графике 1B для всех вариантов базиса. В отличие от индивидуальных моделей, фаза не только первого, но и второго из базисных контекстов для всех экспериментов одинакова и определяется результатом оптимизации. Ненулевой разброс имеют только фазы целевого контекста (таблица 2).

4. Прогнозный режим

Представленная модель может быть использована для прогнозирования вероятности целевого решения в некоторых из рассматриваемых контекстов моделируемого набора на основе вероятностей того же решения в оставшихся контекстах. Метод такого прогноза и результаты его апробации на данных двухэтапной игры представлены в разделах 4.1 и 4.2.

4.1. Метод

Метод прогнозирования строится на основе описательной модели, показанной на рисунке 2. В левой части этой схемы отдельным экспериментам соответствуют столбцы матриц $\mathbf{V}_0$ и $\mathbf{V}_1$, тогда как матрицы $\mathbf{U}_0$ и $\mathbf{U}_1$ кодируют общую алгоритмику поведенческой системы, породившей моделируемые данные. Знание этих матриц $\mathbf{U}$, найденных из n «обучающих» экспериментов как описано в разделе 2, и делает возможным использование модели для прогнозирования части вероятностей рассматриваемого решения в новом эксперименте $n+1$. Уточнение «части» здесь означает, что про-

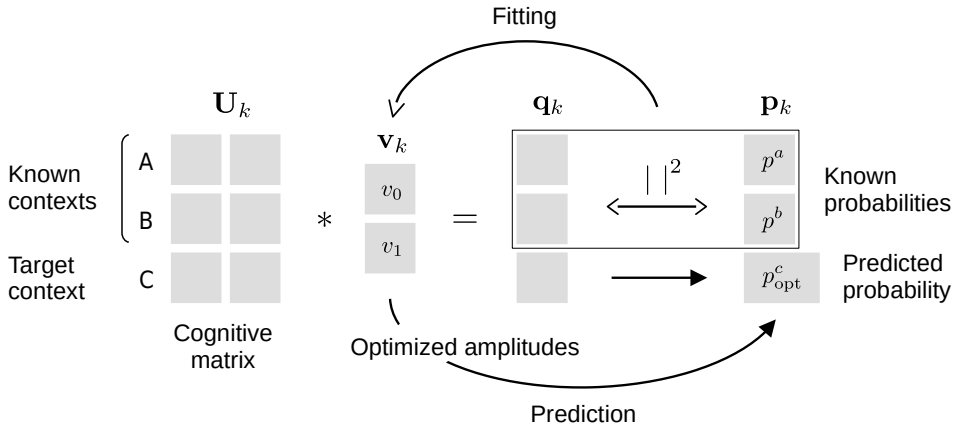


Fig. 4. Prognosis of the basis decision probability in the target context C based on the known contexts A, B and matrices of behavioral algorithmics **U** (12)–(13)

Рис. 4. Схема прогнозирования вероятности базисного решения в целевом контексте C на основе известных контекстов A, B и матриц поведенческой алгоритмики **U** (12)–(13)

гнозируются вероятности не во всех трёх контекстах A, B и C, а только в одном — целевом — из них; вероятности в остальных двух «известных» контекстах необходимо знать.

Для такого предсказания к матрицам $\mathbf{V}_1$ и $\mathbf{V}_0$ необходимо добавить столбцы $\mathbf{v}_1$ и $\mathbf{v}_0$, соответствующие новому эксперименту $n+1$. Согласно схеме 2 и уравнениям (8) перемножение этих столбцов с матрицами $\mathbf{U}_1$ и $\mathbf{U}_0$ добавит к амплитудным матрицам $\widetilde{\mathbf{Q}}_1$ и $\widetilde{\mathbf{Q}}_0$ новые столбцы $\mathbf{q}_1$ и $\mathbf{q}_0$. Квадраты модулей этих амплитуд порождают столбцы прогнозных вероятностей согласно выражениям (6):

$$\mathbf{p}_k^{\text{est}} = |\mathbf{U}_k \mathbf{v}_k|^2, \qquad k = 0, 1. \tag{12}$$

В известной паре контекстов эти вероятности должны максимально точно воспроизводить данные значения, что и является критерием для нахождения оптимальных амплитудных столбцов $\mathbf{v}_k$. При этом используются функции ошибок

$$L_k^{\text{fit}}(\mathbf{v}_k) = \frac{\|\mathbf{p}_k^{\text{est}}[\text{iz}] - \mathbf{p}_k[\text{iz}]\|}{\|\mathbf{p}_k[\text{iz}]\|}, \qquad k = 0, 1, \tag{13}$$

где аргумент [iz] выбирает из векторов строки, соответствующие известным контекстам.

Согласно выражению (12), найденные таким образом амплитуды $\mathbf{v}_k^{\text{opt}}$ определяют вероятности $\mathbf{p}_k^{\text{opt}}$ базисного решения во всех контекстах, включая целевой [neiz] как показано на схеме 4. Финальные прогнозные значения получаются из этих вероятностей с помощью нормировки, аналогичной (10):

$$p_k^{\text{prog}} = \frac{\mathbf{p}_k^{\text{opt}}[\text{neiz}]}{\mathbf{p}_0^{\text{opt}}[\text{neiz}] + \mathbf{p}_1^{\text{opt}}[\text{neiz}]}, \qquad k = 0, 1. \tag{14}$$

Итак, решённая прогнозная задача (12)—(14) состоит в предсказании вероятности базисного решения в одном из рассматриваемых контекстов на основе вероятностей того же решения в остальных контекстах. Для этого необходимо знать матрицы $\mathbf{U}_0$ и $\mathbf{U}_1$, полученные в результате описательного моделирования.

4.2. Испытания

Описанный метод испытан на данных, приведённых в таблице 1. Для каждого из 12 экспериментов прогноз выполнялся для всех трёх вариантов целевого контекста. Необходимые при этом

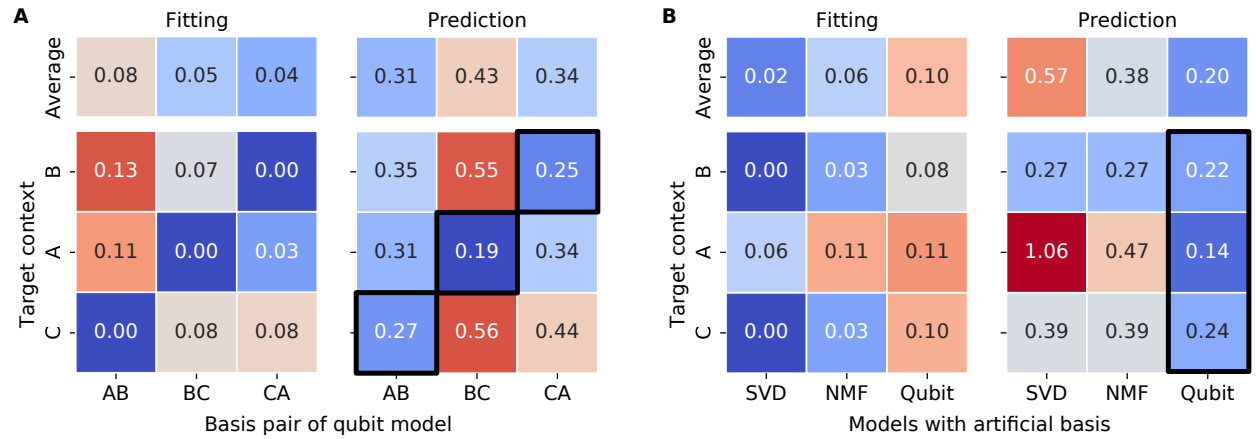


Fig. 5. A: testing of prognostic model for all combinations of the basis context pair and the target context. Fitting: relative standard error (13) of approximation of data in the known contexts. Prediction: prognostic error (15). B: the same values for the prognosis based on singiular value decomposition (SVD), nonnegative matrix factorization (NMF), and on the qubit model with artificial basis. Top row shows average of all columns. Black frame indicates best models with fixed and artificial basis.

Рис. 5. A: результаты прогнозного испытания модели для всех комбинаций базисной пары контекстов и целевого контекста. Аппроксимация: относительная среднеквадратическая ошибка (13) описания известных контекстов. Прогноз: ошибка прогнозирования (15). B: те же величины для прогноза на основе сингулярного (SVD) и неотрицательного (NMF) матричных разложений, а также на основе кубитной модели с искусственным базисом. Среднее по каждому из столбцов показано в верхнем ряду.

матрицы $\mathbf{U}_0$ и $\mathbf{U}_1$ определялись в ходе описательного моделирования 11 оставшихся «обучающих» экспериментов. Как отмечено в разделе 3, эти матрицы могут быть получены на основе любой из трёх пар базисных контекстов: AB, BC и CA (рисунок 3). В итоге, для каждой из 9 комбинаций «целевой контекст x — базисная пара yz» выражение (14) прогнозирует строку вероятностей $\mathbf{p}^x_{yz}[\mathrm{prog}]$ для всех 12 экспериментов таблицы 1. Относительная ошибка прогноза измерялась нормированным среднеквадратическим отклонением

$$\mathrm{STD}^{\mathrm{prog}}(x,yz) = \| \frac{\mathbf{p}^x_{yz}[\mathrm{prog}] - \mathbf{p}^x}{\mathbf{p}^x} \| / \sqrt{n}, \qquad x,y,z \in A,B,C. \tag{15}$$

Эти значения показаны на правом графике рисунка 5A. Приведённые в верхней строке средние (average) соответствуют относительной ошибке прогноза (prediction) целевой вероятности 36 %. Для каждого целевого контекста (target context), однако, существует кубитная модель, дающая существенно более высокую точность (0.27, 0.19, 0.25). Эти значения на диагонали матрицы (чёрная рамка) соответствует выбору базисной пары контекстов (basis pair), дополняющей целевой контекст до тройки ABC. В этих случаях настроенная в обучающем описательном режиме функция матриц $\mathbf{U}_0$ и $\mathbf{U}_1$ совпадает с требуемой для прогноза. При этом, в отличие от остальных комбинаций (x, yz), ошибки описания известных контекстов (fitting) (13) равны нулю как показано на левом графике.

5. Сравнение с методами на искусственном базисе

Аналогичный прогноз был выполнен на основе сингулярного и неотрицательного разложений. Эти разложения представляют матрицу данных $\mathbf{P}_1$ (6) в виде произведения трёх и двух матриц соответственно. В отличие от кубитной модели, эти методы работают не с амплитудами, а непосредственно с действительнозначными данными. Тем не менее, метод прогнозного моделирования (раздел 4.1) применим к этим описательным моделям.

Для неотрицательного разложения (NMF) первая и вторая матрицы соответствует матрицам **U** и **V** на схеме 2. Соответствующий новому эксперименту дополнительный столбец второй матрицы подбирается по условию наиболее точного воспроизведения данных в известных контекстах аналогично схеме на рисунке 4, однако без квадратного модуля (12). Для сингулярного разложения (SVD) процедура отличается тем, что в качестве второй базисной матрицы берётся произведение второй (сингулярной) и третьей матриц разложения.

В отличие от кубитной модели, базисные матрицы в разложениях NMF и SVD берутся не из данных напрямую, а конструируются искусственно. Соответственно, выбор пары базисных контекстов (таблица 2, горизонтальная ось на рисунке 5A) не требуется, а разложение данной матрицы моделируемых величин является единственным. При этом аналогичные **U** на схеме 2 первые матрицы разложений становятся более информативными и также конструируются для наиболее точного воспроизведения данных. При двух базисных компонентах, относительная норма ошибки описания данных в таблице 1 для обоих методов составила 0.255 (раздел 3). Аналогичная модификация кубитной модели представлена в Приложении.

Точность прогнозных моделей на основе SVD и NMF, а также кубитной модели с искусственным базисом представлена на рисунке 5B. Наихудший прогноз со средней относительной ошибкой 0.57 (15) получен с помощью модели на основе SVD. Прогноз на основе NMF даёт среднюю ошибку 0.38, близкую к средним ошибкам кубитных моделей на рисунке 5A, однако уступает кубитным моделям с наилучшим фиксированным базисом для каждого контекста.

Кубитная модель с искусственным базисом (qubit) позволяет добиться ещё большего преимущества, уменьшая ошибку для каждого из конекстов на 3–5 % (чёрная рамка). Ценой этого преимущества является утрата интерпретируемости матриц (рисунок 2), (16). Строки матриц **V** более не соответствуют исходным контекстам задачи, тогда как получаемые кубитные тройки не несут очевидных фазовых закономерностей, показанных на рисунке 3.

Заключение

Наибольшую ценность для разработки новых вычислительных систем представляет семантика и число параметров представленной модели. В отличие от весов связей в современных нейросетях, параметры полученного разложения (в частности, фазы элементов матриц **U** (8)) интерпретируются в процессно-смысловых категориях естественного мышления (экватор сферы на рисунке 1A); эти категории, в свою очередь, связаны с причинно-следственной логикой и субъективно-эмоциональными состояниями поведенческих систем [15, 20]. На этой основе представленная модель в автоматическом режиме выявила процессно-смысловые отношения между поведенческими контекстами, принципиально отличающиеся от корреляционных закономерностей синтаксического уровня. Подтверждение этого свойства в дальнейших испытаниях будет означать существенный прогресс в разработке природоподобных методов семантического анализа данных.

Число параметров представленной модели с фиксированным базисом не зависит от числа столбцов-постановок $n = 12$ и определяется лишь числом строк-контекстов $k = 3$ матрицы исходных данных. При двух базисных контекстах модель имеет всего $k - 1$ параметр: фаза одного из базисных контекстов и $k - 2$ значений фаз X для каждого из не-базисных контекстов. Для сравнения, усечённое до размерности 2 неотрицательное разложение матрицы размером $k \times n$ зависит от $2(k+n) = 30$ параметров; для сингулярного разложения это число на 4 единицы меньше. При этом точность как описания, так и прогноза может существенно превосходить эти классические аналоги как показано в разделах 3 и 4.2 на примере двухэтапной игры. В переложении на тензорную алгебру нейросетей это свойство представляет интерес для разработки более эффективных вычислительных архитектур.

Вариант модели с искусственным базисом (см. Приложение) позволяет ещё более повысить точность прогноза (рисунок 5B), однако утрачивает преимущество по числу параметров. Новым свой-

ством этой модификации является множественность решений соответствующей оптимизационной задачи. Совместно с отмеченными выше интерпретационными свойствами, получаемые таким образом матричные разложения могут соответствовать альтернативным смысловым моделям одних и тех же фактических данных, выбор между которыми обусловлен субъективизмом когнитивных системы. Таким образом нежелательная на первый взгляд многозначность может соответствовать субъективной многовариантности естественного мышления, не находящей выражения в других парадигмах анализа данных. Возможность такого использования является предметом дальнейших исследований.

References

[1] S. D. Larson, P. Gleeson, and A. E. X. Brown, "Connectome to behaviour: Modelling caenorhabditis elegans at cellular resolution", *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 373, no. 1758, p. 20 170 366, 2018. DOI: 10.1098/rstb.2017.0366.

[2] O. P. Kuznetsov, "Nonclassical paradigms in the artificial intelligence", *Teoriya i Sistemy Upravleniya*, no. 5, pp. 3–23, 1995, in Russian.

[3] A. Pavlov, "Fourier holography techniques for artificial intelligence", in *Advances in Information Optics and Photonics*, 2010, pp. 251–269. DOI: 10.1117/3.793309.ch13.

[4] D. Widdows, K. Kitto, and T. Cohen, "Quantum mathematics in artificial intelligence", *Journal of Artificial Intelligence Research*, vol. 72, pp. 1307–1341, 2021. DOI: 10.1613/jair.1.12702.

[5] A. Melnikov, M. Kordzanganeh, A. Alodjants, and R.-K. Lee, "Quantum machine learning: From physics to software engineering", *Advances in Physics: X*, vol. 8, no. 1, 2023. DOI: 10.1080/23746149.2023.2165452.

[6] A. Y. Khrennikov, *Ubiquitous Quantum Structure: From Psychology to Finance*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-05101-2.

[7] I. A. Surov, "Logic of sets and logic of waves in cognitive-behavioral modeling", *Information and Mathematical Technologies in Science and Management*, vol. 32, no. 4, pp. 51–66, 2023. DOI: 10.25729/ESI.2023.32.4.005.

[8] A. K. Guts, *Fundamentals of Quantum Cybernetics*. KAN, 2008, in Russian.

[9] A. Hirose, Ed., *Complex-Valued Neural Networks. Theories and Applications*. World Scientific, 2003.

[10] J. J. Denimal and S. Camiz, "Complex principal component analysis: Theory and geometrical aspects", *Journal of Classification*, vol. 39, no. 2, pp. 376–408, 2022. DOI: 10.1007/s00357-022-09412-0.

[11] S. Kozhisseri and I. A. Surov, "Quantum-probabilistic SVD: Complex-valued factorization of matrix data", *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, vol. 22, no. 3, pp. 567–573, 2022. DOI: 10.17586/2226-1494-2022-22-3-567-573.

[12] B. Wang, Q. Li, M. Melucci, and D. Song, "Semantic Hilbert space for text representation learning", in *Proceedings of the World Wide Web Conference*, ACM Press, 2019, pp. 3293–3299. DOI: 10.1145/3308558.3313516.

[13] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization", *Nature*, vol. 401, no. 6755, pp. 788–791, 1999. DOI: 10.1038/44565.

[14] I. A. Surov, "Natural code of subjective experience", *Biosemiotics*, vol. 15, no. 1, pp. 109–139, 2022. DOI: 10.1007/s12304-022-09487-7.

[15] I. A. Surov, "What is the difference? Pragmatic formalization of meaning", *Artificial intelligence and decision making*, no. 1, pp. 78–89, 2023, in Russian. DOI: 10.14357/20718594230108.

[16] A. Tversky and E. Shafir, “The disjunction effect in choice under uncertainty”, *Psychological Science*, vol. 3, no. 5, pp. 305–309, 1992, ISSN: 14679280. DOI: 10.1111/j.1467-9280.1992.tb00678.x.

[17] I. A. Surov, “Quantum cognitive triad: Semantic geometry of context representation”, *Foundations of Science*, vol. 26, no. 4, pp. 947–975, 2021. DOI: 10.1007/s10699-020-09712-x.

[18] I. A. Surov, “Probabilistic prediction of “irrational” decisions from semantic composition of contexts”, *Journal of Applied Informatics*, vol. 19, no. 1, pp. 125–143, 2024. DOI: 10.37791/2687-0649-2024-19-1-125-143.

[19] N. Gillis, *Nonnegative Matrix Factorization*. Society for Industrial and Applied Mathematics, 2020. DOI: 10.1137/1.9781611976410.

[20] I. A. Surov, “Life cycle: Semantic matrix of process modeling”, *Ontology of designing*, vol. 12, no. 4, pp. 430–453, 2022, in Russian. DOI: 10.18287/2223-9537-2022-12-4-430-453x.

[21] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, “Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound- constrained optimization”, *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, 1997. DOI: 10.1145/279232.279236.

Приложение. Кубитная модель на искусственном базисе

Для моделирования одной из строк матрицы (6) на основе пары других её строк представленная матричная форма избыточна в силу тривиальности первых двух строк матриц $\mathbf{U}_0$, $\mathbf{U}_1$ (8) и совпадения этих строк в матрицах $\widetilde{\mathbf{Q}}_\mathbf{k}$ и $\mathbf{Q}_\mathbf{k}$. Матричная форма, однако, позволяет искать разложение амплитудных матриц (5) более общим образом, когда матрицы $\mathbf{V}_0$ и $\mathbf{V}_1$ не берутся из данных напрямую, а подбираются для получения более точной модели. В этом случае разложения (8) принимают вид

$$\begin{aligned}
\widetilde{\mathbf{Q}}_\mathbf{1} &= \begin{bmatrix} u_{11}e^{i\mathcal{A}} & e^{i(X_a+\mathcal{B})}u_{12} \\ u_{21}e^{i\mathcal{A}} & e^{i(X_b+\mathcal{B})}u_{22} \\ u_{31}e^{i\mathcal{A}} & e^{i(X_c+\mathcal{B})}u_{32} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \end{bmatrix} = \mathbf{U}_1\mathbf{V}_1, \\
\widetilde{\mathbf{Q}}_\mathbf{0} &= \begin{bmatrix} u_{11} & e^{iX_a}u_{12} \\ u_{21} & e^{iX_b}u_{22} \\ u_{31} & e^{iX_c}u_{32} \end{bmatrix} \begin{bmatrix} \sqrt{1-v_{11}^2} & \sqrt{1-v_{12}^2} & \dots & \sqrt{1-v_{1n}^2} \\ \sqrt{1-v_{21}^2} & \sqrt{1-v_{22}^2} & \dots & \sqrt{1-v_{2n}^2} \end{bmatrix} = \mathbf{U}_0\mathbf{V}_0,
\end{aligned} \tag{16}$$

где матрица $\mathbf{U}_0$ имеет 9 параметров, $\mathbf{U}_1$ добавляет к ним фиксированную фазу $\mathcal{A}$ и произвольную $\mathcal{B}$, а $2n$ элементов $\mathbf{V}_1$ и $\mathbf{V}_0$ теперь также являются параметрами модели. Соответственно функции ошибки (9) и нормировки (10) принимают вид

$$L_k(\mathbf{U}_k, \mathbf{V}_k) = \frac{\|\mathbf{P}_k - |\mathbf{U}_k\mathbf{V}_k|^2\|}{\|\mathbf{P}_k\|}, \qquad \mathbf{P}_k[\mathrm{mod}] = \frac{\left|\widetilde{\mathbf{Q}}_\mathbf{k}^{\mathrm{opt}}\right|^2}{\left|\widetilde{\mathbf{Q}}_\mathbf{0}^{\mathrm{opt}}\right|^2 + \left|\widetilde{\mathbf{Q}}_\mathbf{1}^{\mathrm{opt}}\right|^2}, \qquad k \in \{0, 1\}, \tag{17}$$

где матричное деление во втором уравнении выполняется поэлементно, а $\widetilde{\mathbf{Q}}_\mathbf{k}^{\mathrm{opt}}$ есть матрицы (16) с оптимальными параметрами.

Испытания показали, что результат минимизации функции (17) методом L-BFGS-B [21] чувствителен к начальным значениям параметров модели. Соответственно, кубитная модель с искусственным базисом (в отличие от разложений SVD и NMF, а также в отличие от моделей с фиксированным базисом (3)) не является единственной. Модель, результат которой представлен в разделе 5, выбрана из 100 различных решений по критерию наименьшей ошибки прогноза (15).

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 31, NO. 3, 2024
journal homepage: www.mais-journal.ru

THEORY OF COMPUTING

Discovering Hierarchical Process Models: An Approach Based on Events Partitioning

A. K. Begicheva[1], I. A. Lomazova[1], R. A. Nesterov[1]

DOI: 10.18255/1818-1015-2024-3-294-315

[1]National Research University Higher School of Economics, Moscow, Russia

MSC2020: 68Q85
Research article
Full text in English

Received June 25, 2024
Revised July 18, 2024
Accepted July 24, 2024

Process mining is a field of computer science that deals with the discovery and analysis of process models based on automatically generated event logs. Currently, many companies are using this technology to optimize and improve their business processes. However, a discovered process model may be too detailed, sophisticated, and difficult for experts to understand. In this paper, we consider a problem of discovering the hierarchical business process model from a low-level event log, i. e., the problem of the automatic synthesis of more readable and understandable process models based on the data stored in the event logs of information systems.

The discovery of better-structured and more readable process models is extensively studied in the framework of process mining research from different perspectives. In this paper, we present an algorithm for discovering hierarchical process models represented as two-level workflow Petri nets. The algorithm is based on predefined event partitioning so that this partitioning defines a sub-process corresponding to a high-level transition at the top level of a two-level net. In contrast to existing solutions, our algorithm does not impose restrictions on the process control flow and allows for concurrency and iterations.

Keywords: process mining; Petri nets; workflow nets; process discovery; hierarchical process model; event log

INFORMATION ABOUT THE AUTHORS

Begicheva, Antonina K. (corresponding author)	ORCID iD: 0000-0001-6657-1760. E-mail: abegicheva@hse.ru Lecturer, M. Sc.
Lomazova, Irina A.	ORCID iD: 0000-0002-9420-3751. E-mail: ilomazova@hse.ru Professor, Dr. Sc.
Nesterov, Roman A.	ORCID iD: 0000-0002-4162-9070. E-mail: rnesterov@hse.ru Associate professor, PhD

Funding: Basic Research Program at HSE University.

For citation: A. K. Begicheva, I. A. Lomazova, and R. A. Nesterov, “Discovering hierarchical process models: an approach based on events partitioning”, *Modeling and Analysis of Information Systems*, vol. 31, no. 3, pp. 294–315, 2024. DOI: 10.18255/1818-1015-2024-3-294-315.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 31, № 3, 2024
сайт журнала: www.mais-journal.ru

THEORY OF COMPUTING

Синтез иерархических моделей процессов: подход на основе разбиения событий на множества

А. К. Бегичева[1], И. А. Ломазова[1], Р. А. Нестеров[1]

DOI: 10.18255/1818-1015-2024-3-294-315

[1]Национальный исследовательский университет «Высшая школа экономики», Москва, Россия

УДК 004.942
Научная статья
Полный текст на английском языке

Получена 25 июня 2024 г.
После доработки 18 июля 2024 г.
Принята к публикации 24 июля 2024 г.

Process mining — это область компьютерных наук, которая занимается синтезом и анализом моделей процессов на основе автоматически генерируемых журналов событий. В настоящее время многие организации используют эту технологию для оптимизации и совершенствования бизнес-процессов. Однако синтезированная модель процесса может быть слишком подробной, сложной и трудной для понимания экспертами. В работе мы рассматриваем задачу синтеза иерархической модели бизнес-процесса из низкоуровневого журнала событий, то есть, задачу автоматического синтеза более удобочитаемых и понятных моделей процессов на основе данных, хранящихся в журналах событий информационных систем.

Построение более структурированных и удобочитаемых моделей процессов широко изучается в рамках исследований в области process mining с разных точек зрения. В этой статье мы представляем алгоритм синтеза иерархических моделей процессов, представленных в виде двухуровневых сетей потоков работ. Алгоритм основан на предопределенном разбиении событий на множества, которые определяют подпроцессы, соответствующие высокоуровневым переходам на верхнем уровне двухуровневой сети потоков работ. В отличие от существующих решений, представленный алгоритм не накладывает ограничений на поток управления процессом, а также допускает параллелизм и итерации.

Ключевые слова: синтез моделей процессов; сети Петри; сети потоков работ; иерархические модели процессов; журнал событий

ИНФОРМАЦИЯ ОБ АВТОРАХ

Бегичева, Антонина Константиновна (автор для корреспонденции)	ORCID iD: 0000-0001-6657-1760. E-mail: abegicheva@hse.ru Преподаватель, магистр
Ломазова, Ирина Александровна	ORCID iD: 0000-0002-9420-3751. E-mail: ilomazova@hse.ru Профессор, доктор физико-математических наук
Нестеров, Роман Александрович	ORCID iD: 0000-0002-4162-9070. E-mail: rnesterov@hse.ru Доцент, канд. физ.-мат. наук

Финансирование: Программа фундаментальных исследований Национального исследовательского университета «Высшая школа экономики».

Для цитирования: A. K. Begicheva, I. A. Lomazova, and R. A. Nesterov, “Discovering hierarchical process models: an approach based on events partitioning”, *Modeling and Analysis of Information Systems*, vol. 31, no. 3, pp. 294–315, 2024. DOI: 10.18255/1818-1015-2024-3-294-315.

Introduction

Over the past decade, companies whose processes are supported by various information systems have become convinced of the need to store as much potentially useful information about the process executions within a system as possible. This was facilitated by qualitative improvement in the areas related to the extraction of valuable information from the recorded data, which helps to adjust the operation of companies over time and thus save and increase their resources. Process mining is a field of computer science that provides a palette of tools to extract the logic of the system behavior as well as to model and optimize the processes that occur in a system. In particular, process mining methods allow one to find inconsistencies between the planned and actual behavior of a system and to track the occurrence of the inefficient or incorrect behavior.

Despite the fact that increasing attention is being paid to preserving the optimal amount of the necessary information about processes, the actual data on process executions is not always available in a convenient format and with the necessary degree of detail, since system logs are generated for a lot of different purposes.

Process discovery aims at extracting processes from event logs and constructing models of these processes. Most of the available process discovery methods produce a model with the same level of detail provided by the initial event log [1].

Therefore, a promising area of research is the problem of discovering a more readable process model from a detailed event log, while preserving the important information about the process execution for experts. Readability of process models can be achieved in various ways. The most commonly used methods are filtering rare behavior from the original event log, skipping "minor" events (the significance of an event is assessed according to the chosen methodology); and abstraction, when some events are considered indistinguishable from others. We will discuss existing methods in more detail in Section 1. In our study, we consider the latter approach, when more readable models are the result of model abstraction — they are more compact and have the optimal level of detail for the work of experts in comparison to the level of model detail that could be obtained by direct discovery methods. To preserve the important data, we are dealing not only with abstract (high-level) models, but also with hierarchical models storing the low-level information in the form of sub-processes.

In this paper, we propose an algorithm for discovering hierarchical process models from event logs. Processes are represented using workflow nets [2], a special subclass of Petri nets used for modeling the control flow of business processes. This study extends our previously achieved results [3] where we proposed an approach to discovering abstract models for processes without cycles. Here, we provide a more general solution by overcoming the prohibition of cyclic behavior.

Hierarchical models allow us to have a high-level view of the model by "folding" the behavior of an individual sub-process into a high-level transition with the ability to unfold it back. Thus, at the top level, there is a high-level model in which every individual transition corresponds to a sub-process built from low-level events. The history of detailed behavior of the process is recorded in a low-level event log. Regarding the number of levels in the hierarchy, we will only use two levels — high and low, but the algorithm can naturally be extended to any number of levels.

The paper is structured as follows. Section 1 presents the review of related research. Section 2 gives theoretical preliminaries and the definitions used in the text. In Section 3, we discuss the basics of the hierarchical process discovery algorithm. Section 4 presents the main discovery algorithm and the proof of its correctness in the light of the perfect fitness preservation. Section 5 reports the outcomes from the experimental evaluation. In Section 6, we conclude the paper and discuss the possible future work directions.

1. Related work

Research connected with our paper can be classified into approaches to abstracting event logs and process models and approaches to constructing hierarchical process models from event logs.

One of the recent surveys [4] gives a comprehensive review of approaches and methods that can be applied for low-level event abstraction. The authors divide the methods according to: the learning strategy (supervised or unsupervised), the structure of the process models (strictly sequential or with interleaving), the low-level events grouping approach (deterministic or probabilistic) and the nature of the processed data (discrete or continuous data).

For example, the method presented in [5] is a supervised method that aligns the model complexity with the needs of different stakeholders. Another example of a supervised approach to event abstraction was presented in [6]. This method takes a low-level event log and transforms it to an event log at the desired level of abstraction, using the following behavioral patterns: sequence, choice, parallel, interleaving and repetition of events. This technique allows one to obtain a reliable mapping from low-level events to activity patterns automatically and construct a high-level event log using these patterns. Detecting high-level events based on the patterns of behavior in an event log does not make it possible to refine the accuracy of abstraction, based on the general knowledge of the system, or provide it only partially. Patterns provide the ability to change the scale but not to participate in the selection of correct high-level events. This could only be useful for a superficial analysis. However, there is a risk of combining unrelated low-level events into a single high-level event only because they are executed sequentially, but not because they belong to the same logical component of a system.

Another supervised event abstraction method was discussed in [7]. The nature of this method is as follows. The authors annotate a low-level event with the correct high-level event using the domain knowledge from the actual process model by the special attribute in the event log. In addition, this paper assumes that multiple high-level events are executed in parallel. This allows us to interpret a sequence of identical values as a single instance of a high-level event.

Unsupervised techniques do not require additional information beyond the input log. For example, in [8], the authors specify a fully unsupervised framework for partially ordered event data that detects abstraction classes using event data based on its observed execution context. In [9], the authors offer a framework for evaluating unsupervised abstraction techniques and evaluate the state-of-the-art methods using 400 event logs. One of the conclusions drawn from these evaluations is that there is typically a trade-off between high precision and high comprehensibility in the resulting model. The less abstract the model is the higher its calculated precision will be.

An example of the multi-perspective approach that combines features of the unsupervised and supervised methodologies is provided in [10]. After automatic identification of event groups, this method allows users to select the groups that are relevant and can be used for low-level log abstraction.

A general approach to the representation of multi-level event logs and the corresponding multi-level hierarchical models was studied in [11]. The authors highlighted the fact that this approach can combine multiple modeling notation for representing different levels in multi-level process models.

There are many ways of abstracting process models by reducing their size in order to make them more convenient to work with. Each method may be useful depending on a group of interrelated factors: the abstraction purposes, the presence of certain patterns and constructs, and the specifics of modeling notation. Reducing the size of the model by abstraction can be done as the “convolution” of groups of elements, or implemented by throwing some parts of the model away (insignificant in a particular case) [12]. The importance of the low-level event log abstraction is emphasized, among others, in [13].

Researchers determine which level of abstraction is appropriate for a particular case in different ways, but the main criterion is that the model should be readable and understandable. In [14], the abstraction of a process model occurs through “simplification” automatically: the user determines only the desired degree of detail, but not the actual correctness of identifying high-level events. Conversely, the paper [5] stressed the importance of the abstraction level dependence on the domain expert knowledge.

Petri nets [15] can also be extended by adding the hierarchy as, e.g., in Colored Petri nets (CPN) [16]. Hierarchical events allow one to construct more compact, readable and understandable process models. The hierarchy of CPN models can be used as an abstraction, in the case of two levels: a high-level *abstract* model and a low-level *refined* model. In our paper, the high-level model is a model with abstract transitions. An abstract transition refers to a Petri net sub-process which refines the activity represented by this high-level transition. A complete low-level, also referred to as *classical*, process model can be obtained from a high-level model by substituting sub-processes for high-level transitions. By the classical process model, we mean a model that is not loaded with information about the hierarchy, which has the same level of detail as the original event log.

Synthesis of a classical process model is a standard process discovery problem that has been extensively studied in the literature. A wide range of process discovery algorithms supports the automated classical process model synthesis [1].

Inductive Miner [17] is one of the most widely used process discovery algorithms that produces well-structured process models, built recursively from building blocks for standard behavioral patterns. They can be potentially used for constructing high-level process models. However, this technique does not take the actual correspondence between low-level events and sub-processes. In [18], the authors also used the recognition of behavioral patterns in a process by a structural partitioning algorithm and then defined a specific workflow schema for each pattern.

In [19], a two-phase approach to mining hierarchical process models was presented. Process models were considered as interactive and context-dependent maps based on common execution patterns. In the first phase, an event log is abstracted to the desired level by detecting relevant execution patterns. An example of such a pattern is the maximal repeat that captures typical sequences of activities in the log. Every pattern is then estimated by its frequency, significance, or some other metric needed for accurate abstraction. In the second phase, the *Fuzzy Miner* discovery algorithm [14], adapted to process map discovery, is applied to the transformed log.

FlexHMiner [20] is a general algorithm based on process trees implemented in ProM software. The authors stress the flexibility of this approach: to identify the hierarchy of events, the method supports both supervised methods and methods using the general knowledge of a process. The limitations of this method include the fact that each of the sub-processes can be executed only once, which means that the method is not suitable for processes with cycles.

A large volume of literature is devoted to the problem of discovering structured models from event logs. Researchers offer different techniques to improve the structure of discovered models, e. g., in [21], and to produce already well-structured process models [22, 23]. Different ways of detecting sub-processes in event logs, using low-level transition systems, were discussed in [24–26]. However, these works did not consider mining hierarchical process models from event logs.

One way to use process discovery techniques for abstract model synthesis is log pre-processing. For example in [27] the authors divide the initial log into sub-processes using activity instances information. The limitation of the proposed method is in the activity instance partitioning: as the authors only consider cases where a subprocess always begins and ends with fixed events.

In [3], the authors presented an algorithm for the discovery of a high-level process model from the event log for acyclic processes. This method takes the initial data on abstraction in the form of a set of detailed events grouped into high-level ones, which means that any method of identifying abstract events can potentially be used, including those based on expert knowledge. After pre-processing, this algorithm allows the use of any existing process discovery approach that is suitable for the synthesis of a classical process model. The possibility of using existing approaches as components makes the proposed algorithm flexible.

This paper extends the conditions of applicability of the algorithm from [3] since it works only for acyclic models. For the algorithm to find and process potential cycles in the event log, we will reuse the method

for detecting the repetitive behavior in a event log proposed and tested in [28, 29], which partially covers the general solution of the cycle detection problem.

2. Preliminaries

By $\mathbb{N}$ we denote the set of non-negative integers.

Let X be a set. A *multiset* m over the set X is a mapping: $m : X \to \mathbb{N}$, i. e., a multiset may contain several copies of the same element. For an element $x \in X$, we write $x \in m$, if $m(x) > 0$. For two multisets m, m' over X we write $m \subseteq m'$ iff $\forall x \in X : m(x) \leq m'(x)$ (the inclusion relation). The sum, the union and the subtraction of two multisets m and m' are defined as usual: $\forall x \in X : (m + m')(x) = m(x) + m'(x), (m \cup m')(x) = max(m(x), m'(x)), (m - m')(x) = m(x) - m'(x)$, if $m(x) - m'(x) \geq 0$, otherwise $(m - m')(x) = 0$. By $\mathcal{M}(X)$ we denote the set of all multisets over X.

For a set X, by X^* with elements of the form $\langle x_1, \ldots, x_k \rangle$ we denote the set of all finite sequences (words) over X, $\langle\rangle$ denotes the empty word, i. e., the word of zero length. The concatenation of two words w_1 and w_2 is denoted by $w_1 \cdot w_2$.

Let $Q \subseteq X$ be a subset of X. The projection $\restriction_Q$: $X^* \to Q^*$ is defined recursively as follows: $\langle\rangle \restriction_Q = \langle\rangle$, and for $\sigma \in X^*$ and $x \in X$:

$$(\sigma \cdot \langle x \rangle) \restriction_Q = \begin{cases} \sigma \restriction_Q \text{ if } x \notin Q \\ \sigma \restriction_Q \cdot \langle x \rangle \text{ if } x \in Q \end{cases}$$

We say that $X = X_1 \cup X_2 \cup \cdots \cup X_n$ is a partition of the set X if for all $1 \leq i, j \leq n$ such that $i \neq j$ we have $X_i \cap X_j = \emptyset$.

2.1. Petri nets

Let P and T be two finite disjoint sets of places and transitions, respectively, and $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ be an arc-weight function. Let also A be a finite set of *event names* (or *activities*) representing observable actions or events, τ — a special label for *silent* or invisible action, $\lambda : T \to A \cup \{\tau\}$ is a transition labeling function. Then $N = (P, T, F, \lambda)$ is a *labeled Petri net*.

Graphically, a Petri net is designated as a bipartite graph, where places are represented by circles, transitions by boxes, and the flow relation F by directed arcs.

A *marking* in a Petri net $N = (P, T, F, \lambda)$ is a function $m : P \to \mathbb{N}$ mapping each place to some number of tokens (possibly zero). Hence, a marking in a Petri net may be considered as a multiset over its set of places. Tokens are graphically designated by black circles. A current marking m is represented by putting $m(p)$ tokens into each place $p \in P$. A *marked Petri net* (N, m_0) is a Petri net N together with its initial marking m_0.

For transition $t \in T$, its *preset* (denoted ${}^\bullet t$) and its *postset* (denoted $t^\bullet$) are defined as sets of its input and output places respectively, i. e., ${}^\bullet t = \{p \mid F(p, t) \neq 0\}$ and $t^\bullet = \{p \mid F(t, p) \neq 0\}$.

A transition $t \in T$ is *enabled* in a marking m, if for all $p \in {}^\bullet t$, $m(p) \geq F(p, t)$. An enabled transition t may *fire* yielding a new marking m', such that $m'(p) = m(p) - F(p, t) + F(t, p)$ for each $p \in P$ (denoted $m \xrightarrow{\lambda(t)} m'$, or just $m \to m'$). A marking m' is reachable from a marking m, if there exists a sequence of firings $m = m_0 \to m_1 \to \ldots m_k = m'$. By $\mathcal{R}(N, m)$ we denote the set of all markings reachable from marking m in a net N. A transition $t \in T$ is called *dead* for a marked net (N, m_0), if for each reachable marking $m \in \mathcal{R}(N, m_0)$, t is not enabled in m.

Let (N, m_0) be a marked Petri net with transitions labeled with activities from $A \cup \{\tau\}$, and let $m_0 \xrightarrow{a_1} m_1 \xrightarrow{a_2} \ldots$ be a finite or infinite sequence of firings in N, which starts from the initial marking m_0 and cannot be extended. Then a sequence of observable activities ρ, such that $\rho = \langle a_1, a_2, \ldots \rangle \restriction_A$, is called a *run*. For a finite run ρ, which corresponds to a sequence of firings $m_0 \xrightarrow{a_1} \ldots \xrightarrow{a_k} m_k$, we call m_0 and m_k its initial and final markings respectively.

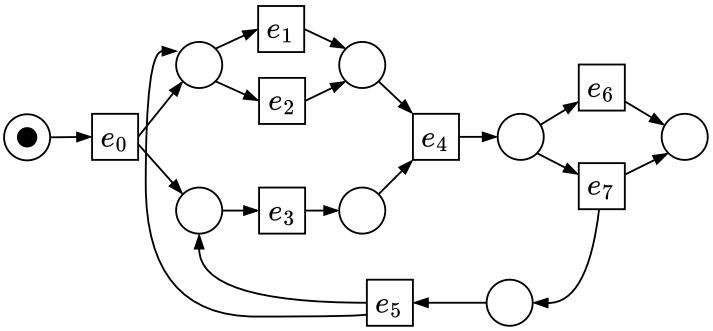


Fig. 1. A workflow net for handling compensation requests

In our study, we consider *workflow nets* — a special subclass of Petri nets [2] for workflow modeling. A *workflow net* is a (labeled) Petri net with two special places: i and f. These places mark the beginning and the ending of a workflow process.

A (labeled) marked Petri net $N = (P, T, F, \lambda, m_0)$ is called a workflow net (WF-net) if the following conditions hold:

1. There is one source place $i \in P$ and one sink place $f \in P$, such that ${}^\bullet i = f^\bullet = \emptyset$.
2. Every node from $P \cup T$ is on a path from i to f.
3. The initial marking m_0 in N contains the only token in its source place.

Given a WF-net, by $[in]$ we denote its initial marking with the only token in place i, and by $[fin]$ — its *final* marking with the only token in place f.

The example of a workflow net that simulates a simple process of handling ticket refund requests, is shown in Fig. 1 [30].

Soundness [2] is the main correctness property for workflow nets. A WF-net $N = (P, T, F, \lambda, [in])$ is called *sound*, if

1. For any marking $m \in R(N, [in])$, $[fin] \in \mathcal{R}(N, m)$;
2. If for some $m \in R(N, [in])$, $[fin] \subseteq m$, then $m = [fin]$;
3. There are no dead transitions in N.

2.2. Event logs

Most information systems record the history of their process execution into event logs. An *event record* usually contains case ID, an activity name, a time step, and some information about resources, data, etc. In the light of our research, we use case IDs for splitting an event log into traces, timestamps — for ordering events within each trace, and abstract from all event attributes except event names (activities).

Let A be a finite set of activities. A *trace* σ is a finite sequence of activities from A, i. e., $\sigma \in A^*$. By $\#a(\sigma)$ we denote the number of occurrences of activity a in trace σ.

An *event log* L is a finite multi-set of traces, i. e., $L \in \mathcal{M}(A^*)$. Let $X \subseteq A$. We extend projection $\restriction_X$ to event logs, i. e., for an event log $L \in M(A^*)$, its projection is the event log $L\restriction_X$, defined as the multiset of projections of all traces in L. In other words, $L\restriction_X(\sigma\restriction_X) = L(\sigma)$ for all $\sigma \in L$.

An important question is whether the event log matches the behavior of the process model and vice versa. There are several metrics to measure conformance between a WF-net and an event log. Specifically, *fitness* defines to what extend the log can be replayed by the model.

Let N be a WF-net with transition labels from A, an initial marking $[in]$, and a final marking $[fin]$. Let σ be a trace over A. We say that trace $\sigma = \langle a_1, \ldots, a_k \rangle$ *perfectly fits* N, if σ is a run in N with initial marking $[in]$ and final marking $[fin]$. An event log L perfectly fits N, if every trace from L perfectly fits N.

3. Discovering hierarchical WF-nets

3.1. Hierarchical WF-nets

Here, we define *hierarchical workflow* (HWF) nets with two levels of representing the process behavior. Transitions in a high-level WF-net are labeled by activities from $\tilde{A}$, while transitions in a set of low-level WF-nets are labeled by the corresponding low-level activities from A.

An HWF-net is a tuple $\mathcal{N} = (\tilde{N}, N_1, N_2, \ldots, N_k, \ell)$, where:

1. $\tilde{N} = (\tilde{P}, \tilde{T}, \tilde{F}, \tilde{\lambda}, [\tilde{in}])$ is a WF-net, called a *high-level WF-net*, where $\tilde{\lambda}\colon \tilde{T} \to \tilde{A}$ is a transition labeling function;
2. $N_i = (P_i, T_i, F_i, \lambda_i, [in]_i)$ is a WF-net, called a *low-level WF-net* for $i = 1, 2, \ldots, k$ with a transition labeling function $\lambda_i\colon T_i \to A_i$, where $A_i \subseteq A$ is a subset of low-level activities for N_i, such that $A_1, A_2, \ldots, A_k$ — is a partitioning of A;
3. $\ell\colon \tilde{A} \to \{N_1, N_2, \ldots, N_k\}$ is a partial injective function mapping certain activities in $\tilde{A}$ to low-level WF-nets.

We refer to $\tilde{N}, N_1, N_2, \ldots, N_k$ as the components of $\mathcal{N}$. A marking for $\mathcal{N}$ is defined by the markings of its components. A *marking* in $\mathcal{N}$ is a set $\mathcal{M} = \{M, m_1, m_2, \ldots, m_k\}$, where M is a marking in $\tilde{N}$ and $m_1, m_2, \ldots, m_k$ are markings in $N_1, N_2, \ldots, N_k$ respectively.

The initial marking $\mathcal{M}_0$ for HWF-net $\tilde{N}$ contains exactly one token in the source place of $\tilde{N}$.

Let $dom(\ell)$ denote the domain of ℓ. We call the activities in $dom(\ell)$ — *high-level activities*. We also call low-level WF-nets *sub-processes*. Accordingly, every transition in a high-level WF-net $\mathcal{N}$ is assigned the corresponding low-level WF-net modeling the behavior of a sub-process.

Transition $t \in \tilde{T} \cup T_1 \cup \ldots T_k$ is enabled if it is enabled in its component by the ordinary firing rule, described in the previous section, for Petri nets.

There are the three following alternatives of transition firing in HWF-nets:

1. Let $t \in (\tilde{T} \setminus dom(\ell)) \cup T_1 \cup \ldots T_k$ be a transition enabled in $\mathcal{M}$. Then, the firing step $\mathcal{M} \xrightarrow{\lambda(t)} \mathcal{M}'$ is completed according to the standard rules, i. e., it only changes the marking in the low-level WF-net containing t.
2. Let $t \in dom(\ell)$ be a transition enabled in $\mathcal{M}$. Then, a silent firing step $\mathcal{M} \xrightarrow{\tau} \mathcal{M}'$ may be done, where $\mathcal{M}' = \{M', m_1', m_2', \ldots, m_k'\}$, such that $M'(p) = M(p) - F(p, t)$ for all $p \in \tilde{P}$, $m_i' = m_i + [in]_i$, where $[in]_i$ — is an initial marking for the sub-process N_i corresponding to the enabled transition t, $m_j' = m_j, \forall j \neq i$, and τ — is the invisible action, that takes all tokens from the input places for t and add the initial marking to the low-level WF-net N_i.
3. Let m_i be a marking in the low-level WF-net N_i that contains a token in its final place f. Then, a silent firing step $\mathcal{M} \xrightarrow{\tau} \mathcal{M}'$ may be done, where $\mathcal{M}' = \{M', m_1', m_2', \ldots, m_k'\}$, such that $M'(p) = M(p) + F(t, p)$ for all $p \in \tilde{P}$, where t is the transition in the high-level WF-net $\tilde{N}$ corresponding to N_i, and $m_i' = m_i' - [fin]_i$, where $[fin]_i$ – is a final marking for N_i, $m_j' = m_j, \forall j \neq i$.

The example of an HWF-net is provided in Fig. 2. We do not impose specific restrictions on the number of input and output places a transition in a high-level WF-net can have. In Fig. 2, we only show the refinement of two transitions t_1 and t_2 in the high-level WF-net $\tilde{N}$ with two low-level WF-nets N_1 and N_2. They represent the low-level behavior of two sub-processes α_1 and α_2, respectively. Note that, if a low-level WF-net corresponding to a high-level activity contains the single transition, we still represent such a sub-process with an individual WF-net.

We next consider the operational semantics of an HWF-net by defining its run. For what follows, let $\mathcal{N} = (\tilde{N}, N_1, N_2, \ldots, N_k, \ell)$ be an HWF-net , where $\tilde{N}$ — is a high-level net and $N_1, N_2, \ldots, N_k$ — are nets for its sub-processes.

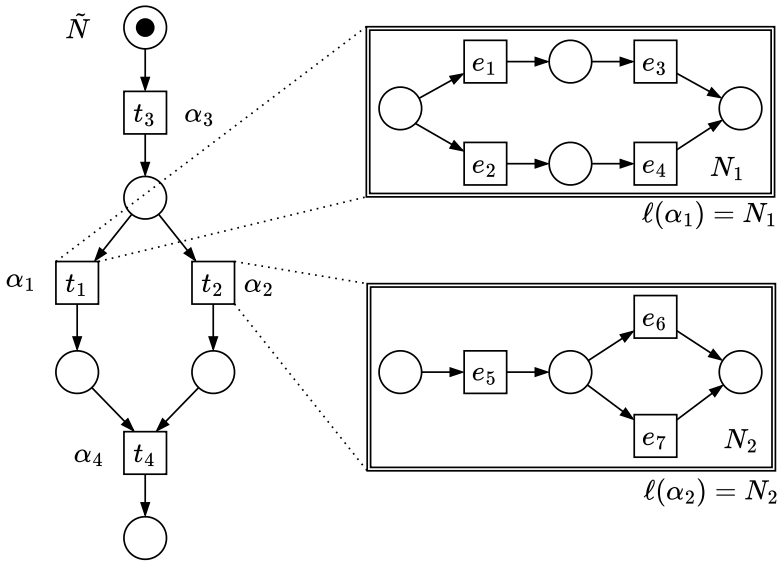


Fig. 2. An HWF-net with two refined transitions

Intuitively, a set of transitions enabled in in a high-level WF-net determines the set of sub-processes for which we can start to fire their low-level transitions. Transition firing, as described above, corresponds to starting, executing and terminating sub-processes, which can be run concurrently.

Let t' be a transition enabled in the current marking $\mathcal{M}$ of an HWF-net. The following options of $\tilde{\mathcal{M}} \xrightarrow{\lambda(t')} \tilde{\mathcal{M}}'$ are possible.

If there are high-level transitions, enabled at $\tilde{m}$ in a high-level WF-net, sharing common places, then there is a *conflict*. We can choose, which sub-process to start, while the other sub-processes corresponding to conflicting transitions in a high-level WF-net will not be able to be executed. For example, high-level transitions t_1 and t_2, once enabled, will be in conflict, and we can start only one of the corresponding sub-processes, N_1 or N_2. Firing a transition in a high-level WF-net is complete if the corresponding low-level WF-net reaches its final marking.

For instance, let us again consider the HWF-net shown in Fig. 2. After firing high-level transition t_3 and executing a corresponding sub-process α_3 (not provided in Fig. 2), two high-level transitions t_1 and t_2 become enabled. They share a common place, i. e., high-level transitions t_1 and t_2 are in conflict. Thus, we can execute exactly one of the corresponding sub-processes α_1 (low-level WF-net N_1) and α_2 (low-level WF-net N_2). We can, for instance, obtain a sequence $\rho = \langle \alpha_3, e_5, e_6, \alpha_4 \rangle$ which will represent a possible run of the HWF-net from Fig. 2. Note that high-level activities α_3 and α_4 should also be replaced with corresponding sub-process runs.

Lastly, we give a straightforward approach to transforming an HWF-net $\mathcal{N} = (\tilde{N}, N_1, N_2, \ldots, N_k, \ell)$ to the corresponding *equivalent* classic WF-net denoted by $\mathbf{eq}(\mathcal{N}) = (P, T, F, \lambda, [in])$. We need to replace transitions in a high-level WF-net with their sub-process implementation given by low-level WF-net corresponding by ℓ. In addition, we need to remove tokens from the input places of low-level WF-nets. When a transition t in a high-level WF-net $\tilde{N}$ is replaced by a low-level WF-net N_i, we need to fuse a source place in N_i with all input places of t and to fuse a sink place in N_i with all output places of t. By construction, $\mathbf{eq}(\mathcal{N})$ is a WF-net.

For instance, the WF-net $\mathbf{eq}(\mathcal{N})$ equivalent the HWF-net $\mathcal{N}$, shown in Fig. 2, is provided in Fig. 3. We replaced transition t_1 with N_1 and transition t_2 with N_2 as determined by the labels of low-level WF-nets. This figure also shows the double-line contours of corresponding high-level transitions.

Proposition 1 gives the main connection between an HWF-net and its classical representation.

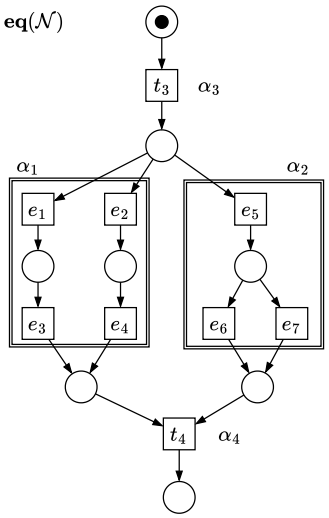


Fig. 3. The WF-net equivalent to the HWF-net in Fig. 2

Proposition 1. *Let $\mathcal{N} = (\tilde{N}, N_1, N_2, \ldots, N_k, \ell)$ be an HWF-net, and* $\mathbf{eq}(\mathcal{N})$ *be the corresponding equivalent WF-net. A sequence ρ of activities is a run in $\mathcal{N}$ if and only if ρ is a run in* $\mathbf{eq}(\mathcal{N})$.

In other words, a run in HWF-net $\mathcal{N}$ is also a run in the corresponding classical WF-net N and vice versa. Proof of this proposition directly follows from the construction of the classical WF-net and from the way we define the sequential semantics of a hierarchical WF-net and from semantic definition.

To sum up, for each HWF-net we can effectively build a classical WF-net having exactly the same behavior.

3.2. Events partitioning

We suppose that partitioning the set of low-level activities A into subsets $A_1, \ldots A_k$ is made either by an expert, or automatically based on some information contained in extended action records, such as resources or data. In Section 5, we give two examples of partitioning activities for a real log. Then we suppose that a sub-process is defined by its set of activities, and we suppose that sets of activities for two sub-processes do not intersect. If it is not the case and two sub-processes include some common activities like "close the file", one can easily distinguish them by appending the resource or file name to the activity identifier.

Let L be a log over a set A of activities, and let $A = A_1 \cup A_2 \cup \cdots \cup A_k$ be a partition of A. Let also $\tilde{A} = \{\alpha_1, \alpha_2, \ldots \alpha_k\}$ be a set of high-level activities (sub-process names).

The problem is to construct an HWF-net $\mathcal{N} = (\tilde{N}, N_1, N_2, \ldots, N_k, l)$, where for each $i \in [1, k]$, N_i is a sub-process (WF-net), labeled by α_i, with transitions labeled by low-level activities from A_i. The runs of $\mathcal{N}$ should conform to traces from L.

Another important remark concerning partitioning activities: we suppose that it does not violate the log control flow. Specifically, if there are iterations in the process, then for a set of iterated activities B and for each sub-process activities set A_i, we assume that either $B \cap A_i = \emptyset$, or $B \subseteq A_i$, or $A_i \subseteq B$. Note that this is a reasonable constraint, taking into account the concept of a sub-process. If it is still not the case, i. e., , only a part of A_i activities are iterated, then the partition can be refined, such that A_i is split into two subsets: a subset of iterated activities and the remainder.

For example, consider a low-level WF-net discovered from an event log shown in Fig. 4. Suppose that the set B of the iterated activities includes $\{a_3, a_4, b_3, b_4\}$ and that the low-level events are partitioned into two subsets $A_1 = \{a_1, a_2, a_3, a_4\}$ and $A_2 = \{b_1, b_2, b_3, b_4\}$.

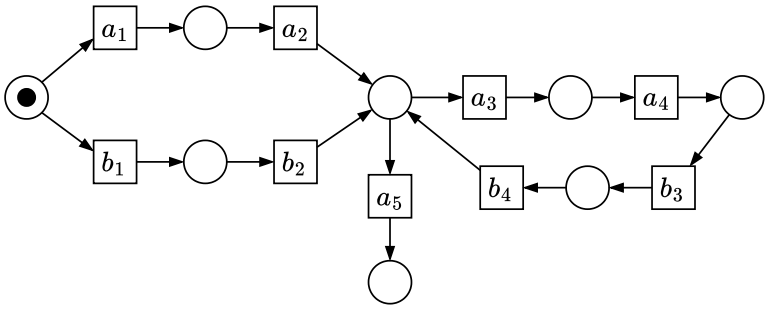


Fig. 4. The example cycle and high-level activity inconsistency

Using the proposed events partitioning, we cannot represent this model as a high-level WF-net, since the iterated activities belong to different high-level events. In addition, in the set of iterated activities B, low-level events a_3 and a_4 are always executed before b_3, b_4. Thus, one needs to revise this partitioning of low-level events in such a way that either B is fully included into a high-level activity A_i, or a high-level activity A_i is a part of a cycle.

3.3. The proposed solution

Here, we describe the main ideas and the structure of the algorithm to discover the hierarchical WF-net from an event log.

Let L be a log with activities from A, and let $A = A_1 \cup A_2 \cup \cdots \cup A_k$ be a partition of A. Let $\tilde{A} = \{\alpha_1, \alpha_2, \ldots \alpha_k\}$ be a set of high-level activities (sub-process names).

A hierarchical WF-net $\mathcal{N}$ (HWF-net) consists of a high-level WF-net $\tilde{N}$ with activities $\tilde{A} = \{\alpha_1, \ldots, \alpha_k\}$, and k sub-process WF-nets $N_1, N_2, \ldots, N_k$, where for each N_i, all its activities belong to A_i.

Sub-process WF-nets $N_1, N_2, \ldots, N_k$ can be discovered directly. To discover N_i, we filter log L to $L_i = L\uparrow_{A_i}$. Then we apply one of popular algorithms (e. g., Inductive Miner) to discover a WF-net from event log L_i. The fitness and precision of the obtained model depend solely on the choice of the discovery algorithm.

Example 1. *Consider an event log L of a business process over the set of low-level activities $A = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}\}$. Let L be an event log, such that $L = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6\}$, where $\sigma_1 = t_1 t_2 t_5 t_3 t_4 t_3 t_6 t_{12} t_7 t_3 t_{11}$, $\sigma_2 = t_1\ t_5\ t_6\ t_2\ t_3\ t_7\ t_8\ t_9\ t_{10}\ t_5\ t_6\ t_{11}$, $\sigma_3 = t_1\ t_2\ t_3\ t_5\ t_6\ t_{11}$, $\sigma_4 = t_1\ t_2\ t_5\ t_3\ t_{12}\ t_6\ t_3\ t_{11}$, $\sigma_5 = t_1\ t_2\ t_5\ t_3\ t_4\ t_6\ t_7\ t_3\ t_{12}\ t_8\ t_9\ t_{10}\ t_5\ t_6\ t_3\ t_{11}$, $\sigma_6 = t_1\ t_2\ t_3\ t_4\ t_3\ t_5\ t_6\ t_{11}$. A partition for this low-level activity set is $A = A_0 \cup A_1 \cup \cdots \cup A_5$, where $A_0 = \{t_1\}$, $A_1 = \{t_2\}$, $A_2 = \{t_3, t_4, t_{12}\}$, $A_3 = \{t_5, t_6\}$, $A_4 = \{t_7, t_8, t_9, t_{10}\}$, $A_5 = \{t_{11}\}$. A set of high-level activities for our example is $\tilde{A} = \{\alpha_0, \ldots, \alpha_5\}$, such that for every high-level activity from $\tilde{A}$ the corresponding sub-process $\alpha_i \in \tilde{A}, i \leq |\tilde{A}|$ contains activities only from $A_i \in A$.*

As the high-level WF-net of the hierarchical WF-net we consider the workflow net shown in Fig. 5 with activities labeled with $\tilde{A}$.

The existing process mining algorithm should be able to discover the workflow net presented in Fig. 6 directly from the low-level event log L. In this low-level net, we can also see that sub-processes, corresponding to high-level activities $\alpha_0, \alpha_1, \ldots \alpha_5$, have the same relations between activities as in the net from Fig. 5. For simplicity, the sub-processes α_0, α_1 and α_5, consisting of the single transition, are not highlighted with double rectangles.

Discovering a high-level WF-net is not so easy and is quite a challenge. The main problem with it is caused by the possible interleaving of concurrent sub-processes and iteration. A naive solution could be to replace each activity $t_j \in A_i$ by α_i — the name of the corresponding sub-process — in the log L. Then we need to remove *stuttering*, i. e., to replace, wherever possible, several sequential occurrences of the same high-level activity by a single activity. Then we apply the one of popular discovery algorithms to the obtained log over the set $\tilde{A}$ of activities. However, this does not work, due to the presence of the patterns of behavior other than the simple sequential execution.

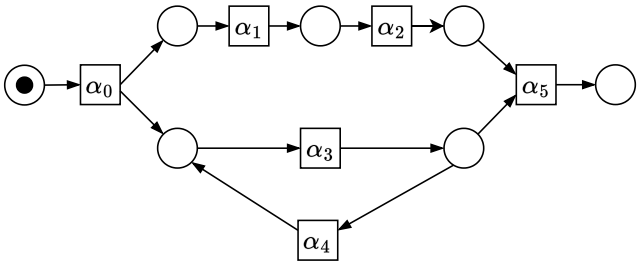


Fig. 5. The high-level workflow net for Fig. 6

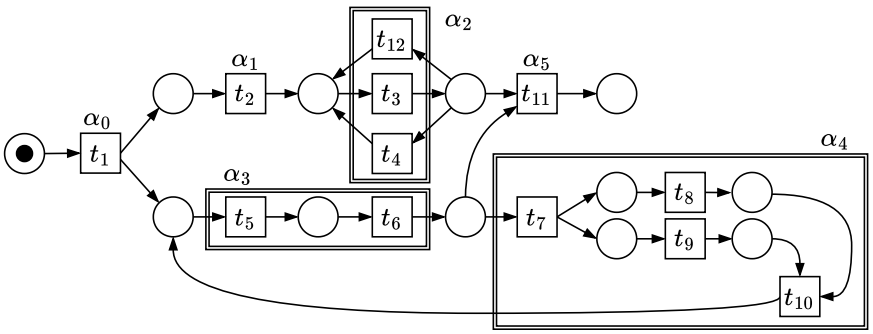


Fig. 6. An example of a low-level net with cycles inside the subprocess and between two subprocesses corresponding to the high-level net from Fig. 5

Consider the examples in Fig. 7. Fragment (a) in Fig. 7 shows two concurrent sub-processes β and γ, executing after sub-process α, which consists of the single transition. After replacing low-level activities with the corresponding sub-process names and removing stuttering, for the fragment (a), we get the following runs: $\langle \alpha, \beta, \gamma, \dots \rangle$, $\langle \alpha, \gamma, \beta, \dots \rangle$, $\langle \alpha, \beta, \gamma, \beta, \gamma, \dots \rangle$, $\langle \alpha, \gamma, \beta, \gamma, \beta, \dots \rangle$ etc. Fragment (b) in Fig. 7 shows a cycle. The body of this cycle is the sequence of two sub-processes β and γ. Among runs for the fragment (b) we also have $\langle \alpha, \beta, \gamma, \dots \rangle$, $\langle \alpha, \beta, \gamma, \beta, \gamma, \dots \rangle$. That is why iterations should be considered separately.

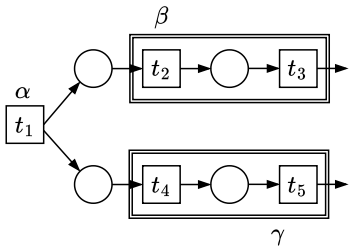
Discovering high-level WF-nets for acyclic processes, i. e., logs without iteration, was studied earlier in [3] where all details can be found. Here, we refer to this algorithm as Algorithm $\mathfrak{A}_0$ and illustrate it with the example in Fig. 7(a). Algorithm $\mathfrak{A}_0$, discovering a high-level WF-model from a log L without cycles, reduces this problem to the classical discovery problem, which can be solved by many popular algorithms, such as Inductive Miner. Therefore, Algorithm $\mathfrak{A}_0$ can be parameterized by Algorithm $\mathfrak{D}$, i. e., an already known algorithm for solving the classical discovery problem.

Algorithm $\mathfrak{A}_0(\mathfrak{D})$:

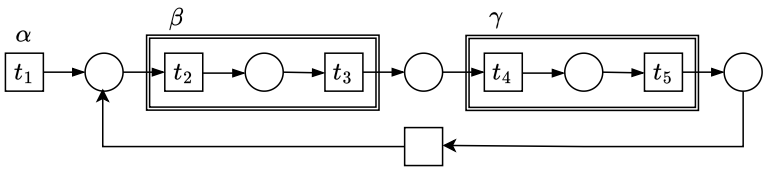
Step 1. For all traces in L, replace each activity with the corresponding sub-process names and remove stuttering.

Step 2. For each trace σ that contains $t_i \in \sigma, i \leq |\sigma|$ such as $\#t_i > 1$, find all occurrence of t_i in σ. For each occurrence's position k create a clone of σ, delete from it every t_i except the one at the position k and remove (newly formed) stuttering. Replace σ with the set of all obtained clones of σ.
For example, the trace $\langle \alpha, \beta, \gamma, \beta, \gamma, \dots \rangle$ will be replaced by two traces $\langle \alpha, \beta, \gamma, \dots \rangle$ and $\langle \alpha, \gamma, \beta, \dots \rangle$ obtained by keeping the first occurrences of β and γ, and correspondingly by keeping the first occurrence of γ and the second occurrence of β. In this example, constructing clones by keeping other occurrences of γ does not generate new traces.

Step 3. Let $\tilde{L}$ be the resulting log from executing two previous steps. To obtain a high-level WF-net $\tilde{N}$, apply the given as the input parameter Algorithm $\mathfrak{D}$, to discover a WF-net from event log $\tilde{L}$.



(a) concurrent sub-processes



(b) sub-processes inside a loop

Fig. 7. Interleaving and iteration of sub-processes

It was proven in [3] that if an algorithm used in Step 3 of Algorithm $\mathfrak{A}_0$ for each input log L discovers a WF-net perfectly fitting L, then Algorithm $\mathfrak{A}_0$, given a log L without repetitive behavior, produces an HWF-net $\mathcal{N}$ such that $\mathbf{eq}(\mathcal{N})$ perfectly fits L.

3.4. Detecting cycles in event logs

Now we come to logs with the repetitive behavior. The main idea here is to represent a loop body as a subset of its activities. Then a body of a loop can be considered as a sub-process with a new loop sub-process name. To discover the repetitive behavior, we use the method from [28], which allow us to determine causal, concurrency, and repetitive relations between events in an event log. Actually, for our purpose we need only repetitive relations and the loop discovery based on them. We call the loop discovery algorithm — Algorithm $\mathfrak{B}$. The strategy of this algorithm includes the pruning of interleaving tasks in an event trace to separate them by the supports of minimal transition invariants (t-invariants) — firing sequences, which represent potential cycles (see [31] for details). The procedure to obtain the t-invariants operates recursively on every task in trace σ_i from the most external cycle in every trace to the smaller nested cycles.

Let us consider the application of Algorithm $\mathfrak{B}$ in the following example.

Consider the log L from Example 1. Extracting the information about cycles is to derive the direct causal and concurrency relations between transitions in the event log L. The result of this step is shown in Table 1.

Algorithm $\mathfrak{B}$ finds all sub-sequences containing the repetitive behaviour, i. e., if in a trace σ there is an event t_i such that $\#t_i > 1$, then the sub-sequence, which contains repetitive behavior, should start with t_i until the next occurrence of t_i in σ. In [28], such sequences are called *cycs*. In L, the following cycs are detected: $cyc_1 = \{t_3\ t_4\}$, $cyc_2 = \{t_3\ t_6\ t_{12}\}$ in σ_1, $cyc_3 = \{t_5\ t_6\ t_2\ t_3\ t_7\ t_8\ t_9\ t_{10}\}$ in σ_2, $cyc_4 = \{t_3\ t_{12}\ t_6\}$ in σ_4, $cyc_5 = \{t_5\ t_3\ t_4\ t_6\ t_7\ t_3\ t_8\ t_9\ t_{10}\}$, $cyc_6 = \{t_3\ t_4\ t_6\ t_7\}$ in σ_5, $cyc_7 = \{t_3\ t_4\}$ in σ_6. We can see that cyc_7 is equal to cyc_1. Thus, we can merge them because the goal is to obtain a set of different cycs, and their frequency is not important in this case. In addition, we can see that there is another cyc in cyc_5. Furthermore, according to [28], cyc_5 is not the elementary cyc because $\exists t_i \in cyc$ such that $\#t_i > 1$, and we could derive a smaller nested elementary $cyc' = \{t_3\ t_4\ t_6\ t_7\}$ from it.

The next step of the Algorithm $\mathfrak{B}$ is to build the causality graph for each cyc found. In the process of building the causality graph, we use the relations between activities. These relations can be extracted from the input event log using any suitable process discovery algorithm. The set of relations between activities for the log L is presented in Table 1.

Now, having all relations, we can easily build the causality graph for each cyc. In [28], it is proposed that a strongly connected component of the causality graph for a cyc is also the support of a minimal t-invariant in the final model. The resulting graphs are shown in Fig. 8. Note that the graphs for cyc_1 and cyc_7 are equal because they contain the same transitions, and this is also true for cyc_2 and cyc_4. Therefore, we have depicted them only once (Fig. 8a and Fig. 8b, respectively).

Let us consider the case of cyc_5 separately because the cyc contains a nested cyc. In this case, first, we processed the nested (elementary) cyc' and built the causality graph for it. The final causality graph

Table 1. Relations between transitions in Example 1

T_i	Causal Relationship	Concurrent Relationship
t_1	t_2, t_5	
t_2	t_3, t_5	
t_3	t_{11}, t_{12}	t_5, t_6, t_7
t_4	t_6	
t_5	t_6	t_3
t_6	t_2, t_7, t_{11}	t_3, t_4, t_{12}
t_7	t_8	t_3
t_8	t_9	
t_9	t_{10}	
t_{10}	t_5	
t_{12}	t_7, t_8	t_6

for cyc' is equal to the causality graph for cyc_6 presented in Fig. 8e. The strongly connected component in this case is equal to cyc_1, and it is already in our set of t-invariant supports. The next step is to remove the detected t-invariant support of a nested cyc from cyc_5. Afterwards, we get the new sequence of transitions $cyc'_5 = \{t_5\ t_6\ t_7\ t_3\ t_8\ t_9\ t_{10}\}$, and we can build the causality graph for this new elementary cyc. The resulting graph is shown in Fig. 8d.

As a result of applying Algorithm $\mathfrak{B}$ to the log L, we obtain the following set of t-invariant supports: $Y_1 = \{t_3, t_4\},\ Y_2 = \{t_3, t_{12}\},\ Y_3 = \{t_5, t_6, t_7, t_8, t_9, t_{10}\}$.

Algorithm $\mathfrak{B}$ allows us to detect the bodies of elementary cycles as sets of their activities and process them recursively, starting with inner elementary cycles. Thus, at each iteration we are dealing with a loop body without internal loops. To obtain a sub-trace, corresponding to the loop body with a set of activities B from a log trace σ, we construct the projection $\sigma\restriction_B$. After filtering all current traces in this way, we get an event log for discovering a WF-net that simulates the behavior of the loop body using Algorithm $\mathfrak{A}_0$.

As mentioned in Section 3.2, partitioning of low-level events should not violate the log control flow from the point of view on the iterated behavior. Here we give a more precise representation of this requirement based on the results produced by Algorithm $\mathfrak{B}$ to detect cycle bodies, in terms of t-invariants.

Let $B = \{b_1, \ldots, b_n\}$ be the cycle bodies found by Algorithm $\mathfrak{B}$ and L be an event log over $A = A_1 \cup A_2 \cup \cdots \cup A_k$, where k is a number of high-level activities. Then, the partition of events $A = A_1 \cup A_2 \cup \cdots \cup A_k$ is *consistent* with B iff $\forall b_i \in B$ and $A_j \in A$ one of the following holds:

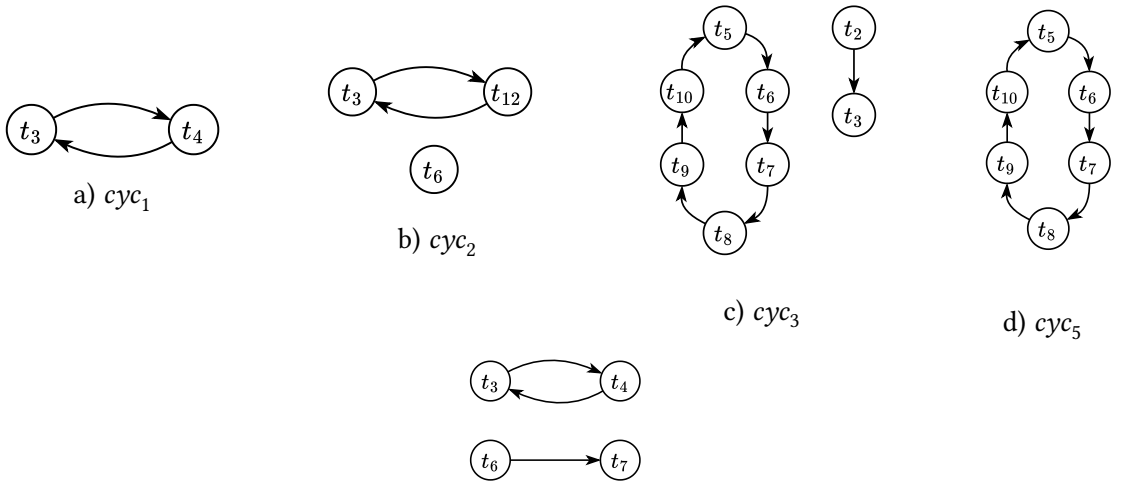
1. $b_i \cap A_j = \emptyset$;
2. $b_i \subseteq A_j$;
3. $A_j \subseteq b_i$.

The example in Fig. 4 discussed earlier in Section 3.2 shows that the inconsistency between the event partitioning and iterated behavior does not allow to construct an WF-net, since high-level events have common parts. Inconsistency can be corrected by revising the initial event partitioning.

The resulting high-level WF-net is then constructed recursively by replacing, the body of each detected loop with the name of its sub-process, starting with the inner loops. Note that if, after this step, in the WF-net there are transitions that are involved in more than one cycle, we need to merge all the same named transitions into a single one with that name. This also applies to places, because logically some of them should also be merged, depending on the activities relations. As the strategy of the merging algorithm, we also use the one proposed in [28]. We call the algorithm for merging transitions by activities correspondence Algorithm $\mathfrak{C}$.

4. Algorithm for discovering HWF-nets from low-level event logs

Here, we describe our main discovery algorithm in more detail.



a) cyc_1 b) cyc_2 c) cyc_3 d) cyc_5 e) cyc_6

Fig. 8. Causality graphs for every cyc from L

Let A be a set of activities and L — a log over A. Let then $A = A_1 \cup \cdots \cup A_k$ be a partition of A. Let $\tilde{A} = \{\alpha_1, \ldots, \alpha_k\}$ be a set of sub-process names. For $i \in [1, k]$, A_i is a set of activities of a sub-process α_i.

Then Algorithm $\mathfrak{A}(\mathfrak{D})$ constructs an HWF-net $\mathcal{N} = (\tilde{N}, N_1, N_2, \ldots, N_k, \ell)$ with high-level WF-net $\tilde{N} = (\tilde{P}, \tilde{T}, \tilde{F}, \tilde{\lambda}, [\tilde{in}])$, where $\tilde{\lambda}\colon \tilde{T} \to \tilde{A}$ and for each $\alpha_i \in \tilde{A}$, $\ell(\alpha_i) = N_i$, i. e., sub-process named with α_i corresponds to low-level WF-net N_i in $\mathcal{N}$.

By $\tilde{B} = \{\beta_0, \beta_1, \ldots, \beta_m\}$ we denote a set of cycle names and by ℓ_B — a function which maps each name from $\tilde{B}$ to a WF-net that implements the cycle with this name. For a WF-net N, denote by $Loop(N)$ a WF-net that is a loop with body N.

Algorithm $\mathfrak{A}(\mathfrak{D})$:

Step 1. Apply Algorithm $\mathfrak{B}$ to L to find a set $B = \{b_1, \ldots, b_m\}$ of cycle bodies, where every $b \in B$ is a set of activities in some cycle and $m = |B|$. A cycle name for each $b \in B$ will be stored in a set $\tilde{B}$ where $|\tilde{B}| = |B|$. The correspondence between every $b \in B$ and its name $\beta \in \tilde{B}$ is defined by index, i. e., for each set of cycle activities $b_q \in B$ a cycle name will be $\beta_q \in \tilde{B}$, where $q = 1, 2, \ldots, m$ is the index.

Step 2. Construct the projection $L\restriction_{b_i}$ for each $b \in B$ and apply Algorithm $\mathfrak{A}_0(\mathfrak{D})$ to it (with respect to the partition $A = A_1 \cup \cdots \cup A_k$). Let $\tilde{N}$ be the resulting high-level WF-net over the set $\tilde{A}$ of sub-process names. Let $N_1, \ldots, N_j$ — resulting WF-nets for sub-processes with names $\alpha_1, \ldots, \alpha_j$. Let $N_{\beta_1}, \ldots, N_{\beta_m}$ — resulting WF-nets for sub-processes within the cycle.

Step 3. Let $\ell_B(\beta_1) = N_{\beta_1}, \ldots, \ell_B(\beta_m) = N_{\beta_j}$. Let also $\ell(A_1) = N_1, \ldots, \ell(A_j) = N_j$.

Step 4. For every $\sigma \in L$ such as $b_j \subseteq \sigma$ and $i = 1, 2, \ldots, |L|$ replace by β_j all occurrences of activities from b_j in σ_i and remove stuttering.

Step 5. Let A_{new} be a current set of activities such as $A_{new} = A \cup \tilde{B}$ and $\tilde{A}_{new}$ be a current partition of A_{new} such as $\tilde{A}_{new} = \tilde{A} \cup \tilde{B}$.

Step 6. Apply Algorithm $\mathfrak{A}_0(\mathfrak{D})$ to the log L obtained at *Step 4* with respect to the current partition of activities $\tilde{A}_{new}$. Let $\tilde{N}$ be a resulting high-level WF-net.

Step 7. For each $\beta \in \tilde{B}$, replace a transition labeled by β in $\tilde{N}$ with the sub-process $Loop(\ell_B(\beta))$.

Step 8. For each pair of transition $\tilde{t_i}, \tilde{t_j} \in \tilde{T}$ from $\tilde{N}$, which are corresponding to the same α sub-process name, apply the merging Algorithm $\mathfrak{C}$.

The resulting net $\tilde{N}$ is a high-level WF-net for the HWF-net constructed by Algorithm. Its low-level WF-nets, which are defined by function ℓ, are also built during Algorithm operation.

Correctness of Algorithm $\mathfrak{A}(\mathfrak{D})$ is justified by the following statement.

Theorem 1. *Let A be a set of activities and L — a log over A. Let also $A = A_1 \cup \cdots \cup A_k$ be a partition of A, and $\tilde{A} = \{\alpha_1, \ldots, \alpha_k\}$ — a set of sub-process names.*

If Algorithm $\mathfrak{D}$, given a log L', discovers a WF-net N' such that N' perfectly fits L', then Algorithm $\mathfrak{A}(\mathfrak{D})$ constructs an HWF-net $\mathcal{N} = (\tilde{N}, N_1, N_2, \ldots, N_k, \ell)$, such that $\mathcal{N}$ perfectly fits L according to the substitution.

Proof. To prove that an HWF-net built using Algorithm $\mathfrak{A}(\mathfrak{D})$ perfectly fits the input log, provided that Algorithm $\mathfrak{D}$ discovers models with perfect fitness, we use three previously proven assertions, namely:

1. The theorem proven in [3] states that when $\mathfrak{D}$ is an discovery algorithm with perfect fitness, Algorithm $\mathfrak{A}_0(\mathfrak{D})$ discovers a high-level WF-net, whose refinement perfectly fits the input log without repetitions (the log of an acyclic process).
2. In [28] it is proven that, given a log L, Algorithm $\mathfrak{B}$ correctly finds in L all repetitive components that correspond to supports of t-invariants in the Petri net model for L.
3. Proposition 1 in Subsection 3.1 justify correctness of refining a high-level WF-net by substituting sub-process modules for high-level transitions.
4. In [28] it is proven that the given merging strategy of Algorithm $\mathfrak{C}$ correctly merges all equally named transitions after substitutions of the processes corresponding to the components of repetitive behavior to the Petri net model for L.

□

Taking the above into consideration, proving the theorem is straightforward, though quite technical. Thus, we informally describe the logic of the proof here.

Let Algorithm $\mathfrak{D}$ be a discovery algorithm that discovers a perfectly fitting WF-net for a given event log.

From *Step 1* to *Step 5* of the algorithm, repetitive components, i. e., cycles are processed. At *Step 1*, all inner elementary repetitive components in the log are discovered using Algorithm $\mathfrak{B}$. The activities of every component are those of some inner loop body, which do not have repetitions. Then, a WF-net N for this loop body is correctly discovered using Algorithm $\mathfrak{A}_0(\mathfrak{D})$, and the loop itself is folded into a high-level activity β, and N is kept as the value $\ell_B(\beta)$. WF-nets for sub-processes within the body of this loop are also discovered by Algorithm $\mathfrak{A}_0(\mathfrak{D})$ and accumulated by ℓ. If a loop activity β is contained in another loop body, then with the one more iteration of Step 1, the upper loop N' is discovered, the transition labeled with β in it is replaced with N, and N' is itself folded into a new high-level activity.

After processing all loops, Algorithm proceeds to *Step 6*, where after reducing all loops to high-level activities, Algorithm $\mathfrak{A}_0(\mathfrak{D})$ is applied to a log without repetitions.

In *Step 7* all transitions labeled with loop activities in a high-level and low-level WF-nets are replaced by WF-nets for these loops, kept by ℓ_B.

Step 8 merges all equally named transitions that corresponds to the same activity in the event log.

That is why we can see that, while Algorithm $\mathfrak{A}_0(\mathfrak{D})$ ensures the perfect fitness between the acyclic fragments of the model (when loops are folded into transitions), Algorithm $\mathfrak{B}$ ensures correct processing of cyclic behavior, and Proposition 1 guarantees that replacing loop activities by the corresponding loop WF-nets does not violate perfect fitness, the main algorithm provides systematic log processing and model construction.

5. Experimental evaluation

In this section, we report the main outcomes from a series of experiments conducted to evaluate the algorithm for discovering two-level hierarchical process models from event logs.

To support the automated experimental evaluation, we implemented the hierarchical process discovery algorithm described in the previous section using the open source library PM4Py [32]. The source files for our implementation are published on the open GitHub repository [33]. We conducted experiments using two kinds of event logs:

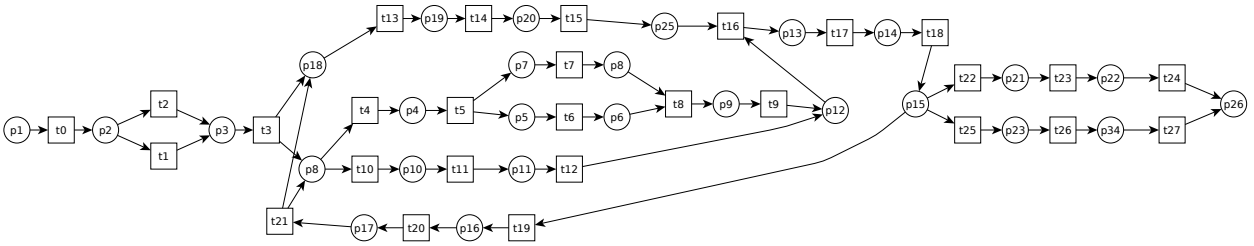


Fig. 9. A classical WF-net with generated by refining the WF-net in Fig. 1

1. *Artificial* event logs generated by manually prepared process models.
2. *Real-life* event logs provided by various information systems.

Event logs are encoded in a standard format as XML-based XES files.

To assess the quality of the algorithm quality, we will use several metrics of *conformance checking*. *Conformance checking* is an important part of process mining along with process discovery [34]. The main aim of conformance checking is to evaluate the quality of process discovery algorithm by estimating the corresponding quality of discovered process models. Conformance checking provides four main quality dimensions. *Fitness* estimates the extent to which a discovered process model can execute traces in an event log. A model perfectly fits an event log if it can execute all traces in an event log. According to Theorem 1, the hierarchical process discovery algorithm yields perfectly fitting process models. *Precision* evaluates the ratio between the behavior allowed by a process model and the one not recorded in an event log. A model with perfect precision can only execute traces in an event log. The perfect precision limits the use of a process model since an event log represents only a finite "snapshot" of all possible process executions. Generalization and precision are two dual metrics. The fourth metric, *simplicity*, captures the structural complexity of a discovered model. We improve simplicity by the two-level structure of a discovered process models.

Within the experimental evaluation, we estimated fitness and precision of process models discovered from artificially generated and real-life event logs. Fitness was estimated using alignments between a process model and an event log as defined in [35]. The precision was estimated using the complex ETC-align measures proposed in [36]. Both measures are values in the interval $[0, 1]$. As a discovery algorithm, we used Inductive Miner.

5.1. Discovering HWF-Nets from Artificial Event Logs

The high-level source for generating artificial low-level event logs was the Petri net shown earlier in Fig. 1. In this model, we refined its transitions with different sub-processes containing sequential, parallel and cyclic executions of low-level events. The example of refining the Petri net from Fig. 1 is shown in Fig. 9, where we show the corresponding classical representation of an HWF-net.

Generation of low-level event logs from the prepared model was implemented using the algorithm presented in [37]. Afterwards, we transform a low-level event log into a high-level event log by grouping low-level events into a single high-level event and by extracting information about cyclic behavior.

The corresponding high-level WF-net discovered from the artificial low-level event log that is generated from the WF-net shown in 9 is provided in Fig. 10. Intuitively, one can see that this high-level WF-net is rather similar to the original Petri net from Fig. 1.

As for the quality evaluation for the above presented high-level model, we have the following:

1. The discovered high-level WF-net perfectly fits the high-level event log obtained from a low-level log, where we identified cycles and grouped activities correspondingly.
2. The classical WF-net obtained by refining transitions in a discovered high-level WF-net by discovered sub-nets perfectly fits the low-level log.

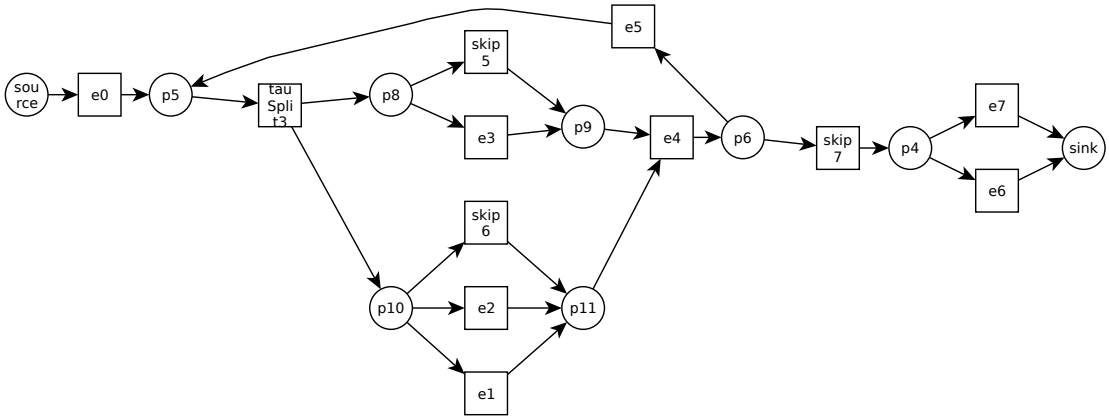


Fig. 10. A high-level WF-net discovered from an event log generated by refining the WF-net in Fig. 9

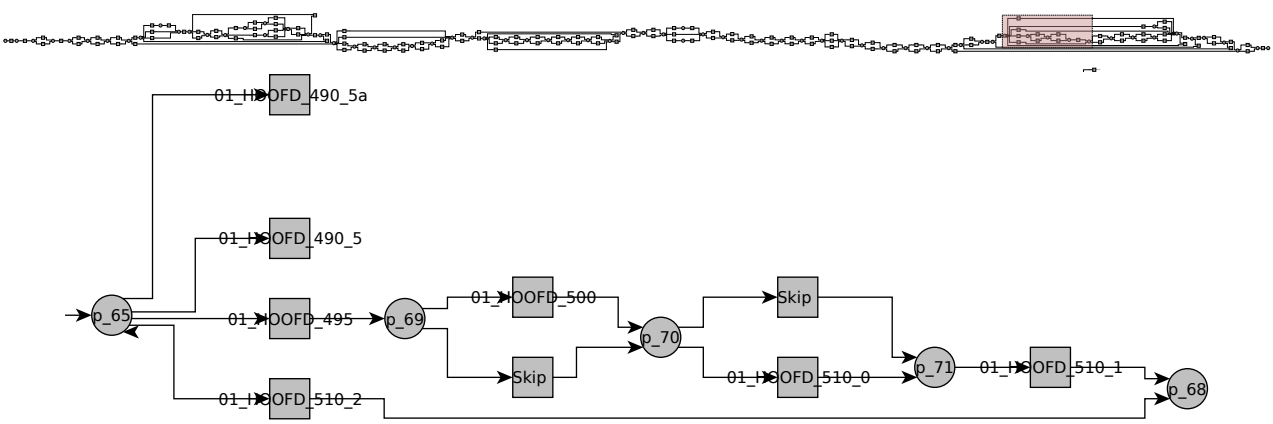


Fig. 11. A classical WF-net discovered from *BPI Challenge 2015* event log

Other examples of process models that were used for the artificial event log generation are also provided in the main repository [33].

5.2. Discovering HWF-nets from real-life event logs

We used two real-life event logs provided by *Business Process Intelligence Challenge (BPI Challenge)* 2015 and 2017 [38]. These event logs were also enriched with additional statistical information about classical process models.

The *BPI Challenge 2015* event log was provided by five Dutch municipalities. The cases in this event log contain information on the main application and objection procedures in various stages. A classical low-level WF-net for case $f1$ discovered using the Inductive miner is shown in Fig. 11. In addition, Fig. 11 shows an enlarged part of the process highlighted in the final model to clearly demonstrate the level of detail. It is easy to see that the resulting model is absolutely inappropriate for visual analysis.

The code of each event in the *BPI Challenge 2015* event log consists of three parts: two digits, a variable number of characters, and three more digits.

Using the event log description, we know that the first two digits and the characters indicate the sub-process the event belongs to, which allows us to assume an option of identifying the sub-processes.

We used the first two parts of the event name to create the mapping between low-level events and sub-process names. After applying our hierarchical process discovery algorithm in combination with the In-

ductive Miner, we obtained a high-level model presented in Fig. 12 that is far more comprehensible than the classical model mainly because of its size.

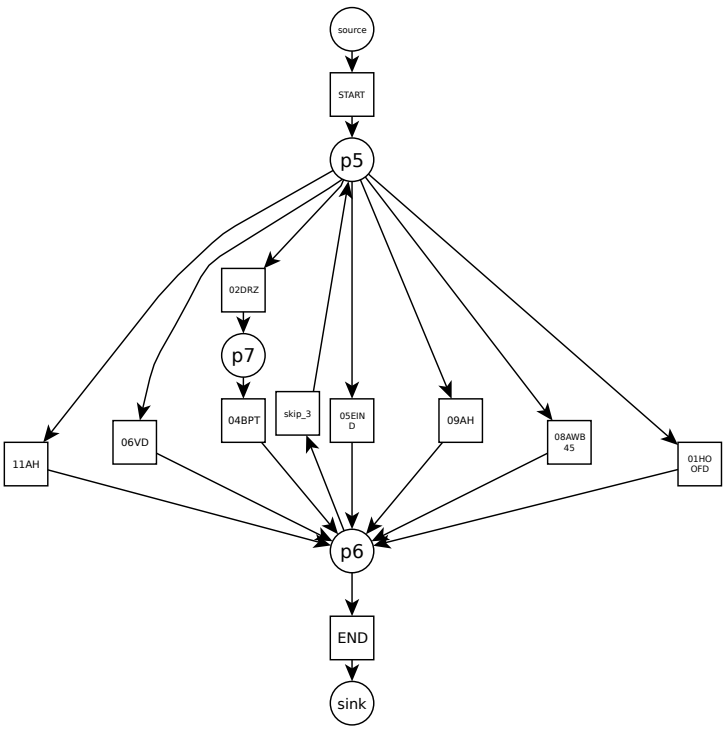


Fig. 12. A high-level WF-net discovered from the *BPI Challenge 2015* event log

The *BPI Challenge 2017* event log pertains to a loan application process of a Dutch financial institute. The data contains all applications filed trough an online system from 2016 till February of 2017. Here, as a base for mapping low-level events to sub-process names, we used the mark of the event type in its name — application, offer or workflow. Thus, a mapping could be based on various features of event data depending on the expert's needs. The classical model for these data is presented in Fig. 13, which is also difficult to read due to its purely sequential representation.

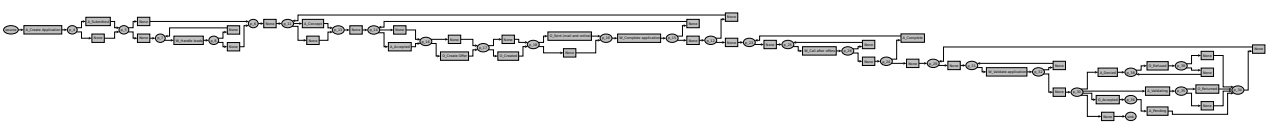


Fig. 13. A classical WF-net discovered from the *BPI Challenge 2017* event log

Applying the principle of mapping low-level events in the *BPI Challenge 2017* event log described above, we obtained the high-level WF-net shown in Fig. 14, which clearly demonstrates sub-processes (if necessary, they can be expanded) and their order.

Table 2 shows the fitness and precision evaluation of classical and high-level WF-nets discovered from real-life *BPI Challenge 2015* and *2017* event logs.

Fitness 1 shows the fitness evaluation between the classical WF-net constructed from the high-level WF-net by refining transitions with low-level sub-processes.

Fitness 2 shows the fitness evaluation between the high-level WF-net and an event log with low-level events grouped into sub-processes. This confirms the formal correctness results of the hierarchical pro-

cess discovery algorithm. Similar to the experimental results for artificial event logs, here we also observe a decrease in the precision for the identification of sub-processes, therefore, generalizing traces in an initial low-level event log.

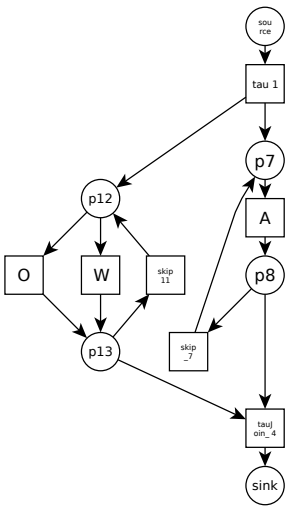


Fig. 14. A high-level WF-net discovered from the *BPI Challenge 2017* event log

Table 2. Comparing metrics for classical and high-level WF-nets discovered from *BPI Challenge* event logs

Event log	High-level WF-net			Classical WF-net	
	Fitness 1	Fitness 2	Precision	Fitness	Precision
BPI Challenge 2015	1	1	0.5835	1	0.5700
BPI Challenge 2017	1	1	0.3898	1	0.7000

6. Conclusion and Future Work

In this study, we propose a new process discovery technique for solving the problem of discovering a hierarchical WF-net model from a low-level event log, based on sub-processes abstraction into high-level transitions according to event partitioning. Unlike the previous solutions, we allow cycles and concurrency in process behavior.

We prove that the proposed technique makes it possible to obtain hierarchical models, which fit event logs perfectly. The technique was also evaluated in real and artificial event logs. Experiments show that fitness and precision of obtained hierarchical models are almost the same as for the standard "classical" case, while hierarchical models are much more compact, more readable and more visual.

To implement our algorithm and check it on real data we used Python and one of the most convenient instruments for process mining at the moment — the PM4Py [32]. The implementation is provided in the public GitHub repository [33].

In further research, we plan to develop and evaluate various event partitioning methods for automatic discovery of hierarchical models.

References

[1] A. Augusto *et al.*, "Automated discovery of process models from event logs: Review and benchmark", *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 4, pp. 686–705, 2018.

[2] W. van der Aalst, “Workflow verification: Finding control-flow errors using Petri-net-based techniques”, in *Business process management: models, techniques, and empirical studies*, Springer, 2002, pp. 161–183.

[3] A. K. Begicheva and I. A. Lomazova, “Discovering high-level process models from event logs”, *Modeling and Analysis of Information Systems*, vol. 24, no. 2, pp. 125–140, 2017.

[4] S. J. van Zelst, F. Mannhardt, M. de Leoni, and A. Koschmider, “Event abstraction in process mining: Literature review and taxonomy”, *Granular Computing*, vol. 6, no. 3, pp. 719–736, 2021.

[5] D. G. Maneschijn, R. H. Bemthuis, F. A. Bukhsh, and M.-E. Iacob, “A methodology for aligning process model abstraction levels and stakeholder needs”, in *Proceedings of the 24th International Conference on Enterprise Information Systems - Volume 1*, 2022, pp. 137–147.

[6] F. Mannhardt, M. de Leoni, H. Reijers, W. van der Aalst, and P. Toussaint, “From low-level events to activities – a pattern-based approach”, in *Business Process Management*, Springer, 2016, pp. 125–141.

[7] N. Tax, N. Sidorova, R. Haakma, and W. van der Aalst, “Event abstraction for process mining using supervised learning techniques”, in *Proceedings of SAI Intelligent Systems Conference 2016*, Springer, 2018, pp. 161–170.

[8] C.-Y. Li, S. J. van Zelst, and W. van der Aalst, “A framework for automated abstraction class detection for event abstraction”, in *Intelligent Systems Design and Applications*, Springer, 2023, pp. 126–136.

[9] G. van Houdt, M. de Leoni, N. Martin, and B. Depaire, “An empirical evaluation of unsupervised event log abstraction techniques in process mining”, *Information Systems*, vol. 121, p. 102 320, 2024.

[10] A. Rebmann, P. Pfeiffer, P. Fettke, and H. v. d. Aa, “Multi-perspective identification of event groups for event abstraction”, in *Process Mining Workshops*, Springer, 2023, pp. 31–43.

[11] S. J. Leemans, K. Goel, and S. J. van Zelst, “Using multi-level information in hierarchical process mining: Balancing behavioural quality and model complexity”, in *Proceedings of the 2nd International Conference on Process Mining*, IEEE, 2020, pp. 137–144.

[12] A. Senderovich, A. Shleyfman, M. Weidlich, A. Gal, and A. Mandelbaum, “To aggregate or to eliminate? Optimal model simplification for improved process performance prediction”, *Information Systems*, vol. 78, pp. 96–111, 2018.

[13] S. Smirnov, H. Reijers, M. Weske, and T. Nugteren, “Business process model abstraction: A definition, catalog, and survey”, *Distributed and Parallel Databases*, vol. 30, pp. 63–99, 2012.

[14] C. W. Günther and W. M. Van Der Aalst, “Fuzzy mining–adaptive process simplification based on multi-perspective metrics”, in *International conference on business process management*, Springer, 2007, pp. 328–343.

[15] W. Reisig, *Understanding Petri nets: Modeling techniques, analysis methods, case studies*. Springer, 2013.

[16] K. Jensen and L. Kristensen, *Coloured Petri nets: modelling and validation of concurrent systems*. Springer, 2009.

[17] S. Leemans, D. Fahland, and W. van der Aalst, “Discovering block-structured process models from event logs – a constructive approach”, in *Application and Theory of Petri Nets and Concurrency*, Springer, 2013, pp. 311–329.

[18] G. Greco, A. Guzzo, and L. Pontieri, “Mining taxonomies of process models”, *Data & Knowledge Engineering*, vol. 67, no. 1, pp. 74–102, 2008.

[19] J. Li, R. Bose, and W. van der Aalst, “Mining context-dependent and interactive business process maps using execution patterns”, in *Business Process Management Workshops. BPM 2010*, Springer, 2010, pp. 109–121.

[20] X. Lu, A. Gal, and H. A. Reijers, “Discovering hierarchical processes using flexible activity trees for event abstraction”, in *Proceedings of the 2nd International Conference on Process Mining*, IEEE, 2020, pp. 145–152.

[21] W. van der Aalst and C. Gunther, "Finding structure in unstructured processes: The case for process mining", in *Seventh International Conference on Application of Concurrency to System Design (ACSD 2007)*, IEEE, 2007, pp. 3–12. DOI: 10.1109/ACSD.2007.50.

[22] J. De Smedt, J. De Weerdt, and J. Vanthienen, "Multi-paradigm process mining: Retrieving better models by combining rules and sequences", in *On the Move to Meaningful Internet Systems: OTM 2014 Conferences*, Springer, 2014, pp. 446–453. DOI: 10.1007/978-3-662-45563-0_26.

[23] J. de San Pedro and J. Cortadella, "Mining structured Petri nets for the visualization of process behavior", in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, ACM, 2016, pp. 839–846. DOI: 10.1145/2851613.2851645.

[24] W. van der Aalst, A. Kalenkova, V. Rubin, and E. Verbeek, "Process discovery using localized events", in *Application and Theory of Petri Nets and Concurrency*, Springer, 2015, pp. 287–308. DOI: 10.1007/978-3-319-19488-2_15.

[25] A. Kalenkova and I. Lomazova, "Discovery of cancellation regions within process mining techniques", *Fundamenta Informaticae*, vol. 133, pp. 197–209, 2-3 2014. DOI: 10.3233/FI-2014-1071.

[26] A. Kalenkova, I. Lomazova, and W. van der Aalst, "Process model discovery: A method based on transition system decomposition", in *Application and Theory of Petri Nets and Concurrency*, Springer, 2014, pp. 71–90. DOI: 10.1007/978-3-319-07734-5_5.

[27] C.-Y. Li, S. J. van Zelst, and W. M. van der Aalst, "An activity instance based hierarchical framework for event abstraction", in *Proceedings of the 3rd International Conference on Process Mining*, 2021, pp. 160–167. DOI: 10.1109/ICPM53251.2021.9576868.

[28] T. Tapia-Flores, E. López-Mellado, A. P. Estrada-Vargas, and J.-J. Lesage, "Discovering Petri net models of discrete-event processes by computing t-invariants", *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 992–1003, 2017.

[29] T. Tapia-Flores and E. Lopez-Mellado, "Discovering workflow nets of concurrent iterative processes", *Acta Informatica*, vol. 61, Sep. 2023. DOI: 10.1007/s00236-023-00445-5.

[30] W. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, Heidelberg, 2011.

[31] K. Lautenbach, "Linear algebraic techniques for place/transition nets", in *Petri Nets: Central Models and Their Properties. ACPN 1986*, Springer, Heidelberg, 1987, pp. 142–167.

[32] A. Berti, S. Van Zelst, and W. van der Aalst, "Process mining for Python (PM4Py): Bridging the gap between process- and data science", in *Proceedings of the ICPM Demo Track 2019*, CEUR-WS.org, 2019, pp. 13–16.

[33] A. Begicheva, *Hierarchical process model discovery – hldiscovery*, 2022. [Online]. Available: https://github.com/gingerabsurdity/hldiscovery.

[34] J. Carmona, B. van Dongen, A. Solti, and M. Weidlich, *Conformance Checking: Relating Processes and Models*. Springer, 2018.

[35] A. Adriansyah, B. F. van Dongen, and W. M. van der Aalst, "Conformance checking using cost-based fitness analysis", in *2011 ieee 15th international enterprise distributed object computing conference*, IEEE, 2011, pp. 55–64.

[36] J. Munoz-Gama and J. Carmona, "A fresh look at precision in process conformance", in *International Conference on Business Process Management*, Springer, 2010, pp. 211–226.

[37] I. Shugurov and A. Mitsyuk, "Generation of a set of event logs with noise", in *Proceedings of the 8th Spring/Summer Young Researchers Colloquium on Software Engineering*, 2014, pp. 88–95.

[38] A. Augusto *et al.*, *Data underlying the paper: Automated discovery of process models from event logs: Review and benchmark (version 1)*, Data set. 4TU.Centre for Research Data, 2019. DOI: 10.4121/uuid:adc42403-9a38-48dc-9f0a-a0a49bfb6371.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 31, NO. 3, 2024
journal homepage: www.mais-journal.ru

DISCRETE MATHEMATICS IN RELATION TO COMPUTER SCIENCE

Estimation of Interpolation Projectors Using Legendre Polynomials

M. V. Nevskii[1]

DOI: 10.18255/1818-1015-2024-3-316-337

[1]P.G. Demidov Yaroslavl State University, Yaroslavl, Russia

MSC2020: 41A05, 52B55, 52C07
Research article
Full text in Russian

Received August 13, 2024
Revised August 26, 2024
Accepted August 28, 2024

We give some estimates for the minimum projector norm under linear interpolation on a compact set in $\mathbb{R}^n$. Let $\Pi_1(\mathbb{R}^n)$ be the space of polynomials in n variables of degree at most 1, Ω is a compactum in $\mathbb{R}^n$, $K = \operatorname{conv}(\Omega)$. We will assume that $\operatorname{vol}(K) > 0$. Let the points $x^{(j)} \in \Omega$, $1 \leqslant j \leqslant n+1$, be the vertices of an n-dimensional nondegenerate simplex. The interpolation projector $P : C(\Omega) \to \Pi_1(\mathbb{R}^n)$ with the nodes $x^{(j)}$ is defined by the equations $Pf\left(x^{(j)}\right) = f\left(x^{(j)}\right)$. By $\|P\|_\Omega$ we mean the norm of P as an operator from $C(\Omega)$ to $C(\Omega)$. By $\theta_n(\Omega)$ we denote the minimal norm $\|P\|_\Omega$ of all operators P with nodes belonging to Ω. By $\operatorname{simp}(E)$ we denote the maximal volume of the simplex with vertices in E. We establish the inequalities $\chi_n^{-1}\left(\frac{\operatorname{vol}(K)}{\operatorname{simp}(\Omega)}\right) \leqslant \theta_n(\Omega) \leqslant n+1$. Here χ_n is the standardized Legendre polynomial of degree n. The lower estimate is proved using the obtained characterization of Legendre polynomials through the volumes of convex polyhedra. More specifically, we show that for every $\gamma \geq 1$ the volume of the set $\left\{x = (x_1, \ldots, x_n) \in \mathbb{R}^n : \sum |x_j| + \left|1 - \sum x_j\right| \leq \gamma\right\}$ is equal to $\chi_n(\gamma)/n!$. In the case when Ω is an n-dimensional cube or an n-dimensional ball, the lower estimate gives the possibility to obtain the inequalities of the form $\theta_n(\Omega) \geqslant c\sqrt{n}$. Also we formulate some open questions.

Keywords: polynomial interpolation; projector; norm; esimate; Legendre polynomials

INFORMATION ABOUT THE AUTHORS

Nevskii, Mikhail V. (corresponding author) | ORCID iD: 0000-0002-6392-7618. E-mail: mnevsk55@yandex.ru
Head of the Chair, Dr. Sc., Docent

For citation: M. V. Nevskii, "Estimation of interpolation projectors using Legendre polynomials", *Modeling and Analysis of Information Systems*, vol. 31, no. 3, pp. 316–337, 2024. DOI: 10.18255/1818-1015-2024-3-316-337.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 31, № 3, 2024
сайт журнала: www.mais-journal.ru

DISCRETE MATHEMATICS IN RELATION TO COMPUTER SCIENCE

Оценивание интерполяционных проекторов с применением многочленов Лежандра

М. В. Невский[1]

DOI: 10.18255/1818-1015-2024-3-316-337

[1]Ярославский государственный университет им. П.Г. Демидова, Ярославль, Россия

УДК 514.17+517.51+519.6
Научная статья
Полный текст на русском языке

Получена 13 августа 2024 г.
После доработки 26 августа 2024 г.
Принята к публикации 28 августа 2024 г.

Приводятся оценки для минимальной нормы проектора при линейной интерполяции на компакте в $\mathbb{R}^n$. Пусть $\Pi_1(\mathbb{R}^n)$ — пространство многочленов от n переменных степени не выше 1, Ω — компакт в $\mathbb{R}^n$, $K = \operatorname{conv}(E)$. Будем предполагать, что $\operatorname{vol}(K) > 0$. Пусть точки $x^{(j)} \in \Omega$, $1 \leqslant j \leqslant n+1$, являются вершинами n-мерного невырожденного симплекса. Интерполяционный проектор $P : C(\Omega) \to \Pi_1(\mathbb{R}^n)$ с узлами $x^{(j)}$ определяется равенствами $Pf\left(x^{(j)}\right) = f\left(x^{(j)}\right)$. Под $\|P\|_\Omega$ будем понимать норму P как оператора из $C(\Omega)$ в $C(\Omega$. Через $\theta_n(\Omega)$ обозначим минимальную норму $\|P\|_\Omega$ из всех операторов P с узлами, принадлежащими Ω. Через $\operatorname{simp}(\Omega)$ обозначим максимальный объём симплекса с вершинами в Ω. Устанавливаются неравенства $\chi_n^{-1}\left(\frac{\operatorname{vol}(K)}{\operatorname{simp}(\Omega)}\right) \leqslant \theta_n(\Omega) \leqslant n+1$. Здесь χ_n — стандартизованный многочлен Лежандра степени n. Нижняя оценка доказывается с применением полученной характеризации многочленов Лежандра через объёмы выпуклых многогранников. Именно, мы показываем, что при $\gamma \geq 1$ объём многогранника $\left\{x = (x_1, \ldots, x_n) \in \mathbb{R}^n : \sum |x_j| + \left|1 - \sum x_j\right| \leq \gamma\right\}$ равен $\chi_n(\gamma)/n!$. В случае, когда Ω — n-мерный куб или n-мерный шар, нижняя оценка даёт возможность получить неравенства вида $\theta_n(\Omega) \geqslant c\sqrt{n}$. Формулируются некоторые открытые вопросы.

Ключевые слова: полиномиальная интерполяция; проектор; норма; оценка; многочлены Лежандра

ИНФОРМАЦИЯ ОБ АВТОРАХ

Невский, Михаил Викторович (автор для корреспонденции) | ORCID iD: 0000-0002-6392-7618. E-mail: mnevsk55@yandex.ru
Заведующий кафедрой, доктор физ.-мат. наук, доцент

Для цитирования: M. V. Nevskii, “Estimation of interpolation projectors using Legendre polynomials”, *Modeling and Analysis of Information Systems*, vol. 31, no. 3, pp. 316–337, 2024. DOI: 10.18255/1818-1015-2024-3-316-337.

Введение

Пусть Ω — компактное подмножество $\mathbb{R}^n$. Для функции $f \in C(\Omega)$ через $E_1(f;\Omega)_{C(\Omega)}$ обозначим величину наилучшего приближения f в $C(\Omega)$-норме многочленами от n переменных степени $\leqslant 1$. Пространство таких многочленов мы обозначаем через $\Pi_1(\mathbb{R}^n)$. Пусть P — полиномиальный проектор на Ω, т. е. линейный ограниченный оператор из $C(\Omega)$ в $\Pi_1(\mathbb{R}^n)$, такой что $P(Pf|_\Omega) = Pf$ для любой функции $f \in C(\Omega)$. Тогда выполняются следующие неравенства:

$$E_1(f;\Omega)_{C(\Omega)} \leqslant \|f - Pf\|_{C(\Omega)} \leqslant (1 + \|P\|_\Omega) E_1(f;\Omega)_{C(\Omega)}. \tag{1}$$

Здесь $\|P\|_\Omega = \sup\{\|Pf|_\Omega\|_{C(\Omega)} : \|f\|_{C(\Omega)} \leqslant 1\}$ есть $C(\Omega)$-операторная норма P.

Первое неравенство в (1) является очевидным, поскольку $Pf \in \Pi_1(\mathbb{R}^n)$. Второе неравенство известно в литературе под названием *леммы Лебега, или неравенства Лебега* (см., например, [1]). Это неравенство показывает, что если норма $\|P\|_\Omega$ невелика, то нет существенной потери точности при замене многочлена наилучшего приближения для f на значение Pf, *которое линейно зависит от функции* f. Этот подход даёт простой инструмент для «линеаризации» элемента наилучшего приближения в различных задачах анализа и вычислительной математики, связанных с приближением непрерывных функций на подмножествах $\mathbb{R}^n$.

В приложениях важны как верхние, так и нижние границы операторных норм интерполяционных проекторов. Обычно верхние оценки минимальных норм проекторов получают, рассматривая проекторы некоторого специального типа. Техника получения нижних оценок минимальных норм принципиально иная — в этом случае необходимо доказать нижнюю оценку для произвольного проектора.

Построение и оценка интерполяционных проекторов является классической темой в теории приближений и её приложениях. Эти задачи рассматривались во многих статьях и монографиях (см., например, [2—9]).

Пусть $K = \mathrm{conv}(\Omega)$ имеет ненулевой объём. В настоящей статье мы доказываем, что существует интерполяционный проектор, норма которого не превосходит $n + 1$ (теорема 1). С другой стороны (теорема 6), для любого интерполяционного проектора $P : C(\Omega) \to \Pi_1(\mathbb{R}^n)$

$$\|P\|_\Omega \geqslant \chi_n^{-1}\left(\frac{\mathrm{vol}(K)}{\mathrm{simp_n}(\Omega)}\right). \tag{2}$$

Здесь χ_n — стандартизованный многочлен Лежандра степени n, $\mathrm{simp_n}(\Omega)$ — максимальный объём симплекса с вершинами в Ω.

Ключевым моментом доказательства неравенства (2) является геометрическая характеризация многочленов Лежандра, приведённая в теореме 3. Эта теорема утверждает, что для $\gamma \geqslant 1$ объём множества

$$\left\{x \in \mathbb{R}^n : \sum_{j=1}^n |x_j| + \left|1 - \sum_{j=1}^n x_j\right| \leqslant \gamma\right\}$$

равен $\chi_n(\gamma)/n!$. Указанное свойство позволяет доказать оценку (2) для случая, когда Ω есть выпуклое тело, т. е. $\Omega = K$. В дальнейшем (2) распространяется на произвольный компакт Ω, для которого $\mathrm{simp_n}(\Omega) > 0$.

Когда Ω является n-мерным кубом или шаром, неравенство (2) даёт возможность получить оценки вида $\|P\|_\Omega \geqslant c\sqrt{n}$. Эти оценки неулучшаемы по порядку размерности для всех n, если Ω — n-мерный шар, и по крайней мере для тех n, когда существует матрица Адамара порядка $n+1$, если Ω — n-мерный куб.

Опишем содержание статьи по разделам. В разделе 1 мы приводим основные обозначения, определения и предварительную информацию. Раздел 2 содержит верхние оценки минимального коэффициента поглощения симплексом компакта Ω, а также минимальной нормы проектора

при линейной интерполяции на Ω. В разделе 3 доказывается упомянутая выше теорема 3. В разделе 4 мы доказываем неравенство (2) для $\Omega = K$. Здесь же приводятся некоторые явные нижние границы минимальной нормы проектора, когда K является n-мерным шаром или n-мерным кубом. Раздел 5 содержит неравенства рассматриваемого типа при интерполяции линейными функциями на произвольном компактном множестве $\Omega \subset \mathbb{R}^n$. Наконец, раздел 6 содержит заключительные замечания и открытые вопросы.

Расширенная версия статьи опубликована на сайте arXiv.org [10] (см. также обзор [11]).

1. Основные определения и предварительные сведения

В этой статье $n \in \mathbb{N}$. Для $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ через $\|x\|$ обозначается обычная евклидова норма:

$$\|x\| = \sqrt{\langle x, x\rangle} = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2}.$$

Здесь и далее для $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n) \in \mathbb{R}^n$ через $\langle x, y\rangle$ обозначается стандартное скалярное произведение в $\mathbb{R}^n$:

$$\langle x, y\rangle = x_1 y_1 + \ldots + x_n y_n.$$

Ниже $Q_n = [0,1]^n$ — единичный n-мерный куб, B_n — единичный n-мерный шар, задаваемый неравенством $\|x\| \leqslant 1$. Запись $L(n) \asymp M(n)$ означает, что существуют абсолютные константы $c_1, c_2 > 0$, такие что $c_1 M(n) \leqslant L(n) \leqslant c_2 M(n)$.

Пусть K — *выпуклое тело в* $\mathbb{R}^n$, т. е. компактное выпуклое подмножество $\mathbb{R}^n$ с непустой внутренностью. Под σK понимается гомотетическая копия K с центром гомотетии в центре тяжести K и коэффициентом гомотетии σ. Символом $\mathrm{vol}(K)$ обозначается объём K. Если K — выпуклый многогранник, то через $\mathrm{ver}(K)$ мы обозначаем множество вершин K.

Пусть Ω — ограниченное замкнутое подмножество $\mathbb{R}^n$. Под $C(\Omega)$ понимается пространство непрерывных функций $f : \Omega \to \mathbb{R}$ с равномерной нормой

$$\|f\|_{C(\Omega)} := \max_{x \in \Omega} |f(x)|.$$

Ниже часто предполагается, что $\mathrm{vol}(\mathrm{conv}(\Omega)) > 0$. Это условие эквивалентно тому, что существует n-мерный невырожденный симплекс с вершинами в Ω.

Через $\Pi_1(\mathbb{R}^n)$ будем обозначать пространство многочленов от n переменных степени $\leqslant 1$.

Пусть S — невырожденный симплекс в $\mathbb{R}^n$. Через $\xi(\Omega; S)$ обозначается минимальное $\sigma \geqslant 1$, для которого $\Omega \subset \sigma S$. По нашей терминологии, $\xi(\Omega; S)$ называется *коэффициентом поглощения множества* Ω *симплексом* S. Равенство $\xi(\Omega; S) = 1$ эквивалентно включению $\Omega \subset S$. Заметим, что $\xi(\mathrm{conv}(\Omega); S) = \xi(\Omega; S)$. Обозначим

$$\xi_n(\Omega) = \min\{\xi(\Omega; S) : S - n\text{-мерный симплекс}, \mathrm{ver}(S) \subset \Omega, \mathrm{vol}(S) \neq 0\}.$$

Пусть S — невырожденный симплекс в $\mathbb{R}^n$ с вершинами $x^{(j)} = \left(x_1^{(j)}, \ldots, x_n^{(j)}\right)$, $1 \leqslant j \leqslant n+1$. *Матрицей вершин* этого симплекса называется матрица

$$\mathbf{A} = \begin{pmatrix} x_1^{(1)} & \ldots & x_n^{(1)} & 1 \\ x_1^{(2)} & \ldots & x_n^{(2)} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(n+1)} & \ldots & x_n^{(n+1)} & 1 \end{pmatrix}.$$

Эта матрица является невырожденной, причём

$$\mathrm{vol}(S) = \frac{|\det(\mathbf{A})|}{n!}. \tag{3}$$

Пусть $\mathbf{A}^{-1} = (l_{ij})$. Линейные многочлены

$$\lambda_j(x) = l_{1j}x_1 + \ldots + l_{nj}x_n + l_{n+1,j}, \quad j = 1, \ldots, n+1,$$

коэффициенты которых составляют столбцы матрицы $\mathbf{A}^{-1}$, называются *базисными многочленами Лагранжа симплекса* S. Справедливы равенства $\lambda_j\left(x^{(k)}\right) = \delta_j^k$, где δ_j^k — δ-символ Кронекера. Заметим также, что

$$\lambda_j(x) = \frac{\Delta_j(x)}{\Delta}. \tag{4}$$

Здесь $\Delta = \det(\mathbf{A})$, а определитель $\Delta_j(x)$ получается из Δ путём замены j-й строки на строку $(x_1 \ \ldots \ x_n \ 1)$.

Для произвольной точки $x \in \mathbb{R}^n$

$$x = \sum_{j=1}^{n+1} \lambda_j(x) x^{(j)}, \quad \sum_{j=1}^{n+1} \lambda_j(x) = 1.$$

Поэтому $\lambda_j(x)$ являются *барицентрическими координатами точки* x *относительно симплекса* S. Уравнения $\lambda_j(x) = 0$ задают $(n-1)$-мерные гиперплоскости, содержащие грани S. Следовательно,

$$S = \left\{x \in \mathbb{R}^n : \ \lambda_j(x) \geqslant 0, \ j = 1, \ldots, n+1\right\}.$$

Коэффициент поглощения $\xi(\Omega; S)$ вычисляется по формуле

$$\xi(\Omega; S) = (n+1) \max_{1 \leqslant j \leqslant n+1} \max_{x \in \Omega} (-\lambda_j(x)) + 1. \tag{5}$$

Для выпуклого Ω равенство (5) доказывается в [12]; в общем случае доказательство проводится по той же схеме.

Совокупность из $n+1$ точек Ω называется *допустимым набором узлов* для интерполяции с помощью $\Pi_1(\mathbb{R}^n)$, если симплекс с вершинами в этих точках является невырожденным. Ниже рассматриваются лишь допустимые наборы узлов и те множества Ω, каждое из которых содержит такой набор.

Пусть $x^{(j)} \in \Omega$, $1 \leqslant j \leqslant n+1$, являются вершинами невырожденного симплекса S. *Интерполяционный проектор* $P : C(\Omega) \to \Pi_1(\mathbb{R}^n)$ *с узлами* $x^{(j)}$ определяется равенствами $Pf\left(x^{(j)}\right) = f_j = f\left(x^{(j)}\right)$. Будем говорить, что проектор P и симплекс S *соответствуют друг другу* и применять обозначения P_S и S_P.

Для проектора $P = P_S$ справедлив аналог интерполяционной формулы Лагранжа:

$$Pf(x) = \sum_{j=1}^{n+1} f\left(x^{(j)}\right) \lambda_j(x), \tag{6}$$

где λ_j — базисные многочлены Лагранжа симплекса $S = S_P$. Обозначим через $\|P\|_\Omega$ норму P как оператора из $C(\Omega)$ в $C(\Omega)$. Из (6) имеем:

$$\begin{aligned} \|P\|_\Omega &= \sup_{\|f\|_{C(\Omega)}=1} \|Pf\|_{C(\Omega)} = \sup_{-1 \leqslant f_j \leqslant 1} \max_{x \in \Omega} \left| \sum_{j=1}^{n+1} f_j \lambda_j(x) \right| \\ &= \max_{x \in \Omega} \sup_{-1 \leqslant f_j \leqslant 1} \left| \sum_{j=1}^{n+1} f_j \lambda_j(x) \right| = \max_{x \in \Omega} \sup_{-1 \leqslant f_j \leqslant 1} \sum_{j=1}^{n+1} f_j \lambda_j(x). \end{aligned}$$

Выражение $\sum f_j\lambda_j(x)$ линейно по x и $f_1,\dots,f_{n+1}$, значит,

$$\max_{x\in\Omega}\sup_{-1\leqslant f_j\leqslant 1}\sum_{j=1}^{n+1} f_j\lambda_j(x)=\max_{x\in\Omega}\max_{f_j=\pm1}\sum_{j=1}^{n+1} f_j\lambda_j(x)=\max_{x\in\Omega}\sum_{j=1}^{n+1}|\lambda_j(x)|.$$

Таким образом,

$$\|P\|_\Omega=\max_{x\in\Omega}\sum_{j=1}^{n+1}|\lambda_j(x)|=\max\left\{\sum_{j=1}^{n+1}|\beta_j|:\ \sum_{j=1}^{n+1}\beta_j=1,\ x=\sum_{j=1}^{n+1}\beta_j x^{(j)}\in\Omega\right\}. \tag{7}$$

Равенство (7) выражает норму проектора P через барицентрические координаты точек множества Ω относительно симплекса с вершинами в узлах интерполяции $x^{(j)}$. Если Ω — выпуклый многогранник в $\mathbb{R}^n$ (например, куб), то верны и более простые равенства

$$\|P\|_\Omega=\max_{x\in\operatorname{ver}(\Omega)}\sum_{j=1}^{n+1}|\lambda_j(x)|=\max\left\{\sum_{j=1}^{n+1}|\beta_j|:\ \sum_{j=1}^{n+1}\beta_j=1,\ x=\sum_{j=1}^{n+1}\beta_j x^{(j)}\in\operatorname{ver}(\Omega)\right\}.$$

Обозначим через $\theta_n(\Omega)$ минимальное значение $\|P_S\|_\Omega$ по всем n-мерным невырожденным симплексам S с вершинами в Ω. Интерполяционный проектор $P:C(\Omega)\to\Pi_1(\mathbb{R}^n)$ называется *минимальным,* если $\|P\|_\Omega=\theta_n(\Omega)$.

В [13] доказано, что для любого интерполяционного проектора $P:C(\Omega)\to\Pi_1(\mathbb{R}^n)$ и соответствующего симплекса S выполняются неравенства

$$\frac{n+1}{2n}\Big(\|P\|_\Omega-1\Big)+1\leqslant\xi(\Omega;S)\leqslant\frac{n+1}{2}\Big(\|P\|_\Omega-1\Big)+1. \tag{8}$$

Благодаря (8) имеем соотношения

$$\frac{n+1}{2n}\Big(\theta_n(\Omega)-1\Big)+1\leqslant\xi_n(\Omega)\leqslant\frac{n+1}{2}\Big(\theta_n(\Omega)-1\Big)+1. \tag{9}$$

Очевидно, что если проектор P удовлетворяет равенству

$$\xi_n(\Omega)=\frac{n+1}{2}\Big(\|P\|_\Omega-1\Big)+1, \tag{10}$$

то P является минимальным и правое соотношение в (9) становится равенством.

Иногда мы будем рассматривать случай, когда $n+1$ есть *число Адамара.* По определению, это означает, что существует матрица Адамара порядка $n+1$. Напомним, что *матрицей Адамара порядка m* называется квадратная матрица $\mathbf{H}$ с элементами 1 или -1, для которой

$$\mathbf{H}^{-1}=\frac{1}{m}\mathbf{H}^T.$$

Это равенство означает, что строки $\mathbf{H}$ попарно ортогональны относительно стандартного скалярного произведения в $\mathbb{R}^m$. Порядок матрицы Адамара равен 1, 2 или кратен 4 (см. [14]). До сих пор неизвестно, существует ли матрица Адамара любого порядка вида $m=4k$. Это одна из самых давних открытых проблем в математике. Порядки ниже 1500, кратные 4, для которых матрицы Адамара пока не известны, суть 668, 716, 892, 956, 1132, 1244, 1388 и 1436 (см., например, [15, 16]).

Обозначим через h_n максимальное значение определителя порядка n с элементами 0 или 1. Пусть ν_n есть максимальный объём n-мерного симплекса, содержащегося в Q_n. Эти числа связаны равенством $h_n=n!\nu_n$ (см. [17]). Для $n>1$

$$\frac{1}{2}\left(1-\frac{\log(4/3)}{\log n}\right)n\log n<\log(2^{n-1}h_{n-1})\leqslant\frac{1}{2}\,n\log n. \tag{11}$$

Правое неравенство в (11) доказано Адамаром [18]; левое неравенство установлено Клементсом и Линдстрёмом [19]. Следовательно, для любого n

$$\left(\frac{3}{4}\right)^{(n+1)/2} \frac{(n+1)^{(n+1)/2}}{2^n} < h_n \leqslant \frac{(n+1)^{(n+1)/2}}{2^n}, \tag{12}$$

$$\left(\frac{3}{4}\right)^{(n+1)/2} \frac{(n+1)^{(n+1)/2}}{2^n n!} < \nu_n \leqslant \frac{(n+1)^{(n+1)/2}}{2^n n!}. \tag{13}$$

Правое равенство в каждом из соотношений выполняется тогда и только тогда, когда $n+1$ есть число Адамара [17]. Для многих n точные значения ν_n и h_n известны. Первые 12 чисел ν_n равны

$$\nu_1 = 1, \quad \nu_2 = \frac{1}{2}, \quad \nu_3 = \frac{1}{3}, \quad \nu_4 = \frac{1}{8}, \quad \nu_5 = \frac{1}{24}, \quad \nu_6 = \frac{1}{80}, \quad \nu_7 = \frac{2}{315},$$

$$\nu_8 = \frac{1}{720}, \quad \nu_9 = \frac{1}{2520}, \quad \nu_{10} = \frac{1}{11340}, \quad \nu_{11} = \frac{9}{246400}, \quad \nu_{12} = \frac{3}{394240}.$$

Через $\varkappa_n$ обозначим объём единичного шара B_n, через σ_n — объём правильного n-мерного симплекса, вписанного в B_n. Числа $\varkappa_n$ и σ_n известны для всех n. Именно (см. [12, 20]),

$$\varkappa_n = \frac{\pi^{n/2}}{\Gamma\left(n/2+1\right)}, \qquad \sigma_n = \frac{1}{n!}\sqrt{n+1}\left(\frac{n+1}{n}\right)^{n/2}, \tag{14}$$

$$\varkappa_{2k} = \frac{\pi^k}{k!}, \qquad \varkappa_{2k+1} = \frac{2^{k+1}\pi^k}{(2k+1)!!} = \frac{2(k!)(4\pi)^k}{(2k+1)!}. \tag{15}$$

Под $\mathrm{simp_n}(\Omega)$ будем понимать максимальный объём невырожденного n-мерного симплекса с вершинами в Ω. Очевидно, $\mathrm{simp_n}(Q_n) = \nu_n$. Правильный симплекс, вписанный в n-мерный шар, имеет максимальный объём из всех симплексов, содержащихся в этом шаре, причём других симплексов с этим свойством нет (см. [21—23]). Значит, $\mathrm{simp_n}(B_n) = \sigma_n$.

2. Неравенства $\xi_n(\Omega) \leqslant n+2$, $\theta_n(\Omega) \leqslant n+1$

Теорема 1. *Пусть Ω — компакт в $\mathbb{R}^n$, для которого $\mathrm{vol}(\mathrm{conv}(\Omega)) > 0$, S — симплекс с вершинами в Ω, имеющий максимальный объём. Тогда*

$$\xi(\Omega; S) \leqslant n+2, \quad \|P_S\|_\Omega \leqslant n+1. \tag{16}$$

Доказательство. Пусть K — произвольное выпуклое тело в $\mathbb{R}^n$. Используя чисто геометрический подход, Лассак [24] показал, что для любого симплекса S' максимального объёма в K справедливы включения

$$S' \subset K \subset (n+2)S'. \tag{17}$$

Отсюда следует, что $\xi(K; S') \leqslant n+2$. Применим это неравенство к выпуклому телу $K = \mathrm{conv}(\Omega)$. В качестве S' возьмём симплекс S из условия теоремы. Поскольку $\mathrm{simp_n}(K) = \mathrm{simp_n}(\Omega)$ (см. ниже доказательство теоремы 6), S является симплексом максимального объёма и в K. Из включения $\Omega \subset K$ имеем $\xi(\Omega; S) \leqslant \xi(K; S) \leqslant n+2$.

Заметим, что левое неравенство в (16) легко следует также из формулы (5). Действительно, так как S имеет максимальный объём из всех симплексов с вершинами в Ω, то $|\Delta_j(x)| \leqslant |\Delta|$ для любых $j = 1, \ldots, n+1$ и $x \in \Omega$. (Это сразу вытекает из (3).) Благодаря (4)

$$-\lambda_j(x) \leqslant |\lambda_j(x)| = \frac{|\Delta_j(x)|}{|\Delta|} \leqslant 1, \quad x \in \Omega. \tag{18}$$

Здесь λ_j — базисные многочлены Лагранжа для S. По формуле (5),

$$\xi(\Omega;S) = (n+1)\max_{1\leqslant j\leqslant n+1}\max_{x\in\Omega}(-\lambda_j(x)) + 1 \leqslant n+2,$$

что доказывает левое неравенство в (16).

Правое неравенство в (16) следует из (7). Поскольку $|\lambda_j(x)| \leqslant 1$, имеем

$$\|P_S\|_\Omega = \max_{x\in\Omega}\sum_{j=1}^{n+1}|\lambda_j(x)| \leqslant n+1.$$

Теорема доказана. □

Теорема 2. *Для произвольного компакта $\Omega \subset \mathbb{R}^n$, выпуклая оболочка которого имеет ненулевой объём,*

$$\xi_n(\Omega) \leqslant n+2, \qquad \theta_n(\Omega) \leqslant n+1. \tag{19}$$

Сразу следует из теоремы 1.

3. Многочлены Лежандра и мера множества $E_{n,\gamma}$

Стандартизованным многочленом Лежандра степени n называется многочлен

$$\chi_n(t) = \frac{1}{2^n n!}\left[(t^2-1)^n\right]^{(n)}, \quad t\in\mathbb{R}$$

(формула Родрига). По поводу свойств χ_n см., например, [25, 26]. Многочлены Лежандра ортогональны на отрезке $[-1,1]$ с весом $w(t)\equiv 1$. Первые многочлены Лежандра имеют вид

$$\chi_0(t)=1, \quad \chi_1(t)=t, \quad \chi_2(t)=\frac{1}{2}\left(3t^2-1\right), \quad \chi_3(t)=\frac{1}{2}\left(5t^3-3t\right),$$

$$\chi_4(t)=\frac{1}{8}\left(35t^4-30t^2+3\right), \quad \chi_5(t)=\frac{1}{8}\left(63t^5-70t^3+15t\right).$$

Справедливо рекуррентное соотношение

$$\chi_{n+1}(t) = \tfrac{2n+1}{n+1}\,t\cdot\chi_n(t) - \tfrac{n}{n+1}\,\chi_{n-1}(t). \tag{20}$$

Отсюда, в частности, $\chi_n(1)=1$. Напомним также, что если $n\geqslant 1$, то $\chi_n(t)$ возрастает при $t\geqslant 1$. Эти свойства легко получаются и из приводимой ниже формулы (22). Через χ_n^{-1} обозначим функцию, обратную к χ_n на полуоси $[1,+\infty)$.

Одним из ключевых утверждений нашего подхода к оцениванию интерполяционных проекторов является формулируемая ниже теорема 3. Эта теорема обнаруживает довольно неожиданные связи между многочленами Лежандра и объёмами выпуклых многогранников.

Для $\gamma\geqslant 1$ определим множество $E_{n,\gamma}$ равенством

$$E_{n,\gamma} = \left\{x\in\mathbb{R}^n : \sum_{j=1}^n|x_j| + \left|1-\sum_{j=1}^n x_j\right| \leqslant \gamma\right\}. \tag{21}$$

Теорема 3. *Справедливы соотношения*

$$\mathrm{mes}_n(E_{n,\gamma}) = \frac{1}{2^n n!}\sum_{i=0}^n\binom{n}{i}^2(\gamma-1)^{n-i}(\gamma+1)^i = \frac{\chi_n(\gamma)}{n!}. \tag{22}$$

Это утверждение доказано автором в 2003 г. и опубликовано в статье, которая сегодня практически недоступна широкой аудитории. Для удобства читателя мы приводим доказательство ниже.

Доказательство. Сначала докажем левое равенство в (22). Пусть

$$E^{(1)} = \{x \in E_{n,\gamma} : \sum x_i > 1\}, \quad E^{(2)} = \{x \in E_{n,\gamma} : \sum x_i \leqslant 1\}.$$

Получим явные формулы для объёмов $m_1 = \mathrm{mes_n}(E^{(1)})$ и $m_2 = \mathrm{mes_n}(E^{(2)})$.

Зафиксируем k, $1 \leqslant k \leqslant n$, и рассмотрим непустое подмножество $G \subset E^{(1)}$, состоящее из всех $x = (x_1, \ldots, x_n)$, для которых $x_1, \ldots, x_k \geqslant 0$ и $x_{k+1}, \ldots, x_n < 0$. Пусть $y_i = x_i$ для $i = 1, \ldots, k$ и $y_i = -x_i$ для $i = k+1, \ldots, n$. Положим $y = (y_1, \ldots, y_n)$, тогда

$$G = \left\{ y : 1 + y_{k+1} + \ldots + y_n \leqslant y_1 + \ldots + y_k \leqslant \frac{\gamma+1}{2},\ y_i \geqslant 0 \right\}.$$

Поэтому

$$\begin{aligned} \mathrm{mes_n}(G) \;=\; & \int_1^{\alpha} dy_1 \int_1^{\alpha-y_1} dy_2 \ldots \int_1^{\alpha-y_1-\ldots-y_{k-1}} dy_k \\ & \int_0^{y_1+\ldots+y_k-1} dy_{k+1} \int_0^{y_1+\ldots+y_k-1-y_{k+1}} dy_{k+2} \ldots \int_0^{y_1+\ldots+y_k-1-y_{k+1}-\ldots-y_{n-1}} dy_n. \end{aligned}$$

В нашем доказательстве $\alpha = (\gamma+1)/2$. Если $b > 0$, то

$$\int_0^{b} dz_1 \int_0^{b-z_1} dz_2 \ldots \int_0^{b-z_1-\ldots-z_{l-1}} dz_l = \frac{b^l}{l!}.$$

Значит,

$$\begin{aligned} \mathrm{mes_n}(G) \;&=\; \int_1^{\alpha} dy_1 \int_1^{\alpha-y_1} dy_2 \ldots \int_1^{\alpha-y_1-\ldots-y_{k-1}} \frac{1}{(n-k)!}(y_1+\ldots+y_k-1)^{n-k}\, dy_k \\ &=\; \left(\int_{y_1+\ldots+y_k \leqslant \alpha} - \int_{y_1+\ldots+y_k \leqslant 1} \right) \frac{1}{(n-k)!}(y_1+\ldots+y_k-1)^{n-k}\, dy_1 \ldots dy_k \\ &=\; J_1 - J_2. \end{aligned}$$

Первый интеграл равен

$$J_1 = \sum_{j=1}^{k} (-1)^{j+1} \frac{(\alpha-1)^{n-k+j}}{(n-k+j)!} \frac{\alpha^{k-j}}{(k-j)!} + \frac{(-1)^{n+k}}{n!}.$$

Значение J_2 получается из этого выражения, если вместо α взять 1. Следовательно,

$$\mathrm{mes_n}(G) = J_1 - J_2 = \sum_{j=1}^{k} (-1)^{j+1} \frac{(\alpha-1)^{n-k+j}}{(n-k+j)!} \frac{\alpha^{k-j}}{(k-j)!} = \frac{(-1)^{k+1}}{n!} \sum_{i=0}^{k-1} \binom{n}{i} (\alpha-1)^{n-i} (-\alpha)^i. \tag{23}$$

Множество $E^{(1)}$ является объединением всех таких множеств G с различными $k = 1, \ldots, n$. Поэтому мера $E^{(1)}$ равна

$$m_1 = \sum_{k=1}^{n} \binom{n}{k} \frac{(-1)^{k+1}}{n!} \sum_{i=0}^{k-1} \binom{n}{i} (\alpha - 1)^{n-i} (-\alpha)^i.$$

Меняя порядок суммирования и используя тождество

$$\sum_{k=0}^{i} (-1)^k \binom{n}{k} = (-1)^i \binom{n-1}{i} \tag{24}$$

(см., например, [27]), получим

$$m_1 = \frac{1}{n!} \sum_{i=0}^{n-1} \binom{n}{i} (\alpha - 1)^{n-i} (-\alpha)^i \sum_{k=0}^{i} (-1)^k \binom{n}{k} = \frac{1}{n!} \sum_{i=0}^{n-1} \binom{n}{i} \binom{n-1}{i} (\alpha - 1)^{n-i} \alpha^i. \tag{25}$$

Теперь перейдём к $E^{(2)}$. Прежде всего заметим, что $E^{(2)}$ содержит область $S = \{x_i \geqslant 0, \sum x_i \leqslant 1\}$, мера которой равна $1/n!$. Далее, зафиксируем $k \in \{1, \ldots, n\}$ и рассмотрим подмножество $G' \subset E^{(2)}$, соответствующее неравенствам $x_1, \ldots, x_k < 0$; $x_{k+1}, \ldots, x_n \geqslant 0$. Положим $y_1 = -x_1, \ldots, y_k = -x_k$; $y_{k+1} = x_{k+1}, \ldots, y_n = x_n$. Тогда

$$G' = \{y : y_{k+1} + \ldots + y_n \leqslant 1 + y_1 + \ldots + y_k \leqslant \frac{\gamma - 1}{2},\ y_i \geqslant 0\}.$$

Обозначим $\beta = (\gamma - 1)/2$. Имеют место равенства:

$$\begin{aligned}
\mathrm{mes_n}(G') &= \int_0^{\beta} dy_1 \int_0^{\beta - y_1} dy_2 \ldots \int_0^{\beta - y_1 - \ldots - y_{k-1}} dy_k \\
&\quad \int_0^{1 + y_1 + \ldots + y_k} dy_{k+1} \int_0^{1 + y_1 + \ldots + y_k - y_{k+1}} dy_{k+2} \ldots \int_0^{1 + y_1 + \ldots + y_k - y_{k+1} - \ldots - y_{n-1}} dy_n \\
&= \int_0^{\beta} dy_1 \int_0^{\beta - y_1} dy_2 \ldots \int_0^{\beta - y_1 - \ldots - y_{k-1}} \frac{(1 + y_1 + \ldots + y_k)^{n-k}}{(n-k)!} dy_k \\
&= \left[\sum_{j=0}^{k-1} (-1)^{k-1-j} \frac{(1+\beta)^{n-j} \beta^j}{(n-j)! j!} \right] + \frac{(-1)^k}{n!} = \frac{(-1)^{k+1}}{n!} \left(\left[\sum_{j=0}^{k-1} \binom{n}{j} (1+\beta)^{n-j} (-\beta)^j \right] - 1 \right).
\end{aligned}$$

Множество $E^{(2)}$ есть объединение всех таких множеств G', соответствующих различным $k = 1, \ldots, n$, а также симплекса S. Значит,

$$m_2 = \mathrm{mes_n}(E^{(2)}) = \frac{1}{n!} \left(\left\{ \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} \left(\left[\sum_{j=0}^{k-1} \binom{n}{j} (1+\beta)^{n-j} (-\beta)^j \right] - 1 \right) \right\} + 1 \right).$$

Заметим, что $1 + \beta = (\gamma + 1)/2 = \alpha$ и $\beta = (\gamma - 1)/2 = \alpha - 1$. Во внутренней сумме сделаем замену $i = n - j$ и учтём, что

$$\sum_{k=0}^{n} (-1)^k \binom{n}{k} = \sum_{k=1}^{n} (-1)^k \binom{n}{k} + 1 = 0.$$

Мы получим следующее:

$$\begin{aligned} m_2 &= \frac{1}{n!}\left(1+\sum_{k=1}^{n}(-1)^{k+1}\binom{n}{k}\left((-1)^n\sum_{i=n-k+1}^{n}\binom{n}{i}(\alpha-1)^{n-i}(-\alpha)^i-1\right)\right) \\ &= \frac{(-1)^n}{n!}\sum_{k=1}^{n}(-1)^{k+1}\binom{n}{k}\sum_{i=n-k+1}^{n}\binom{n}{i}(\alpha-1)^{n-i}(-\alpha)^i. \end{aligned}$$

Меняя порядок суммирования, придём к равенству

$$m_2=\frac{(-1)^n}{n!}\sum_{i=1}^{n}\binom{n}{i}(\alpha-1)^{n-i}(-\alpha)^i\sum_{k=n+1-i}^{n}(-1)^{k+1}\binom{n}{k}.$$

Применяя (24), запишем

$$\sum_{k=n+1-i}^{n}(-1)^{k+1}\binom{n}{k}=\sum_{k=n+1-i}^{n}(-1)^{k+1}\binom{n}{n-k}=\sum_{j=0}^{i-1}(-1)^{n-j+1}\binom{n}{j}=(-1)^{n+i}\binom{n-1}{i-1}.$$

Следовательно,

$$m_2=\frac{1}{n!}\sum_{i=1}^{n}\binom{n}{i}\binom{n-1}{i-1}(\alpha-1)^{n-i}\alpha^i. \tag{26}$$

Равенства (25) и (26) означают, что

$$\begin{aligned} \mathrm{mes_n}(E_{n,\gamma}) &= m_1+m_2 \\ &= \frac{1}{n!}\sum_{i=0}^{n-1}\binom{n}{i}\binom{n-1}{i}(\alpha-1)^{n-i}\alpha^i+\frac{1}{n!}\sum_{i=1}^{n}\binom{n}{i}\binom{n-1}{i-1}(\alpha-1)^{n-i}\alpha^i \\ &= \frac{1}{n!}\sum_{i=1}^{n-1}\binom{n}{i}\left(\binom{n-1}{i}+\binom{n-1}{i-1}\right)(\alpha-1)^{n-i}\alpha^i+\frac{1}{n!}\left((\alpha-1)^n+\alpha^n\right) \\ &= \frac{1}{n!}\sum_{i=0}^{n}\binom{n}{i}^2(\alpha-1)^{n-i}\alpha^i=\frac{1}{2^n n!}\sum_{i=0}^{n}\binom{n}{i}^2(\gamma-1)^{n-i}(\gamma+1)^i. \end{aligned}$$

Мы приняли во внимание, что

$$\binom{n-1}{i}+\binom{n-1}{i-1}=\binom{n}{i}.$$

Левое равенство в (22) доказано.

Правое равенство в (22) следует из тождества

$$\sum_{i=0}^{n}\binom{n}{i}^2 t^i=(1-t)^n\chi_n\left(\frac{1+t}{1-t}\right)$$

(см. [27]). Положим $t=(\gamma-1)/(\gamma+1)$, тогда

$$(1-t)^n=2^n(\gamma+1)^{-n},\quad \frac{1+t}{1-t}=\gamma.$$

Таким образом,

$$\begin{aligned} \mathrm{mes_n}(E_{n,\gamma}) &= \frac{1}{2^n n!}\sum_{i=0}^{n}\binom{n}{i}^2(\gamma-1)^{n-i}(\gamma+1)^i=\frac{1}{2^n n!}\sum_{i=0}^{n}\binom{n}{i}^2(\gamma+1)^{n-i} \\ &= \frac{1}{2^n n!}(\gamma+1)^n\sum_{i=0}^{n}\binom{n}{i}^2\left(\frac{\gamma-1}{\gamma+1}\right)^i=\frac{\chi_n(\gamma)}{n!}. \end{aligned}$$

Теорема 3 полностью доказана. □

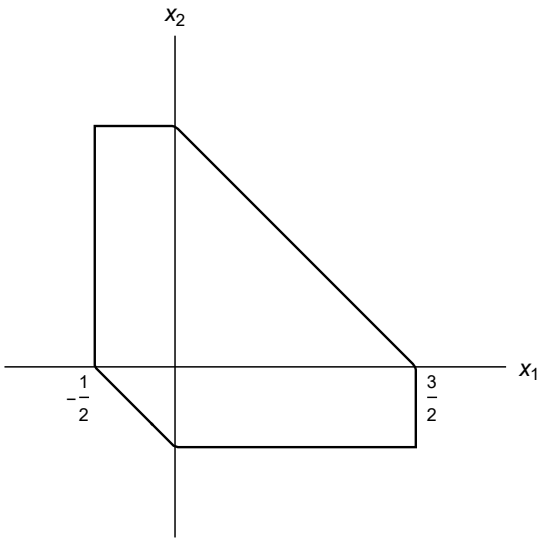


Fig. 1. The set $E_{2,2}$

Рис. 1. Множество $E_{2,2}$

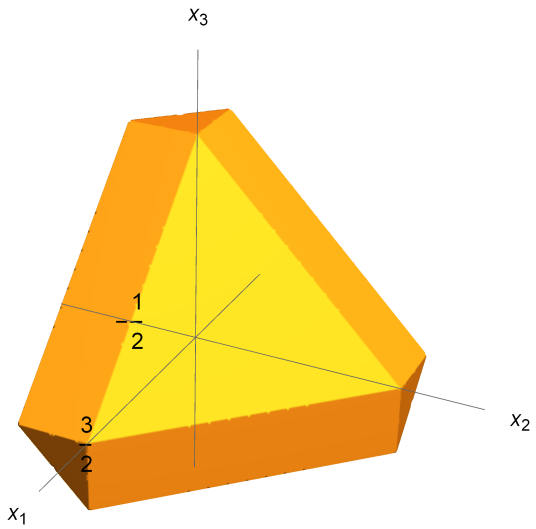


Fig. 2. The set $E_{3,2}$

Рис. 2. Множество $E_{3,2}$

Приведём простые примеры. Возьмём $\gamma = 2$. Множество $E_{1,2} = \{x \in \mathbb{R} : |x|+|1-x| \leqslant 2\}$ есть отрезок $[-1/2, 3/2]$ длины $\mathrm{mes}_1(E_{1,2}) = \chi_1(2)/1! = 2$. Множество $E_{2,2} = \{x \in \mathbb{R}^2 : |x_1| + |x_2| + |1 - x_1 - x_2| \leqslant 2\}$ — шестиугольник на плоскости с площадью $\mathrm{mes}_2(E_{2,2}) = \chi_2(2)/2! = 11/4$. См. рис. 1.

Трёхмерная область $E_{3,2} = \{x \in \mathbb{R}^3 : |x_1| + |x_2| + |x_3| + |1 - x_1 - x_2 - x_3| \leqslant 2\}$ изображена на рис. 2. Объём этого многогранника $\mathrm{mes}_3(E_{3,2}) = \chi_3(2)/3! = 17/6$.

Отметим интересную открытую проблему, связанную с равенством (22). Наряду с формулой Родрига и другими известными соотношениями, это равенство даёт характеризацию многочленов Лежандра — их можно определить и через объёмы выпуклых многогранников. Именно, для $t \geqslant 1$

$$\chi_n(t) = n!\,\mathrm{mes}_n(E_{n,t}), \tag{27}$$

где $E_{n,t}$ — многогранник, задаваемый соотношением (21). Возникает вопрос об аналогах (27) для других классов ортогональных многочленов, таких как многочлены Чебышёва или, в более общем слу-

чае, многочлены Якоби. *Является ли равенство* (27) *проявлением более общей закономерности?* Автор будет благодарен за любую информацию по этому вопросу.

Сделаем ещё одно замечание. Из (27) и (20) следует

$$\mathrm{mes}_{n+1}(E_{n+1,t}) = \frac{2n+1}{(n+1)^2}\, t\, \mathrm{mes}_n(E_{n,t}) - \frac{1}{(n+1)^2}\mathrm{mes}_{n-1}(E_{n-1,t}).$$

Прямое установление этого рекуррентного соотношения для мер множеств $E_{n,t}$ могло бы дать новое доказательство теоремы 3.

4. Неравенство $\theta_n(K) \geqslant \chi_n^{-1}\left(\dfrac{\mathrm{vol}(\mathrm{conv}(K))}{\mathrm{simp_n}(K)}\right)$

Основываясь на теореме 3, получим нижние оценки для нормы интерполяционного проектора на произвольном выпуклом теле $K \subset \mathbb{R}^n$.

Теорема 4. *Пусть K — выпуклое тело в $\mathbb{R}^n$, $P : C(K) \to \Pi_1(\mathbb{R}^n)$ — произвольный интерполяционный проектор. Тогда для соответствующих симплекса $S_P \subset K$ и матрицы узлов* **A** *выполняются соотношения*

$$\|P\|_K \geqslant \chi_n^{-1}\left(\frac{n!\mathrm{vol}(K)}{|\det(\mathbf{A})|}\right) = \chi_n^{-1}\left(\frac{\mathrm{vol}(K)}{\mathrm{vol}(S_P)}\right). \tag{28}$$

Доказательство. Для каждого $i = 1, \ldots, n$ вычтем из i-й строки матрицы **A** её $(n+1)$-ю строку. Обозначим через **B** квадратную подматрицу порядка n, стоящую в первых n строках и столбцах получившейся матрицы. Справедливы равенства

$$|\det(\mathbf{B})| = |\det(\mathbf{A})| = n!\mathrm{vol}(S_P) \leqslant n!\mathrm{vol}(K),$$

поэтому

$$\frac{|\det(\mathbf{B})|}{n!\mathrm{vol}(K)} \leqslant 1. \tag{29}$$

Пусть $x^{(j)}$ — вершины, λ_j — базисные многочлены Лагранжа симплекса S_P. По формуле (7),

$$\|P\|_K = \max_{x\in K}\sum_{j=1}^{n+1}|\lambda_j(x)| = \max\left\{\sum_{j=1}^{n+1}|\beta_j| : \sum_{j=1}^{n+1}\beta_j = 1,\ \sum_{j=1}^{n+1}\beta_j x^{(j)} \in K\right\}.$$

Заменим β_{n+1} на равное значение $1 - \sum\limits_{j=1}^{n}\beta_j$. Условие $\sum\limits_{j=1}^{n+1}\beta_j x^{(j)} \in K$ эквивалентно условию

$$\sum_{j=1}^{n}\beta_j(x^{(j)} - x^{(n+1)}) \in K' = K - x^{(n+1)}.$$

Значит,

$$\|P\|_K = \max\left\{\sum_{j=1}^{n}\left|\beta_j\right| + \left|1 - \sum_{j=1}^{n}\beta_j\right|\right\}, \tag{30}$$

где максимум берётся по всем β_j, таким что $\sum\limits_{j=1}^{n}\beta_j(x^{(j)} - x^{(n+1)}) \in K'$. Очевидно, $\mathrm{vol}(K') = \mathrm{vol}(K)$.

Рассмотрим невырожденный линейный оператор $F : \mathbb{R}^n \to \mathbb{R}^n$, который ставит в соответствие точке $\beta = (\beta_1, \ldots, \beta_n)$ точку $x = F(\beta)$ в соответствии с правилом

$$x = \sum_{j=1}^{n}\beta_j\left(x^{(j)} - x^{(n+1)}\right).$$

Справедливо матричное равенство

$$F(\beta) = (\beta_1, \ldots, \beta_n)\mathbf{B},$$

где $\mathbf{B}$ — введённая выше матрица порядка n с элементами $b_{ij} = x_j^{(i)} - x_j^{(n+1)}$. Обозначим

$$\gamma^* = \chi_n^{-1}\left(\frac{n!\mathrm{vol}(K)}{|\det \mathbf{B}|}\right).$$

Благодаря (29) $n!\mathrm{vol}(K)/|\det(\mathbf{B})| \geqslant 1$, поэтому число γ^* определено корректно. Заметим также, что $\chi_n(\gamma^*) = n!\mathrm{vol}(K)/|\det(\mathbf{B})|$.

Для данного $\gamma \geqslant 1$ введём в рассмотрение множество

$$E_{n,\gamma} = \left\{\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{R}^n : \sum_{j=1}^n |\beta_j| + \left|1 - \sum_{j=1}^n \beta_j\right| \leqslant \gamma\right\}.$$

Покажем, что при $\gamma < \gamma^*$ выполняется $K' \not\subset F(E_{n,\gamma})$. Достаточно убедиться в справедливости неравенства $\mathrm{mes_n}(F(E_{n,\gamma})) < \mathrm{vol}(K')$. По теореме 3 $\mathrm{mes_n}(E_{n,\gamma^*}) = \chi_n(\gamma^*)/n!$, следовательно,

$$\begin{aligned}\mathrm{mes_n}(F(E_{n,\gamma})) &< \mathrm{mes_n}(F(E_{n,\gamma^*})) = |\det \mathbf{B}| \cdot \mathrm{mes_n}(E_{n,\gamma^*}) \\ &= |\det \mathbf{B}| \cdot \frac{\chi_n(\gamma^*)}{n!} = \mathrm{vol}(K) = \mathrm{vol}(K').\end{aligned}$$

Итак, для любого $\varepsilon > 0$ существует точка $z^{(\varepsilon)}$ со свойствами:

$$z^{(\varepsilon)} = \sum \beta_j^{(\varepsilon)}\left(x^{(j)} - x^{(n+1)}\right) \in K' \quad \text{и} \quad \left|\sum \beta_j^{(\varepsilon)}\right| + \left|1 - \sum \beta_j^{(\varepsilon)}\right| \geqslant \gamma^* - \varepsilon.$$

В связи с (30) это означает, что $\|P\|_K \geqslant \gamma^* - \varepsilon$. Поскольку $\varepsilon > 0$ — произвольно, мы получаем

$$\|P\|_K \geqslant \gamma^* = \chi_n^{-1}\left(\frac{n!\mathrm{vol}(K)}{|\det(\mathbf{B})|}\right) = \chi_n^{-1}\left(\frac{n!\mathrm{vol}(K)}{|\det(\mathbf{A})|}\right) = \chi_n^{-1}\left(\frac{\mathrm{vol}(K)}{\mathrm{vol}(S_P)}\right).$$

Теорема доказана. □

Напомним, что $\mathrm{simp_n}(K)$ обозначает максимальный объём симплекса с вершинами в K.

Теорема 5. *Пусть K — произвольное выпуклое тело в $\mathbb{R}^n$. Тогда*

$$\theta_n(K) \geqslant \chi_n^{-1}\left(\frac{\mathrm{vol}(K)}{\mathrm{simp_n}(K)}\right). \tag{31}$$

Доказательство. Из (28) следует, что для любого проектора $P : C(K) \to \Pi_1(\mathbb{R}^n)$

$$\|P\|_K \geqslant \chi_n^{-1}\left(\frac{\mathrm{vol}(K)}{\mathrm{vol}(S_P)}\right) \geqslant \chi_n^{-1}\left(\frac{\mathrm{vol}(K)}{\mathrm{simp_n}(K)}\right).$$

Это немедленно даёт (31). □

Если симплекс $S \subset K$ имеет максимальный объём, то $K \subset (n+2)S$ (см. (17)), значит,

$$\mathrm{vol}(K) \leqslant (n+2)^n \mathrm{vol}(S) \leqslant (n+2)^n \mathrm{simp_n}(K).$$

Следовательно, отношение $\mathrm{vol}(K)/\mathrm{simp_n}(K)$ в правой части (31) ограничено сверху величиной $(n+2)^n$.

Приведём здесь следствия оценки (31) для куба и шара. Сначала отметим доказанные автором неравенства для $\chi_n^{-1}(s)$ (см. [12]). Если n — чётное, то

$$\chi_n^{-1}(s) > \left(\frac{s\left((n/2)!\right)^2}{n!}\right)^{1/n}. \tag{32}$$

Если же n — нечётное, то

$$\chi_n^{-1}(s) > \left(\frac{s\,\frac{n+1}{2}!\,\frac{n-1}{2}!}{n!}\right)^{1/n}. \tag{33}$$

Пусть K есть куб $Q_n = [0,1]^n$. Тогда $\mathrm{vol}(K) = 1, \mathrm{simp_n}(K) = \nu_n$, поэтому (31) даёт

$$\theta_n(Q_n) \geqslant \chi_n^{-1}\left(\frac{1}{\nu_n}\right). \tag{34}$$

Величина ν_n может быть оценена сверху с помощью неравенства (13). Применение (32) и (33) позволяет получить из (34) следующий результат (см. [12]).

Следствие 1. *Для всех n*

$$\theta_n(Q_n) > \frac{\sqrt{n-1}}{e}.$$

Отсюда получается, что $\theta_n(Q_n) > c\sqrt{n}$. Например, как отмечено в [10], выполняется

$$\theta_n(Q_n) > \frac{2\sqrt{2}}{3e}\sqrt{n}.$$

Заметим, что $\dfrac{2\sqrt{2}}{3e} = 0.3468\ldots$. Оценка $\theta_n(Q_n) > c\sqrt{n}$ точна по n по крайней мере в случае, когда $n+1$ есть число Адамара: для таких n верно $\theta_n(Q_n) \asymp \sqrt{n}$.

Перейдём к случаю, когда K есть единичный шар B_n. Здесь $\mathrm{vol}(K) = \varkappa_n, \mathrm{simp_n}(K) = \sigma_n$, Тем самым (31) принимает вид

$$\theta_n(B_n) \geqslant \chi_n^{-1}\left(\frac{\varkappa_n}{\sigma_n}\right). \tag{35}$$

Применяя (32), (33), (14) и (15), получаем из (35) такую оценку (см. [28]).

Следствие 2. *Существует абсолютная константа $c > 0$, для которой $\theta_n(B_n) > c\sqrt{n}$. Подходящим значением c является*

$$c = \frac{\sqrt[3]{\pi}}{\sqrt{12e}\cdot\sqrt[6]{3}} = 0.2135\ldots$$

Оценка $\theta_n(B_n) > c\sqrt{n}$ является точной по размерности n. Именно, имеет место соотношение $\theta_n(B_n) \asymp \sqrt{n}$. Точное значение $\theta_n(B_n)$ найдено в [29]. При этом установлено, что $\theta_n(B_n) \geqslant \sqrt{n}$ с равенством только при $n = 1$. См. подробнее раздел 6.

5. Линейная интерполяция на произвольном компакте

Перейдём к обобщению неравенства (31) на (необязательно выпуклое) компактное множество. Пусть Ω — компакт в $\mathbb{R}^n$. Всюду далее в этом разделе через K обозначается выпуклая оболочка Ω. Будем предполагать, что $\mathrm{vol}(K) > 0$.

Нам понадобится следующая элементарная лемма.

Лемма 1. *Если $\varphi : \mathbb{R}^n \to \mathbb{R}$ — выпуклая непрерывная функция, то* $\max\limits_K \varphi = \max\limits_\Omega \varphi$.

Доказательство. Максимум из условия леммы существует, поскольку φ — непрерывная функция. Так как $K = \operatorname{conv}(\Omega)$, для любого $y \in K$ существуют натуральное m, точки $y^{(1)}, \ldots, y^{(m)} \in \Omega$ и числа $\mu_1, \ldots, \mu_m$, такие что

$$y = \sum_{i=1}^{m} \mu_i y^{(i)}, \quad \mu_i \geqslant 0 \quad \text{и} \quad \sum_{i=1}^{m} \mu_i = 1.$$

Очевидно, $\varphi(y^{(i)}) \leqslant \max\limits_{\Omega} \varphi$. Из выпуклости φ следует

$$\varphi(y) = \varphi\left(\sum_{i=1}^{m} \mu_i y^{(i)}\right) \leqslant \sum_{i=1}^{m} \mu_i \varphi(y^{(i)}) \leqslant \left(\sum_{i=1}^{m} \mu_i\right) \max_{\Omega} \varphi = \max_{\Omega} \varphi.$$

Поэтому $\max\limits_{K} \varphi \leqslant \max\limits_{\Omega} \varphi$. Обратное неравенство тривиально. □

Результат леммы 1 известен. Он сразу получается из следующего максимального принципа Бауэра [30]. *Любая выпуклая непрерывная функция, заданная на выпуклом компактном множестве, достигает максимума в некоторой экстремальной точке этого множества.* Следовательно, максимум φ на $K = \operatorname{conv}(\Omega)$ достигается в экстремальной точке Ω.

Теорема 6. *Пусть Ω — произвольный компакт в $\mathbb{R}^n$ с условием $\operatorname{vol}(K) > 0$, где $K = \operatorname{conv}(\Omega)$. Тогда*

$$\theta_n(\Omega) \geqslant \chi_n^{-1}\left(\frac{\operatorname{vol}(K)}{\operatorname{simp_n}(\Omega)}\right). \tag{36}$$

Доказательство. Прежде всего отметим, что для произвольного многочлена $p \in \Pi_1(\mathbb{R}^n)$

$$\|p\|_{\Omega} = \|p\|_{K}. \tag{37}$$

Это немедленно следует из леммы 1 для выпуклой непрерывной функции $\varphi(x) = |p(x)|$.

Пусть $P : C(\Omega) \to \Pi_1(\mathbb{R}^n)$ — произвольный интерполяционный проектор с узлами в Ω. Будем рассматривать P также как оператор на $C(K)$. Из (37) следует, что $\|P\|_{\Omega} = \|P\|_{K}$. Значит, $\theta_n(\Omega)$ не меньше, чем $\theta_n(K)$. Применяя неравенство (31) теоремы 5, получаем

$$\theta_n(\Omega) \geqslant \theta_n(K) \geqslant \chi_n^{-1}\left(\frac{\operatorname{vol}(K)}{\operatorname{simp_n}(K)}\right).$$

Остаётся заметить, что $\operatorname{simp_n}(K) = \operatorname{simp_n}(\Omega)$. Для доказательства рассмотрим любой симплекс $S \subset K$ с некоторой вершиной $x \notin \Omega$. Не изменяя других вершин, мы можем заменить x на вершину $x' \in \Omega$ так, что объём получившегося симплекса не уменьшится. Действительно, этот объём возрастает вместе с $\operatorname{dist}(x; \Gamma)$, где Γ — $(n-1)$-мерная гиперплоскость, содержащая все вершины симплекса, за исключением x. Пусть Γ задаётся уравнением $q(z) = \langle a, z\rangle + a_0 = 0$, $a = (a_1, \ldots, a_n) \in \mathbb{R}^n$, $a_0 \in \mathbb{R}$. Тогда

$$\operatorname{dist}(x; \Gamma) = \frac{|q(x)|}{\|a\|},$$

очевидно, есть выпуклая непрерывная функция. По лемме 1, максимум $\operatorname{dist}(x; \Gamma)$ на K достигается в точке $x' \in \Omega$. Применяя эту процедуру последовательно ко всем вершинам симплекса, не принадлежащим Ω, мы построим новый симплекс с вершинами в Ω без уменьшения первоначального объёма. Таким образом, $\operatorname{simp_n}(K) = \operatorname{simp_n}(\Omega)$. Это завершает доказательство теоремы. □

Объединим неравенство (36) теоремы 6 с правым неравенством (19) теоремы 2. Пусть $\Omega \subset \mathbb{R}^n$ — произвольное компактное множество, удовлетворяющее условию $\mathrm{vol}(\mathrm{conv}(\Omega)) > 0$. Для минимальной нормы интерполяционного проектора с узлами в Ω справедливы оценки

$$\chi_n^{-1}\left(\frac{\mathrm{vol}(\mathrm{conv}(\Omega))}{\mathrm{simp_n}(\Omega)}\right) \leqslant \theta_n(\Omega) \leqslant n+1. \tag{38}$$

По схеме, изложенной в [12], неравенства (38) переносятся на интерполяцию с помощью многочленов из пространств, более широких, чем $\Pi_1(\mathbb{R}^n)$. В настоящей статье эта тематика не рассматривается.

6. Заключительные замечания и открытые вопросы

Коротко отметим некоторые результаты о числах $\theta_n(K)$ и $\xi_n(K)$ для $K = Q_n$ и $K = B_n$. В случае, когда K есть n-мерный куб, в этой тематике имеются интересные открытые вопросы. Более детальный обзор даётся в [10].

Несмотря на простоту формулировки, задача нахождения точных значений $\theta_n(Q_n)$ очень трудна. С 2006 г. они известны только для четырёх размерностей n, а именно для $n = 1, 2, 3$ и 7 (см. [12]):

$$\theta_1(Q_1) = 1, \quad \theta_2(Q_2) = \frac{2\sqrt{5}}{5} + 1 = 1.8944\ldots, \quad \theta_3(Q_3) = 2, \quad \theta_7(Q_7) = \frac{5}{2}.$$

Соответствующие числа $\xi(Q_n)$ суть

$$\xi_1(Q_1) = 1, \quad \xi_2(Q_2) = \frac{3\sqrt{5}}{5} + 1 = 2.3416\ldots, \quad \xi_3(Q_3) = 3, \quad \xi_7(Q_7) = 7.$$

Для $n = 1, 2, 3, 7$ правое соотношение в (9) является равенством:

$$\xi_n(Q_n) = \frac{n+1}{2}\,(\theta_n(Q_n) - 1) + 1. \tag{39}$$

Всегда $\xi_n(Q_n) \geqslant n$; если $n+1$ — число Адамара, то $\xi_n(Q_n) = n$ (см. [12, 31, 32]). Благодаря (9) неравенство $\xi_n(Q_n) \geqslant n$ даёт

$$\theta_n(Q_n) \geqslant 3 - \frac{4}{n+1}. \tag{40}$$

Если $n = 1, 3$ или 7, то в (40) выполняется равенство. Для $1 \leqslant n \leqslant 3$ и $n = 7$ симплексы, соответствующие минимальным проекторам, в точности те же, что и симплексы, экстремальные относительно $\xi_n(Q_n)$ (см. [12, 33], а также недавний обзор [11]).

Пусть $n+1$ — число Адамара и S — n-мерный правильный симплекс, вершины которого совпадают с вершинами куба Q_n. Тогда для соответствующего проектора $P_S : C(Q_n) \to \Pi_1(\mathbb{R}^n)$ выполняется

$$\|P_S\|_{Q_n} \leqslant \sqrt{n+1}. \tag{41}$$

Различные доказательства даются в [12]. Статья [34] содержит доказательство (41), существенно использующее структуру матрицы Адамара. Интересно заметить, что равенство $\|P_S\|_{Q_n} = \sqrt{n+1}$ может выполняться как для всех правильных симплексов с вершинами в вершинах куба ($n = 1, n = 3$), так и для части из них ($n = 15$), а может и не выполняться вовсе.

В следствии 1 отмечалось, что для любого n верно $\theta_n(Q_n) \geqslant (1/e)\sqrt{n-1}$. Поэтому, если $n+1$ — число Адамара, то

$$\frac{\sqrt{n-1}}{e} \leqslant \theta_n(Q_n) \leqslant \sqrt{n+1}.$$

Другими словами, верхняя оценка $\theta_n(Q_n) \geqslant c\sqrt{n}$ является точной по n по крайней мере, когда $n+1$ есть число Адамара. Для этих размерностей $\theta_n(Q_n) \asymp \sqrt{n}$.

Верхние оценки чисел $\theta_n(Q_n)$ для конкретных n были улучшены А. Ю. Ухаловым и его учениками с применением компьютерных методов. Для этого, в частности, обсчитывались симплексы максимального объёма в кубе. Во всех ситуациях, когда $n+1$ — число Адамара, осуществлялся перебор всех известных матриц Адамара соответствующего порядка. Например, для получения оценки $\theta_{23}(Q_{23})$ рассматривались все описанные 60 матриц Адамара порядка 24. Для оценивания $\theta_{27}(Q_{27})$ были рассмотрены 487 матриц Адамара порядка 28. Лучшие верхние оценки для $1 \leqslant n \leqslant 27$ приведены в [35]. Вот эти оценки (для краткости мы пишем $\theta_n = \theta_n(Q_n)$):

$$\theta_1 = 1, \quad \theta_2 = \frac{2\sqrt{5}}{5} + 1, \quad \theta_3 = 2, \quad \theta_4 \leqslant \frac{3(4+\sqrt{2})}{7}, \quad \theta_5 \leqslant 2.448804,$$

$$\theta_6 \leqslant 2.6000\ldots, \quad \theta_7 = \frac{5}{2}, \quad \theta_8 \leqslant \frac{22}{7}, \quad \theta_9 \leqslant 3, \quad \theta_{10} \leqslant \frac{19}{5}, \quad \theta_{11} \leqslant 3,$$

$$\theta_{12} \leqslant \frac{17}{5}, \quad \theta_{13} \leqslant \frac{49}{13}, \quad \theta_{14} \leqslant \frac{21}{5}, \quad \theta_{15} \leqslant \frac{7}{2}, \quad \theta_{16} \leqslant \frac{21}{5}, \quad \theta_{17} \leqslant \frac{139}{34},$$

$$\theta_{18} \leqslant 5.1400\ldots, \quad \theta_{19} \leqslant 4, \quad \theta_{20} \leqslant 4.68879\ldots, \quad \theta_{21} \leqslant \frac{251}{50}, \quad \theta_{22} \leqslant \frac{1817}{335},$$

$$\theta_{23} \leqslant \frac{9}{2}, \quad \theta_{24} \leqslant \frac{103}{21}, \quad \theta_{25} \leqslant 5, \quad \theta_{26} \leqslant \frac{474}{91}, \quad \theta_{27} \leqslant 5.$$

Лучшая известная нижняя оценка чисел $\theta_n(Q_n)$ для всех n имеет вид

$$\theta_n(Q_n) \geqslant \max\left[3 - \frac{4}{n+1},\ \chi_n^{-1}\left(\frac{1}{\nu_n}\right)\right], \tag{42}$$

где χ_n — стандартизованный многочлен Лежандра степени n. Значения правой части (42) для $1 \leqslant n \leqslant 54$ даются в [35].

Как отмечалось выше (см. (9)), для всех n

$$\xi_n(Q_n) \leqslant \frac{n+1}{2}\,(\theta_n(Q_n) - 1) + 1. \tag{43}$$

До сих пор известны только четыре значения n, для которых это соотношение становится равенством: $n = 1, 2, 3$ и 7. Это как раз те случаи, в которых точные значения $\theta_n(Q_n)$ и $\xi_n(Q_n)$ известны одновременно. Вполне возможно, что минимальное n, для которого неравенство (43) является строгим, равно 4, но это остаётся открытой проблемой.

Приведённая выше оценка $\xi_n(Q_n) \geqslant n$ является точной по порядку n. Если $n > 2$, то

$$\xi_n(Q_n) \leqslant \frac{n^2 - 3}{n - 1} \tag{44}$$

(см. [12]). При $n > 1$ правая часть (44) строго меньше $n+1$. Неравенство $\xi_n < n+1$ верно также и для $n = 1, 2$. Поэтому всегда $n \leqslant \xi_n(Q_n) < n+1$, т. е. $\xi_n(Q_n) - n \in [0, 1)$. Вместе с тем точные значения константы $\xi_n(Q_n)$ пока удалось найти лишь для тех n, когда $n+1$ — число Адамара, а также для $n = 2$, $n = 5$ и $n = 9$. Во всех этих случаях, кроме $n = 2$, выполняется равенство $\xi_n(Q_n) = n$. В [32] даётся доказательство, существенно использующее структуру матрицы Адамара порядка $n+1$. Там же найдены точные значения $\xi_n(Q_n)$ для $n = 5$ и $n = 9$, а также построены бесконечные семейства экстремальных симплексов для $n = 5, 7, 9$.

Подчеркнём, что пока $n = 2$ является единственной известной размерностью, когда $\xi_n(Q_n) > n$. Далее, для нечётных $1 \leqslant n \leqslant 11$ верно $\xi_n(Q_n) = n$. Неясно, так ли это для всех нечётных n.

Благодаря эквивалентности $\xi_n(Q_n) \asymp n$ и неравенству $\theta_n \geqslant c\sqrt{n}$, для всех достаточно больших n

$$\xi_n(Q_n) < \frac{n+1}{2}\,(\theta_n(Q_n) - 1) + 1. \tag{45}$$

Обозначим через n_0 минимальное натуральное число, такое что для $n \geqslant n_0$ выполняется (45). Задача о точном значении n_0 также является трудной. Известные нижняя и верхняя границы n_0 различаются весьма значительно. Из предыдущего имеем $n_0 \geqslant 8$. В 2009 г. автор показал, что $n_0 \leqslant 57$ (см. [12, 33]). Достаточным условием для справедливости (45) для $n > 2$ является неравенство

$$\chi_n\left(\frac{3n-5}{n-1}\right) \cdot \nu_n < 1. \tag{46}$$

В [33] установлено, что (46) выполняется при $n \leqslant 57$. Более поздние вычисления позволили несколько понизить верхнюю границу для n_0. Именно, в [36] отмечается, что $n_0 \leqslant 53$. Таким образом, сегодня мы знаем, что $8 \leqslant n_0 \leqslant 53$. Уточнение интервала для n_0 — актуальная задача.

Перейдём теперь к шару B_n. По сравнению с кубом Q_n здесь наблюдается разительный контраст, поскольку *значения $\theta_n(B_n)$ и $\xi_n(B_n)$ найдены точно для любого n.*

В [37] показано, что $\xi_n(B_n) = n$. Более того, для симплекса $S \subset B_n$ равенство $\xi(B_n; S) = n$ эквивалентно тому, что S — правильный симплекс, вписанный в шар. Из (8) следует, что для любого интерполяционного проектора $P : C(B_n) \to \Pi_1(\mathbb{R}^n)$

$$\|P\|_{B_n} \geqslant 3 - \frac{4}{n+1}. \tag{47}$$

Правое равенство в (9), т. е. равенство

$$\xi_n(B_n) = \frac{n+1}{2}\Big(\theta_n(B_n) - 1\Big) + 1, \tag{48}$$

равносильно

$$\theta_n(B_n) = 3 - \frac{4}{n+1}. \tag{49}$$

Как показано в [38], равенства (48) и (49) имеют место при $1 \leqslant n \leqslant 4$, а начиная с $n = 5$ выполняется строгое неравенство

$$\xi_n(B_n) < \frac{n+1}{2}\Big(\theta_n(B_n) - 1\Big) + 1.$$

Для $1 \leqslant n \leqslant 4$ равенство в (47) верно только если S_P — правильный вписанный симплекс.

То, что нижняя оценка $\theta_n(B_n) > c\sqrt{n}$, отмеченная в следствии 2, точна по размерности n, было впервые установлено в [38]: *имеет место эквивалентность $\theta_n(B_n) \asymp \sqrt{n}$.*

Полное решение задачи о точном значении $\theta_n(B_n)$ дано в [29]. Опишем эти результаты. Пусть функция $\psi : [0, n+1] \to \mathbb{R}$ задаётся равенством

$$\psi(t) = \frac{2\sqrt{n}}{n+1}\Big(t(n+1-t)\Big)^{1/2} + \left|1 - \frac{2t}{n+1}\right|.$$

Обозначим

$$a = a_n = \left\lfloor \frac{n+1}{2} - \frac{\sqrt{n+1}}{2} \right\rfloor.$$

Пусть S — правильный симплекс, вписанный в B_n, p_n — $C(B_n)$-операторная норма проектора P_S. В [38] доказано, что $p_n = \max\{\psi(a), \psi(a+1)\}$ и справедливы неравенства $\sqrt{n} \leqslant p_n \leqslant \sqrt{n+1}$. Более того, $p_n = \sqrt{n}$ только для $n = 1$ и $p_n = \sqrt{n+1}$ тогда и только тогда, когда $\sqrt{n+1}$ — целое число.

Равенство $\theta_n(B_n) = p_n$ было получено сначала для $1 \leqslant n \leqslant 4$ (различные доказательства даются в [38] и [39]). В качестве гипотезы для всех n это утверждение было сформулировано в [39]. Наконец, в [29] был применён новый геометрический подход, который позволил доказать эту гипотезу, т. е. равенство $\theta_n(B_n) = p_n$, для произвольного n. В частности, имеем $\sqrt{n} \leqslant \theta_n(B_n) \leqslant \sqrt{n+1}$. Минимальным является проектор, соответствующий правильному симплексу, вписанному в граничную сферу, и других минимальных проекторов не существует.

Пусть k_n совпадает с тем из чисел a_n и $a_n + 1$, на котором $\psi(t)$ принимает большее значение. Числа k_n растут с n, но не строго монотонно. Если $n \geqslant 2$, то $k_n \leqslant n/2$. В качестве примера приведём числа k_n для $1 \leqslant n \leqslant 15$, $n = 50$, $n = 100$ и $n = 1000$ ([38]):

$$k_1 = k_2 = k_3 = k_4 = 1, \quad k_5 = k_6 = 2, \quad k_7 = k_8 = k_9 = 3, \quad k_{10} = k_{11} = 4,$$

$$k_{12} = k_{13} = 5, \quad k_{14} = k_{15} = 6, \quad k_{50} = 22, \quad k_{100} = 45, \quad k_{1000} = 485.$$

Отметим, что равенство (48) выполняется для тех и только тех размерностей n, когда $k_n = 1$.

Благодарности

Автор выражает благодарность П. А. Шварцману за полезные замечания и предложения.

References

[1] R. A. DeVore and G. G. Lorentz, *Constructive Approximation*. Springer-Verlag: Berlin – Heidelberg, 1993.

[2] V. Barthelmann, E. Novak, and K. Ritter, "High dimensional polynomial interpolation on sparse grids", *Advances in Computational Mathematics*, vol. 12, no. 4, pp. 273–288, 2000. DOI: 10.1023/A:1018977404843.

[3] C. De Boor, "Polynomial interpolation in several variables", in *Studies in Computer Science*, Plenum Press, 1994, pp. 87–119.

[4] S. De Marchi, *Lectures on Multivariate Polynomial Interpolation*. Göttingen – Padova, 2015.

[5] M. Gasca and R. Sauer, "Polynomial interpolation in several variables", *Advances in Computational Mathematics*, vol. 12, no. 4, pp. 377–410, 2000. DOI: 10.1023/A:1018981505752.

[6] M. Gunzburger and A. Teckentrup, *Optimal point sets for total degree polynomial interpolation in moderate dimensions*, 2014. arXiv: 1407.3291 [math.NA]. [Online]. Available: https://arxiv.org/abs/1407.3291.

[7] V. Kaarnioja, *On applying the maximum volume principle to a basis selection problem in multivariate polynomial interpolation*, 2017. arXiv: 1512.07424 [math.NA]. [Online]. Available: https://arxiv.org/abs/1512.07424.

[8] S. Pashkovskij, *Vychislitel'nye Primeneniya Mnogochlenov i Ryadov Chebysheva*. Nauka, 1983, in Russian.

[9] T. J. Rivlin, *The Chebyshev Polynomials*. John Wiley & Sons, 1974.

[10] M. Nevskii, *Optimal lagrange interpolation projectors and legendre polynomials*, 2024. arXiv: 2405.01254 [math.MG]. [Online]. Available: https://arxiv.org/abs/2405.01254.

[11] M. Nevskii, *Geometric estimates in linear interpolation on a cube and a ball*, 2024. arXiv: 2402.11611 [math.CA]. [Online]. Available: https://arxiv.org/abs/2402.11611.

[12] M. V. Nevskii, *Geometricheskie Ocenki v Polinomial'noj Interpolyacii*. P. G. Demidov Yaroslavl State University, 2012, 218 pp., in Russian.

[13] M. V. Nevskii, "Inequalities for the norms of interpolation projectors", *Modeling and Analysis of Information Systems.*, vol. 15, no. 3, pp. 28–37, 2008, in Russian.

[14] M. Hall Jr., *Combinatorial Theory*. Blaisdall Publishing Company, 1967.

[15] K. J. Horadam, *Hadamard Matrices and Their Applications*. Princeton University Press, 2007.

[16] P. K. Manjhi and M. K. Rama, "Some new examples of circulant partial Hadamard matrices of type $4 - H(k \times n)$", *Advances and Applications in Mathematical Sciences*, vol. 21, no. 5, pp. 2559–2564, 2022.

[17] M. Hudelson, V. Klee, and D. Larman, "Largest j-simplices in d-cubes: some relatives of the Hadamard maximum determinant problem", *Linear Algebra and its Applications*, vol. 241–243, pp. 519–598, 1996.

[18] J. Hadamard, "Résolution d'une question relative aux déterminants", *Bulletin des Sciences Mathématiques*, vol. 2, pp. 240–246, 1893.

[19] G. F. Clements and B. Lindström, "A sequence of (± 1) determinants with large values", *Proceedings of the American Mathematical Society*, vol. 16, no. 3, pp. 548–550, 1965.

[20] G. M. Fikhtengol'ts, *Kurs Differencial'nogo i Integral'nogo Ischisleniya. Tom 3*. Moskva: Fizmatlit, 2001, in Russian.

[21] L. Fejes Tót, *Regular Figures*. New York: Macmillan/Pergamon, 1964.

[22] D. Slepian, "The content of some extreme simplices", *Pacific Journal of Mathematics*, vol. 31, pp. 795–808, 1969.

[23] D. Vandev, "A minimal volume ellipsoid around a simplex", *Proceedings of the Bulgarian Academy of Sciences*, vol. 45, no. 6, pp. 37–40, 1992.

[24] M. Lassak, "Approximation of convex bodies by inscribed simplices of maximum volume", *Beiträge zur Algebra und Geometrie / Contributions to Algebra and Geometry*, vol. 52, pp. 389–394, 2011. DOI: 10.1007/s13366-011-0026-x.

[25] P. K. Suetin, *Klassicheskie ortogonal'nye mnogochleny*. Nauka, 1979, in Russian.

[26] G. Szegö, *Orthogonal Polynomials*. American Mathematical Society: Providence, 1975.

[27] A. P. Prudnikov, Y. A. Brychkov, and O. I. Marichev, *Integraly i Ryady*. Nauka, 2002, in Russian.

[28] M. V. Nevskii, "Geometric estimates in interpolation on an n-dimensional ball", *Modeling and Analysis of Information Systems*, vol. 26, no. 3, pp. 441–449, 2019, in Russian. DOI: 10.18255/1818-1015-2019-3-441-449.

[29] M. V. Nevskii, "On the minimal norm of the projection operator for linear interpolation on an n-dimensional ball", *Matematicheskie Zametki*, vol. 114, no. 3, pp. 477–480, 2023, in Russian. DOI: 10.4213/mzm14044.

[30] H. Bauer, "Minimalstellen von funktionen und extremal punkte", *Archiv der Mathematik*, vol. 9, no. 4, pp. 389–393, 1958, in German. DOI: 10.1023/A:1018977404843.

[31] M. Nevskii, "Properties of axial diameters of a simplex", *Discrete & Computational Geometry*, vol. 46, no. 2, pp. 301–312, 2011. DOI: 10.1007/s00454-011-9355-7.

[32] M. Nevskii and A. Ukhalov, "Perfect simplices in $\mathbb{R}^5$", *Beiträge zur Algebra und Geometrie / Contributions to Algebra and Geometry*, vol. 59, no. 3, pp. 501–521, 2018. DOI: 10.1007/s13366-018-0386-6.

[33] M. V. Nevskii, "On a certain relation for the minimal norm of an interpolation projector", *Modeling and Analysis of Information Systems.*, vol. 16, no. 1, pp. 24–43, 2009, in Russian.

[34] M. V. Nevskii, "On some estimate for the norm of an interpolation projector", *Modeling and Analysis of Information Systems*, vol. 29, no. 2, pp. 92–103, 2022, in Russian. DOI: 10.18255/1818-1015-2022-2-92-103.

[35] M. V. Nevskii and A. Y. Ukhalov, *Izbrannye Zadachi Analiza i Vychislitel'noj Geometrii. Chast' 2*. P. G. Demidov Yaroslavl State University, 2022, in Russian.

[36] M. V. Nevskii and A. Y. Ukhalov, "On optimal interpolation by linear functions on an n-dimensional cube", *Modeling and Analysis of Information Systems*, vol. 25, no. 3, pp. 291–311, 2018, in Russian. DOI: 10.18255/1818-1015-2018-3-291-311.

[37] M. V. Nevskii, "On some problems for a simplex and a ball in $\mathbb{R}^n$", *Modeling and Analysis of Information Systems*, vol. 25, no. 6, pp. 680–691, 2018, in Russian. DOI: 10.18255/1818-1015-2018-6-680-691.

[38] M. V. Nevskii and A. Y. Ukhalov, "Linear interpolation on a Euclidean ball in $\mathbb{R}^n$", *Modeling and Analysis of Information Systems*, vol. 26, no. 2, pp. 279–296, 2019, in Russian. DOI: 10.18255/1818-1015-2019-2-279-296.

[39] M. V. Nevskii, "On properties of a regular simplex inscribed into a ball", *Modeling and Analysis of Information Systems*, vol. 28, no. 2, pp. 186–197, 2021, in Russian. DOI: 10.18255/1818-1015-2021-2-186-197.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 31, NO. 3, 2024
journal homepage: www.mais-journal.ru

DISCRETE MATHEMATICS IN RELATION TO COMPUTER SCIENCE

Some Polynomial Subclasses of the Eulerian Walk Problem for a Multiple Graph

A. V. Smirnov[1]

DOI: 10.18255/1818-1015-2024-3-338-356

[1]P.G. Demidov Yaroslavl State University, Yaroslavl, Russia

MSC2020: 05C45, 05C65, 05C85
Research article
Full text in Russian

Received August 3, 2024
Revised August 16, 2024
Accepted August 21, 2024

In this paper, we study undirected multiple graphs of any natural multiplicity $k > 1$. There are edges of three types: ordinary edges, multiple edges and multi-edges. Each edge of the last two types is a union of k linked edges, which connect 2 or $(k + 1)$ vertices, correspondingly. The linked edges should be used simultaneously. If a vertex is incident to a multiple edge, it can be also incident to other multiple edges and it can be the common end of k linked edges of some multi-edge. If a vertex is the common end of some multi-edge, it cannot be the common end of another multi-edge.

We study the problem of finding the Eulerian walk (the cycle or the trail) in a multiple graph, which generalizes the classical problem for an ordinary graph. The multiple Eulerian walk problem is NP-hard. We prove the polynomiality of two subclasses of the multiple Eulerian walk problem and elaborate the polynomial algorithms. In the first subclass, we set a constraint on the ordinary edges reachability sets, which are the subsets of vertices joined by ordinary edges only. In the second subclass, we set a constraint on the quasi-vertices degrees in the graph with quasi-vertices. The structure of this ordinary graph reflects the structure of the multiple graph, and each quasi-vertex is determined by k indices of the ordinary edges reachability sets, which are incident to some multi-edge.

Keywords: multiple graph; divisible graph; covering trails; edge-disjoint paths; eulerian trail; eulerian cycle; graph with quasi-vertices; ordinary edges reachability set; polynomial subclass

INFORMATION ABOUT THE AUTHORS

Smirnov, Alexander V. (corresponding author)	ORCID iD: 0000-0002-0980-2507. E-mail: alexander_sm@mail.ru PhD, Associate Professor

Funding: Yaroslavl State University (project VIP-016).

For citation: A. V. Smirnov, "Some polynomial subclasses of the Eulerian walk problem for a multiple graph", *Modeling and Analysis of Information Systems*, vol. 31, no. 3, pp. 338–356, 2024. DOI: 10.18255/1818-1015-2024-3-338-356.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 31, № 3, 2024
сайт журнала: www.mais-journal.ru

DISCRETE MATHEMATICS IN RELATION TO COMPUTER SCIENCE

Некоторые полиномиальные подклассы задачи об эйлеровом маршруте в кратном графе

А. В. Смирнов[1]

DOI: 10.18255/1818-1015-2024-3-338-356

[1]Ярославский государственный университет им. П.Г. Демидова, Ярославль, Россия

УДК 519.17+519.161
Научная статья
Полный текст на русском языке

Получена 3 августа 2024 г.
После доработки 16 августа 2024 г.
Принята к публикации 21 августа 2024 г.

В статье рассматриваются неориентированные кратные графы произвольной натуральной кратности $k > 1$. Кратный граф содержит ребра трех типов: обычные, кратные и мультиребра. Ребра последних двух типов представляют собой объединение k связанных ребер, которые соединяют 2 или $(k + 1)$ вершину соответственно. Связанные ребра могут использоваться только согласованно. Если вершина инцидентна кратному ребру, то она может быть инцидентна другим кратным ребрам, а также она может быть общим концом k связанных ребер мультиребра. Если вершина является общим концом мультиребра, то она не может быть общим концом никакого другого мультиребра.

Рассматривается задача об эйлеровом маршруте (цикле или цепи) в кратном графе, которая обобщает классическую задачу для обычного графа. Задача о кратном эйлеровом маршруте является NP-трудной. Обоснована полиномиальность двух подклассов задачи о кратном эйлеровом маршруте, разработаны полиномиальные алгоритмы. В первом подклассе задано ограничение на множества достижимости по обычным ребрам, которые представляют собой подмножества вершин, соединенных только обычными ребрами. Во втором подклассе задано ограничение на степень квазивершин в графе с квазивершинами. Структура этого обычного графа отражает структуру кратного графа, а каждая квазивершина определяется k индексами множеств достижимости по обычным ребрам, которые инцидентны какому-то мультиребру.

Ключевые слова: кратный граф; делимый граф; покрывающие цепи; пути, не пересекающиеся по ребрам; эйлерова цепь; эйлеров цикл; граф с квазивершинами; множество достижимости по обычным ребрам; полиномиальный подкласс

ИНФОРМАЦИЯ ОБ АВТОРАХ

Смирнов, Александр Валерьевич (автор для корреспонденции) | ORCID iD: 0000-0002-0980-2507. E-mail: alexander_sm@mail.ru
Канд. физ.-мат. наук, доцент

Финансирование: ЯрГУ (проект VIP-016).

Для цитирования: A. V. Smirnov, “Some polynomial subclasses of the Eulerian walk problem for a multiple graph”, *Modeling and Analysis of Information Systems*, vol. 31, no. 3, pp. 338–356, 2024. DOI: 10.18255/1818-1015-2024-3-338-356.

Введение

В данной статье мы рассмотрим задачу об *эйлеровом маршруте* (цикле или цепи) в кратном графе. Кратные графы содержат три типа ребер (обычные, кратные и мультиребра) и являются обобщением обычных графов — по сути, обычный граф имеет кратность $k = 1$. Определения кратного графа кратности $k > 1$ и делимого кратного графа были сформулированы в статье [1]. Там же были введены понятия кратного пути и кратного цикла.

Отметим, что частным случаем кратного графа является кратная сеть (см. [2, 3]). Задача о наибольшем потоке в кратной сети обобщает классическую задачу (см. [4]) и имеет ряд приложений в сфере экономики, управления, финансов. В частности, кратные сети и потоки используются для поиска решения NP-трудной задачи целочисленного сбалансирования трех- и четырехмерной матрицы (см., например, [5, 6]).

В статье [7] была поставлена задача об эйлеровом маршруте (цепи или цикле) в неориентированном кратном графе. Главное отличие от эйлерова маршрута в обычном графе — необходимость согласования кратного эйлерова маршрута на связанных ребрах, что означает, что в таком маршруте связанные ребра каждого кратного и мультиребра могут проходиться только одновременно и только в одинаковом направлении. Рассмотрены необходимые условия существования эйлерова маршрута в кратном графе, которые, однако, не будут достаточными из-за того, что при их выполнении не всегда возможно обеспечить согласованность всех связанных ребер в маршруте. Предложен экспоненциальный алгоритм решения задачи о кратном эйлеровом маршруте, выделен один полиномиальный подкласс задачи.

Задача об эйлеровом маршруте в классической постановке для обычного графа полиномиальна; для нее существуют быстрые полиномиальные алгоритмы, например, описанный в работе [8] алгоритм Хиргольцера. Однако при переходе к обобщению обычного графа либо к варианту постановки с введением дополнительных ограничений задача зачастую становится NP-трудной.

Так происходит, например, в случае одного из наиболее популярных обобщений графов — гиперграфов. Напомним, что гиперграф содержит гиперребра, соединяющие k вершин (см. [9]). Понятие гиперребра, таким образом, в какой-то мере родственно понятию мультиребра в кратном графе, исследуемом в данной статье. Однако есть и существенное отличие двух концепций: все вершины гиперребра «равноправны», в то время как мультиребро определяется как множество из k связанных ребер, соединяющих одну вершину — общий конец с k отдельными вершинами. В работе [10] показано, что задача распознавания эйлерова маршрута в гиперграфе NP-полна даже в случае k-однородного гиперграфа ($k > 2$), в котором все ребра соединяют ровно k вершин.

В статье [11] для темпоральных графов рассмотрены различные варианты задачи об эйлеровом маршруте, среди них выделены полиномиальные и NP-полные. В темпоральном графе каждое ребро e доступно лишь в моменты времени, определяемые функцией $\lambda(e) \in 2^\tau$ ($\tau \geqslant 2$ — это общее время жизни графа), соответственно, любой маршрут должен учитывать это обстоятельство. Отметим, что NP-полной будет задача в постановке, наиболее близкой к классической, когда требуется найти цепь, проходящую каждое ребро темпорального графа ровно один раз. Это справедливо в том числе для $\tau = 2$.

В работе [12] рассмотрена классическая задача об эйлеровом цикле с дополнительным ограничением: граф изображен на плоскости и для каждой вершины все ребра, инцидентные ей, пронумерованы по часовой стрелке; при этом любые два ребра, идущие подряд в эйлеровом цикле, должны иметь соседние номера относительно общей вершины. В указанной статье доказано, что задача распознавания такого эйлерова цикла будет NP-полной даже в случае планарного графа.

Еще один пример классической задачи с дополнительным ограничением — это распознавание эйлерова цикла, в котором каждый подцикл имеет длину не меньше k. В статье [13] обоснована NP-полнота сужения этой задачи, откуда следует NP-полнота общей задачи.

Точно так же задача об эйлеровом маршруте становится труднорешаемой и в случае кратных графов. NP-полнота соответствующей задачи распознавания доказана в [14]. При этом задача будет NP-полной для любой кратности $k \geqslant 2$ даже для специального класса кратных графов — делимых графов. Этот класс характеризуется тем, что кратный граф может быть представлен в виде объединения k обычных графов, каждый из которых содержит ровно одно связанное ребро каждого кратного и мультиребра.

В данной статье мы выделим два полиномиальных подкласса задачи об эйлеровом маршруте в кратном графе и получим полиномиальные алгоритмы для них.

1. Необходимые определения и постановка задачи об эйлеровом маршруте в кратном графе

Напомним несколько понятий, связанных с кратными графами. Поскольку данная статья продолжает исследование, описанное в работах [7, 14], здесь будут сформулированы только самые важные определения. Поясняющие примеры и ряд связанных определений были подробно рассмотрены в статьях [1, 7].

Определение 1. *Кратный граф* G произвольной натуральной кратности $k > 1$ — это граф, вершины которого могут соединяться ребрами одного из 3 видов:

1. *Обычное ребро* e^o; множество обычных ребер обозначим через E^o.
2. *Кратное ребро* e^k между двумя вершинами, которое состоит из k одинаковых связанных ребер; связанные ребра кратного ребра могут использоваться только согласованно; множество кратных ребер обозначим через E^k.
3. *Связанное ребро* e между двумя вершинами, имеющее один общий конец с другим $(k-1)$ ребром (у любых двух из k связанных ребер только один конец является общим); множество связанных общей вершиной ребер будем называть *мультиребром* e^m; связанные ребра мультиребра могут использоваться только согласованно; множество мультиребер обозначим через E^m.

Если вершина инцидентна какому-либо кратному ребру, то она может быть инцидентна другим кратным ребрам, а также она может быть общим концом какого-либо мультиребра.

Если вершина является общим концом какого-либо мультиребра, то она не может быть общим концом никакого другого мультиребра.

Если вершина является отдельным концом мультиребра или инцидентна обычному ребру, то она не может быть общим концом мультиребра и не может быть инцидентна кратному ребру.

Множества вершин и ребер графа G обозначим через V и E соответственно. Заметим, что $E = E^o \cup E^k \cup E^m$. Обычное или кратное ребро, соединяющее две вершины x и y, обозначается стандартным образом: $\{x, y\}$. Мультиребро, соединяющее общую вершину x с k отдельными вершинами $y_1, \ldots, y_k$, обозначается так: $e^m_x = \{x, \{y_1, \ldots, y_k\}\}$.

Рис. 1 и 2 иллюстрируют определение 1. В левой части рис. 1 кратное ребро представлено в виде объединения k одинаковых ребер между двумя вершинами, что показано штрихами. Равенство (или согласованность) связанных ребер предполагает, что все характеристики этих ребер (например, длина) одинаковы, и эти ребра могут использоваться только одновременно. Так, если осуществляется проход в определенном направлении по одному из связанных ребер, то одновременно с этим все остальные ребра проходятся в том же самом направлении. Кратное ребро может включаться в какие-либо новые структуры только целиком. В дальнейшем мы будем обозначать кратные ребра жирными линиями, как в правой части рис. 1.

В левой части рис. 2 мультиребро $\{x_0, \{x_1, \ldots, x_k\}\}$ представлено в виде объединения k одинаковых ребер, связывающих общую вершину x_0 с k разными вершинами $x_1, \ldots, x_k$. Как и на рис. 1, равенство ребер показано штрихами. Согласованность связанных ребер имеет тот же смысл,

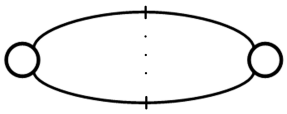


Fig. 1. Multiple edge

Рис. 1. Кратное ребро

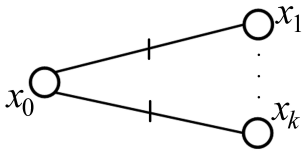


Fig. 2. Multi-edge

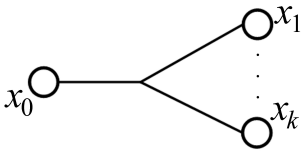


Рис. 2. Мультиребро

что и для кратных ребер. В дальнейшем мультиребра мы будем изображать при помощи расщепляющихся на k частей линий, как в правой части рис. 2.

В данной статье рассматриваются только неориентированные кратные графы.

Определение 2. *Обычной вершиной* назовем вершину, которая инцидентна обычному ребру или является отдельным концом мультиребра.

Кратной вершиной назовем вершину, которая инцидентна кратному ребру или является общим концом мультиребра.

Из определения 1 следует, что множества обычных и кратных вершин не пересекаются. При этом кратная вершина может быть соединена с обычными только посредством мультиребра.

Определение 3. Для любой вершины $x \in V$ определена ее *степень* $\deg x$ — количество обычных или связанных ребер инцидентных x.

Таким образом, каждое обычное ребро вида $\{x, y\}$ добавляет 1 к $\deg x$, каждое кратное ребро вида $\{x, y\}$ добавляет k к $\deg x$, а каждое мультиребро вида $\{x, \{y_1, \ldots, y_k\}\}$ добавляет k к $\deg x$ и 1 к каждому из $\deg y_i$. Очевидно, что степень любой кратной вершины будет кратна k.

Определение 4. *Делимым кратным графом* назовем такой граф, в котором между двумя концами одного мультиребра не существует пути, проходящего только по обычным ребрам.

При удалении всех мультиребер делимый граф распадется на n компонент связности (связность здесь понимается в том же смысле, что и для обычных графов), каждая из которых содержит только кратные ребра либо только обычные ребра. При этом связанные ребра каждого мультиребра можно пронумеровать от 1 до k таким образом, что каждой компоненте связности, содержащей только обычные ребра, будут инцидентны связанные ребра мультиребер с одинаковыми номерами.

Определение 5. *Частью* G_i $(i \in \overline{1,k})$ делимого графа $G(V, E)$ назовем подграф, содержащий связанные ребра с номером i всех кратных и мультиребер, а также компоненты связности, состоящие из обычных ребер и инцидентные i-ым связанным ребрам всех мультиребер.

В статье [1] вводится понятие *кратного пути из вершины x в вершину y*. В данной работе нам не потребуется получать кратные пути, поэтому не будем приводить определение полностью, напомним лишь основные особенности кратного пути:

- кратный путь представляет собой объединение k обычных (некратных) путей, где связанные ребра кратных и мультиребер используются согласованно: в каждом из обычных путей встречается ровно одно связанное ребро используемого кратного или мультиребра, причем порядок вхождения связанных ребер разных кратных и мультиребер одинаков во всех k путях;

- обычные вершины и ребра могут встречаться многократно;
- каждая кратная вершина встречается не более одного раза.

В отличие от обычных графов, *связность* кратного графа не предполагает наличие кратных путей из каждой вершины в каждую. Фактически в связном кратном графе между каждой парой вершин должен существовать обычный (некратный) путь, использующий связанные ребра кратных и мультиребер несогласованно, а кратные пути обязательно должны существовать только для пар кратных вершин. Критерий связности кратного графа и полиномиальный алгоритм его проверки подробно рассматривались в работе [1].

Эйлеров маршрут отличается от кратного пути тем, что кратные вершины можно использовать повторно, но каждое ребро используется только один раз. Эйлеров маршрут может быть найден только в связном кратном графе, поэтому далее по умолчанию предполагается связность рассматриваемого графа.

Определение 6. *Эйлеровым маршрутом* h в кратном графе $G(V, E)$ назовем такой обход графа $G(V, E)$, в котором каждое ребро из E встречается ровно один раз, а связанные ребра каждого кратного и мультиребра из $E^k \cup E^m$ используются только согласованно (одновременно).

Для эйлерова маршрута выполнено: $h = \cup_{i=1}^{k} h^i$, то есть каждый эйлеров маршрут h в кратном графе представляет собой объединение k обычных маршрутов h^i ($i \in \overline{1,k}$), в каждом из которых присутствует ровно одно связанное ребро каждого кратного и мультиребра, причем порядок обхода связанных ребер одинаков во всех h^i.

Фактически это значит, что если e_1 и e_2 — это два ребра маршрута h, каждое из которых либо кратное, либо мультиребро, и в проекции h^i связанное ребро из e_1 проходится раньше связанного ребра из e_2, то во всех остальных проекциях h^j связанные ребра из e_2 могут проходиться только после связанных ребер из e_1.

Определение 7. Замкнутый эйлеров маршрут в кратном графе (начальная вершина равна конечной) называется *эйлеровым циклом*.

Незамкнутый эйлеров маршрут в кратном графе называется *эйлеровой цепью*.

Кратный граф назовем *эйлеровым*, если в нем существует эйлеров маршрут (цикл или цепь).

Пример 1. Рассмотрим представленный на рис. 3 кратный граф G кратности 2 со следующими множествами обычных, кратных и мультиребер (вершины будем обозначать их номерами):

$$E^k = \big\{\{1,3\},\ \{1,4\},\ \{2,3\},\ \{2,4\},\ \{3,4\},\ \{15,16\}\big\};$$

$$E^m = \big\{\{3,\{5,13\}\},\ \{4,\{8,10\}\},\ \{15,\{6,14\}\},\ \{16,\{9,11\}\}\big\};$$

$$E^o = \big\{\{5,6\},\ \{5,7\},\ \{5,8\},\ \{6,7\},\ \{6,9\},\ \{7,8\},\ \{7,9\},\ \{8,9\},\ \{10,11\},$$
$$\{10,12\},\ \{10,13\},\ \{11,12\},\ \{11,14\},\ \{12,13\},\ \{12,14\},\ \{13,14\}\big\}.$$

Граф G является делимым. Части G_1 и G_2 этого графа показаны на рис. 4, связанные ребра всех кратных и мультиребер изображены пунктирными линиями.

Заметим, что граф перестанет быть делимым, если добавить в него обычное ребро между любой парой вершин из множеств $\{5,6,7,8,9\}$ и $\{10,11,12,13,14\}$.

Нетрудно убедиться, что в этом графе существует эйлеров цикл h, представленный на рис. 5.

Следует отметить, что в эйлеровом цикле h две кратные вершины 3 и 4 проходятся дважды (на рисунке отмечены серым), поэтому цикл h является эйлеровым маршрутом, но не является кратным циклом.

Задача 1 (эйлеров маршрут в кратном графе)**.** *Для данного кратного графа $G(V, E)$ установить, существует ли в нем эйлеров маршрут (цикл или цепь), и найти этот маршрут.*

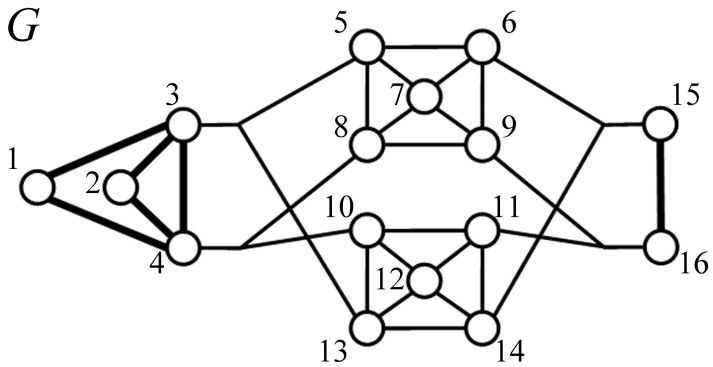


Fig. 3. Divisible graph of multiplicity 2

Рис. 3. Делимый граф кратности 2

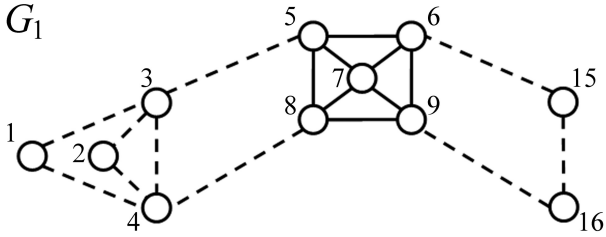


Fig. 4. Partition of the divisible graph

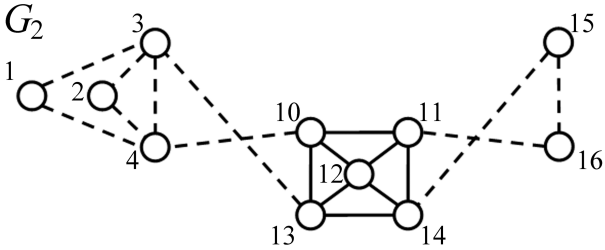


Рис. 4. Части делимого графа

В дальнейшем мы будем обозначать задачу 1 как MEW (multiple eulerian walk).

Определение 8. *Множеством достижимости по обычным ребрам* для некоторой обычной вершины x назовем множество R^o_x всех вершин y таких, что существует путь из x в y, проходящий только по обычным ребрам.

Будем обозначать через G^o_x подграф, образованный всеми вершинами из R^o_x и всеми обычными ребрами $e \in E^o$, которые соединяют вершины из R^o_x.

Если $y \in R^o_x$, то $R^o_y = R^o_x$, поэтому для удобства будем выбирать вершину с *минимальным* номером для задания индекса x каждого R^o_x.

Напомним необходимые условия существования кратного эйлерова цикла. Условия подробно рассматривались и обосновывались в статье [7], там же показано, что в общем случае необходимые условия не являются достаточными, что также отличает задачу для кратного графа от классической.

Теорема 1. *Если в кратном графе $G(V, E)$ существует эйлеров цикл, то все его обычные вершины имеют четную степень, а степень каждой кратной вершины v может быть представлена как* $\deg v = k \cdot a_v$, *где k — это кратность графа, a_v — четное число.*

Теорема 2. *Если в делимом кратном графе $G(V, E)$ существует эйлерова цепь, то одновременно выполнены следующие условия:*

- *все обычные вершины имеют четную степень;*
- *степень двух кратных вершин x, y представляется в виде* $\deg x = k \cdot a$, $\deg y = k \cdot b$, *где k — это кратность графа, a и b — нечетные числа;*
- *степень всех остальных кратных вершин v представляется в виде* $\deg v = k \cdot c_v$, *где k — это кратность графа, c_v — четное число.*

Теорема 3. *Если в неделимом кратном графе $G(V, E)$ существует эйлерова цепь, то найдутся две вершины x, y, для которых выполняется одно из условий:*

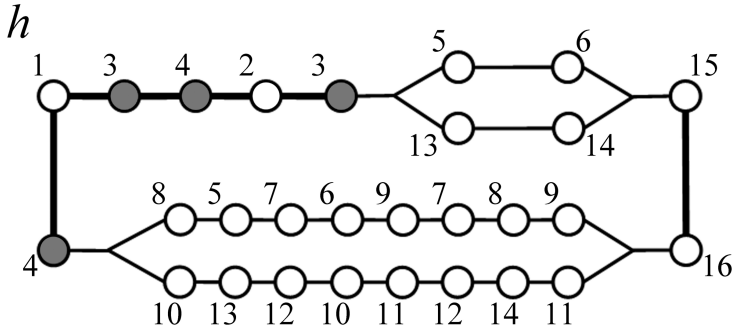


Fig. 5. Eulerian cycle in the multiple graph

Рис. 5. Эйлеров цикл в кратном графе

1. *x, y — кратные вершины;* $\deg x = k \cdot a$, $\deg y = k \cdot b$, *где k — это кратность графа, a и b — нечетные числа.*
2. *x — кратная вершина, y — обычная вершина и найдется мультиребро* $\{y_0, \{y_1, \ldots, y_k\}\} \in E^m$ *такое, что* $y_i \in R^o_y$ $(i \in \overline{1,k})$; $\deg x = k \cdot a$, $\deg y = k + b$, *где k — это кратность графа, a — нечетное число, b — четное неотрицательное число.*
3. *x, y — обычные вершины и найдутся два мультиребра* $\{x_0, \{x_1, \ldots, x_k\}\}, \{y_0, \{y_1, \ldots, y_k\}\} \in E^m$ *такие, что* $x_i \in R^o_x$, $y_i \in R^o_y$ $(i \in \overline{1,k})$; $\deg x = k + a$, $\deg y = k + b$, *где k — это кратность графа, a и b — четные неотрицательные числа.*

Степень всех остальных обычных вершин четна, а степень всех остальных кратных вершин v представляется в виде $\deg v = k \cdot c_v$, *где k — это кратность графа, c_v — четное число.*

Условия 1–3 теоремы 3 определяют выбор начальной и конечной вершины кратной эйлеровой цепи (вершины x и y) в случае, когда граф не является делимым.

2. Граф с квазивершинами и задача о покрывающих цепях

В статье [7] показано, что кратный граф $G(V, E)$ можно свести к обычному графу $G_{ord}(V_{ord}, E_{ord})$, структура которого будет отражать структуру исходного графа. Часть вершин будут соответствовать множествам подграфов G^o_x. Такие вершины называются *квазивершинами*, а граф G_{ord} — *графом с квазивершинами*. Преобразование выполняется с помощью простого полиномиального алгоритма, который трансформирует все кратные вершины и ребра в обычные, а затем создает квазивершины и ребра инцидентные им. Для каждого мультиребра $e^m = \{x_0, \{x_1, \ldots, x_k\}\} \in E^m$ находятся множества достижимости $R^o_{y_i}$ такие, что $x_i \in R^o_{y_i}$ $(i \in \overline{1,k})$. Номера этих множеств достижимости определяют квазивершину $q_{y_1,\ldots,y_k} \in V_{ord}$, а мультиребро e^m определяет обычное ребро $\{x_0, q_{y_1,\ldots,y_k}\} \in E_{ord}$.

Из необходимых условий существования кратного эйлерова цикла следует, что если кратный граф $G(V, E)$ является эйлеровым, то эйлеровым является и граф с квазивершинами $G_{ord}(V_{ord}, E_{ord})$, причем каждому эйлерову маршруту в исходном кратном графе соответствует эйлеров маршрут в графе с квазивершинами. Обратное в общем случае неверно: граф с квазивершинами может быть эйлеровым, тогда как исходный граф не эйлеров; также в графе $G_{ord}(V_{ord}, E_{ord})$ может существовать эйлеров цикл или цепь, которым не соответствует эйлеров цикл или цепь в графе $G(V, E)$.

Пример 2. Снова рассмотрим кратный граф из примера 1 (рис. 3). Нетрудно заметить, что в этом графе всего два множества достижимости по обычным ребрам:

$$R^o_5 = \{5, 6, 7, 8, 9\}, \qquad R^o_{10} = \{10, 11, 12, 13, 14\}.$$

При этом все мультиребра имеют ровно по одному концу в каждом из R^o_5 и R^o_{10}, следовательно, потребуется создать только одну квазивершину $q_{5,10}$, а граф $G_{ord}(V_{ord}, E_{ord})$ будет иметь простую структуру (см. рис. 6). Квазивершина отмечена на рисунке серым.

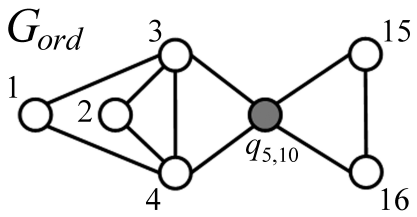


Fig. 6. Graph with a quasi-vertex

Рис. 6. Граф с квазивершиной

Кратному эйлерову циклу, представленному на рис. 5, будет соответствовать обычный эйлеров цикл $h_{ord} = (1, 3, 4, 2, 3, q_{5,10}, 15, 16, q_{5,10}, 4, 1)$ в графе G_{ord}.

Определение 9. *Покрывающими цепями* в обычном графе $G^o(V^o, E^o)$ называется множество цепей, не пересекающихся по ребрам и не содержащих повторяющихся ребер; при этом множество покрывающих цепей содержит все ребра графа.

Задача 2 (покрывающие цепи с заданными концами в обычном графе)**.** *Дан обычный связный граф $G^o(V^o, E^o)$, в котором выделено p начальных вершин $s_1, \ldots, s_p$ и p конечных вершин $t_1, \ldots, t_p$. Требуется установить, существуют ли в графе $G^o(V^o, E^o)$ p покрывающих цепей $\mu_1(s_1, t_1), \ldots, \mu_p(s_p, t_p)$, и найти эти цепи.*

Задачу 2 обозначим через CT (covering trails). Начальные и конечные вершины будем также называть *терминальными*.

В статье [7] показано, что каждый эйлеров маршрут в графе с квазивершинами $G_{ord}(V_{ord}, E_{ord})$ определяет задачу CT в каждом подграфе G^o_x исходного кратного графа $G(V, E)$. Если каждая такая задача CT имеет решение, то эти решения в объединении с найденным эйлеровым маршрутом в графе $G_{ord}(V_{ord}, E_{ord})$ порождают кратный эйлеров маршрут в графе $G(V, E)$. В указанной статье предложен точный алгоритм поиска кратного эйлерова маршрута, который последовательно перебирает все возможные эйлеровы маршруты в графе G_{ord} (для этого используется описанная в работе [15] операция *каппа-трансформации*) и пытается решить возникающие при этом задачи CT для графов G^o_x. Однако и построение всех возможных эйлеровых маршрутов, и решение задач CT в общем случае требуют экспоненциального количества шагов. Поэтому при поиске полиномиальных подклассов задачи MEW нам нужно выделить случаи, где указанные подзадачи могут быть решены за полиномиальное время.

Напомним также классическую задачу о путях, не пересекающихся по ребрам (см. [16]), которая близка к задаче CT.

Задача 3 (пути, не пересекающиеся по ребрам)**.** *Дан обычный связный граф $G^o(V^o, E^o)$, в котором выделено p начальных вершин $s_1, \ldots, s_p$ и p конечных вершин $t_1, \ldots, t_p$. Требуется установить, существуют ли в графе $G^o(V^o, E^o)$ p путей (цепей) $\mu_1(s_1, t_1), \ldots, \mu_p(s_p, t_p)$, попарно не пересекающихся по ребрам, и найти эти пути.*

Будем использовать для задачи 3 стандартное обозначение EDP (edge-disjoint paths). В работе [16] доказано, что задача EDP является NP-трудной, если параметр p не фиксирован (является частью входных данных задачи).

Часто наряду с графом $G^o(V^o, E^o)$ рассматривают граф $H^o(V^o, F^o)$, где $F^o = \{\{s_1, t_1\}, \ldots, \{s_p, t_p\}\}$. Тогда для графа $G^o \cup H^o$ с множеством вершин V^o и множеством ребер $E^o \cup F^o$ можно рассмотреть следующую задачу, которая будет эквивалентна задаче EDP: *найти в графе $G^o \cup H^o$ множество из $|F^o|$ не пересекающихся по ребрам циклов, каждый из которых содержит ровно одно ребро из F^o.*

Пользуясь свойствами графа $G^o \cup H^o$, можно выделить различные подклассы исходной задачи EDP. В частности, нас будет интересовать случай, когда граф $G^o \cup H^o$ эйлеров.

3. Графы с ограничением на множества достижимости по обычным ребрам

Лемма 1. *Пусть для связного обычного графа G^o соответствующий граф $G^o \cup H^o$ является эйлеровым. Тогда задача CT для него всегда имеет решение при $p = 1$ или $p = 2$, которое может быть найдено за полиномиальное время.*

Доказательство. В случае $p = 1$ утверждение очевидно: задача CT сводится к поиску эйлерова цикла в графе $G^o \cup H^o$, который по условию является эйлеровым. Для этого можно использовать любой из известных полиномиальных алгоритмов, например, алгоритм Хиргольцера.

Пусть $p = 2$. Заметим, что если $s_1 = t_1$ или $s_2 = t_2$, задача сводится к случаю $p = 1$. Поэтому будем считать, что $s_1 \neq t_1$ и $s_2 \neq t_2$.

Пусть сначала вершины s_1, s_2, t_1, t_2 попарно различны. В силу эйлеровости все степени вершин графа $G^o \cup H^o$ четны, а в графе G^o есть ровно четыре вершины нечетной степени: s_1, s_2, t_1, t_2.

С помощью алгоритма Дейкстры (см. [17]) найдем кратчайшую цепь $\mu_1(s_1, t_1)$. При удалении цепи μ_1 из графа G^o мы получим граф $G^o \setminus \mu_1 = C_1 \cup \ldots \cup C_r$, где C_i $(i \in \overline{1,r})$ — это компоненты связности. Пусть для определенности $s_2 \in C_1$.

Удаление цепи μ_1 из графа G^o уменьшает на 1 степень вершин s_1 и t_1. Степень всех остальных вершин уменьшается на 0 или на 2. В итоговом графе степени только двух вершин будут нечетны, следовательно, они попадут в одну компоненту связности, поэтому $t_2 \in C_1$. В компоненте C_1 существует эйлерова цепь $\mu_2(s_2, t_2)$, которую можно найти с помощью алгоритма Хиргольцера.

Во всех остальных компонентах C_i $(i \in \overline{2,r})$ найдем эйлеровы циклы h_i. Каждая компонента C_i имеет хотя бы одну общую вершину v_i с цепью $\mu_1(s_1, t_1)$. Перестроим цепь μ_1, заменяя в ней вершины v_i на найденные циклы h_i (обход цикла будем начинать с v_i).

Полученные в результате цепи $\mu_1(s_1, t_1)$ и $\mu_2(s_2, t_2)$ являются покрывающими для графа G^o, причем все действия по их построению выполнялись с помощью полиномиальных алгоритмов.

Случаи, когда какие-то из вершин s_1, s_2, t_1, t_2 совпадают, обосновываются аналогично, отличие только в том, что у двух или четырех из этих вершин изначально будут четные степени в графе G^o. □

Следствие 1. *Найденные в доказательстве леммы 1 цепи $\mu_1(s_1, t_1)$ и $\mu_2(s_2, t_2)$ имеют хотя бы одну общую вершину.*

Это утверждение очевидно ввиду связности графа G^o.

Лемма 2. *Пусть для связного обычного графа G^o соответствующий граф $G^o \cup H^o$ является эйлеровым, при этом в задаче CT $p = 3$ и $t_1 = t_2 = t_3 = t$. Если в графе G^o есть мост, удаление которого оставляет s_1, s_2 и s_3 в одной компоненте связности, а t — в другой, то задача CT не имеет решения. Иначе задача CT всегда имеет решение, которое может быть найдено за полиномиальное время.*

Доказательство. Сначала рассмотрим случай, когда в графе G^o есть мост $e = \{x, y\}$ нужного вида. По теореме 3.2 из книги [18] любая цепь $\mu(s_i, t)$ $(i \in \overline{1,3})$ будет содержать ребро e, а значит, любые цепи между терминальными вершинами будут пересекаться по ребрам, следовательно, получить покрывающие цепи с заданными концами не удастся.

Отметим, что проверить наличие такого моста очень просто: достаточно с помощью алгоритма Дейкстры найти кратчайшую цепь $\mu(s_1, t)$ и, поочередно удаляя ее ребра, с помощью поиска в ширину проверить, не является ли какое-то из них мостом нужного вида. Описанная процедура проверки очевидно полиномиальна.

Пусть теперь в графе нет моста указанного вида. Если $t = s_1$, или $t = s_2$, или $t = s_3$, то одна из покрывающих цепей имеет нулевую длину и поиск двух оставшихся выполняется с помощью процедуры из доказательства леммы 1.

Пусть $t \neq s_i$ ($i \in \overline{1,3}$). В силу эйлеровости графа $G^o \cup H^o$ степени всех вершин в нем четны, а значит в графе G^o степень вершины t нечетна (при этом $\deg t = 3 + 2a$, $a \geqslant 0$) и степень либо одной, либо всех трех вершин s_i нечетна. Степень всех остальных вершин графа G^o четна.

С помощью алгоритма Дейкстры найдем кратчайшую цепь $\mu_1(s_1, t)$. При удалении цепи μ_1 из графа G^o мы получим граф $G^o \setminus \mu_1 = C_1 \cup \ldots \cup C_r$, где C_i ($i \in \overline{1,r}$) — это компоненты связности. Пусть для определенности $t \in C_1$.

Удаление цепи $\mu_1(s_1, t)$ влияет на степени вершин так же, как в лемме 1. Поэтому степень t станет четной. Если $s_2 \neq s_3$, то это будут единственные две вершины нечетной степени, следовательно, они попадут в одну компоненту связности C_j. Все компоненты C_i ($i \in \overline{1,r}$) имеют хотя бы одну общую вершину с цепью μ_1. Если предположить, что $j \neq 1$, то тогда между C_j и C_1 в цепи μ_1 есть мост, отделяющий s_1, s_2 и s_3 от t, но этого быть не может. Поэтому $s_2 \in C_1$ и $s_3 \in C_1$, значит, к C_1 применима лемма 1.

С оставшимися компонентами C_i ($i \in \overline{2,r}$) поступаем как в доказательстве леммы 1. В итоге мы за полиномиальное время получаем искомые покрывающие цепи. □

Теорема 4. *Пусть в кратном графе $G(V, E)$ выполнено какое-то из необходимых условий существования эйлерова маршрута (теоремы 1–3). Пусть при этом каждому множеству достижимости по обычным ребрам R^o_x инцидентно не более 4 связанных ребер каких-то мультиребер. Тогда задача MEW для этого графа разрешима за полиномиальное время.*

Кратный эйлеров маршрут в таком графе существует, если граф с квазивершинами $G_{ord}(V_{ord}, E_{ord})$ эйлеров. Исключение составляет единственный случай:

- *выполнено условие 2 или условие 3 теоремы 3;*
- *граф имеет кратность $k = 3$;*
- *в подграфе G^o_x, в который попадает начальная (конечная) вершина t кратной эйлеровой цепи, существует мост такой, что его удаление оставляет вершину t в одной компоненте связности, а три вершины, являющиеся концами одного мультиребра, — в другой.*

Доказательство. Как уже отмечалось выше, если кратный граф эйлеров, то эйлеровым будет и граф с квазивершинами, поэтому эйлеровость графа $G_{ord}(V_{ord}, E_{ord})$ является необходимым условием разрешимости задачи MEW.

1. Рассмотрим сначала случай, когда выполнено либо условие теоремы 1, либо условие теоремы 2, либо условие 1 теоремы 3. В этом случае нам нужно искать либо кратный эйлеров цикл, либо кратную эйлерову цепь с началом и концом в кратных вершинах. Эйлеровость графа G_{ord} при выполнении одного из указанных условий приводит к тому, что степени всех квазивершин четны. Это значит, что каждое множество достижимости по обычным ребрам R^o_x будет инцидентно либо 2, либо 4 связанным ребрам каких-то мультиребер. При прохождении эйлерова маршрута в графе G_{ord} в каждом подграфе G^o_x будет возникать столько же начальных вершин для задачи CT, сколько и конечных (при этом параметр $p = 1$ или $p = 2$). Из выполнения необходимых условий эйлеровости кратного графа G следует эйлеровость графа $G^o_x \cup H^o_x$. По лемме 1 каждая такая задача имеет решение, которое находится за полиномиальное время. Следовательно, любому эйлерову маршруту h_{ord} в графе G_{ord} будет соответствовать кратный эйлеров маршрут h в графе G. При этом маршрут h_{ord} можно найти с помощью полиномиального алгоритма Хиргольцера.

2. Пусть теперь выполнено условие 2 теоремы 3. В частности, это значит, что y — обычная вершина и найдется мультиребро $\{y_0, \{y_1, \ldots, y_k\}\} \in E^m$ такое, что $y_i \in R^o_y$ ($i \in \overline{1,k}$); $\deg y = k + b$, где k — это кратность графа, b — четное неотрицательное число. Заметим, что в этом случае кратность графа может принимать значения только от 2 до 4, иначе множеству R^o_y будет инцидентно более 4 связанных ребер какого-то мультиребра. Рассмотрим случаи с каждым вариантом кратности.

2.1. Если $k = 2$, то в графе G_{ord} есть квазивершина $q_{y,y}$ нечетной степени, с которой начинается эйлерова цепь, и эта квазивершина — висячая (значение $\deg q_{y,y} = 2$ невозможно, поскольку тогда данная квазивершина не будет являться начальной вершиной эйлеровой цепи в графе G_{ord}; а при $\deg q_{y,y} \geqslant 3$ вершины из R^o_y будут инцидентны минимум 6 связанным ребрам мультиребер). Выберем вершину y_1 в качестве начальной вершины кратной эйлеровой цепи, а при решении задачи СТ для подграфа G^o_y положим $s_1 = y_1$, $t_1 = y_2$. Если в графе G_{ord} есть квазивершина $q_{y,z}$ степени 2, в задаче СТ для подграфа G^o_y появятся еще терминальные вершины s_2, t_2. Как и в рассмотренном выше случае 1, граф $G^o_y \cup H^o_y$ — эйлеров и подзадача СТ для подграфа G^o_y соответствует условию леммы 1. Все остальные возникающие подзадачи решаются так же, как в случае 1.

2.2. Если $k = 3$, то в графе G_{ord} есть квазивершина $q_{y,y,y}$ нечетной степени, с которой начинается эйлерова цепь, и эта квазивершина — висячая. В соответствующем подграфе G^o_y однозначно выбирается начальная вершина t эйлеровой цепи (по сути, это единственная обычная вершина нечетной степени). Заметим, что в графе G_{ord} не может быть никакой квазивершины $q_{y,w,z}$: в силу эйлеровости графа G_{ord} такая вершина должна иметь четную степень, что приведет к появлению избыточного связанного ребра мультиребра, инцидентного R^o_y. В графе $G^o_y \cup H^o_y$ степень вершины t увеличится на 3 по сравнению со степенью этой вершины в кратном графе G, а остальные степени останутся неизменными (связанные ребра мультиребер заменятся на ребра (y_i, t), $i \in \overline{1,3}$), поэтому граф $G^o_y \cup H^o_y$ — эйлеров. Таким образом, для подграфа G^o_y задача СТ сводится к получению трех покрывающих цепей от t к y_i ($i \in \overline{1,3}$), что соотвествует условию леммы 2. Сформулированный в условии данной теоремы случай неразрешимости, по сути, определяется условием неразрешимости задачи СТ из леммы 2. Все остальные возникающие подзадачи решаются так же, как в рассмотренном выше случае 1.

2.3. Если $k = 4$, то в графе G_{ord} есть квазивершина $q_{y,y,y,y}$ нечетной степени, с которой начинается эйлерова цепь, и эта квазивершина — висячая. Аналогично случаю 2.2, в графе G_{ord} не может быть никакой другой квазивершины, содержащей y в качестве индекса, а граф $G^o_y \cup H^o_y$ — эйлеров. Рассмотрим G^o_y и решим задачу СТ, полагая $s_1 = y_1$, $t_1 = y_2$, $s_2 = y_3$, $t_2 = y_4$. По лемме 1 решение существует и находится за полиномиальное время. По следствию 1 покрывающие цепи будут иметь общую вершину. Выберем ее в качестве начальной вершины t кратного эйлерова цикла и преобразуем две найденные покрывающие цепи в G^o_y в четыре покрывающие цепи от t к y_i ($i \in \overline{1,4}$). Все остальные возникающие подзадачи решаются так же, как в рассмотренном выше случае 1.

3. Пусть теперь выполнено условие 3 теоремы 3. Тогда все рассуждения относительно начальной вершины кратной эйлеровой цепи из случая 2 нужно повторить и для конечной вершины этой цепи. Все остальные возникающие подзадачи решаются так же, как в рассмотренном выше случае 2.

Заметим, что во всех случаях мы разбивали нашу задачу на подзадачи СТ и поиска обычных эйлеровых маршрутов, каждая из которых рассматривалась однократно и решалась с помощью полиномиальных алгоритмов. Количество подзадач СТ определяется количеством множеств R^o_x, а это количество заведомо не превосходит количества обычных вершин в графе. Следовательно, задача MEW для графов рассматриваемого вида разрешима за полиномиальное время. □

Из доказательства теоремы 4 можно получить следующий полиномиальный алгоритм для рассмотренного подкласса кратных графов. Отметим, что этот алгоритм будет работать как для делимых, так и для произвольных кратных графов.

Алгоритм 1 (поиск кратного эйлерова маршрута в графе с ограничением на R^o_x)**.**

1. Проверяем выполнение необходимых условий существования кратного эйлерового цикла или цепи (теоремы 1–3). Если они не выполнены, кратного эйлерова маршрута не существует, выход.

2. Находим все R_x^o (полиномиальный алгоритм 2 из статьи [1]), строим граф с квазивершинами G_{ord} (полиномиальный алгоритм 1 из статьи [7]).
3. Проверяем, что каждому множеству достижимости по обычным ребрам R_x^o инцидентно не более 4 связанных ребер каких-то мультиребер.
4. С помощью алгоритма Хиргольцера находим в графе G_{ord} эйлеров цикл или эйлерову цепь h_{ord} (в зависимости от того, какое из необходимых условий истинно). Если h_{ord} не существует, задача MEW не имеет решения, выход.
5. Обходя маршрут h_{ord}, определяем стартовые и конечные вершины для задач CT в подграфах G_x^o.
6. Решаем каждую из задач CT с помощью полиномиальных алгоритмов так, как это описано в доказательстве лемм 1 и 2 и теоремы 4. Если реализовался негативный случай из теоремы 4, задача MEW не имеет решения, выход.
7. Находим искомый эйлеров маршрут h в кратном графе $G(V, E)$, перенося в кратный граф маршрут h_{ord} и объединяя его со всеми найденными покрывающими цепями в нужном порядке.

Оценим трудоемкость алгоритма 1.

На шаге 1 нам нужно посчитать степени всех вершин, однократно просмотрев каждое ребро, а затем проверить значение степени для каждой вершины на соответствие условиям теорем 1–3. Следовательно, трудоемкость данного шага — $O(|V| + |E|) = O(E)$.

На шаге 2 для получения всех R_x^o нужно однократно просмотреть все обычные ребра, а также просмотреть обычные вершины, инцидентные только отдельным связанным ребрам мультиребер, что дает трудоемкость $O(|V^o| + |E^o|) \leqslant O(|E|)$. Здесь V^o — множество обычных вершин. Построение графа с квазивершинами требует однократного просмотра всех кратных и мультиребер, однако для каждого из мультиребер нужно просмотреть отдельно k концов (k — кратность графа), поэтому трудоемкость этой части алгоритма составляет $O(|E^k| + k|E^m|) \leqslant O(k|E|)$.

Шаг 3 требует обхода всех концов мультиребер с проверкой, каким R_x^o они инцидентны, что требует $O(k|E^m|) \leqslant O(k|E|)$ действий.

Шаг 4 выполняет алгоритм Хиргольцера трудоемкости $O(|E_{ord}|) \leqslant O(|E|)$.

Обход маршрута h_{ord} на шаге 5 требует однократного просмотра всех ребер графа G_{ord} и соответствующих им ребер графа G, поэтому его трудоемкость — $O(|E|)$.

На шаге 6 требуется решить $s < |V|$ задач CT, подходящих под условие теоремы 4. Каждая такая задача решается в своем подграфе $G_x^o(R_x^o, E_x^o)$. При этом потребуется найти одну кратчайшую цепь (трудоемкость $O(|R_x^o|^2)$) и r эйлеровых маршрутов в непересекающихся подграфах $C_i \subset G_x^o$ ($i \in \overline{1, r}$), что в совокупности потребует $O(|E_x^o| + |R_x^o|^2) = O(|R_x^o|^2)$ действий. Решение всех задач CT потребует $O(s \cdot \max\{|R_x^o|^2\})$ шагов.

Перенос найденных маршрутов в кратный граф на шаге 7 потребует однократного просмотра ребер кратного графа и соответствующих им ребер графа G_{ord}, количество шагов — $O(|E|)$.

Каждый шаг выполняется однократно, следовательно, трудоемкость алгоритма 1 составляет $O(k|E| + s \cdot \max\{|R_x^o|^2\})$.

Отметим, что граф из примера 1 (рис. 3) соответствует условию теоремы 4, поэтому кратный эйлеров цикл, представленный на рис. 6, может быть найден с помощью алгоритма 1.

Пример 3. Рассмотрим более сложный пример на применение алгоритма 1. Пусть имеется кратный граф кратности $k = 2$, представленный на рис. 7. Концы мультиребра $\{13, \{35, 37\}\}$ лежат в одном множестве достижимости по обычным ребрам R_{31}^o, поэтому граф не является делимым.

Для этого графа выполняется условие 2 теоремы 3, при этом одним из концов возможной кратной эйлеровой цепи должна быть вершина 6, а другим — какая-то из вершин множества R_{31}^o.

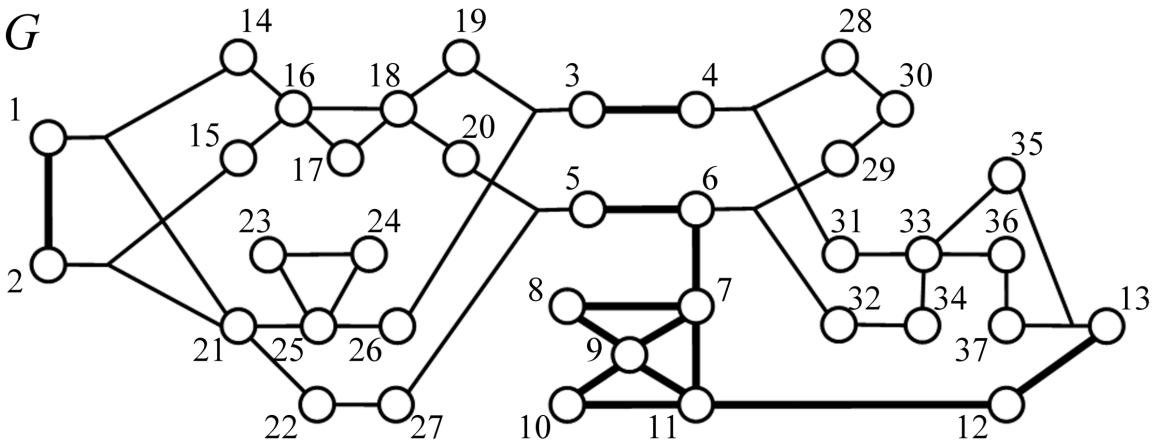


Fig. 7. Nondivisible graph of multiplicity 2

Рис. 7. Неделимый граф кратности 2

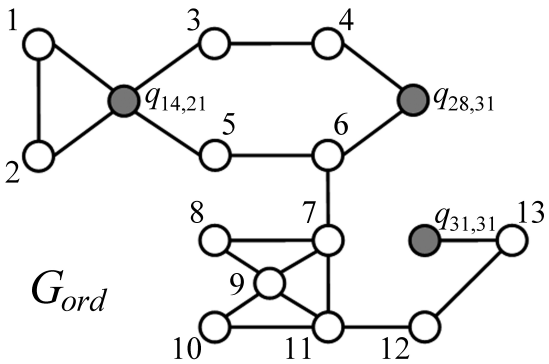


Fig. 8. Graph with quasi-vertices

Рис. 8. Граф с квазивершинами

Найдем все множества достижимости по обычным ребрам:

$$R^o_{14} = \{14, \ldots, 20\}, \quad R^o_{21} = \{21, \ldots, 27\}, \quad R^o_{28} = \{28, 29, 30\}, \quad R^o_{31} = \{31, \ldots, 37\}.$$

Рассмотрев все мультиребра, сформируем квазивершины $q_{14,21}$, $q_{28,31}$, $q_{31,31}$ и построим граф G_{ord} (рис. 8).

В исходном кратном графе G каждому множеству R^o_x инцидентно не больше 4 связанных ребер мультиребер, а граф G_{ord} содержит эйлерову цепь, поэтому задача соответствует условию теоремы 4. Найдем эйлерову цепь в графе G_{ord}:

$$h_{ord}(q_{31,31}, 6) = (q_{31,31}, 13, 12, 11, 10, 9, 11, 7, 9, 8, 7, 6, q_{28,31}, 4, 3, q_{14,21}, 2, 1, q_{14,21}, 5, 6).$$

Обходя эту цепь, определим терминальные пары для задач СТ в каждом подграфе G^o_x.

В G^o_{31} сначала нам нужно выбрать начальную вершину кратной эйлеровой цепи. Согласно алгоритму 1 это вершина 35. Тогда в данном множестве будет две терминальные пары: $(35, 37)$, $(32, 31)$. Для первой пары строим кратчайшую цепь $\mu(35, 37) = (35, 33, 36, 37)$. Удаляя эту цепь из G^o_{31}, получаем для второй пары простую эйлерову цепь $\mu(32, 31) = (32, 34, 33, 31)$.

В G^o_{21} терминальными являются пары $(26, 21)$ и $(21, 27)$. Для первой пары строим кратчайшую цепь $\mu(26, 21) = (26, 25, 21)$. Удаляя эту цепь из G^o_{21}, получаем для второй пары эйлерову цепь $\mu(21, 27) = (21, 22, 27)$, а также цикл $(25, 24, 23, 25)$, который нужно встроить в $\mu(26, 21)$, используя общую вершину 25: $\mu(26, 21) = (26, 25, 24, 23, 25, 21)$.

Аналогичным образом получаем две покрывающие цепи $\mu(19, 15) = (19, 18, 16, 15)$ и $\mu(14, 20) = (14, 16, 17, 18, 20)$ в G^o_{14}, а также одну покрывающую цепь $\mu(29, 28) = (29, 30, 28)$ в G^o_{28}.

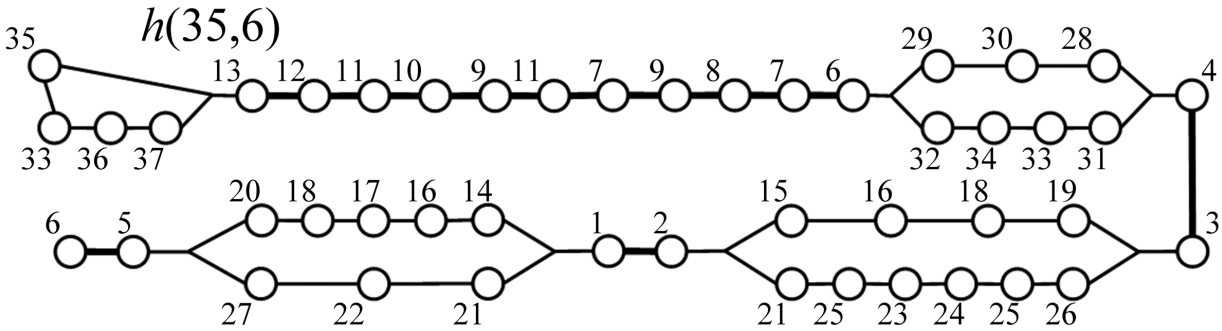


Fig. 9. Eulerian trail in the multiple graph G

Рис. 9. Эйлерова цепь в кратном графе G

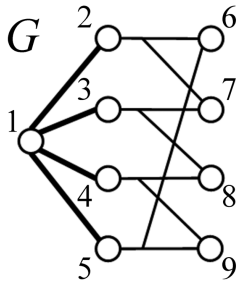


Fig. 10. Multiple graph G and graph with quasi-vertices G_{ord}

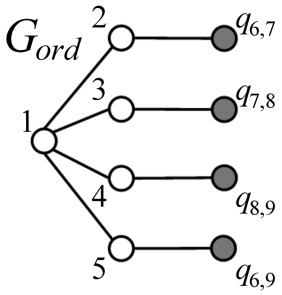


Рис. 10. Кратный граф G и граф с квазивершинами G_{ord}

Перенося эйлерову цепь $h_{ord}(q_{31,31}, 6)$ в граф G и объединяя ее с найденными покрывающими цепями в соответствующих G_x^o, получим кратную эйлерову цепь $h(35, 6)$, показанную на рис. 9.

Пример 4. Покажем также, что существуют кратные графы, для которых выполняется какое-то из необходимых условий эйлеровости, однако соответствующий граф с квазивершинами не эйлеров и, следовательно, в кратном графе невозможно получить эйлеров маршрут.

Слева на рис. 10 представлен делимый кратный граф, для которого выполнено условие теоремы 1. Более того, в каждой части G_1 и G_2 этого графа существует эйлеров цикл. Однако соответствующий граф с квазивершинами G_{ord} (справа на рис. 10) не является эйлеровым. Действительно, какие бы эйлеровы циклы мы ни получали в частях G_1 и G_2, их невозможно будет согласовать на общей части кратного графа G.

4. Делимые графы с ограничением на степень квазивершин

Теорема 5. *Пусть имеется делимый кратный граф $G(V, E)$, для которого выполнено необходимое условие существования эйлерова маршрута (теоремы 1–2). Пусть в графе с квазивершинами $G_{ord}(V_{ord}, E_{ord})$ степени всех квазивершин равны 2:* $\deg q_{y_1,\ldots,y_k} = 2$. *Пусть при этом каждый подграф G_x^o соответствует какому-либо случаю, когда задача CT для него может быть решена за полиномиальное время. Тогда задача MEW для графа $G(V, E)$ также может быть решена за полиномиальное время.*

Доказательство. Заметим, что выполнение условия теоремы 1 или теоремы 2 в сочетании с условием $\deg q_{y_1,\ldots,y_k} = 2$ влечет эйлеровость графа $G_{ord}(V_{ord}, E_{ord})$. При этом условие $\deg q_{y_1,\ldots,y_k} = 2$ гарантирует, что при обходе любого эйлерова маршрута в графе G_{ord} пары терминальных вершин (s_i, t_i) для задачи CT будут определяться однозначно в каждом подграфе G_x^o (с точностью до перестановки элементов внутри пар, которая не имеет значения ввиду неориентированности графа G).

Поэтому для нахождения кратного эйлерова маршрута в графе G достаточно с помощью полиномиального алгоритма Хиргольцера найти эйлеров маршрут h_{ord} в графе G_{ord}, затем произвести

обход этого маршрута для определения терминальных пар для каждой задачи СТ в подграфах G_x^o и решить полученные задачи СТ с помощью соответствующих полиномиальных алгоритмов. Если какая-то из задач СТ окажется неразрешимой, то тогда в графе G не существует кратного эйлерова маршрута. В противном случае объединение маршрута h_{ord} с решениями всех задач СТ порождает кратный эйлеров маршрут h в графе G.

Заметим, что мы разбили задачу поиска кратного эйлерова маршрута на одну подзадачу поиска обычного эйлерова маршрута и несколько подзадач СТ, каждая из которых решалась однократно с помощью полиномиального алгоритма. Количество подзадач СТ определяется количеством множеств R_x^o, а это количество заведомо не превосходит количества обычных вершин в графе. Следовательно, описанная процедура поиска кратного эйлерова маршрута полиномиальна. □

Из доказательства теоремы 5 можно получить следующий полиномиальный алгоритм для рассмотренного подкласса делимых кратных графов.

Алгоритм 2 (поиск кратного эйлерова маршрута в делимом графе с ограничением на степень квазивершин)**.**

1. Проверяем выполнение необходимых условий существования кратного эйлерового цикла или цепи (теоремы 1–2). Если они не выполнены, кратного эйлерова маршрута не существует, выход.
2. Находим все R_x^o (полиномиальный алгоритм 2 из статьи [1]), строим граф с квазивершинами G_{ord} (полиномиальный алгоритм 1 из статьи [7]).
3. Проверяем для всех квазивершин условие $\deg q_{y_1,\ldots,y_k} = 2$.
4. С помощью алгоритма Хиргольцера находим в графе G_{ord} эйлеров цикл или эйлерову цепь h_{ord} (в зависимости от того, какое из необходимых условий истинно).
5. Обходя маршрут h_{ord}, определяем стартовые и конечные вершины для задач СТ в подграфах G_x^o.
6. Решаем каждую из задач СТ с помощью соответствующих полиномиальных алгоритмов. Если какая-то из задач СТ не имеет решения, задача MEW также не имеет решения, выход.
7. Находим искомый эйлеров маршрут h в кратном графе $G(V, E)$, перенося в кратный граф маршрут h_{ord} и объединяя его со всеми найденными покрывающими цепями в нужном порядке.

Трудоемкость алгоритма 2 оценивается по тому же принципу, что и трудоемкость алгоритма 1. Единственное отличие состоит в том, что сложность шага 6 определяется сложностью самой трудоемкой из подзадач СТ (пусть это будет $O(A)$). В худшем случае сложность шага 6 будет составлять $O(s \cdot A)$, где s — количество подграфов G_x^o (но заведомо $s < |V|$), а сложность алгоритма в целом — $O(k|E| + s \cdot A)$.

Из делимости графа G и выполнения условия теоремы 1 или теоремы 2 (четность степеней обычных вершин) следует эйлеровость всех графов $G_x^o \cup H_x^o$. Выделим для таких графов некоторые случаи, когда задача СТ решается за полиномиальное время:

1. $p = 1$ или $p = 2$ (лемма 1).
2. $p = 3$ и $t_1 = t_2 = t_3 = t$ (лемма 2).
3. G_x^o — дерево. В этом случае между любыми двумя вершинами в графе G_x^o существует только одна цепь без повторяющихся ребер. Поэтому для решения задачи СТ найдем с помощью алгоритма Дейкстры кратчайшие цепи между всеми парами терминальных вершин и проверим, являются ли они покрывающими.
4. Задача EDP с теми же терминальными парами (s_i, t_i) $(i \in \overline{1, p})$, что и задача СТ, разрешима за полиномиальное время. В этом случае нужно найти решение задачи EDP — не пересекающиеся по ребрам пути $\mu_i(s_i, t_i)$ $(i \in \overline{1, p})$. При удалении найденных путей из графа G_x^o мы получим

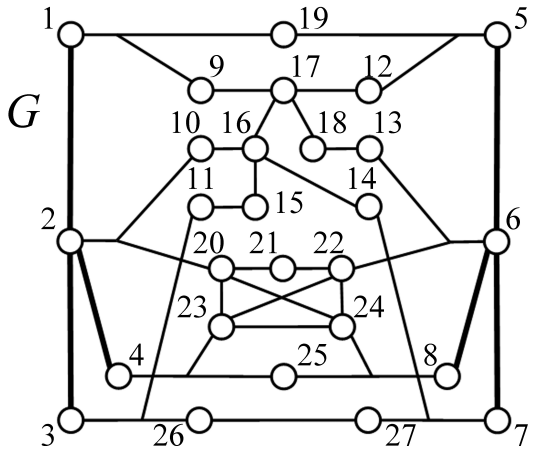


Fig. 11. Divisible graph G and graph with quasi-vertices G_{ord}

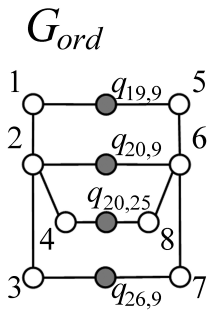


Рис. 11. Делимый граф G и граф с квазивершинами G_{ord}

граф $G_x^o \setminus (\mu_1 \cup \ldots \cup \mu_p) = C_1 \cup \ldots \cup C_r$, где C_j $(j \in \overline{1,r})$ — какие-то компоненты связности. Как и в доказательстве леммы 1, каждая компонента C_j является эйлеровой. При этом она имеет хотя бы одну общую вершину v_j с каким-то путем μ_i. Поэтому с помощью алгоритма Хиргольцера найдем эйлеров цикл h_j в каждой компоненте C_j и встроим его в соответствующий путь μ_i, заменяя вершину v_j на цикл h_j (обход начинаем с v_j). В результате мы получим искомое решение задачи CT.

В случае когда все подграфы G_x^o соответствуют пунктам 1–3, трудоемкость алгоритма 2 будет составлять $O(k|E| + s \cdot \max\{|R_x^o|^2\})$.

Относительно последнего пункта заметим, что отыскание полиномиальных подклассов задачи EDP представляет собой весьма нетривиальную проблему. Так, даже в случае когда граф $G_x^o \cup H_x^o$ является эйлеровым, задача EDP остается NP-трудной, что обосновывается в статье [19]. В работе [20] показано, что для любого фиксированного p (p не является входным параметром) можно получить полиномиальный алгоритм решения задачи EDP, однако этот алгоритм будет практически не реализуем. Как правило, графы, для которых существуют хорошие полиномиальные алгоритмы решения задачи EDP, имеют весьма специфическую структуру (см., например, [21, 22]).

Отметим также, что условие теоремы 5 нельзя ослабить. Если мы уберем условие полиномиальности задачи CT, то тогда ослабленному условию будет соответствовать граф, который использовался в доказательстве NP-полноты задачи распознавания MEW в статье [14]. Если же допустить, что для квазивершин может быть $\deg q_{y_1,\ldots,y_k} = 4$, это приведет к неоднозначности выбора терминальных пар в задачах CT. Тогда в случае неразрешимости какой-то из них нужно будет рассмотреть иной вариант выбора терминальных пар, для чего потребуется выполнить одну или несколько каппа-трансформаций эйлерова маршрута в графе G_{ord}; в худшем случае число каппа-трансформаций будет экспоненциально.

Пример 5. Рассмотрим пример на применение алгоритма 2. Пусть имеется делимый кратный граф кратности $k = 2$, представленный слева на рис. 11. Этот граф соответствует условию теоремы 1, будем искать в нем кратный эйлеров цикл. Как и ранее, найдем все множества достижимости по обычным ребрам, сформируем соответствующие квазивершины и построим граф с квазивершинами G_{ord} (справа на рис. 11).

Степени всех квазивершин равны 2, подграфы G_{19}^o, G_{20}^o, G_{25}^o и G_{26}^o подходят под условие леммы 1, а подграф G_9^o является деревом. Следовательно, графы G и G_{ord} соответствуют условию теоремы 5.

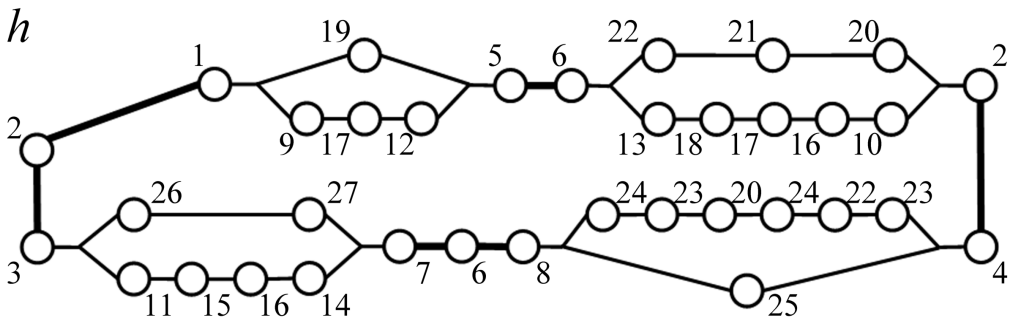


Fig. 12. Eulerian cycle in the divisible graph G

Рис. 12. Эйлеров цикл в делимом графе G

Найдем в графе G_{ord} эйлеров цикл:

$$h_{ord} = (1, q_{19,9}, 5, 6, q_{20,9}, 2, 4, q_{20,25}, 8, 6, 7, q_{26,9}, 3, 2, 1).$$

Обходя этот цикл, определим терминальные пары для задач СТ в каждом подграфе G^o_x.

В G^o_9 терминальными являются пары $(9, 12)$, $(13, 10)$ и $(14, 11)$. Поскольку G^o_9 — дерево, найдем кратчайшие цепи между вершинами всех терминальных пар и убедимся, что они являются покрывающими. Это цепи $\mu(9, 12) = (9, 17, 12)$, $\mu(13, 10) = (13, 18, 17, 16, 10)$ и $\mu(14, 11) = (14, 16, 15, 11)$.

Две покрывающие цепи $\mu(22, 20) = (22, 21, 20)$ и $\mu(23, 24) = (23, 22, 24, 20, 23, 24)$ в G^o_{20} получаются так же, как в примере 3, а для подграфов G^o_{19}, G^o_{25} и G^o_{26} задача СТ тривиальна.

Перенося эйлеров цикл h_{ord} в граф G и объединяя его с найденными покрывающими цепями в соответствующих G^o_x, получим кратный эйлеров цикл h, показанный на рис. 12.

References

[1] A. V. Smirnov, “The shortest path problem for a multiple graph”, *Automatic Control and Computer Sciences*, vol. 52, no. 7, pp. 625–633, 2018. DOI: 10.3103/S0146411618070234.

[2] V. S. Rublev and A. V. Smirnov, “Flows in multiple networks”, *Yaroslavsky Pedagogichesky Vestnik*, vol. 3, no. 2, pp. 60–68, 2011, in Russian.

[3] A. V. Smirnov, “The problem of finding the maximum multiple flow in the divisible network and its special cases”, *Automatic Control and Computer Sciences*, vol. 50, no. 7, pp. 527–535, 2016. DOI: 10.3103/S0146411616070191.

[4] L. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton University Press, 1962, 216 pp.

[5] V. S. Roublev and A. V. Smirnov, “The problem of integer-valued balancing of a three-dimensional matrix and algorithms of its solution”, *Modeling and Analysis of Information Systems*, vol. 17, no. 2, pp. 72–98, 2010, in Russian.

[6] A. V. Smirnov, “Network model for the problem of integer balancing of a four-dimensional matrix”, *Automatic Control and Computer Sciences*, vol. 51, no. 7, pp. 558–566, 2017. DOI: 10.3103/S0146411617070185.

[7] A. V. Smirnov, “The algorithms for the Eulerian cycle and Eulerian trail problems for a multiple graph”, *Modeling and Analysis of Information Systems*, vol. 30, no. 3, pp. 264–282, 2023, in Russian. DOI: 10.18255/1818-1015-2023-3-264-282.

[8] C. Hierholzer, “Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren”, *Mathematische Annalen*, vol. 6, no. 1, pp. 30–32, 1873, in German. DOI: 10.1007/BF01442866.

[9] C. Berge, *Graphs and Hypergraphs*. North-Holland Publishing Company, 1973, 528 pp.

[10] Z. Lonc and P. Naroski, "On tours that contain all edges of a hypergraph", *The Electronic Journal of Combinatorics*, vol. 17, R144, 2010. DOI: 10.37236/416.

[11] A. Marino and A. Silva, "Eulerian walks in temporal graphs", *Algoritmica*, vol. 85, no. 3, pp. 805–830, 2023. DOI: 10.1007/s00453-022-01021-y.

[12] S. W. Bent and U. Manber, "On non-intersecting Eulerian circuits", *Discrete Applied Mathematics*, vol. 18, no. 1, pp. 87–94, 1987. DOI: 10.1016/0166-218X(87)90045-X.

[13] S. Jimbo, "The NP-completeness of Eulerian recurrent length for 4-regular Eulerian graphs", in *Proceedings of the 2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology*, 2014, pp. 155–159. DOI: 10.1109/ICAIET.2014.34.

[14] A. V. Smirnov, "NP-completeness of the Eulerian walk problem for a multiple graph", *Modeling and Analysis of Information Systems*, vol. 31, no. 1, pp. 102–114, 2024, in Russian. DOI: 10.18255/1818-1015-2024-1-102-114.

[15] J. Abrham and A. Kotzig, "Transformations of Euler tours", *Annals of Discrete Mathematics*, vol. 8, pp. 65–69, 1980. DOI: 10.1016/S0167-5060(08)70852-5.

[16] R. M. Karp, "On the computational complexity of combinatorial problems", *Networks*, vol. 5, no. 1, pp. 45–68, 1975. DOI: 10.1002/net.1975.5.1.45.

[17] E. W. Dijkstra, "A note on two problems in connexion with graphs", *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959. DOI: 10.1007/BF01386390.

[18] F. Harary, *Graph theory*. Addison-Wesley Pub. Co., 1969, 274 pp.

[19] M. Middendorf and F. Pfeiffer, "On the complexity of the disjoint paths problem", *Combinatorica*, vol. 13, pp. 97–107, 1993. DOI: 10.1007/BF01202792.

[20] N. Robertson and P. D. Seymour, "Graph minors. XIII. The disjoint paths problem", *Journal of Combinatorial Theory, Series B*, vol. 63, no. 1, pp. 65–110, 1995. DOI: 10.1006/jctb.1995.1006.

[21] N. Alon and M. Capalbo, "Finding disjoint paths in expanders deterministically and online", in *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, 2007, pp. 518–524. DOI: 10.1109/FOCS.2007.19.

[22] D. Wagner and K. Weihe, "A linear-time algorithm for edge-disjoint paths in planar graphs", *Combinatorica*, vol. 15, no. 1, pp. 135–150, 1995. DOI: 10.1007/BF01294465.