MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 27, NO. 3, 2020
journal homepage: www.mais-journal.ru

THEORY OF COMPUTING

Efficient Equivalence Checking Technique for Some Classes of Finite-State Machines

V. A. Zakharov[1]

DOI: 10.18255/1818-1015-2020-3-260-303

[1]National Research University Higher School of Economics (HSE), 20 Myasnitskaya ulitsa, Moscow, 101000, Russia.

MSC2020: 68Q70
Research article
Full text in Russian

Received August 25, 2020
After revision September 7, 2020
Accepted September 9, 2020

Finite transducers, two-tape automata, and biautomata are related computational models descended from the concept of Finite-State Automaton. In these models an automaton controls two heads that read or write symbols on the tapes in the one-way mode. The computations of these three types of automata show many common features, and it is surprising that the methods for analyzing the behavior of automata developed for one of these models do not find suitable utilization in other models. The goal of this paper is to develop a uniform technique for building polynomial-time equivalence checking algorithms for some classes of automata (finite transducers, two-tape automata, biautomata, single-state pushdown automata) which exhibit certain features of the deterministic or unambiguous behavior. This new technique reduces the equivalence checking of automata to solvability checking of certain systems of equations over the semirings of languages or transductions. It turns out that such a checking can be performed by the variable elimination technique which relies on some combinatorial and algebraic properties of prefix-free regular languages. The main results obtained in this paper are as follows:

1. Using the algebraic approach a new algorithm for checking the equivalence of states of deterministic finite automata is constructed; time complexity of this algorithm is $O(n \log n)$.
2. A new class of prefix-free finite transducers is distinguished and it is shown that the developed algebraic approach provides the equivalence checking of transducers from this class in quadratic time (for real-time prefix-free transducers) and cubic (for prefix-free transducers with ε-transitions) relative to the sizes of analysed machines.
3. It is shown that the equivalence problem for deterministic two-tape finite automata can be reduced to the same problem for prefix-free finite transducers and solved in cubic time relative to the size of the analysed machines.
4. In the same way it is proved that the equivalence problem for deterministic finite biautomata can be solved in cubic time relative to the sizes of analysed machines.
5. By means of the developed approach an efficient equivalence checking algorithm for the class of simple grammars corresponding to deterministic single-state pushdown automata is constructed.

Keywords: transducer; two-tape automaton; biatomaton; simple grammar; equivalence checking; prefix-free language; language equation; decision procedure.

INFORMATION ABOUT THE AUTHORS

Vladimir Anatolyevich Zakharov
correspondence author

orcid.org/0000-0002-3794-9565. E-mail: zakh@cs.msu.ru
Doctor of Science, Professor.

Funding: This work was supported by the Russian Foundation for Basic Research, Grant N 18-01-00854.

For citation: V. A. Zakharov, "Efficient Equivalence Checking Technique for Some Classes of Finite-State Machines", *Modeling and analysis of information systems*, vol. 27, no. 3, pp. 260-303, 2020.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 27, № 3, 2020
сайт журнала: www.mais-journal.ru

THEORY OF COMPUTING

Эффективные алгоритмы проверки эквивалентности для некоторых классов автоматов

В. А. Захаров[1]

DOI: 10.18255/1818-1015-2020-3-260-303

[1]Национальный исследовательский университет «Высшая школа экономики», ул. Мясницкая, 20, г. Москва, 101000 Россия.

УДК 517.9
Научная статья
Полный текст на русском языке

Получена 25 августа 2020 г.
После доработки 7 сентября 2020 г.
Принята к публикации 9 сентября 2020 г.

Конечные преобразователи, двухленточные автоматы и биавтоматы — взаимосвязанные вычислительные модели, ведущие свое происхождение от концепции конечного автомата. В вычислениях этих машин проявляется много общих черт, и удивительно, что методы анализа, разработанные для одной из указанных моделей, не находят подходящего применения в других моделях. Целью данной статьи является разработка единой методики построения быстрых алгоритмов проверки эквивалентности для некоторых классов автоматов (конечных преобразователей, двухленточных автоматов, биавтоматов, магазинных автоматов), которые демонстрируют определенные черты детерминированного или однозначное поведение. Этот новый метод сводит проверку эквивалентности автоматов к проверке разрешимости систем уравнений над полукольцами языков или бинарных отношений. Как оказалось, такую проверку достаточно просто провести методом исключения переменных, используя некоторые комбинаторные и алгебраические свойства регулярных префиксных языков. Основные результаты, полученные в этой статье, таковы.

1. При помощи алгебраического метода построен новый алгоритм проверки эквивалентности детерминированных конечных автоматов, имеющий сложность по времени $O(n \log n)$.
2. Выделен новый класс префиксных конечных трансдьюсеров и показано, что проверка эквивалентности трансдьюсеров из этого класса может быть осуществлена новым методом за время, квадратичное (для префиксных трансдьюсеров реального времени) и кубическое (для префиксных трансдьюсеров с ε-переходами) относительно размеров анализируемых автоматов.
3. Показано, что проблема эквивалентности для детерминированных двухленточных конечных автоматов сводится к задаче проверки эквивалентности префиксных конечных трансдьюсеров и может быть решена за время, кубическое относительно их размеров.
4. Аналогичным образом установлена разрешимость проблемы эквивалентности для детерминированных конечных биавтоматов за время, кубическое относительно их размеров.
5. При помощи нового метода построен алгоритм проверки эквивалентности для простых грамматик, соответствующих детерминированным магазинным автоматам с одним состоянием.

Ключевые слова: трансдьюсер; двухленточный автомат; биавтомат; простая грамматика; проверка эквивалентности; префиксный язык; языковое уравнение; разрешающая процедура.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Владимир Анатольевич Захаров
автор для корреспонденции

orcid.org/0000-0002-3794-9565. E-mail: zakh@cs.msu.ru
Доктор физ.-мат. наук, профессор.

Финансирование: Работа выполнена при финансовой поддержке гранта РФФИ N 18-01-00854.

Для цитирования: V. A. Zakharov, “Efficient Equivalence Checking Technique for Some Classes of Finite-State Machines”, *Modeling and analysis of information systems*, vol. 27, no. 3, pp. 260-303, 2020.

Введение

Конечные автоматы-преобразователи (трансдьюсеры), двухленточные автоматы с конечным числом состояний (2-FSAs) и конечные биавтоматы — это взаимосвязанные модели вычислений, ведущие свое происхождение от концепции конечного автомата. В этих моделях программа с конечным числом состояний управляет двумя головками, каждая из которых считывает или записывает буквы на лентах, перемещаясь при этом в одну сторону. Конечные трансдьюсеры и двухленточные автоматы вычисляют один и тот же класс рациональных бинарных отношений на словах, но вычисления в этих моделях проводятся по-разному: трансдьюсер считывает слова на входной ленте и делает записи на выходной ленте, а 2-FSA считывает слова попеременно на двух входных лентах. Обе модели допускают взаимную трансляцию друг в друга, и поэтому многие авторы не делают различия между ними, когда речь заходит о рациональных отношениях. Биавтоматы проводят вычисления на единственной входной ленте и способны распознавать линейные контекстно-свободные языки, прочитывая входные слова сразу с двух концов. Взаимосвязь между конечными трансдьюсерами и биавтоматами основана на простом факте, отмеченном в статье [1]: для каждого рационального бинарного отношения R множество слов $L = \{uv^{-1} : (u, v) \in R\}$ является линейным контекстно-свободным языком.

Указанные модели, несмотря на сходство их вычислительных возможностей, имеют разные области применения. При изучении и применении автоматов-преобразователей обычно ограничиваются рассмотрением трансдьюсеров реального времени; эти автоматы проводят запись на выходной ленте только в ответ на считывание очередного символа на входной ленте. Трансдьюсеры реального времени находят применение во многих областях информатики, таких как: вычислительная лингвистика и обработка текстов [2], биоинформатика [3], проектирование реагирующих и распределенных систем [4—7]. Задачи оптимизации и верификации информационных систем и математических моделей, возникающие в этих приложениях, могут быть сведены к проблемам проверки эквивалентности и минимизации конечных автоматов-преобразователей (см. [7—9]).

Область применения многоленточных автоматов (m-FSAs) не столь обширна. В статье [10] было показано, что задача оптимизации строковых предикатов, которые используются для представления и обработки строк в базах данных, может быть сведена к проблеме минимизации для m-FSAs. В статье [11] было отмечено, что проблема проверки эквивалентности для стандартных схем программ, содержащих только унарные предикаты вида $p(x)$ и операторы присваивания вида $x := f(x)$, взаимосводима к проблеме эквивалентности для детерминированных многоленточных автоматов (m-DFSAs). Трудность решения последней из указанных задач оказала значительное влияние на развитие математических методов оптимизации программ.

Биавтоматы были впервые введены в статье [12] и независимо в работе [13] (в первых публикациях они были названы линейными автоматами) как более простая альтернатива магазинным автоматам (PDAs) для обработки контекстно-свободных (к-с) языков, порожденных линейными грамматиками. Впоследствии в нескольких публикациях было проведено подробное изучение вычислительных возможностей биавтоматов [14—16] и алгоритмических аспектов задач анализа их поведения [14, 17]. Биавтоматы относятся к числу наиболее простых моделей вычислений, способных распознавать нерегулярные к-с языки, и можно полагать, что они найдут подходящее применение в решении задач синтаксического анализа и обработки текстов. В этом случае будет весьма полезно иметь эффективные алгоритмы проверки эквивалентности и минимизации для биавтоматов.

Упомянутые здесь три модели вычислений имеют немало общих черт, и поэтому вызывает удивление то обстоятельство, что алгоритмические задачи анализа их поведения до сих пор не рассматривались с единой точки зрения. Главная цель этой статьи — представить вниманию читателей общий метод построения эффективных (в том числе полиномиальных по времени) алгоритмов про-

верки эквивалентности для нескольких взаимосвязанных классов автоматов (конечных трансдьюсеров, двухленточных автоматов, биавтоматов, магазинных автоматов с единственным состоянием управления), вычисления которых обладают свойствами детерминированности или однозначности. Чтобы повернуть Землю, нужна точка опоры. В поисках ее целесообразно провести обзор известных ранее результатов, касающихся разрешимости проблемы эквивалентности для конечных трансдьюсеров, многоленточных и магазинных автоматов, и попытаться разработать такой подход, который был бы пригоден для всех этих автоматных моделей. Вначале мы рассмотрим ранее известные результаты исследования задачи проверки эквивалентности конечных автоматов-преобразователей, поскольку изучению и решению этой проблемы для разнообразных классов трансдьюсеров было посвящено довольно много работ.

Исследование проблемы эквивалентности для трансдьюсеров (машин с конечным множеством состояний) проводится с начала 60-х. Его можно условно разделить на три периода. Вначале П. Фишер и А. Розенберг [18] доказали алгоритмическую неразрешимость этой задачи для конечных трансдьюсеров произвольного вида путем сведения к ней проблемы соответствий Поста. Вслед за этим Т. Гриффитс [19] получил аналогичный результат для класса трансдьюсеров реального времени, и немного позднее О. Ибарра [20] сумел показать, что алгоритмическая неразрешимость проблемы эквивалентности сохраняется даже в том случае, когда трансдьюсеры работают над однобуквенным выходным алфавитом.

Доказательства, приведенные в статьях [18—20], показывают, что неразрешимость возникает в тех случаях, когда некоторые входные слова могут иметь произвольно много образов. Поэтому на следующем этапе исследований внимание было сосредоточено на изучении проблемы эквивалентности для трансдьюсеров с ограниченным числом значений. Вначале удалось установить, что эта проблема разрешима для функциональных [21, 22], детерминированных [23] и конечно неоднозначных [24] трансдьюсеров. Кроме того, в статье [24] было показано, что проверку эквивалентности можно провести за полиномиальное время, если анализируемые трансдьюсеры являются однозначными и, в частности, детерминированными. Были также предложены полиномиальные по времени алгоритмы проверки того, является ли заданный тренсдьюсер реального времени ограниченно-значным [25], или k-значным для любого фиксированного k [24]. Трансдьюсер считается k-значным, если для любого входного слова множество его образов содержит не более k выходных слов. Что касается проблемы эквивалентности для k-значных трансдьюсеров, ее разрешимость была установлена в статье [26], однако предложенная в этой работе разрешающая процедура не позволяла получить оценки сложности решения этой проблемы. Более совершенный подход, основанный на представлении произвольного k-значного трансдьюсера в виде прямой суммы конечного числа функциональных трансдьюсеров, был предложен А. Вебером в статье [27] (см. также [28]). Ему удалось получить верхнюю оценку длины входного слова, на котором может проявляться неэквивалентность заданных k-значных трансдьюсеров и, таким образом, показать, что проблема эквивалентности для этого класса трансдьюсеров разрешима за время, ограниченное двойной экспонентой от размера анализируемых автоматов. Разрешающая процедура, описанная в статье [28], основывается исключительно на комбинаторных соображениях, и этим объясняется столь высокая верхняя оценка ее сложности.

В стремлении уменьшить сложность процедур проверки эквивалентности для ограниченно-значных трансдьюсеров Ж. Сакарович и Р. де Суза провели систематическую ревизию методов, предложенных в работах [27, 28]. В серии статей [29—33] им удалось модифицировать метод декомпозиции k-значных трансдьюсеров и разработать улучшенный способ проверки эквивалентности, который в значительно большей мере принимает в расчет структуру анализируемых трансдьюсеров. Предложенные усовершенствования позволили построить полиномиальные по времени процедуры проверки свойства k-значности трансдьюсеров реального времени и экспоненциальные

по времени разрешающие процедуры для проверки эквивалентности k-значных трансдьюсеров. Существенно более простой метод верификации был описан в статье [34]. Для его применения не требуется проводить предварительную декомпозицию рассматриваемых трансдьюсеров, он работает непосредственно с их отношениями переходов и позволяет анализировать поведение трансдьюсеров реального времени, которые оперируют не только над выходными словами, но и над полугруппами. В работе [34] было показано, что эквивалентность k-значных трансдьюсеров можно проверить за экспоненциальное время в том случае, если полугруппа выходных элементов вложима в группу, в которой проблема проверки тождеств разрешима за полиномиальное время.

Как видно из приведенного здесь краткого обзора по проблеме эквивалентности для конечных автоматов-преобразователей, ограниченно-значные трансдьюсеры долгое время находились в центре внимания. Главная цель исследований состояла в проведении как можно более точной демаркации границы между разрешимыми и неразрешимыми случаями проблемы эквивалентности. Оказалось, что наиболее обширным среди известных классов преобразователей, для которых проблема эквивалентности разрешима, является класс преобразователей с ограниченной степенью длины (см. [35]). В статье [36] было показано, что эквивалентность трансдьюсеров из этого класса можно проверить за время, которое оценивается сверху тройной экспонентой, зависящей от размера анализируемых автоматов. Столь высокая трудоемкость разрешающей процедуры вряд ли позволит использовать ее в каких-либо приложениях. Возможно, разумнее было бы изменить цель исследования и попытаться обнаружить классы трансдьюсеров, для которых проблема эквивалентности допускает решение за практически приемлемое время. В статье [24] было показано, что эквивалентность однозначных трансдьюсеров может быть проверена за полиномиальное время. Однако авторы статьи [24] рассматривали свойство однозначности (unambiguity) конечных трансдьюсеров лишь применительно к действиям считывания символов на входной ленте. Возможно, что ослабление этого свойства позволит выделить новый, более обширный класс автоматов, для которого проблема эквивалентности будет также разрешима за полиномиальное время.

Неразрешимость проблемы эквивалентности для недетерминированных 2-FSAs была доказана в статье [18]. Но вскоре М. Берд [37] показал, что эта задача разрешима для 2-DFSAs. Аналогичный вывод следует также из результатов работы Л. Вэлианта [38]. Интересная особенность 2-DFSAs состоит в том, что проблема включения для этого класса автоматов неразрешима (см. [37]). Автор статьи [39] улучшил алгоритм Вэлианта и получил верхнюю оценку временно́й сложности $2^{O(n^6)}$. Позднее Э. Фридман и Ш. Грейбах обнаружили, что эквивалентность 2-DFSAs можно проверять за полиномиальное время. В статье [40] они предложили два алгоритма. Первый из них прост для понимания, но он имеет экспоненциальную временну́ю сложность. Более изощренный второй алгоритм основан на идеях первого и проверяет эквивалентность 2-DFSA за время $O(n^{12})$. Авторы статьи [40] также коротко пояснили, как можно улучшить этот алгоритм, чтобы достичь временно́й сложности $O(n^4)$. И, наконец, в 1991 г. разрешимость проблемы эквивалентности для детерминированных автоматов с произвольным числом лент была установлена Т. Харью и Ю. Кархюмяки в статье [41]. Используя серию замечательных результатов из теории групп, они показали, как проверить, имеют ли два недетерминированных многоленточных автомата одинаковое количество допускающих прогонов на каждом входе. Из описания разрешающей процедуры, предложенной в [41], видно, что проблема эквивалентности для n-DFSA принадлежит классу сложности **co-NP**. Как ни странно, но за прошедшие три десятилетия в изучении этой проблемы не было достигнуто значительного прогресса. Тем не менее, необходимо отметить статью [42]; ее автор переформулировал стратегию принятия решений, разработанную в работе [41], в терминах матричной алгебры и построил рандомизированный алгоритм для проверки эквивалентности n-DFSA за полиномиальное время.

Возможно, дальнейший прогресс в изучении n-DFSAs может быть достигнут путем установления и использования отношений между многоленточными автоматами и другими вычислительными моделями. Учитывая тесную взаимосвязь между конечными преобразователями и 2-FSAs, можно предположить, что процедуры проверки эквивалентности, разработанные для определенных классов конечных преобразователей, могут быть адаптированы для анализа 2-DFSAs.

Вычислительные возможности биавтоматов изучались в работах [12, 14, 43]. Р. Луканова [12] доказала, что недетерминированные биавтоматы могут распознавать все к-с языки, порождаемые линейными грамматиками. Что касается вычислительных возможностей детерминированных биавтоматов, в статье [43] было показано, что эта модель вычислений несравнима с детерминированными магазинными автоматами (DPDAs): при помощи детерминированных биавтоматов можно распознавать язык палиндромов, который не принадлежит семейству детерминированных к-с языков, хотя детерминированные биавтоматы не могут распознавать нелинейные к-с языки. Авторы статьи [14] провели подробное исследование детерминированных биавтоматов и получили много интересных алгоритмических и теоретико-множественных результатов о свойствах этой модели вычислений. Одна из наиболее значимых задач, которая оставалась нерешенной (см. [14]), — это проблема эквивалентности для детерминированных биавтоматов. И хотя глубокое изучение этой задачи фактически еще не начиналось, интенсивные исследования аналогичной проблемы были проведены для некоторых других родственных вычислительных моделей, таких как магазинные автоматы и к-с грамматики.

В статье [44] было доказано, что проблема эквивалентности для к-с грамматик и магазинных автоматов алгоритмически неразрешима. Из доказательства видно, что неразрешимость имеет место для линейных к-с грамматик (а, значит, и для недетерминированных биавтоматов). Но спустя несколько лет А. Кореньяк и Дж. Хопкрофт [45] обнаружили, что проблема эквивалентности разрешима для $LL(1)$-грамматик в нормальной форме Грейбах (т.н. простых грамматик), которые соответствуют детерминированным магазинным автоматам с единственным состоянием управления. Разрешающая процедура, предложенная в статье [45], проверяет разрешимость систем языковых уравнений, соответствующих правилам вывода анализируемых грамматик. Было показано, что $LL(1)$-языки обладают свойством префиксности, и оно позволяет проводить проверку разрешимости указанных систем уравнений с помощью равносильных преобразований. Авторы статьи [45] показали, что эти равносильные преобразования можно применять так, чтобы размер систем уравнений экспоненциально зависел от размера анализируемых грамматик; таким образом, время работы разрешающего алгоритма оценивается сверху двойной экспонентой от размера проверяемых грамматик. Стоит отметить также, что, как и в случае многоленточных автоматов, проблема включения для простых грамматик алгоритмически неразрешима (см. [46]). Д. Кокаль в статье [47] сумел преодолеть эффект двойного экспоненциального взрыва. Он следовал тем же путем, что и авторы работы [45], но использовал более изощренные свойства префиксных языков в равносильных преобразованиях систем языковых уравнений. Кокаль показал, как можно проверять разрешимость таких систем уравнений с помощью метода исключения переменных, и сократил временну́ю сложность разрешающей процедуры до одинарной экспоненты. Присутствие экспоненциального множителя в этой оценке обусловлено тем, что длина кратчайшего слова, которое порождается к-с грамматикой и определяет размер системы уравнений, может зависеть экспоненциально от размера грамматики. Исследовательская группа в составе С. Бастьена, Ю. Чижовича и др. в статье [48] сумела уменьшить оценку сложности алгоритма Кокаля, воспользовавшись полиномиальным алгоритмом сравнения пар слов, представленных в сжатой форме т.н. “прямыми программами” (к-с грамматиками, порождающими языки, которые состоят из одного слова), и получили, таким образом, полиномиальный по времени алгоритм проверки эквивалентности простых грамматик (ранее полиномиальный алгоритм более высокой сложности был предложен в статье [49]).

Приведенный обзор позволяет сделать несколько важных выводов.

За редким исключением, все работы, посвященные исследованию проблемы эквивалентности для указанных выше автоматных моделей вычислений, имеют фрагментарный характер: авторы выделяют некоторый класс автоматов и пытаются построить алгоритм проверки эквивалентности, существенно использующий особенности вычислений автоматов этого класса. Несмотря на хорошо известные и вполне очевидные взаимосвязи вычислений автоматов из разных классов, очень редко предпринимались попытки перенести результаты и адаптировать методы решения проблемы эквивалентности, полученные для одних классов автоматов, на другие классы моделей вычислений. В связи с этим можно задаться вопросом о возможности создания такого метода проверки эквивалентности автоматных моделей вычислений, который, с одной стороны, обладал бы определенной универсальностью и был бы в равной мере применим к автоматам разных типов, а, с другой стороны, позволял бы получать эффективные алгоритмы решения проблемы эквивалентности за счет учета специфических особенностей вычислений некоторых классов автоматов.

В большинстве работ авторы преследовали вполне естественную цель выделить как можно более обширные классы автоматов с разрешимой проблемой эквивалентности. Однако по мере расширения “области разрешимости” этой проблемы алгоритмы проверки эквивалентности автоматов становились все более трудоемкими и менее практичными. Подходящим примером здесь могут служить магазинные автоматы. Разрешимость проблемы эквивалентности для сравнительно узких классов DPDAs была установлена в статьях [38, 45, 50]; предложенные алгоритмы имели не менее чем экспоненциальную временну́ю сложность. Впоследствии для этих классов автоматов были построены полиномиальные по времени или имеющие сложность, близкую к этой оценке, алгоритмы проверки эквивалентности (см. [48, 51, 52]). В то же время, многие исследователи [53—59] неустанно расширяли “область разрешимости” проблемы эквивалентности для DPDAs. Методы построения разрешающих алгоритмов становились все более изощренными и, наконец, Ж. Синезерг [60] сумел доказать разрешимость проблемы эквивалентности для произвольных DPDAs. Однако предложенная разрешающая процедура оказалась чрезвычайно трудоемкой и совершенно непрактичной: единственное, что удалось выяснить о ее сложности — это то, что она может быть оценена примитивно рекурсивной функцией [61]. Этот пример показывает, что нередко по мере расширения области применения разрешающих алгоритмов их прикладная ценность значительно снижается. Таким образом, желательно разработать такой метод построения алгоритмов проверки эквивалентности различных типов автоматов, который позволял бы гарантировать высокую эффективность разрешающих процедур, построенных с его помощью, при выполнении определенных требований, которые налагаются на устройство анализируемых автоматов. Как можно видеть из статьи [62], такой подход может приводить к довольно интересным практическим результатам.

Для достижения поставленной цели нужно выбрать оптимальный подход к построению разрешающих алгоритмов. Среди разнообразия методов проверки эквивалентности программ можно выделить три подхода, имеющих наиболее широкое применение: метод оценки длины контрпримера, метод анализа совместных вычислений и метод проверки разрешимости систем уравнений. В первом из них оценивается длина кратчайшего контрпримера — входных данных, на которых неэквивалентные программы заданного размера или структуры вычисляют разные результаты. Если такая оценка известна, то для проверки эквивалентности двух программ (автоматов) достаточно сравнить результаты их вычислений на всех входных данных, размер которых не превосходит этой оценки. Этот метод был успешно использован в работах [21, 27, 28, 41]. Недостаток метода состоит в том, что он требует переборной проверки большого объема входных данных, и с его помощью нельзя построить быстрых алгоритмов верификации. В методе совместных вычислений для пары проверяемых программ (автоматов) строится система переходов, вычисления которой моделируют

всевозможные пары вычислений этих программ на одинаковых входных данных. В построенной системе переходов допускающим объявляется такое состояние совместных вычислений пары программ, достижение которого явно свидетельствует о том, что эти программы способны получать на одинаковых данных разные результаты. Таким образом, задача проверки эквивалентности программ сводится к задаче проверки пустоты в том классе систем переходов, которые моделируют совместные вычисления этих программ.

Метод совместных вычислений нашел применение во многих работах [33, 34, 36—38, 40, 51, 55, 57, 59]. Наибольшую трудность в его использовании составляет выбор подходящей системы переходов для моделирования совместных вычислений двух программ: такие системы должны быть достаточно "подробными", чтобы соответствовать семантике проверяемых программ и различать результаты их вычислений, и, вместе с тем, они должны быть достаточно простыми для эффективного решения проблемы пустоты. Например, совместные вычисления двух DPDAs можно очень просто моделировать при помощи автоматов с двумя магазинами, но проблема пустоты для этой модели вычислений неразрешима. Одна из заслуг Л. Вэлианта состоит в том, что в статье [38] ему удалось выделить такой класс DPDAs, совместные вычисления которых можно моделировать при помощи автоматов с одним магазином, и, таким образом, получить решение проблемы эквивалентности для DPDAs с конечным числом поворотов магазина. Изучению необходимых и достаточных условий, которым должны удовлетворять системы переходов для успешного их применения в методе совместных вычислений, посвящена статья [63].

В алгебраическом методе проверки эквивалентности вычисления любой программы (автомата) описываются при помощи системы уравнений: каждой точке (состоянию управления) программы ставится в соответствие переменная, и уравнения системы описывают зависимость результатов, вычисляемых программой в одних состояниях, от данных, которые были вычислены программой в смежных состояниях. Если к такой системе добавить уравнение, приравнивающее друг другу переменные, соответствующие выходам этих программ, то получим систему уравнений, разрешимость которой равносильна эквивалентности анализируемых программ. Поэтому для проверки эквивалентности программ достаточно уметь проверять разрешимость соответствующих систем уравнений. Преимущества этого подхода ясны: алгебраизация задачи позволяет воспользоваться для ее решения обширным арсеналом выразительных средств и методов современной алгебры. Алгебраический метод проверки эквивалентности использовался в статьях [45, 47, 48, 52, 58, 60, 64]. Взаимосвязь проблем эквивалентности для автоматов и разрешимости уравнений над множествами слов была исследована в статье [65].

Произвольные системы языковых уравнений трудны для решения — с их помощью можно задавать любые рекурсивно перечислимые языки. Но, как показано в статье [66], даже системы линейных языковых уравнений в общем случае эффективно неразрешимы. Существенную роль здесь играют свойства языков, используемых в уравнениях в качестве коэффициентов. Одним из таких полезных свойств является свойство префиксности языков — язык L называется префиксным, если ни одно слово w из L не является префиксом других слов этого языка. Опираясь именно на свойство префиксности языков, авторам статьи [45] удалось доказать разрешимость проблемы эквивалентности для простых к-с языков. В последующих работах [47, 48], посвященных изучению этой задачи, префиксное свойство также сыграло решающую роль. Важное значение этого свойства для эффективного решения языковых уравнений отмечалось в работе [65]: была доказана разрешимость проблемы эквивалентности конечных префиксных подстановок на заданных к-с языков, а также проблемы эквивалентности для одного класса недетерминированных трансдьюсеров. При помощи префиксного свойства Ж. Сенизергу [52] удалось показать, что проблема эквивалентности для DPDA с конечным числом поворотов магазина принадлежит классу сложности **co-NP**. Перечисленные результаты свидетельствуют о том, что для построения эффективных алгоритмов проверки

эквивалентности целесообразно обратить внимание на классы автоматов, устройство которых тесно связано с префиксными языками.

Цель данной статьи — выделить такие классы автоматов (трансдьюсеров, многоленточных автоматов, биавтоматов, магазинных автоматов), для которых можно получить эффективное решение проблемы эквивалентности при помощи алгебраического метода, опираясь на свойства префиксности регулярных языков.

Основные результаты, полученные в этой статье, таковы.

1. Разработан новый метод проверки эквивалентности автоматов различных типов; в этом методе задача проверки эквивалентности автоматов сводится к задаче проверки разрешимости систем линейных языковых уравнений, и эта задача решается при помощи традиционных алгебраических приемов с существенным использованием свойств регулярных префиксных языков.
2. При помощи алгебраического метода построен новый алгоритм проверки эквивалентности состояний детерминированных конечных автоматов, имеющий сложность по времени $O(n \log n)$.
3. Выделен новый класс префиксных конечных трансдьюсеров и показано, что проверка эквивалентности трансдьюсеров из этого класса может быть осуществлена новым методом за время, квадратичное (для префиксных трансдьюсеров реального времени) и кубическое (для префиксных трансдьюсеров общего вида) относительно размеров анализируемых автоматов.
4. Показано, что проблема эквивалентности для детерминированных двухленточных конечных автоматов сводится к задаче проверки эквивалентности префиксных конечных трансдьюсеров и может быть решена за время, кубическое относительно их размеров.
5. Установлена разрешимость проблемы эквивалентности для детерминированных конечных биавтоматов за время, кубическое относительно их размеров.

В этой статье также показано, что предложенный метод проверки разрешимости систем языковых уравнений можно также использовать для анализа систем нелинейных уравнений и с его помощью получать простые алгоритмы проверки эквивалентности для некоторых классов магазинных автоматов и к-с грамматик, в частности, $LL(1)$-грамматик и соответствующих им детерминированных магазинных автоматов с единственным состоянием управления.

Статья организована следующим образом. Раздел 1 содержит минимальный набор основных понятий из теории формальных языков, которые являются общими для всех последующих разделов. В разделе 2 на примере простейшей модели детерминированных конечных автоматов описаны основные положения того алгебраического метода проверки эквивалентности автоматных моделей вычислений, которому посвящена эта статья. Применение описанного метода для более сложных типов автоматов, которые рассматриваются в данной статье, требует использования некоторых свойств регулярных префиксных языков. Поэтому следующий раздел 3 посвящен префиксным языкам: доказаны свойства замкнутости класса префиксных языков относительно некоторых теоретико-языковых операций (конкатенации, итерации, деления на слова слева и справа), рассмотрены автоматные способы описания префиксных языков и установлены оценки сложности выполнения указанных операций над префиксными языками, описанными при помощи конечных автоматов. В разделе 4 введены основные понятия теории конечных автоматов-преобразователей (обобщенных машин с конечным числом состояний, трансдьюсеров). В разделе 5 описан новый класс трансдьюсеров реального времени — префиксных трансдьюсеров — и показано, как проблема эквивалентности для трансдьюсеров из этого класса может быть сведена к задаче проверки разрешимости систем линейных языковых уравнений, в которых коэффициентами и свободными членами являются конечные слова, образующие конечные префиксные языки. В этом же разделе описан квадратичный по сложности алгоритм проверки разрешимости указанных систем

языковых уравнений; в его основу положен метод Гаусса исключения переменных, используемый в линейной алгебре для решения систем линейных уравнений. В этом алгоритме существенно используются свойства префиксных языков, которые были установлены в разделе 3. В следующем разделе 6 исследован еще более обширный класс недетерминированных трансдьюсеров, в которых допустимы переходы без считывания входных букв — класс префиксных однородных трансдьюсеров с ε-переходами. В линейных уравнениях, описывающих поведение трансдьюсеров из этого класса, коэффициентами и свободными членами являются регулярные префиксные языки, которые представлены детерминированными конечными автоматами. Процедура проверки разрешимости таких систем уравнений, основанная на методе исключения переменных, несколько усложняется. Она так же использует особенности префиксных языков, отмеченные в разделе 3, и позволяет проверять эквивалентность префиксных однородных трансдьюсеров с ε-переходами за время, кубическое относительно их размеров. Далее, в разделе 7 исследована проблема эквивалентности для детерминированных двухленточных автоматов. Показано, что класс бинарных отношений (трансдукций), распознаваемых автоматами из этого класса, совпадает с классом бинарных отношений, вычисляемых префиксными однородными трансдьюсерами с ε-переходами. Обнаруженное соответствие позволяет провести трансляцию детерминированных двухленточных автоматов в префиксные однородные трансдьюсеры и, воспользовавшись результатами раздела 6, построить кубический по времени алгоритм проверки эквивалентности детерминированных двухленточных автоматов. В разделе 8 метод проверки разрешимости систем линейных языковых уравнений, разработанный в разделе 3, подходящим образом адаптирован для решения проблемы эквивалентности детерминированных конечных биавтоматов; показано, что эта задача может быть решена за время, кубическое относительно размеров проверяемых биавтоматов. Метод исключения переменных применим не только для проверки разрешимости систем линейных языковых уравнений; в разделе 9 показано, как при помощи этого метода можно эффективно проверять разрешимость систем языковых уравнений, порожденных $LL(1)$-грамматиками. Здесь так же решающую роль играет свойство префиксности $LL(1)$-языков. Перспективы дальнейшего развития и применения предложенного в этой статье алгебраического подхода к разработке алгоритмов проверки эквивалентности различных типов автоматов обсуждаются в заключительном разделе 9.

1. Основные понятия

Алфавитом называется всякое конечное непустое множество букв $\Sigma = \{a_1, \ldots, a_k\}$. *Слово* в алфавите Σ — это любая конечная последовательность $w = a_{i_1} a_{i_2} \ldots a_{i_n}$ букв из Σ. Пустое слово обозначим символом ε, а множество всех слов в алфавите Σ обозначим записью Σ^*. Длина $|w|$ слова w — это количество букв в w. Для произвольной пары слов $u = a_{i_1} \ldots a_{i_n}$ и $v = a_{j_1} \ldots a_{j_m}$ запись uv будет обозначать их *конкатенацию*, которая является словом $a_{i_1} \ldots a_{i_n} a_{j_1} \ldots a_{j_m}$. Для любого слова $w = a_{i_1} a_{i_2} \ldots a_{i_n}$ запись w^{-1} будет обозначать *обратное слово* $a_{i_n} \ldots a_{i_2} a_{i_1}$.

Предположим, что слово w является конкатенацией слов u и v, т.е. $w = uv$. Тогда слово u является *префиксом* w, слово v — *суффиксом* w, а слово w — *расширением* слова u. В том случае, если $w = uv$ мы будем называть слово u *частным при делении справа* слова w на слово v (и обозначать записью $u = w/v$), а слово v — *частным при делении слева* слова w на слово u (и обозначать записью $v = w \setminus u$. Два слова w_1 и w_2 будут называться *совместными*, если одно из них является префиксом другого; в противном случае эти слова будут считаться *несовместными*.

Язык L в алфавите Σ — это любое подмножество множества слов Σ^*. *Конкатенацией языков* L_1 и L_2 является язык $L_1 L_2 = \{uv \ : \ u \in L_1, v \in L_2\}$. Если $L_1 = \emptyset$ или $L_2 = \emptyset$, то $L_1 L_2 = \emptyset$. Если $L_1 = \{u\}$, то мы будем использовать запись uL_2 для обозначения конкатенации L_1 и L_2. *Итерацией языка* L будем называть язык $L^* = \{\varepsilon\} \cup L \cup LL \cup \ldots$. *Обращением языка* L будем называть язык $L^{-1} = \{w^{-1} : w \in L\}$. *Частным при делении справа* языка L на слово w считается язык $L/w = \{u : uw \in L\}$ частных при делении справа всех слов языка L на слово w, а *частным при делении слева* языка L на слово w

считается язык $L \setminus w = \{u : wu \in L\}$ частных при делении слева всех слов языка L на слово w. Необходимо иметь в виду, что в том случае, если ни одно слово из L не имеет w в качестве суффикса (префикса), то $L/w = \emptyset$ (соответственно, $L \setminus w = \emptyset$).

Трансдукцией в алфавитах Σ и Δ называется всякое подмножество T множества пар слов $\Sigma^* \times \Delta^*$. *Областью определения* трансдукции T называется язык $Dom(T) = \{u : \exists w : (u, w) \in T\}$, а *образом* трансдукции T называется язык $Im(T) = \{w : \exists u : (u, w) \in T\}$. Язык $Im(T, u) = \{w : (u, w) \in T\}$ называется *образом слова u в трансдукции T*. *Конкатенацией трансдукций* R_1 и R_2 является трансдукция $R_1R_2 = \{(u_1u_2, v_1v_2) : (u_1, v_1) \in R_1, (u_2, v_2) \in R_2\}$.

Обозначим записью $Reg(\Sigma)$ семейство регулярных языков в алфавите Σ. Регулярные языки обычно описываются с помощью конечных автоматов. *Детерминированный конечный автомат* (DFA) в алфавите Σ задается системой переходов $A = \langle Q, q_0, F, \varphi \rangle$, в которой Q — это конечное множество *состояний*, q_0 — *начальное состояние*, $F \subseteq Q$ — подмножество *допускающих состояний*, и $\varphi : Q \times \Sigma \longrightarrow Q$ — частичная функция переходов. Тройки (q, x, q'), в которых $\varphi(q, x) = q'$, называются *переходами* DFA A и обозначаются записями $q \xrightarrow{x} q'$. DFA A *допускает* слово $w = a_1a_2 \dots a_n$, если существует такая последовательность переходов $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} q_{n-1} \xrightarrow{a_n} q_n$, в которой q_0 является начальным состоянием, а q_n — допускающим состоянием автомата A. DFA A *описывает* регулярный язык $L(A)$, представляющий собой множество всех слов w, которые допускаются автоматом A. *Размер* DFA A — это число его состояний $|Q|$. Далее DFA A с начальным состоянием q условимся обозначать записью $A(q)$.

Для описания конечных семейств регулярных языков мы будем использовать *детерминированные конечные автоматы с несколькими множествами допускающих состояний* (multi-DFAs) в алфавите Σ. Такой автомат задается системой переходов $D = \langle Q, q_0, F_1, \dots, F_m, \varphi \rangle$, в которой Q, q_0, и φ имеют тот же смысл, что и для DFAs, а $F_1, \dots, F_m$ — это подмножества допускающих состояний. Такой multi-DFA D описывает семейство регулярных языков $\{L(D_1), \dots, L(D_m)\}$, где $D_i = \langle Q, q_0, F_i, \varphi \rangle$ — это обычный DFA для каждого $i, 1 \le i \le m$.

Вполне естественно ожидать, что DFAs и multi-DFAs не имеют бесполезных состояний. DFA (multi-DFA) A называется *сокращенным*, если его допускающие состояния достижимы из каждого состояния автомата A. В дальнейшем по умолчанию будем подразумевать, что все DFAs и multi-DFAs, которые рассматриваются в этой статье, являются сокращенными.

2. Алгебраический метод проверки эквивалентности автоматов

Алгебраический метод проводит проверку эквивалентности двух машин (автоматов, программ) в два этапа. На первом этапе формируется система уравнений, которая служит алгебраической спецификацией задачи проверки эквивалентности двух заданных машин. Выбор алгебры, в которой может быть построена подходящая система уравнений, зависит, прежде всего, от операционной семантики того типа машин, для которого изучается проблема эквивалентности. На втором этапе проверяется разрешимость построенной системы уравнений. Проверка проводится путем применения равносильных преобразований; эти преобразования либо приводят проверяемую систему к заранее выделенной приведенной форме, которая наверняка имеет решение, либо обнаруживают присутствие в системе несовместных уравнений, которые свидетельствуют о том, что такая система не имеет решений. Применяемые равносильные преобразования могут учитывать специфические особенности устройства проверяемых машин, такие как детерминированность, однозначность, префиксность и др.

Основные положения этого метода объясним на примере решения проблемы эквивалентности для детерминированных конечных автоматов: для заданного DFA A и пары его состояний p и q проверить эквивалентность этих состояний автомата. Два состояния p и q некоторого DFA $A = \langle Q, q_0, F, \varphi\rangle$ считаются эквивалентными, если для DFAs $A(p) = \langle Q, p, F, \varphi\rangle$ и $A(q) = \langle Q, q, F, \varphi\rangle$, имеющих состояния p и q в качестве начальных, верно равенство $L(A(p)) = L(A(q))$.

Пусть задан DFA $A = \langle Q, q_0, F, \varphi\rangle$, работающий в алфавите Σ. Для проверки эквивалентности пары состояний p и q этого автомата построим систему уравнений $\mathcal{E}(A, p, q) = \mathcal{E}(A) \cup \{X_p = X_q\}$, состоящую из двух частей. Подсистема $\mathcal{E}(A)$ представляет собой спецификацию DFA A в алгебре регулярных выражений, а уравнение $X_p = X_q$ выражает требование эквивалентности состояний p и q. Эту систему уравнений можно составить и эффективно проверить ее разрешимость для любого DFA, однако, для упрощения изложения далее будем полагать, что рассматриваемый DFA A является сокращенным, т.е. из любого его состояния достижимо некоторое допускающее состояние.

Для построения системы уравнений $\mathcal{E}(A)$ каждому состоянию $r \in Q$ поставим в соответствие индивидуальную переменную X_r и константу δ_r, которая равна 1, если r — допускающее состояние, и 0 в противном случае. Уравнения системы $\mathcal{E}(A)$ формируются из регулярных выражений (р-выражений). Значения р-выражений — это языки в алфавите Σ. Р-выражения строятся из переменных X_r, соответствующих состояниям $r \in Q$, и констант при помощи операций конкатенации $\cdot$ и альтернации +; константами являются 0, 1, а также буквы алфавита Σ. Значениями констант 0 и 1 являются соответственно пустой язык $L_0 = \emptyset$ и язык $L_1 = \{\varepsilon\}$, состоящий только из пустого слова, а значением каждой константы $a \in \Sigma$ является язык $L_a = \{a\}$, состоящий только из однобуквенного слова a. Мы будем неявно использовать тождества алгебры р-выражений: $0 \cdot H = H \cdot 0 = 0$, $0 + H = H + 0 = H$

Система уравнений (1) хорошо известна в теории автоматов — она используется в некоторых алгоритмах построения регулярных выражений для языков, допускаемых DFAs (см., например, [67]). Она имеет вид

$$\mathcal{E}(A) = \{X_r = \sum_{a\in\Sigma} a \cdot H_{r,a} + \delta_r \ : \ r \in Q\}, \tag{1}$$

$$\text{где } H_{r,a} = \begin{cases} X_{\varphi(r,a)}, & \text{если значение функции переходов } \varphi(r,a) \text{ определено,} \\ 0, & \text{в противном случае.} \end{cases}$$

Как показано в учебнике [67], для любого сокращенного DFA A система уравнений (1) имеет единственное решение $\{X_r = L(A(r)) \ : \ r \in Q\}$, причем все языки $L(A(r))$ непусты. Отсюда следует, что состояния p и q эквивалентны тогда и только тогда, когда система уравнений $\mathcal{E}(A, p, q)$ имеет решение. Так задача проверки эквивалентности состояний DFA сводится к задаче проверки разрешимости системы линейных уравнений в алгебре р-выражений.

Решение системы линейных уравнений $\mathcal{E}(A, p, q)$ проводится традиционным для линейной алгебры методом Гаусса полного исключения переменных. В случае применения этого метода для проверки эквивалентности состояний DFA системы уравнений $\mathcal{E}_t$, которые образуются на каждой итерации t, состоят из двух частей: первая часть — это подсистема *базовых уравнений* $\mathcal{E}(A')$ для некоторого сокращенного DFA A' (этот автомат может быть отличен от исходного заданного DFA A), а вторая часть — это множество *проверочных уравнений* вида $X_i = X_j$, в обеих частях которых присутствуют только переменные. Проверочное уравнение $X_i = X_j$ из системы уравнений $\mathcal{E}_t$ указанного вида и переменная X_i из левой части этого уравнения называются *приведенными*, если X_i не имеет других вхождений в системе. Система уравнений $\mathcal{E}$ называется *приведенной*, если все проверочные уравнения этой системы являются приведенными. Нетрудно заметить, что любая приведенная система уравнений имеет решение.

В неприведенной системе уравнений $\mathcal{E}_t$ переменная X_i из левой части хотя бы одного проверочного уравнения $X_i = X_j$ имеет другие вхождения в уравнения системы $\mathcal{E}_t$. Такую переменную X_i будем называть *активной* переменной этой системы уравнений. В частности, исходная система уравнений $\mathcal{E}_1 = \mathcal{E}(A, p, q)$, соответствующая задаче проверки эквивалентности состояний p, q заданного DFA A, не является приведенной, и переменная X_p из левой части проверочного уравнения $X_p = X_q$ — это активная переменная.

Проверка разрешимости системы уравнений $\mathcal{E}(A, p, q)$ проводится итеративно путем применения равносильных преобразований. Эти преобразования деактивизируют одну за другой переменные проверяемой системы, до тех пор пока не будет получена либо приведенная система уравнений, которая, как было отмечено, всегда имеет решение, либо система, содержащая уравнения вида $X_i = 0$ или $0 = 1$, которые несовместны с остальными уравнениями системы и, тем самым, свидетельствуют о неразрешимости этой системы.

В начале первой итерации полагаем $\mathcal{E}_1 = \mathcal{E}(A, p, q)$. На каждой итерации t в неприведенной системе уравнений $\mathcal{E}_t$ выбирается активная переменная, проводится ее деактивизация ("исключение"), и затем восстанавливается (если это необходимо) канонический вид системы уравнений $\mathcal{E}_{t+1}$. Правила равносильных преобразований на t-й итерации таковы.

1) *Удаление тождеств.* Уравнения вида $X = X$ удаляются из системы $\mathcal{E}_t$.

2) *Проверка активности.* Если в систем $\mathcal{E}_t$ нет активных уравнений, то $\mathcal{E}_t$ — приведенная система уравнений. Тогда процедура проверки разрешимости завершает свое выполнение и объявляет о разрешимости исходной системы уравнений $\mathcal{E}(A, p, q)$.

3) *Деактивизация переменных.* В системе $\mathcal{E}_t$ выбирается проверочное уравнение $X_i = X_j$, содержащее активную переменную X_i, и во всех остальных уравнениях системы $\mathcal{E}_t$ все вхождения переменной X_i замещаются переменной X_j за счет применения подстановки $\{X_i/X_j\}$.

После этого шага уравнение $X_i = X_j$ становится приведенным, и число приведенных переменных в системе $\mathcal{E}_t$ увеличивается на 1. Однако применение подстановки $\{X_i/X_j\}$ к уравнениям системы $\mathcal{E}_t$ может давать побочный эффект: два базовых уравнения вида

$$\begin{aligned} X_j &= \sum_{a\in\Sigma} a \cdot H'_{j,a} + \delta'_j \\ X_j &= \sum_{a\in\Sigma} a \cdot H''_{j,a} + \delta''_j \end{aligned}$$

с одной и той же переменной X_j в левых частях появляются в системе $\mathcal{E}_t$. Это отклонение системы уравнений от канонической формы должно быть устранено.

4) *Устранение дублирующих базовых уравнений.* Удалим из системы $\mathcal{E}_t$ одно из двух указанных выше уравнений с переменной X_j в левой части и добавим в систему $|\Sigma|$ уравнений $H'_{j,a} = H''_{j,a}$ для каждой буквы $a \in \Sigma$, а также уравнение $\delta'_j = \delta''_j$. Нетрудно заметить, что после применения этого преобразования будет получена равносильная система уравнений, в которой могут содержаться "нестандартные" уравнения вида $X_r = 0$ или $c_1 = c_2$, где c_1, c_2 — константы.

5) *Устранение "нестандартных" уравнений.* Если среди добавленных уравнений есть уравнения вида $X_r = 0$ или $0 = 1$, то процедура проверки завершает свое выполнение и объявляет о неразрешимости исходной системы уравнений $\mathcal{E}(A, p, q)$. Если таких уравнений нет, т.е. среди добавленных уравнений есть только одно "нестандартное" уравнение (тождество) вида $0 = 0$ или $1 = 1$, то оно удаляется из системы.

Если в результате применения указанных правил процедура проверки еще не завершила свою работу, то будет получена система уравнений $\mathcal{E}_{t+1}$, которая равносильна системе $\mathcal{E}_t$, но имеет большее число приведенных переменных. После этого процедура проверки переходит к следующей итерации $t + 1$.

Утверждение 1. *Для любого сокращенного DFA A и пары его состояний p, q предложенная процедура после применения к системе уравнений $\mathcal{E}_1 = \mathcal{E}(A, p, q)$ завершает работу и дает правильный ответ на вопрос о разрешимости системы $\mathcal{E}_1$.*

Доказательство. После каждой итерации t число разрешенных переменных в системе уравнений $\mathcal{E}_{t+1}$ увеличивается, по меньшей мере, на единицу, и поэтому процедура проверки обязательно завершает работу. Преобразования систем уравнений $\mathcal{E}_t$ по правилам 1) и 3)–5) являются равносильными преобразованиями, и поэтому на каждой итерации t система уравнений $\mathcal{E}_t$ равносильна исходной системе $\mathcal{E}(A, p, q)$. Так как приведенная система уравнений всегда имеет решение, заключение о разрешимости системы уравнений $\mathcal{E}(A, p, q)$, которое процедура принимает на шаге 2), является правильным. Так как для сокращенного DFA система уравнений $\mathcal{E}(A)$ имеет единственное решение $X_r = L(A(r)) \neq \emptyset$ для любого состояния $r \in Q$, заключение о неразрешимости системы уравнений $\mathcal{E}(A, p, q)$, которое процедура принимает на шаге 5), является правильным. □

Таким образом, верна

Теорема 1. *Проблема эквивалентности для сокращенных DFAs разрешима за время $O(n \log n)$, где n — размер DFA.*

Доказательство. Согласно утверждению 1 описанная выше разрешающая процедура корректно проверяет эквивалентность пары состояний любого DFA A. Число итераций этой процедуры не превосходит числа состояний этого автомата. Чтобы сократить вычислительные издержки, данные, с которыми работает процедура проверки разрешимости системы уравнений $\mathcal{E}(A, p, q)$, можно представить при помощи ссылочных структур. Например, множество вхождений каждой переменной в p-выражения системы уравнений $\mathcal{E}(A, p, q)$ можно задать в виде слабо сбалансированного дерева: корень дерева представляет имя переменной X_r, а листовые вершины используются для ссылки на них всех вхождений переменной X_r в p-выражения проверяемой системы уравнений. Две основные операции, которые выполняет процедура проверки разрешимости систем уравнений, — сравнение имен переменных и подстановка одной переменной вместо всех вхождений другой, — могут быть выполнены за время, линейное относительно высоты деревьев, представляющих эти переменные. Применение подстановки X/Y можно осуществить двумя способами:

1. Корень дерева, представляющего одну из этих переменных, присоединяется к одной из наиболее близких к корню листовых вершин дерева, представляющего другую переменную, и образованное в результате дерево получает имя X;
2. Корни обоих деревьев, представляющих переменные X и Y, присоединяются к новому корню, и образованное в результате дерево получает имя X.

Так как общее число вхождений всех переменных в уравнения системы $\mathcal{E}(A, p, q)$ не превосходит величины $|\Sigma|n$, такое моделирование операции подстановки позволяет поддерживать во всех деревьях, представляющих вхождения переменных, высоту, не превосходящую величины $O(\log n)$. Таким образом, каждая итерация процедуры проверки разрешимости системы уравнений $\mathcal{E}(A, p, q)$ может быть выполнена за время, логарифмически зависящее от размера DFA A. □

Хотя в утверждении 1 говорится о применении процедуры проверки разрешимости только к системам уравнений, порожденным сокращенными DFAs, это ограничение не является принципиальным. Чтобы проверять разрешимость систем уравнений $\mathcal{E}(A, p, q)$ для произвольных DFAs, достаточно оснастить описанную процедуру двумя дополнительными правилами равносильных преобразований:

6) *Подстановка константы* 0. Если система $\mathcal{E}_t$ содержит уравнение $X_i = 0$, то во всех остальных уравнениях системы $\mathcal{E}_t$ все вхождения переменной X_i замещаются константой 0 за счет применения подстановки $\{X_i/0\}$.

7) *Распространение константы* 0. Если система $\mathcal{E}_t$ содержит уравнение $0 = a_{i_1} \cdot X_{j_1} + \cdots + a_{i_k} \cdot X_{j_k}$, то это уравнение удаляется из системы, и вместо него в систему добавляются уравнения $X_{j_1} = 0, \ldots, X_{j_k} = 0$.

Наиболее важным для успешного применения описанного метода проверки разрешимости систем уравнений является правило устранения дублирующих уравнений (правило 4), которое восстанавливает канонический вид системы после применения подстановки. Правила такого вида существенно зависят от специфических особенностей устройства автоматов рассматриваемого класса. В случае конечных автоматов такой особенностью является детерминированность автомата. Благодаря этому свойству уравнение

$$a_{i_1} \cdot X_{j_1} + \cdots + a_{i_k} \cdot X_{j_k} = a_{i_1} \cdot Y_{j_1} + \cdots + a_{i_k} \cdot Y_{j_k}$$

распадается на серию более простых уравнений $X_{j_1} = Y_{j_1}, \ldots, X_{j_k} = Y_{j_k}$, в левых частях которых стоят переменные, и система уравнений вновь принимает канонический вид. Например, для уравнений, описывающих недетерминированный конечный автомат, такого расщепления сделать невозможно. Поэтому метод исключения переменных применим для проверки разрешимости систем линейных языковых уравнений, которые порождены лишь такими автоматами (машинами, программами), которые обладают некоторой "однозначностью" поведения. Одним из проявлений этой особенности поведения автоматов является свойство префиксности тех языков, которые фигурируют в системах уравнений, описывающих работу автомата. Изучению этого свойства посвящен следующий раздел статьи.

3. Префиксные языки и их основные свойства

Предложенный в этой статье метод проверки эквивалентности конечных автоматов различных типов существенно опирается на некоторые простые свойства семейств попарно вполне несовместных префиксных регулярных языков. Язык L называется *префиксным*, если все его слова попарно несовместны. Согласно приведенному определению пустой язык $\emptyset$ является префиксным, и язык $L = \{\varepsilon\}$ — это единственный префиксный язык, содержащий пустое слово ε. Два языка L_1 и L_2 называются *совместными*, если каждое слово из одного языка совместно с некоторым словом из другого. Будем говорить, что язык L_1 *не имеет расширений в языке* L_2, если ни одно слово из языка L_2 не является расширением каких-либо слов из языка L_1. Языки L_1 и L_2 назовем *вполне несовместными* ни один из них не имеет расширений в другом, т.е. любые два слова $w_1 \in L_1$ и $w_2 \in L_2$ несовместны.

Сформулированные ниже утверждения следуют непосредственно из определений свойств несовместности, префиксности и отсутствия расширений в языках. Они являются ключевыми при построении эффективных разрешающих алгоритмов в разделах 5, 6, и 8.

Утверждение 2. *Если язык L_1 не имеет расширений в языке L_2, то для любого слова w и языка L верно, что*

- *язык L_1 не имеет расширений в языке L_2/w,*
- *язык $L_1 \setminus w$ не имеет расширений в языке $L_2 \setminus w$, и*
- *язык $L_1 L$ не имеет расширений в языке L_2.*

Утверждение 3. *Если языки L_1 и L_2 не имеют расширений в обоих языках L_3 и L_4, то $L_1 \cup L_2$ не имеет расширений в языке $L_3 \cup L_4$.*

Утверждение 4. *Если языки L_1 и L_2 вполне несовместны, то для любого языка L языки $L_1 L$ и L_2 также вполне несовместны.*

Утверждение 5. *Для любых префиксных языков L_1 и L_2, и всякого слова w языки $L_1 L_2$ и $L_1 \setminus w$ являются префиксными.*

Утверждение 6. *Для любой пары вполне несовместных префиксных языков L_1 и L_2 язык $L_1 \cup L_2$ является префиксным.*

Утверждение 7. *Если язык L_1 является префиксным, а язык L_2 не имеет расширений в языке L_3, то язык L_1L_2 также не имеет расширений в языке L_1L_3.*

Утверждение 8. *Для любой пары вполне несовместных префиксных языков L_1 и L_2 язык $L_1^*L_2$ является префиксным.*

Доказательство. Возьмем пару вполне несовместных префиксных языков L_1 и L_2.

1. Если $\varepsilon \in L_1$, то $L_2 = \emptyset$, поскольку пустое слово ε совместно с любым словом. Значит, язык $L_1^*L_2 = \emptyset$ является префиксным.

2. Рассмотрим случай $\varepsilon \notin L_1$. Предположим, что язык $L_1^*L_2$ не является префиксным, и пусть слово $w' = u'_1 \dots u'_k v'$, где $\{u'_1, \dots, u'_k\} \subseteq L_1$, $v' \in L_2$, — это кратчайшее слово из множества $L_1^*L_2$, которое совместно с некоторым другим словом $w'' = u''_1 \dots u''_m v''$ из множества $L_1^*L_2$, где $\{u''_1, \dots, u''_m\} \subseteq L_1$, $v'' \in L_2$.

Если $k > 0$, то непустое слово u'_1 может быть совместно либо со словом v'' (в случае $m = 0$), либо со словом u''_1 (в случае $m > 0$). Первая альтернатива невозможна, поскольку языки L_1 и L_2 вполне несовместны. Так как язык L_1 является префиксным, вторая альтернатива возможна только тогда, когда $u'_1 = u''_1$. Но отсюда следует, что более короткие слова $u'_2 \dots u'_k v'$ и $u''_2 \dots u''_m v''$ также совместны вопреки выбору слова w'.

Если $k = 0$, то v' совместно либо со словом u''_1 (в случае $m > 0$), либо со словом v'' (в случае $m = 0$). Первая альтернатива невозможна, поскольку языки L_1 и L_2 вполне несовместны. А поскольку язык L_2 является префиксным, вторая альтернатива возможна лишь при условии $v' = v''$. Отсюда следует $w' = w''$, что противоречит выбору слова w'.

Полученные противоречия приводят к заключению о том, что язык $L_1^*L_2$ является префиксным. □

Утверждение 9. *Если префиксный язык L_1 вполне несовместен с языком L_2, и при этом оба языка L_1 и L_2 не имеют расширений в языке L_3, то язык $L_1^*L_2$ также не имеет расширений в языке $L_1^*L_3$.*

Доказательство утверждения 9 проводится по той же схеме *reductio ad absurdum*, что и доказательство утверждения 8.

Таким образом, некоторые теоретико-языковые операции, такие как объединение, конкатенация, деление слева и справа, сохраняют свойства несовместности, префиксности и отсутствия расширений в языках.

Утверждение 10. *Пусть L' и L'' — конечные непустые префиксные совместные языки. Тогда существует единственное разбиение $L' = \bigcup_{i=1}^{n} L'_i$ и $L'' = \bigcup_{i=1}^{n} L''_i$ этих двух языков, при котором для каждого $i, 1 \le i \le n$, одно из подмножеств L'_i или L''_i состоит из одного слова $\{w\}$, а все слова из другого подмножества являются расширениями этого слова w.*

Доказательство. Воспользуемся индукцией по числу слов в множестве $L' \cup L''$. Базис индукции — случай, когда $|L'| = 1$ или $|L''| = 1$, — очевиден. Для обоснования индуктивного перехода рассмотрим кратчайшее слово u в множестве $L' \cup L''$. Предположим для определенности, что $w \in L'$. Тогда возьмем множество слов $\{w\}$ в качестве L'_1 и все слова из L'', которые совместны со словом w, в качестве L''_1. Так как w — это одно из кратчайших слов множества $L' \cup L''$, а L' и L'' — совместные языки, множество слов L''_1 непусто и состоит только из расширений слова w. Так как язык L' является префиксным, ни одно слово из L''_1 не является расширением или префиксом какого-либо

слова из множества $L' - \{w\}$. Таким образом, пара подмножеств L'_1 и L''_1 удовлетворяют требованиям утверждения 10 и определяется однозначно выбором слова $w \in L'_1$. Множества $L' - \{w\}$ и $L'' - L''_1$ — это непустые префиксные и совместные языки; по индуктивному предположению они имеют единственное разбиение, удовлетворяющее требованиям утверждения 10. □

Такое разбиение пары совместных префиксных языков $L' = \bigcup_{i=1}^{n} L'_i$ и $L'' = \bigcup_{i=1}^{n} L''_i$ будем называть *расщеплением* языков L' и L''. Пары соответствующих друг другу подмножеств L'_i и L''_i, $1 \le i \le n$, такого расщепления будут называться *долями*. Утверждение 10 сыграет ключевую роль в построении алгоритма проверки эквивалентности трансдьюсеров реального времени в разделе 5.

Обозначим записью $PFReg(\Sigma)$ семейство всех префиксных регулярных языков. Конечные автоматы, которые описывают префиксные языки, обладают важным характерным свойством, которое существенно облегчает работу с этими автоматами.

Утверждение 11 ([68, 69]). *Пусть $A = \langle Q, q_0, F, \varphi \rangle$ — сокращенный DFA в алфавите Σ. Тогда регулярный язык $L(A)$ является префиксным тогда и только тогда, когда значение функции переходов $\varphi(q, x)$ не определено для любого допускающего состояния q из F и любой буквы x из Σ, т.е. из допускающих состояний автомата A не исходит никаких переходов.*

Как следует из утверждения 11, если язык $L(A)$ является префиксным, то все допускающие состояния автомата A можно безопасно объединить в одно единственное допускающее тупиковое состояние f, у которого нет исходящих из него переходов. Автомат $A = \langle Q, q_0, \{f\}, \varphi \rangle$ такого вида будет называться *префиксным DFA*. Аналогично, каждое семейство регулярных языков $\mathcal{F} = \{L_1, \dots, L_m, L_{m+1}\}$, в котором $L_1, \dots, L_m$ — это попарно вполне несовместные префиксные языки, которые не имеют расширений в языке L_{m+1}, можно описать при помощи multi-DFA $D = \langle Q, q_0, \{f_1\}, \dots, \{f_m\}, F_{m+1}, \varphi \rangle$, в котором допускающие состояния $f_1, \dots, f_m$ не имеют исходящих из них переходов. Такой multi-DFA, описывающий семейство языков $\mathcal{F}$, будет называться *префиксным* multi-DFA.

В разрешающих алгоритмах, представленных в разделах 5, 6, и 8, будут применяться пять операций над регулярными языками: деление регулярного языка на слово слева $L \setminus w$ и справа L/a, объединение языков $L_1 \cup L_2$, конкатенация языков $L_1 L_2$, и комбинированная итерация языков $L_1^* L_2$. Префиксные DFA и multi-DFA, используемые для описания префиксных регулярных языков, предоставляют значительные преимущества для эффективного выполнения этих операций.

Для заданного слова $w = x_1 \dots x_n$ и multi-DFA $D = \langle Q, q_0, F_1, \dots, F_m, \varphi \rangle$, который описывает семейство регулярных языков $\{L_1, \dots, L_m\}$, достаточно всего лишь отыскать w-последователя q' начального состояния q_0 (т.е. такое состояние q', для которого имеется последовательность переходов $q_0 \xrightarrow{x_1} \cdots \xrightarrow{x_n} q'$), чтобы получить multi-DFA representation $D' = \langle Q, q', F_1, \dots, F_m, \varphi \rangle$, описывающий семейство языков $\{L_1 \setminus w, \dots, L_m \setminus w\}$.

Для заданной буквы $a \in \Sigma$ и DFA $A = \langle Q, q_0, F, \varphi \rangle$, нетрудно отыскать подмножество всех a-предшественников $F' = \{q' : q' \xrightarrow{a} q, q \in F\}$ допускающих состояний и построить DFA $A' = \langle Q, q_0, F', \varphi \rangle$, описывающий частное от деления $L(A)/a$.

Если multi-DFA $D = \langle Q, q_0, F_1, F_2, \varphi \rangle$ описывает пару регулярных языков L_1 и L_2, то их объединение $L_1 \cup L_2$ можно описать при помощи DFA $B' = \langle Q, q_0, F_1 \cup F_2, \varphi \rangle$.

Объединив допускающее состояние префиксного DFA, описывающего язык L_0, и начальное состояние multi-DFA, который описывает семейство языков $L_1, \dots, L_m$, можно построить автоматное описание семейства конкатенаций $\{L_0 L_1, \dots, L_0 L_m\}$.

И, наконец, если имеется префиксный multi-DFA $D = \langle Q, q_0, \{f_1\}, \ldots, \{f_m\}, F, \varphi\rangle$, то 1) добавив к нему копию q_0' начального состояния q_0 вместе с исходящими из него переходами, 2) перенаправив в состояние q_0' все те переходы. которые ранее вели в состояние q_0, и 3) объединив начальное состояние q_0 и допускающее состояние f_1, можно построить multi-DFA $D' = \langle Q \cup \{q_0'\}, q_0, \{f_2\}, \ldots, \{f_m\}, F, \varphi'\rangle$, который описывает семейство языков $L_1^* L_2, \ldots, L_1^* L_m, L_1^* L_{m+1}$.

Все эти операции над языками можно выполнить за время, линейно зависящее от размера заданных автоматных описаний языков, и, кроме того, размер (число состояний) полученных при этом автоматов увеличивается не более чем на 1.

4. Конечные автоматы-преобразователи

Конечный автомат-преобразователь (далее коротко, трансдьюсер), работающий в конечном входном алфавите Σ и конечном выходном алфавите Δ, представляет собой систему переходов $\pi = \langle Q, F, \longrightarrow\rangle$, в которой Q — конечное множество *состояний*, $F \subseteq Q$ — подмножество *финальных состояний*, и $\longrightarrow$ — конечное *отношение переходов* типа $Q \times (\Sigma \cup \{\varepsilon\}) \times \Delta^* \times Q$. Иногда для обозначения трансдьюсеров будет использоваться запись $\pi(q_0)$, чтобы выделить некоторое состояние q_0 из множества Q как *начальное состояние* π. Четверки (q, x, u, q') из отношения $\longrightarrow$ называются *переходами* и обозначаются записью $q \xrightarrow{x|u} q'$. Если x — это буква алфавита Σ, то переход $q \xrightarrow{x|u} q'$ будем называть Σ-переходом. Переходы вида $q \xrightarrow{\varepsilon|u} q'$ называются *ε-переходами*. Нетрудно понять, что Σ-переход $q \xrightarrow{a|u} q'$ означает, что трансдьюсер, пребывающий в состоянии q, считывает букву a на входной ленте, записывает на выходной ленте слово u и переходит в состояние q'. Когда в трансдьюсере срабатывает ε-переход $q \xrightarrow{\varepsilon|u} q'$, то машина, не обращаясь к входной ленте, записывает слово u на выходной ленте и переходит в состояние q'. Размером перехода $q \xrightarrow{x|u} q'$ будет считаться число $|u| + 1$. Размером $|\pi|$ трансдьюсера π является число, равное сумме размеров всех переходов машины π плюс число ее состояний $|Q|$.

Прогоном трансдьюсера π из состояния q в состояние q' на входном слове h называется всякая конечная последовательность переходов

$$q \xrightarrow{x_1|u_1} q_1 \xrightarrow{x_2|u_2} \cdots \xrightarrow{x_{n-1}|u_{n-1}} q_{n-1} \xrightarrow{x_n|u_n} q' , \tag{2}$$

для которой выполняется равенство $h = x_1 x_2 \ldots x_n$. Пара слов (h, u), в которой $u = u_1 u_2 \ldots u_n$, называется *пометкой* прогона (2). Мы будем использовать запись $q \xrightarrow{h|u}_* q'$ для обозначения того факта, что трансдьюсер π имеет прогон (2) с пометкой (h, u) из состояния q в состояние q'. Если $q' \in F$, то прогон (2) называется *финальным*. Прогон (2) трансдьюсера π с пометкой (ε, u) называется *ε-прогоном*. Для каждого состояния q существует *пустой прогон* $q \xrightarrow{\varepsilon|\varepsilon}_* q$ из состояния q в то же самое состояние, представляющий собой пустую последовательность переходов.

Множество пар слов $TR(\pi, q) = \{(h, u) : q \xrightarrow{h|u}_* q', q' \in F\}$, которыми помечены все финальные прогоны машины π из состояния q, называется *отношением трансдукции, вычисляемое трансдьюсером π в состоянии q*. Заметим, что если q — это финальное состояние трансдьюсера π, отношение трансдукции $TR(\pi, q)$ содержит пометку $(\varepsilon, \varepsilon)$. Состояния q_1 и q_2 трансдьюсера π называются *эквивалентными* (и этот факт обозначается записью $\pi(q_1) \sim \pi(q_2)$), если верно равенство $TR(\pi, q_1) = TR(\pi, q_2)$. *Проблема эквивалентности* для трансдьюсеров состоит в том, чтобы для произвольного заданного трансдьюсерв π и пары его состояний q_1 и q_2 проверить выполнимость отношения $\pi(q_1) \sim \pi(q_2)$.

Автомат-преобразователь π называется *трансдьюсером реального времени*, если он не содержит ε-переходов, т.е. $\longrightarrow \subseteq Q \times \Sigma \times \Delta^* \times Q$. В отличие от конечных автоматов, в некоторых трансдьюсерах невозможно избавиться от ε-переходов, и поэтому, вычислительные возможности трансдьюсеров реального времени существенно уступают возможностям трансдьюсеров с ε-переходами.

Трансдьюсер реального времени π называется

- *детерминированным* трансдьюсером, если для каждой входной буквы a и состояния q он имеет не более одного перехода вида $q \xrightarrow{a|u} q'$;
- *функциональным* (или однозначным) трансдьюсером, если для каждого состояния q отношение трансдукции $TR(\pi, q)$ является функцией;
- *k-неоднозначным* трансдьюсером, если для каждого состояния q и любого входного слова h имеется не более k финальных прогонов машины π из состояния q на слове h;
- *k-значным* трансдьюсером, если для каждого состояния q и любого входного слова h отношение $TR(\pi, q)$ содержит не более k образов слова h, т.е. $|Im(TR(\pi, q), h)| \leq k$;
- трансдьюсером *ограниченного диапазона длин* k, если для каждого состояния q и любого входного слова h, множество длин образов w слова h в отношении $TR(\pi, q)$ содержит не более k чисел, т.е. $|\{|w| : w \in Im(TR(\pi, q), h)\}| \leq k$.

Для всех перечисленных выше типов трансдьюсеров реального времени была доказана разрешимость проблемы эквивалентности (см. [21—24, 28, 35]). В следующем разделе вводится в рассмотрение новый класс трансдьюсеров реального времени, для которого эта проблема также оказывается разрешимой.

5. Проверка эквивалентности префиксных трансдьюсеров реального времени

Рассмотрим произвольный трансдьюсер реального времени $\pi = \langle Q, q_0, F, \longrightarrow \rangle$. Для любой пары состояний q, q' и входной буквы x обозначим записью $Out_\pi(q, q', x)$ множество всех выходных слов u на переходах вида $q \xrightarrow{x|u} q'$ в системе π. Трансдьюсер π назовем *префиксным*, если для любого его состояния q, любой входной буквы x, и произвольной пары различных переходов вида $q \xrightarrow{x|u'} q'$ и $q \xrightarrow{x|u''} q''$ выходные слова u' и u'' несовместны. Префиксные трансдьюсеры обладают очевидным характеристическим свойством: для каждого состояния $q \in Q$ и любой входной буквы $x \in \Sigma$ все языки семейства $\{Out_\pi(q, x, p) : p \in Q\}$ являются префиксными и попарно вполне несовместными.

Очевидно, что каждый детерминированный трансдьюсер является префиксным, хотя обратное утверждение неверно. Префиксные трансдьюсеры обладают особым свойством “однозначности”: для каждого состояния q префиксного трансдьюсера π и любой пары $(h, u) \in TR(\pi, q)$ существует единственный прогон машины π из состояния q с пометкой (h, u). Однако следует иметь в виду, что это свойство существенно отличается от свойства однозначности автоматов-преобразователей в его традиционном смысле (см. [24, 25, 27, 28, 31]): в обычном понимании свойство однозначности (ambiguity) требует, чтобы для каждого входного слова h существовало не более одного финального прогона трансдьюсера π помеченного парой (h, w) для некоторого выходного слова w. Необходимо также заметить, что некоторые префиксные трансдьюсеры реального времени не обладают такими свойствами как k-неоднозначность, k-значность или даже ограниченность диапазона длин, т.е. класс префиксных трансдьюсеров не подпадает ни под один из ранее известных случаев разрешимости проблемы эквивалентности для трансдьюсеров реального времени.

Пример 1. *Рассмотрим простой префиксный трансдьюсер реального времени* $\pi = \langle \{q_0, q_1\}, \{q_1\}, \longrightarrow \rangle$, *который работает в алфавитах* $\Sigma = \{a, b\}$ *и* $\Delta = \{c, d\}$ *и имеет три перехода* $q_0 \xrightarrow{a/c} q_0$, $q_0 \xrightarrow{a/dd} q_0$, *и* $q_0 \xrightarrow{b/\varepsilon} q_1$. *Для каждого входного слова* $h_n = a^n b$ *длины выходных слов на финальных прогонах трансдьюсера* π *из состояния* q_0 *являются всевозможными целыми числами в диапазоне от* n *до* $2n$. *Следовательно, трансдьюсер* π *не обладает свойством ограниченного диапазона длин.*

Предложенный в этой статье метод проверки эквивалентности для префиксных трансдьюсеров реального времени основан на преобразованиях регулярных выражений специального вида. Точно также, как это было сделано для проверки эквивалентности DFAs в разделе 1, с каждым трансдьюсером π будем ассоциировать систему линейных уравнений $\mathcal{E}_\pi$, которая специфицирует трансдукции, вычисляемые во всех состояниях машины π. Чтобы проверить эквивалентность $\pi(q') \sim \pi(q'')$ добавим к системе уравнений $\mathcal{E}_\pi$ требование эквивалентности состояний, которое представлено уравнением вида $X' = X''$, а затем попробуем выяснить, имеет ли решение полученная система уравнений. Для этого, как и в случае проверки эквивалентности конечных автоматов, воспользуемся методом исключения Гаусса. Для произвольных трансдьюсеров преимущества такого подхода сомнительны, но для префиксных трансдьюсеров утверждение 10 о расщеплении совместных пар префиксных языков предоставляет удобное средство для эффективного решения систем уравнений, соответствующих задаче проверки эквивалентности.

Чтобы упростить изложение этого метода, сделаем некоторые предположения относительно преобразователя, подлежащего анализу:

- входной алфавит $\Sigma = \{a_1, \ldots, a_k\}$ и выходной алфавит Δ не имеют общих букв; символы x, y, z будут использоваться для обозначения произвольных букв из множества Σ, а символы u, v, w — для обозначения произвольных слов из множества Δ^*,
- мы будем рассматривать только *сокращенные* трансдьюсеры π, у которых из каждого состояния достижимо хотя бы одно финальное состояние.

Регулярные выражения (далее, для краткости, р-выражения) строятся из переменных $X_1, X_2, \ldots$, констант 0, 1, букв из множества Σ, и слов из множества Δ^* при помощи операций конкатенации $\cdot$ и альтернации $+$. Р-выражения интерпретируются в полукольце трансдукций. Значениями констант 0, 1, каждой буквы $x \in \Sigma$, и каждого слова $u \in \Delta^*$ служат соответственно трансдукции $\emptyset$, $\{(\varepsilon, \varepsilon)\}$, $\{(x, \varepsilon)\}$ и $\{(\varepsilon, u)\}$. Конкатенация $\cdot$ и альтернация $+$ истолковываются соответственно как конкатенация и объединение трансдукций. Очевидно, что операция альтернации $+$ коммутативна, и равенство $x \cdot u = u \cdot x$ верно для любой буквы $x \in \Sigma$ и слова $u \in \Delta^*$. Мы также будем использовать следующие очевидные тождества: $0 \cdot T = T \cdot 0 = 0, 0 + T = T, 1 \cdot T = T \cdot 1 = T$.

В центре внимания будут линейные р-выражения двух типов. Δ-*выражением* будем называть константу 0, а также всякое р-выражение вида $u_1 \cdot X_{i_1} + u_2 \cdot X_{i_2} + \cdots + u_m \cdot X_{i_m}$, в котором одна и та же переменная может иметь несколько вхождений. Если слова $u_1, u_2, \ldots, u_m$ попарно несовместны, то такое Δ-выражение называется префиксным. Σ-*выражением* называется всякое выражение вида $a_1 \cdot E_1 + a_2 \cdot E_2 + \cdots + a_k \cdot E_k + w_1 + \cdots + w_m$, в котором E_i, $1 \le i \le k$, — это Δ-выражения, и при этом входной алфавит $\Sigma = \{a_1, a_2, \ldots, a_k\}$ и $\{w_1, \ldots, w_m\} \subseteq \Sigma^*$. Мы будем использовать записи $E(X_1, \ldots, X_n)$ (для Δ-выражений) или $G(X_1, \ldots, X_n)$ (для Σ-выражений), чтобы подчеркнуть тот факт, что эти р-выражения могут содержать только переменные $X_1, \ldots, X_n$.

Рассмотрим произвольный трансдьюсер реального времени $\pi = \langle Q, F, \longrightarrow \rangle$, работающий в алфавитах Σ и Δ. С каждым его состоянием p будем ассоциировать переменную X_p, и для каждой пары $p \in Q$ и $x \in \Sigma$ построим Δ-выражение $E_{p,x} = \sum_{q \in Q} \sum_{u \in Out_\pi(p,x,q)} u \cdot X_q$. Ясно, что если трансдьюсер π является префиксным, то все определенные таким образом Δ-выражения $E_{p,x}$ также являются префиксными. Для каждого состояния p обозначим записью δ_p либо константу 1 в случае $p \in F$,

или константу 0 в противном случае. Тогда алгебраической спецификацией трансдьюсера π может служить система уравнений

$$\mathcal{E}_\pi = \{X_p = \sum_{x\in\Sigma} x \cdot E_{p,x} + \delta_p \ : \ p \in Q\} \ .$$

Утверждение 12. *Для любого трансдьюсера реального времени π система уравнений $\mathcal{E}_\pi$ имеет единственное решение $\{X_p = TR(\pi, p) \ : \ p \in Q\}$.*

Доказательство. То, что множество трансдукций $TR(\pi, p), p \in Q$, взятых в качестве значений соответствующих переменных X_p, — это одно из решений системы уравнений $\mathcal{E}_\pi$, следует непосредственно из определения отношений $TR(\pi, p)$ и устройства уравнений этой системы.

Чтобы доказать единственность этого решения, мы рассмотрим произвольное решение $X_p = T_p$, $p \in Q$, системы уравнений $\mathcal{E}_\pi$ и покажем при помощи индукции по длине входного слова h, что для любой пары (h, w) и состояния $p \in Q$ пара (h, w) принадлежит отношению T_p тогда и только тогда, когда $(h, w) \in TR(\pi, p)$.

Как можно заметить из описания уравнений системы $\mathcal{E}_\pi$, пара (ε, w) может содержаться в T_p в том и только том случае, если $w = \varepsilon$ и $p \in F$; последнее равносильно включению $(\varepsilon, w) \in TR(\pi, p)$.

Предположим, что соотношение $(h, w) \in T_p \iff (h, w) \in TR(\pi, p)$ выполняется для каждого состояния p, любого выходного слова w, и произвольного входного слова h, длина которого не превосходит n. Рассмотрим произвольное входное слово $h' = yh$ длины $n + 1$. Так как отношение T_p является составной частью решения системы уравнений $\mathcal{E}_\pi$, верно равенство $T_p = \sum\limits_{x\in\Sigma} \sum\limits_{q\in Q} \sum\limits_{u\in Out_\pi(p,x,q)} x \cdot u \cdot T_q + c_p$. Поэтому для любого выходного слова w пара (h', w) содержится в отношении T_p тогда и только тогда, когда $(h, w \setminus u) \in T_q$ для некоторого состояния q и выходного слова $u \in Out_\pi(p, y, q)$. Согласно индуктивному предположению последнее условие равносильно тому, что $(h, w \setminus u) \in TR(\pi, q)$, и при этом в трансдьюсере π имеется переход $p \xrightarrow{y|u} q$; это означает, что $(h, w) \in TR(\pi, p)$. □

Следствие 1. *Для каждой пары состояний $p, q \in Q$ эквивалентность $\pi(p) \sim \pi(q)$ имеет место тогда и только тогда система уравнений $\mathcal{E}_\pi \cup \{X_p = X_q\}$ имеет решение.*

Как показывает полученное следствие, для проверки эквивалентности трансдьюсеров реального времени достаточно уметь эффективно проверять разрешимость систем уравнений, которые являются расширением систем уравнений $\mathcal{E}_\pi$, соответствующих трансдьюсерам. Разрешимость некоторых расширений совершенно очевидна. Будем говорить, что система линейных уравнений

$$\mathcal{E} = \mathcal{E}_\pi(X_1, \dots, X_n) \cup \{Y_j = E_j(X_1, \dots, X_n) \ : \ 1 \le j \le m\},$$

является *приведенной*, если множества переменных $\{X_1, \dots, X_n\}$ и $\{Y_1, \dots, Y_m\}$ не пересекаются, и правые части $E_j, 1 \le j \le m$, — это некоторые p-выражения.

Утверждение 13. *Каждая приведенная система уравнений $\mathcal{E}$ имеет единственное решение.*

Доказательство. Следует непосредственно из утверждения 12. □

Некоторые другие расширения систем уравнений $\mathcal{E}_\pi$ не имеют решений.

Утверждение 14. *Если система уравнений* $\mathcal{E}_\pi(X_1,\dots,X_n) \cup \{\sum_{i=1}^{\ell} u_i \cdot X_i = \sum_{j=1}^{m} v_j \cdot X_j\}$ *имеет решение, то языки* $L_1 = \{u_1,\dots,u_\ell\}$ *и* $L_2 = \{v_1,\dots,v_m\}$ *совместны.*

Доказательство. Согласно утверждению 12 система уравнений $\mathcal{E}_\pi$ имеет ненулевое решение $X_1 = T_1,\dots,X_n = T_n$. Если слово u_i несовместно со словами $v_1,\dots,v_m$, то $(\varepsilon,u_i)T_i \cap \bigcup_{j=1}^{m}(\varepsilon,v_j)T_j = \emptyset$, и поэтому $\sum_{i=1}^{\ell} u_i \cdot T_i \neq \sum_{j=1}^{m} v_j \cdot T_j$ □

Утверждение 15. *Если система уравнений* $\mathcal{E}_\pi(X_1,\dots,X_n) \cup \{X_1 = \sum_{i=1}^{\ell} u_i \cdot X_i\}$ *имеет решение, и множество слов* $\{u_1,\dots,u_\ell\}$ *является префиксным, то* $\ell = 1$ *и* $u_1 = \varepsilon$, *т.е.* $\sum_{i=1}^{\ell} u_i \cdot X_i \equiv X_1$.

Доказательство. Согласно утверждению 12 система уравнений $\mathcal{E}_\pi$ имеет единственное решение $X_1 = T_1,\dots,X_n = T_n$ и при этом $T_i \neq \emptyset$ для каждого $i, 1 \le i \le n$. Предположим, что верно равенство $T_1 = \bigcup_{i=1}^{\ell}(\varepsilon,u_i)T_i$. Рассмотрим произвольное входное слово h из области определения $Dom(T_1)$ и конечные языки из множеств значений $Im(T_i,h), 1 \le i \le \ell$. С одной стороны, мы имеем очевидное соотношение

$$T_1 = \bigcup_{i=1}^{\ell}(\varepsilon,u_i)T_i \implies Im(T_1,h) = \bigcup_{i=1}^{\ell} u_i Im(T_i,h).$$

С другой стороны, поскольку множество слов $\{u_1,\dots,u_\ell\}$ является префиксным, языки $u_1 Im(T_1,h)$ и $\bigcup_{i=2}^{\ell} u_i Im(T_i,h)$ не имеют общих слов. Поэтому мощность множества $\bigcup_{i=2}^{\ell} u_i Im(T_i,h)$ равна 0. Однако из равенства $Im(T_1,h) = u_1 Im(T_1,h)$ следует $u_1 = \varepsilon$. А так как пустое слово ε совместно с любым словом, мы приходим к выводу о том, что $\ell = 1$. □

Здесь нужно особо отметить, что в приведенном доказательстве решающую роль сыграл тот факт, что π — это трансдьюсер реального времени, и поэтому для каждого входного слова h трансдукции T_i, являющиеся решениями системы уравнений $\mathcal{E}_\pi$, имеют конечное число образов слова h. Когда в разделе 6 речь пойдет о трансдьюсерах с ε-переходами, утверждением 15 уже нельзя будет воспользоваться.

Далее опишем итеративную процедуру проверки разрешимости системы уравнений $\mathcal{E}_1 = \mathcal{E}_\pi \cup \{X_p = X_q\}$ для произвольного префиксного трансдьюсера π путем преобразования этой системы в равносильную приведенную форму. В начале каждой итерации t эта процедура имеет на входе систему уравнений вида

$$\mathcal{E}_t = \mathcal{E}_{\pi_t} \cup \{X_{j_i} = E_i \,:\, 1 \le i \le N_t\},$$

где

- π_t — это некоторый префиксный трансдьюсер (не обязательно совпадающий с анализируемым трансдьюсером π),
- $X_{j_i}, 1 \le i \le N_t$, — переменные (не обязательно попарно различные),
- $E_i, 1 \le i \le N_t$, — префиксные ненулевые Δ-выражения, зависящие только от тех переменных, которые содержатся в уравнениях подсистемы $\mathcal{E}_{\pi_t}$.

Уравнения подсистемы $\mathcal{E}_{\pi_t}$ будем называть *базовыми уравнениями*, а все остальные уравнения — *проверочными уравнениями*. Если переменная X_i из левой части проверочного уравнения $X_i = E_i$ не встречается где-либо еще в системе $\mathcal{E}_t$, то X_i будет называться *приведенной переменной*, а соответствующее уравнение — *приведенным уравнением*. Все остальные проверочные уравнения и переменные из их левых частей будут называться *активными*. Очевидно, что система уравнений $\mathcal{E}_t$, не содержащая активных переменных, является приведенной. Наш подход основан на идее метода исключения Гаусса и деактивизирует одну за другой все переменные системы $\mathcal{E}_t$ за счет применения подходящих равносильных преобразований к системе $\mathcal{E}_t$. Этот процесс завершается в двух случаях:

- в системе уравнений $\mathcal{E}_t$ отсутствуют активные переменные; тогда согласно утверждению 13, система $\mathcal{E}_t$ имеет решение, и отсюда следует $\pi(p) \sim \pi(q)$;
- одно из активных уравнений системы $\mathcal{E}_t$ противоречит заключению утверждения 14 или утверждения 15; тогда система $\mathcal{E}_t$ не имеет решений, и поэтому состояния p и q трансдьюсера π могут быть сочтены неэквивалентными.

На t-й итерации следующие равносильные преобразования применяются к системе уравнений $\mathcal{E}_t$.

1) *Удаление тождеств.* Уравнения вида $X = X$ удаляются из системы $\mathcal{E}_t$.

2) *Проверка активности переменных.* Если в систем $\mathcal{E}_t$ нет активных уравнений, то процедура завершает свое выполнение и объявляет о разрешимости исходной системы уравнений в соответствии с утверждением 13.

3) *Приведение переменных.* Выбирается активное уравнение $X_i = E_i$ в $\mathcal{E}_t$. Если X_i входит в состав Δ-выражения E_i с ненулевым коэффициентом, то процедура завершает свое выполнение и объявляет о разрешимости исходной системы уравнений в соответствии с утверждением 15. В противном случае во всех остальных уравнениях системы $\mathcal{E}_t$ все вхождения переменной X_i замещаются p-выражением E_i за счет применения подстановки $\{X_i/E_i\}$.

После этого шага переменная X_i и уравнение $X_i = E_i$ становятся приведенными, а число активных переменных в системе $\mathcal{E}_t$ сокращается. Однако применение $\{X_i/E_i\}$ к уравнениям системы дает и побочный эффект: нестандартные уравнения одного из следующих видов

A) $E = G$, где E — это Δ-выражение, отличное от переменных, и G — это Σ-выражение,

B) $E' = E''$, где E', E'' — это Δ-выражения, отличные от переменных,

могут появиться в системе $\mathcal{E}_t$. Может случиться также, что

C) сразу несколько базовых уравнений вида $X = G$ с одной и той же переменной X в левых частях появляются в системе $\mathcal{E}_t$. Эти отклонения системы уравнений от установленной канонической формы должны быть устранены.

4) *Устранение нестандартных уравнений вида $E = G$.* Для этого

- в каждом уравнении вида $E(X_1, \dots, X_\ell) = G$ заменить все вхождения переменных $X_i, 1 \le i \le \ell$, в ненулевом Δ-выражении E на Σ-выражения G_i из правых частей базовых уравнений $X_i = G_i$ подсистемы $\mathcal{E}_{\pi_t}$, а затем привести полученное выражение $E(G_1, \dots, G_\ell)$ к стандартной форме Σ-выражения при помощи закона коммутативности $u \cdot x = x \cdot u$ для букв алфавитов Σ и Δ и закона дистрибутивности $T \cdot R + T \cdot S = T \cdot (R + S)$;
- для каждой пары базовых уравнений $X = G'$ и $X = G''$ с одинаковыми левыми частями заменить одно из этих уравнений на уравнение $G' = G''$.

После этого шага все уравнения вида $E = G$ исчезнут, и все базовые уравнения $X = G$ будут иметь попарно различные переменные в левых частях. Однако это достигается за счет того, что в системе появляются нестандартные уравнения вида

D) $G' = G''$, где G', G'' — это Σ-выражения.

5) *Устранение нестандартных уравнений вида* $G' = G''$. Рассмотрим уравнение

$$\sum_{i=1}^{k} a_i \cdot E'_i + w'_1 + \cdots + w'_m = \sum_{i=1}^{k} a_i \cdot E''_i + w''_1 + \cdots + w''_n .$$

Если $\{w'_1, \ldots, w'_m\} \neq \{w''_1, \ldots, w''_n\}$, то процедура завершается и объявляет о неразрешимости системы. То же самое решение будет принято, если для некоторого i, $1 \leq i \leq k$, одно из Δ-выражений E'_i или E''_i равно 0, тогда как другое отлично от нуля. Легко видеть, что в обоих случаях левая и правая части этого уравнения оцениваются по-разному для любых непустых трансдукций, взятых в качестве значений переменных.

В противном случае удалить это уравнение из системы и подставить вместо него уравнение $E'_i = E''_i, 1 \leq i \leq k$, где E'_i и E''_i — ненулевые Δ-выражения. Таким образом, все уравнения вида $G' = G''$ исчезнут из системы, а вместо них в системе появятся новые уравнения вида $E' = E''$. После этого шага уравнения вида B) — это единственные нестандартные уравнения, которые еще остаются в системе.

6) *Устранение нестандартных уравнений* $E' = E''$. Решающее значение здесь имеет тот факт, что Δ-выражения E' и E'' являются префиксными. Это обусловлено тем, что исходная система уравнений $\mathcal{E}_1$ была построена для префиксного трансдьюсера π, и все преобразования, которые применяются в дальнейшем к этой системе, сохраняют, как это следует из утверждений 4–7, префиксное свойство Δ-выражений, входящих в уравнения. Для устранения нестандартного уравнения типа B)

$$\sum_{i=1}^{\ell} u_i \cdot X'_i = \sum_{j=1}^{m} v_j \cdot X''_j \tag{3}$$

проверяется совместность префиксных языков $L' = \{u_1, \ldots, u_\ell\}$ и $L'' = \{v_1, \ldots, v_m\}$. Если какое-нибудь слово одного из этих языков несовместно с любым словом другого языка, то процедура проверки завершает работу и объявляет о неразрешимости исходной системы уравнений на основании утверждения 14. В противном случае процедура осуществляет расщепление $L' = \bigcup_{i=1}^{n} L'_i$ и $L'' = \bigcup_{i=1}^{n} L''_i$ этих языков (см. утверждение 10), удаляет уравнение (3) из системы $\mathcal{E}_t$, и добавляет вместо него для каждой доли $L'_i = \{u_{i_0}\}$ и $L''_i = \{v_{i_1}, \ldots, v_{i_r}\}$ (или $L'_i = \{u_{i_1}, \ldots, u_{i_r}\}$ и $L''_i = \{v_{i_0}\}$) новое уравнение $X'_{i_0} = (v_{i_1} \setminus u_{i_0}) \cdot X''_{i_1} + \cdots + (v_{i_r} \setminus u_{i_0}) \cdot X''_{i_r}$ (или, соответственно, $X''_{i_0} = (u_{i_1} \setminus v_{i_0}) \cdot X'_{i_1} + \cdots + (u_{i_r} \setminus v_{i_0}) \cdot X'_{i_r}$). Утверждение 5 гарантирует при этом, что Δ-выражения в правых частях добавленных уравнений будут префиксными.

После этого шага будет получена система уравнений $\mathcal{E}_{t+1}$, которая равносильна системе $\mathcal{E}_t$, но имеет меньшее число активных переменных, чем $\mathcal{E}_t$.

Можно обратить внимание на тот факт, что описанная процедура проверки разрешимости систем уравнений очень похожа на алгоритм Мартелли–Монтанари [70] вычисления наиболее общего унификатора двух термов. И так же, как и в этом алгоритме унификации, правила 1)–6) преобразования систем уравнений в приведенную форму могут применяться в произвольном порядке.

Утверждение 16. *Для любого префиксного трансдьюсера реального времени* π *и пары его состояний* p, q *предложенная процедура после применения к системе уравнений* $\mathcal{E}_1 = \mathcal{E}_\pi \cup \{X_p = X_q\}$ *завершает работу и корректно проверяет разрешимость системы* $\mathcal{E}_1$.

Доказательство. Так как после каждого этапа t выполнения процедуры число активных переменных уменьшается по меньшей мере на единицу, процедура обязательно завершает работу. Преобразования по правилам 1) и 3)–6), которым подвергается система уравнений $\mathcal{E}_t$, являются равносильными преобразованиями; поэтому на каждом этапе t система уравнений $\mathcal{E}_t$ равносильна

исходной системе $\mathcal{E}_1$. Согласно утверждениям 13–15 решение относительно разрешимости системы уравнений $\mathcal{E}_t$, которое принимает процедура, является правильным. □

Таким образом, верна

Теорема 2. *Проблема эквивалентности для префиксных трансдьюсеров реального времени разрешима за квадратичное время.*

Доказательство. Согласно утверждению 16 описанная выше разрешающая процедура корректно проверяет эквивалентность $\pi(p) \sim \pi(q)$ для любого префиксного трансдьюсера реального времени π. Число этапов работы этой процедуры не превосходит числа состояний трансдьюсера π. Все Δ-выражения, входящие в систему уравнений $\mathcal{E}_t$, можно представить в виде словарного дерева с использованием ссылочных структур данных. Такие структуры данных позволяют выполнять все операции над уравнениями, такие как применение подстановки, проверка совместности, расщепление уравнений и др., за время, линейное относительно размера системы уравнений $\mathcal{E}_t$. Необходимо заметить, что использование ссылочных структур данных позволяет выполнять эти операции так, чтобы размер системы уравнений $\mathcal{E}_t$ не увеличивался. Поскольку исходную систему уравнений $\mathcal{E}_1$ можно построить за время, линейное относительно размера трансдьюсера π, проверку эквивалентности $\pi(p) \sim \pi(q)$ можно выполнить за время, квадратичное относительно размера π. □

6. Проверка эквивалентности префиксных трансдьюсеров с ε-переходами

В этом разделе выделен новый класс префиксных трансдьюсеров с ε-переходами и показано, как можно проверять эквивалентность машин из этого класса при помощи того же метода проверки разрешимости систем языковых уравнений, который был разработан в разделе 5. Но для трансдьюсеров более общего вида коэффициентами этих уравнений будут не отдельные слова, а регулярные префиксные языки.

Для трансдьюсера $\pi = \langle Q, F, \longrightarrow \rangle$, работающего в входном алфавите Σ и выходном алфавите Δ, назовем состояние q этого трансдьюсера Σ-состоянием (ε-состоянием), если ни один ε-переход не исходит (соответственно, только ε-переходы исходят) из состояния q. Состояния обоих типов будем называть однородными. Трансдьюсер π называется *однородным*, если все его состояния являются однородными. Каждый трансдьюсер можно легко преобразовать в однородный: если из состояния q исходят переходы обоих типов, то его можно расщепить на два состояния q' и q'' так, чтобы q' унаследовало от q только исходящие из него Σ-переходы, а q'' унаследовало от q только исходящие из этого состояния ε-переходы. Переходы, которые прежде вели в состояние q, дублируются и направляются в оба состояния q' и q''. Далее в этом разделе мы будем рассматривать только однородные трансдьюсеры.

Множество состояний Q однородного трансдьюсера можно разделить на подмножество Q_Σ, которое содержит все Σ-состояния, и подмножество Q_ε, которое содержит все ε-состояния. Точно так же, множество финальных состояний F разделяется на подмножество F_Σ, содержащее все финальные Σ-состояния, и подмножество F_ε, содержащее все финальные ε-состояния. Так как однородный трансдьюсер может иметь циклы, состоящие только из ε-переходов, он способен, в отличие от трансдьюсера реального времени, после прочтения одной буквы на входной ленте записать сколь угодно много различных слов на выходной ленте. Чтобы описать реакцию однородного трансдьюсера π в ответ на событие начала вычисления или считывания буквы на входной ленте, мы вводим четыре класса языков:

1) для каждой пары Σ-состояний p, q и входной буквы $a \in \Sigma$ определим множество слов

$$Out_\pi^\Sigma(p, a, q) = \{ w : p \xrightarrow{a|w}_* q \},$$

которые π может записать на входной ленте при совершении прогона из состояния p в состояние q в результате считывания единственной буквы a;

2) для каждого Σ-состояния p и входной буквы $a \in \Sigma$ определим множество слов

$$Fin^{\Sigma}_{\pi}(p, a) = \{w : p \xrightarrow{a|w}_{*} q, q \in F_{\varepsilon}\},$$

которые π может записать на входной ленте при совершении прогона из состояния p в некоторое финальное ε-состояние в результате считывания единственной буквы a;

3) для каждого ε-состояния p и Σ-состояния q определим множество слов

$$Out^{\varepsilon}_{\pi}(p, q) = \{w : p \xrightarrow{\varepsilon|w}_{*} q\},$$

которые π может записать на входной ленте при совершении некоторого ε-прогона из состояния p в состояние q;

4) для каждого ε-состояния p определим множество слов

$$Fin^{\varepsilon}_{\pi}(p) = \{w : p \xrightarrow{\varepsilon|w}_{*} q, q \in F_{\varepsilon}\},$$

которые π может записать на входной ленте при совершении некоторого ε-прогона из состояния p в некоторое финальное ε-состояние.

Подобно тому, как это было определено для трансдьюсеров реального времени, будем называть однородный трансдьюсер π *префиксным*, если для любого $x \in \Sigma \cup \{\varepsilon\}$, состояния q, и пары различных переходов $q \xrightarrow{x|u'} q'$ и $q \xrightarrow{x|u''} q''$ выходные слова u' и u'' несовместны. Следующее утверждение вытекает непосредственно из приведенного определения.

Утверждение 17. *Если $\pi = \langle Q_{\Sigma} \cup Q_{\varepsilon}, F_{\Sigma} \cup F_{\varepsilon}, \longrightarrow \rangle$ — это однородный префиксный трансдьюсер , то*

1. *для каждого Σ-состояния p и входной буквы $a \in \Sigma$ все языки из семейства $\{Out^{\Sigma}_{\pi}(p, a, q) : q \in Q_{\Sigma}\}$ являются попарно вполне несовместными регулярными языками, которые не имеют расширений в регулярном языке $Fin^{\varepsilon}_{\pi}(p, a)$;*
2. *для каждого ε-состояния p все языки из семейства $\{Out^{\varepsilon}_{\pi}(p, q) : q \in Q_{\Sigma}\}$ являются попарно вполне несовместными регулярными языками, которые не имеют расширений в регулярном языке $Fin^{\varepsilon}_{\pi}(p)$.*

Доказательство. Рассматривая каждый ε-переход $p \xrightarrow{\varepsilon|w} q$ трансдьюсера π, где $w = x_1 \dots x_m$, как последовательную композицию $p \xrightarrow{x_1} \cdots \xrightarrow{x_m} q$ переходов, можно построить для каждого ε-состояния p такой конечный мультиавтомат, который 1) работает в выходном алфавите Δ, 2) имеет p в качестве начального состояния, и 3) имеет в качестве семейства множеств допускающих состояний все одноэлементные множества $\{q\}, q \in Q_{\Sigma}$, а также множество F_{ε}. Так как трансдьюсер π обладает свойством префиксности, указанный мультиавтомат является префиксным multi-DFA, который описывает семейство языков $\{Out^{\varepsilon}_{\pi}(p, q) : q \in Q_{\Sigma}\}$ и язык $Fin^{\varepsilon}_{\pi}(p)$.

Следуя тем же рассуждениям можно построить для каждого Σ-состояния p трансдьюсера π и любой входной букве $a \in \Sigma$ префиксный multi-DFA, который описывает семейство языков $\{Out^{\Sigma}_{\pi}(p, a, q) : q \in Q_{\Sigma}\}$ и язык $Fin^{\varepsilon}_{\pi}(p, a)$. □

Заметим, что multi-DFAs, предложенные в доказательстве утверждения 17 можно построить за время, линейное относительно размера трансдьюсера π, и размер каждого такого multi-DFA не превосходит размера машины π. Алгоритм проверки эквивалентности префиксных однородных трансдьюсеров проводит анализ и преобразования семейств регулярных языков $Out^{\Sigma}_{\pi}(p, a, q)$, $Fin^{\Sigma}_{\pi}(p, a)$, $Out^{\varepsilon}_{\pi}(p, q)$, и $Fin^{\varepsilon}_{\pi}(p)$, используя указанные multi-DFAs как их формальные спецификации.

Вначале покажем, как построить систему уравнений $\mathcal{E}_\pi$, которая специфицирует отношения трансдукции, реализуемые в Σ-состояниях однородного трансдьюсера $\pi = \langle Q_\varepsilon \cup Q_\Sigma, F_\varepsilon \cup F_\Sigma, \longrightarrow\rangle$. Уравнения состоят из p-выражений, которые, как и в случае трансдьюсеров реального времени, строятся из переменных $X_1, X_2, \dots$, констант 0, 1, и букв алфавита Σ, но теперь вместо букв из алфавита Δ в качестве констант будут использоваться регулярные языки из класса $Reg(\Delta)$. Значением каждой такой константы $L \in Reg(\Delta)$ является трансдукция $\{(\varepsilon, w) : w \in L\}$. Разумеется, для этих новых констант будут верны тождества $L_1 + L_2 = L_3$ или $L_1 \cdot L_2 = L_3$ всякий раз, когда $L_1 \cup L_2 = L_3$ или $L_1 L_2 = L_3$. К каждой константе $L \in Reg(\Delta)$ будем обращаться по имени (описанию) какого-либо DFA, который задает язык L.

Обобщенное Δ-выражение — это всякое выражение вида $L_0 + L_1 \cdot X_1 + \dots + L_n \cdot X_n$, где $L_i \in Reg(\Delta)$ для каждого $i, 0 \le i \le n$. Обобщенное Δ-выражение такого вида называется префиксным, если языки $L_1, \dots, L_n$ являются префиксными, попарно вполне несовместными и не имеющими расширений в языке L_0. Ясно, что каждое обобщенное Δ-выражение E может быть описано multi-DFA D_E, который задает семейство регулярных языков $L_0, L_1, \dots, L_n$. Обобщенное Σ-выражение — это всякое выражение вида $a_1 \cdot E_1 + \dots + a_k \cdot E_k + L$, где $\Sigma = \{a_1, \dots, a_k\}$, $L \in Reg(\Delta)$, и $E_1, \dots, E_k$ — это обобщенные Δ-выражения.

Для заданного однородного трансдьюсера $\pi = \langle Q_\varepsilon \cup Q_\Sigma, F_\varepsilon \cup F_\Sigma, \longrightarrow\rangle$ определим систему уравнений $\mathcal{E}_\pi$ точно так же, как это было сделано для трансдьюсеров реального времени. С каждым Σ-состоянием p будем ассоциировать переменную X_p, и для каждой входной буквы $a \in \Sigma$ построим обобщенное Δ-выражение $E_{p,a} = \sum\limits_{q \in Q_\Sigma} Out^\Sigma_\pi(p, a, q) \cdot X_q + Fin^\Sigma_\pi(p, a)$. Нетрудно заметить, что если π — префиксный трансдьюсер, то всякое обобщенное Δ-выражение $E_{p,a}$ является префиксным. Поведение трансдьюсера π можно описать следующей системой уравнений

$$\mathcal{E}_\pi = \{X_p = \sum_{a \in \Sigma} a \cdot E_{p,x} + L_p \ : \ p \in Q_\Sigma\},$$

в которых $L_p = 1$, если $p \in F_\Sigma$, и $L_p = 0$ в противном случае.

Утверждение 18. *Для каждого однородного трансдьюсера π система уравнений $\mathcal{E}_\pi$ имеет единственное решение $\{X_p = TR(\pi, p) \ : \ p \in Q_\Sigma\}$.*

Доказательство. Проводится посредством тех же самых рассуждений, что и доказательство аналогичного утверждения 12. □

Для каждого состояния $p \in Q_\Sigma \cup Q_\varepsilon$ однородного трансдьюсера π определим обобщенное Δ-выражение $E_{\pi,p}$:

$$E_{\pi,p} = \begin{cases} X_p, \text{ если } p \in Q_\Sigma, \\ \sum\limits_{q \in Q_\Sigma} Out^\varepsilon_\pi(p, q) X_q + Fin^\varepsilon_\pi(p), \text{ если } p \in Q_\varepsilon \ . \end{cases}$$

Утверждение 19. *Для каждого однородного трансдьюсера π и пары его состояний $p', p'' \in Q_\Sigma \cup Q_\varepsilon$ система уравнений $\mathcal{E}_\pi \cup \{E_{\pi,p'} = E_{\pi,p''}\}$ имеет решение тогда и только тогда, когда $\pi(p') \sim \pi(p'')$.*

Доказательство. Согласно утверждению 18 система уравнений $\mathcal{E}_\pi$ имеет единственное решение $X_p = TR(\pi, p)$ для каждого Σ-состояния p. Тогда, как видно из определения множеств $Out^\varepsilon_\Pi(p, q)$ и $Fin^\varepsilon_\Pi(p)$, для этого набора значений переменных Δ-выражения $E_{\pi,p'}$ и $E_{\pi,p''}$ принимают значения $TR(\pi, p')$ и $TR(\pi, p'')$. □

Таким образом, проблема эквивалентности $\pi(p') \overset{?}{\sim} \pi(p'')$ для однородных трансдьюсеров сводится к задаче проверки разрешимости систем уравнений $\mathcal{E}_\pi \cup \{E_{\pi,p'} = E_{\pi,p''}\}$. И хотя в общем случае эта задача не имеет алгоритмов решения, для нее существует эффективная разрешающая процедура, если известно, что π — префиксный трансдьюсер.

Далее будет представлена процедура проверки разрешимости систем уравнений вида $\mathcal{E}_\pi \cup \{E'_1 = E''_1, \dots, E'_m = E''_m\}$, где π — префиксный однородный трансдьюсер, и E'_i, E''_i — префиксные Δ-выражения для всех $1 \le i \le m$,. Как и в случае трансдьюсеров реального времени, процедура основана на методе Гаусса исключения переменных и заключается в преобразовании исходной системы $\mathcal{E}_1$ к равносильной системе уравнений, представленной в приведенной форме

$$\mathcal{E}_{\pi'}(X_1, \dots, X_n) \cup \{Y_j = E_j(X_1, \dots, X_n) \,:\, 1 \le j \le r\},$$

путем поочередной деактивизации переменных системы $\mathcal{E}_1$.

Уравнения каждой такой системы $\mathcal{E}_t = \mathcal{E}_\pi \cup \{E'_1 = E''_1, \dots, E'_m = E''_m\}$ подразделяются, как и в случае трансдьюсеров реального времени, на базовые, приведенные и активные. На каждой итерации к системе уравнений $\mathcal{E}_t$ применяются следующие преобразования.

1) *Удаление тождеств.* Все активные уравнения $E' = E''$, являющиеся тождествами, удаляются из системы $\mathcal{E}_t$. Для этой цели достаточно проверить эквивалентность multi-DFAs $D_{E'}$ и $D_{E''}$, которые описывают Δ-выражения E' и E''.

2) *Проверка активности переменных.* Если $\mathcal{E}_t$ не содержит активных уравнений, то процедура завершает свое выполнение и объявляет о разрешимости системы уравнений в соответствии с утверждением 18.

3) *Обработка активных уравнений.* Из системы $\mathcal{E}_t$ выбирается какое-нибудь активное уравнение

$$\sum_{i=1}^{n} L'_i X_i + L'_0 = \sum_{i=1}^{n} L''_i X_i + L''_0 \,. \tag{4}$$

Если $L'_i = L''_i$ для всех $i, 1 \le i \le n$, то процедура завершает свое выполнение и объявляет о неразрешимости системы уравнений. Основания для этого решения таковы. Так как уравнение (4) не является тождеством, L'_0 и L''_0 — это различные языки. Поскольку языки $L'_i, 1 \le i \le n$, не имеют расширений в обоих языках L'_0 и L''_0, для любых трансдукций $T_1, \dots, T_n$ верно, что

$$L'_0 \cap \bigcup_{i=1}^{n} L'_i T_i = L''_0 \cap \bigcup_{i=1}^{n} L''_i T_i = \varnothing \,.$$

Поэтому левая и правая части уравнения (4) всегда будут иметь разные значения.

Если $L'_j \ne L''_j$ для некоторого $j, 1 \le j \le n$, то выберем самое короткое свидетельство того, что эти языки не равны, т.е. кратчайшее выходное слово u в непустом языке $\bigcup_{i=1}^{n}(L'_i \oplus L''_i)$. Будем далее полагать для определенности, что $u \in L'_1$ и при этом $u \notin L''_1$. Так как Δ-выражение $\sum_{i=1}^{n} L'_i X_i + L'_0$ является префиксным, все языки $L'_2, \dots, L'_n$ не имеют расширений в u. Поэтому, принимая во внимание тот факт, что u — это кратчайшее слово в множестве слов $\bigcup_{i=1}^{n}(L'_i \oplus L''_i)$, можно заключить, что языки $L''_1, \dots, L''_n$ также не имеют расширений в u. Поэтому, ввиду того, что u не имеет расширений в языке L'_0, для любых трансдукций $T_1, \dots, T_n$ верно, что

$$\sum_{i=1}^{n} L'_i T_i + L'_0 = \sum_{i=1}^{n} L''_i T_i + L''_0 \implies T_1 = \sum_{i=1}^{n} (L''_i \setminus u) T_i + (L''_0 \setminus u) \,.$$

Таким образом, $\mathcal{E}_t$ равносильна системе уравнений $\mathcal{E}_t \cup \{X_1 = \sum_{i=1}^{n}(L_i'' \setminus u)X_i + (L_0'' \setminus u)\}$.

Рассмотрим это дополнительное уравнение

$$X_1 = \sum_{i=1}^{n}(L_i'' \setminus u)X_i + (L_0'' \setminus u)\,. \tag{5}$$

Если $L_0'' \setminus u = L_1'' \setminus u = \cdots = L_n'' \setminus u = \emptyset$, то процедура завершает свое выполнение и объявляет о неразрешимости системы уравнений $\mathcal{E}_t$, так как согласно утверждению 18, подсистема $\mathcal{E}_\pi$ имеет своими решениями только непустые трансдукции.

В противном случае, необходимо заметить, что соотношение $\varepsilon \notin L_1'' \setminus u$ имеет место, ввиду указанной ранее особенности выбора слова u. Поэтому на основании теоремы Ардена [67, 71], уравнение (5) равносильно уравнению

$$X_1 = \sum_{i=2}^{n}(L_1'' \setminus u)^*(L_i'' \setminus u)X_i + (L_1'' \setminus u)^*(L_0'' \setminus u)\,. \tag{6}$$

Таким образом, уравнение (6) может быть введено в систему $\mathcal{E}_t$ вместо уравнения (5), и после этого все вхождения переменной X_1 в других уравнениях системы $\mathcal{E}_t$ можно заменить на Δ-выражение из правой части уравнения (6). В результате будет получена равносильная система уравнений, в которой появляется новое приведенное уравнение (6) и, тем самым, число приведенных переменных увеличивается на 1.

Но применение указанной подстановки к уравнениям системы $\mathcal{E}_t$ имеет те же побочные эффекты, что и в случае с трансдьюсерами реального времени: возникают нестандартные уравнения вида

A) $E = G$, где E — это отличное от переменной Δ-выражение, а G — это Σ-выражение, и

B) $E' = E''$, где E', E'' — это отличные от переменных Δ-выражения,

и, кроме того,

C) несколько базовых уравнений вида $X = G$ с одной и той же переменной X в их левых частях

могут появиться $\mathcal{E}_t$. Тогда каноническую форму системы уравнений можно восстановить при помощи равносильных преобразований 4) и 5), описанных в разделе 5. Утверждения 2–9 гарантируют, что после применения этих преобразований (включая и применение подстановки) все Δ-выражения в полученной системе уравнений $\mathcal{E}_{t+1}$ остаются префиксными.

Утверждение 20. *Для каждого префиксного однородного трансдьюсера π и пары его состояний p', p'' описанная процедура после применения к системе уравнений $\mathcal{E}_1 = \mathcal{E}_\pi \cup \{E_{\pi,p'} = E_{\pi,p''}\}$ всегда завершает работу и корректно устанавливает разрешимость системы $\mathcal{E}_1$.*

Доказательство. На каждой итерации t процедура либо завершает свою работу и корректно обнаруживает (не)разрешимость системы уравнений $\mathcal{E}_t$, либо строит равносильную систему уравнений $\mathcal{E}_{t+1}$, которая имеет большее количество разрешенных переменных, чем система $\mathcal{E}_t$. В результате процедура проверки либо обнаруживает, что система уравнений $\mathcal{E}_1$ не имеет решений, либо строит равносильную приведенную систему уравнений, которая согласно утверждению 18, свидетельствует о разрешимости исходной системы уравнений $\mathcal{E}_1$. □

Теорема 3. *Проблема эквивалентности для префиксных однородных трансдьюсеров разрешима за кубическое время.*

Доказательство. Разрешимость проблемы эквивалентности следует из утверждений 19 и 20. Число итераций, которые должна совершить описанная в этом разделе процедура проверки разрешимости, не превосходит числа состояний анализируемого трансдьюсера π. Как можно видеть из определения языков $Out_\pi^\Sigma(p,a,q)$, $Out_\pi^\varepsilon(p), Fin_\pi^\Sigma(p,a), Fin_\Sigma^\varepsilon(p)$, multi-DFAs, которые описывают Δ-выражения

в системе уравнений $\mathcal{E}_\pi$ и верифицирующем уравнении $E_{\pi,p'} = E_{\pi,p''}$ могут быть извлечены без всяких вычислительных издержек из системы переходов проверяемого трансдьюсера π. В разделе 3 было показано, что все преобразования Δ-выражений, которые осуществляет процедура проверки разрешимости, можно выполнить за линейное время без увеличения общего размера используемых multi-DFAs. И только две операции — проверка равенства регулярных языков $L(A') = L(A'')$, представленных автоматами, и вычисление кратчайшего слова, свидетельствующего о том, что эти языки не равны, т.е. слова $u \in L(A') \oplus L(A'')$ из симметричной разности двух регулярных языков, описанных при помощи DFAs A' и A'', — требуют бóльших вычислительных затрат. Обе эти операции можно выполнить за время, зависящее квадратично от размеров участвующих в этих операциях DFAs (см. [67, 72]). □

В следующих разделах будет показано, как можно воспользоваться префиксными однородными трансдьюсерами для проверки эквивалентности детерминированных двухленточных автоматов и детерминированных биавтоматов.

7. Проверка эквивалентности детерминированных двухленточных автоматов

Двухленточный конечный автомат (далее, 2-FSA), работающий в алфавитах Σ и Δ, задается системой переходов $M = \langle S_1, S_2, F, \dashrightarrow \rangle$; эта система определяется подмножествами состояний S_1, S_2, на которые разбито конечное множество *состояний* $S = S_1 \cup S_2$, подмножеством *финальных состояний* $F \subseteq S$, и *отношением переходов* $\dashrightarrow$, которое имеет тип $(S_1 \times \Sigma \times S) \cup (S_2 \times \Delta \times S)$. Переходы будем обозначать записями $s \overset{z}{\dashrightarrow} s'$, где $z \in \Sigma$ в случае $s \in S_1$, и $z \in \Delta$ в случае $s \in S_2$. Условимся использовать обозначение $\dashrightarrow_\Delta$ для проекции отношения переходов $\dashrightarrow$ на алфавит Δ, т.е. $\dashrightarrow_\Delta = \dashrightarrow \cap\, S_2 \times \Delta \times S$.

Прогон 2-FSA M — это любая последовательность переходов вида

$$s \overset{z_1}{\dashrightarrow} s_1 \overset{z_2}{\dashrightarrow} \cdots \overset{z_{n-1}}{\dashrightarrow} s_{n-1} \overset{z_n}{\dashrightarrow} s' \,. \tag{7}$$

Прогон (7) называется *финальным*, если $s' \in F$. Говорят, что 2-FSA M *допускает* пару слов $(h, w) \in \Sigma^* \times \Delta^*$, начиная работу в состоянии s, если у автомата M есть такой финальный прогон (7), что h — это проекция слова $z_1 z_2 \ldots z_{n-1} z_n$ на алфавит Σ, а w — это проекция того же самого слова $z_1 z_2 \ldots z_{n-1} z_n$ на алфавит Δ. *Трансдукция, распознаваемая 2-FSA M из состояния s* — это множество $TR(M, s)$ всех пар слов (h, w), допускаемых машиной M, которая начинает работу из указанного состояния. 2-FSA M называется *детерминированным* (2-DFSA), если для каждой буквы z и состояния s он имеет не более одного перехода вида $s \overset{z}{\dashrightarrow} s'$.

Состояния s' и s'' из множества состояний 2-FSAs M считаются *эквивалентными* (для обозначения этого отношения используем запись $M(s') \sim M(s'')$), если $TR(M, s') = TR(M, s'')$. *Проблема эквивалентности* для 2-FSAs состоит в том, чтобы для заданного 2-FSA M и заданной пары его состояний s' и s'' выяснить выполнимость отношения $M(s') \sim M(s'')$.

Взаимосвязь между 2-FSAs и трансдьюсерами очевидна: если потребовать, чтобы двухленточный автомат вместо считывания букв на второй ленте, записывал те же самые буквы этой ленты, то в результате 2-FSA превращается в однородный трансдьюсер, который вычисляет в своих состояниях те же самые отношения трансдукции. Если система переходов $M = \langle S_1, S_2, F, \dashrightarrow \rangle$ не имеет циклов, проходящих только по состояниям множества S_2 (т.е. автомат не может сколь угодно долго считывать буквы на второй ленте), то 2-FSA M можно превратить в эквивалентный трансдьюсер реального времени. Однако оставалось неясным, какому классу трансдьюсеров соответствуют 2-DFSAs. Выделив класс префиксных однородных трансдьюсеров, мы можем дать ответ на этот вопрос.

Рассмотрим произвольный 2-FSA $M = \langle S_1, S_2, F, \dashrightarrow \rangle$, и построим соответствующий ему трансдьюсер $\pi_M = \langle S_1 \cup S_2, F, \longrightarrow \rangle$, определив для любой пары состояний s, s' и всякой буквы $x \in \Sigma \cup \Delta$ отношение переходов по следующим правилам:

- π_M имеет переход $s \xrightarrow{x|\varepsilon} s'$ тогда и только тогда, когда $s \in S_1$, $x \in \Sigma$, и M имеет переход $s \overset{x}{\dashrightarrow} s'$,
- π_M имеет переход $s \xrightarrow{\varepsilon|y} s'$ тогда и только тогда, когда $s \in S_2$, $y \in \Delta$, и M имеет переход $s \overset{y}{\dashrightarrow} s'$.

Утверждение 21. *Для произвольного заданного 2-DFSA M соответствующий ему трансдьюсер π_M является префиксным и однородным, и при этом равенство $TR(M, s) = TR(\pi_M, s)$ выполняется для любого состояния $s \in S_1 \cup S_2$.*

Доказательство. Как показывает определение отношения переходов трансдьюсера π_M, из любого его состояния s существует финальный прогон, помеченный парой слов (h, w), в том и только том случае, когда 2-DFSA M, начиная работу в состоянии s, допускает эту пару слов. Поэтому $TR(M, s) = TR(\pi_M, s)$.

Что касается свойства префиксности, то для его подтверждения достаточно заметить, что если $S_1 = \{s_1, \dots, s_m\}$, то для каждого состояния $s \in S_2$ multi-DFA $D = \langle S, s, \{s_1\}, \dots, \{s_m\}, F \cap S_2, \dashrightarrow_\Delta \rangle$ является префиксным. □

Объединив утверждение 21 и теорему 3, получаем решение проблемы эквивалентности для 2-DFSAs.

Теорема 4. *Проблема эквивалентности для детерминированных двухленточных конечных автоматов разрешима за время, кубическое относительно их размеров.*

8. Проблема эквивалентности для детерминированных биавтоматов

Конечный *биавтомат* (BA), работающий в алфавите Σ, задается системой переходов $B = \langle Q, F, Left, Right \rangle$, в которой Q — это конечное множество *состояний*, $F \subseteq Q$ — это подмножество *финальных состояний*, а $Left$ и $Right$ — это *отношения переходов* типа $Q \times \Sigma \times Q$.

Биавтомат B имеет две головки, которые считывают на ленте буквы входного слова. Левая головка считывает буквы с левого конца слова, а правая головка — с правого конца этого же слова. На каждом шаге вычисления биавтомат B выбирает одну из головок, считывает с ее помощью очередную букву входного слова и переходит в новое состояние в соответствии с отношением переходов $Left$ (если для считывания была выбрана левая головка) или $Right$ (если букву считывала правая головка). Вычисление завершается, как только головки встречаются на ленте друг с другом после прочтения всех букв входного слова. Если по окончании вычисления биавтомат B оказывается в финальном состоянии, то он допускает входное слово.

Вычисления биавтомата удобнее всего определить в терминах конфигураций. *Конфигурацией* биавтомата B называется пара $\alpha = (q, w)$, в которой q — это состояние B, а w — это слово в алфавите Σ (та часть входного слова, которую еще предстоит прочитать биавтомату). Отношение $\vdash$ на множестве конфигураций биавтомата B определяется для каждой пары его состояний q, q', слова w, и буквы a следующим образом: $(q, aw) \vdash (q', w)$ выполняется, если $(q, a, q') \in Left$, и $(q, wa) \vdash (q', w)$ выполняется, если $(q, a, q') \in Right$. *Прогоном* биавтомата B на входном слове w из состояния q называется всякая последовательность конфигураций

$$\alpha_0 \vdash \alpha_1 \vdash \dots \alpha_{n-1} \vdash \alpha_n, \tag{8}$$

в которой $\alpha_0 = (q, w)$. Прогон (8) называется *финальным* в том случае, когда $\alpha_n = (q', \varepsilon)$ для некоторого состояния $q' \in F$. Биавтомат B *допускает* входное слово w из состояния q, если существует его финальный прогон на слове w из состояния q. *Язык, распознаваемый биавтоматом B* из состояния

q, — это множество $L(B, q)$ всех слов, которые биавтомат B допускает из этого состояния. Условимся использовать запись $\mathcal{L}(BA)$ для обозначения класса языков, распознаваемых биавтоматами.

Состояния q' и q'' биавтомата B считаются *эквивалентными* (и этот факт обозначается записью $B(q') \sim B(q'')$), если $L(B, q') = L(B, q'')$. *Проблема эквивалентности* для биавтоматов состоит в том, чтобы для произвольного заданного биавтомата B и произвольной заданной пары его состояния q' и q'' выяснить, выполняется ли отношение $B(q') \sim B(q'')$.

В статьях [12, 14, 43] были выделены некоторые специальные классы биавтоматов. Биавтомат $B = \langle Q, F, Left, Right\rangle$ называется *детерминированным* (DBA), если множество его состояний Q разбито на два подмножества Q_L и Q_R так, что отношения переходов $Left$ и $Right$ представляют собой (частичные) функции $Left : Q_L \times \Sigma \longrightarrow Q$ и $Right : Q_R \times \Sigma \longrightarrow Q$. DBA B называется *ослабленным справа* (DWBA), если $F \subseteq Q_L$ и для любого состояния $q \in Q_R$ существует единственная буква $a \in \Sigma$, для которой определено значение функции $Right(q, a)$. Луканова в статье [12] доказала, что $\mathcal{L}(BA)$ совпадает с классом **Lin** линейных к-с языков, и поэтому проблема эквивалентности для произвольных биавтоматов неразрешима. Йираскова и Клима установили в статье [14], что $\mathcal{L}(DWBA)$ совпадает с классом **DLin** детерминированных линейных к-с языков, и это означает, что проблема эквивалентности для DWBAs разрешима. Однако в работе [43] было показано, что $\mathcal{L}(DBA)$ несравнимо с семейством детерминированных к-с языков; это означает, что метод проверки эквивалентности DPDAs, разработанный в статье [60], неприменим для решения проблемы эквивалентности для DBAs, и поэтому вопрос об алгоритмической разрешимости этой задачи оставался открытым.

Мы покажем, как можно эффективно проверять эквивалентность DBAs, используя взаимосвязь между DBAs и префиксными трансдьюсерами и применяя метод исключения переменных для проверки разрешимости систем линейных уравнений.

В статье [1] Розенберг обнаружил простое соответствие между 2-FSAs и линейными к-с языками: для любого линейного к-с языка L существуют такие 2-FSA M и его состояние s, для которых выполняется равенство $L = \{uv^{-1} : (u, v) \in TR(M, s)\}$. Так как биавтоматы распознают линейные к-с языки, а 2-FSAs — бинарные отношения, вычислимые однородными трансдьюсерами, появляется возможность транслировать биавтоматы в трансдьюсеры.

Для заданного биавтомата $B = \langle Q, F, Left, Right\rangle$, работающего в алфавите Σ, рассмотрим трансдьюсер $\pi_B = \langle Q, F, \longrightarrow\rangle$, который имеет одно и то же множество букв Σ в качестве входного и выходного алфавитов и отношение переходов которого определено следующим образом: для каждой пары состояний p, q и буквы a четверка $p \xrightarrow{a|\varepsilon} q$ (или $p \xrightarrow{\varepsilon|a} q$) является переходом трансдьюсера π_B в том и только том случае, если $(p, a, q) \in Left$ (или $(p, a, q) \in Right$ соответственно).

Утверждение 22. *Для любого биавтомата $B = \langle Q, F, Left, Right\rangle$, соответствующего ему трансдьюсера π_B и состояния $q \in Q$ верно, что*

1. $L(B, q) = \{uv^{-1} : (u, v) \in TR(\pi_B, q)\}$;
2. *Если биавтомат B детерминированный, то π_B — это префиксный однородный трансдьюсер.*

Доказательство. Следует непосредственно из определений DBA, языка $L(B, q)$, распознаваемого биавтоматом BA B, отношения трансдукции $TR(\pi, q)$, вычисляемого произвольным трансдьюсером π, и указанного выше соответствия между биавтоматом B и трансдьюсером π_B. □

Будем говорить, что трансдьюсер π *порождает язык сцепления* $LC(\pi, q) = \{uv^{-1} : (u, v) \in TR(\pi, q)\}$ из состояния q. Состояния p и q трансдьюсера π считаются *ctn-эквивалентными* (обозначается $\pi(p) \sim_{ctn} \pi(q)$), если $LC(\pi, p) = LC(\pi, q)$. Таким образом, согласно утверждению 22, проблема эквивалентности для DBAs сводится к проблеме ctn-эквивалентности для префиксных однородных трансдьюсеров.

Для проверки ctn-эквивалентности $\pi(p) \sim_{ctn} \pi(q)$ префиксных однородных трансдьюсеров можно применить тот же метод, при помощи которого в разделе 6 осуществлялась проверка эквивалентности $\pi(p) \sim \pi(q)$: построить систему языковых уравнений $\hat{\mathcal{E}}_\pi$, описывающую языки сцепления, порождаемые трансдьюсером π, добавить к этой системе проверочное уравнение $X_p = X_q$ и проверить разрешимость полученной системы уравнений методом Гаусса исключения переменных. Но на этот раз регулярные выражения, из которых формируются уравнения, будут интерпретироваться в полукольце языков в алфавите Σ.

Уравнения системы $\hat{\mathcal{E}}_\pi$ строятся из p-выражений, а сами выражения образуются из переменных $X_1, X_2, \dots$, констант 0, 1, и регулярных языков из семейства $Reg(\Sigma)$. Регулярные языки расцениваются как константы, их именами служат те DFAs, которые описывают эти языки. Условимся считать $\hat{\Delta}$-выражением $\hat{E}$ всякое выражение вида $L_0 + L_1 \cdot X_1 + \dots + L_n \cdot X_n$, где $L_i \in Reg(\Sigma)$ для каждого $i, 0 \le i \le n$. И хотя здесь рассматриваются трансдьюсеры, у которых входные и выходные слова задаются в одном и том же алфавите Σ, в целях сохранения единообразия введенная ранее классификация p-выражений будет сохранена. Будем называть $\hat{\Delta}$-выражение $\hat{E}$ префиксным, если языки $L_1, \dots, L_n$ являются префиксными, попарно вполне несравнимыми и свободными от расширения в языке L_0. Для представления любого $\hat{\Delta}$-выражения такого рода можно использовать multi-DFA $D_{\hat{E}}$, который описывает семейство регулярных языков $L_1, \dots, L_n, L_0$. Будем называть $\hat{\Sigma}$-выражением всякое выражение $\hat{G}$ вида $\hat{E}_1 \cdot a_1 + \dots + \hat{E}_k \cdot a_k + L$, где $\Sigma = \{a_1, \dots, a_k\}$, $L \in Reg(\Sigma)$, и $\hat{E}_1, \dots, \hat{E}_k$ — это $\hat{\Delta}$-выражения.

Для произвольного заданного однородного трансдьюсера $\pi = \langle Q_\varepsilon \cup Q_\Sigma, F_\varepsilon \cup F_\Sigma, \longrightarrow \rangle$ система уравнений $\hat{\mathcal{E}}_\pi$ определяется так. Для каждого состояния $p \in Q$ и буквы $a \in \Sigma$ конструируется $\hat{\Delta}$-выражение $\hat{E}_{p,a} = \sum\limits_{q \in Q_\Sigma} Out^\Sigma_\pi(p, a, q) \cdot X_q + Fin^\Sigma_\pi(p, a)$. Нетрудно заметить, что если π — это префиксный трансдьюсер, то каждое $\hat{\Delta}$-выражение $\hat{E}_{p,a}$ также является префиксным. Семейство языков сцепления, порождаемое трансдьюсером π задается следующей системой уравнений

$$\hat{\mathcal{E}}_\pi = \{X_p = \sum_{a \in \Sigma} \hat{E}_{p,a} \cdot a + L_p \ : \ p \in Q_\Sigma\},$$

в которой $L_p = 1$, если $p \in F_\Sigma$, и $L_p = 0$ в противном случае. Следуя той же схеме доказательства, что и для утверждения 12 и принимая во внимание изменения, которые были внесены в определение p-выражений, можно легко доказать

Утверждение 23. *Для каждого однородного трансдьюсера π система уравнений $\hat{\mathcal{E}}_\pi$ имеет единственное решение $\{X_p = LC^{-1}(\pi, p) \ : \ p \in Q_\Sigma\}$.*

Чтобы записать проверочные уравнения, определим для каждого состояния p однородного трансдьюсера π специальное $\hat{\Delta}$-выражение $\hat{E}_{\pi,p}$:

$$\hat{E}_{\pi,p} = \begin{cases} X_p, \text{ если } p \in Q_\Sigma, \\ \sum\limits_{q \in Q_\Sigma} Out^\varepsilon_\pi(p, q) \cdot X_q + Fin^\varepsilon_\pi(p), \text{ если } p \in Q_\varepsilon \ . \end{cases}$$

Утверждение 24. *Для каждого однородного трансдьюсера π и любой пары состояний $p', p'' \in Q_\Sigma \cup Q_\varepsilon$ система уравнений $\hat{\mathcal{E}}_\pi \cup \{\hat{E}_{\pi,p'} = \hat{E}_{\pi,p''}\}$ имеет решение тогда и только тогда, когда $\pi(p') \sim_{ctn} \pi(p'')$.*

Доказательство. Как следует из утверждения 23, система уравнений $\hat{\mathcal{E}}_\pi$ имеет единственное решение $X_p = LC^{-1}(\pi, p)$ для каждого состояния $p \in Q_\Sigma$. Обратившись к определениям множеств $Out^\varepsilon_\pi(p, q)$ и $Fin^\varepsilon_\pi(p)$, можно заметить, что для такого решения $\hat{\Delta}$-выражения $\hat{E}_{\pi,p'}$ и $\hat{E}_{\pi,p''}$ в обеих частях проверочного уравнения принимают значения $LC^{-1}(\pi, p')$ и $LC^{-1}(\pi, p'')$. □

Разрешимость систем уравнений $\hat{\mathcal{E}}_\pi \cup \{\hat{E}'_1 = \hat{E}''_1, \ldots, \hat{E}'_m = \hat{E}''_m\}$, где π — это префиксный однородный трансдьюсер, и $\hat{E}'_i, \hat{E}''_i, 1 \le i \le m$, — это префиксные $\hat{\Delta}$-выражения, можно проверять, придерживаясь того же самого сценария по деактивизации переменных, который был описан в разделе 6, т.е. преобразуя системы уравнений к равносильной приведенной форме. На каждой итерации процедура проверки применяет следующие равносильные преобразования к сложившейся системе уравнений $\hat{\mathcal{E}}_t$.

1) *Удаление тождеств* $\hat{E} = \hat{E}$.
2) *Проверка активности переменных.*
3) *Обработка активных уравнений* $\hat{E}' = \hat{E}''$.
4) *Удаление нестандартных уравнений вида* $\hat{E} = \hat{G}$.

Эти преобразования совершенно аналогичны тем правилам, которые описаны в разделе 6. Единственное отличие имеет правило удаления нестандартных уравнений вида $\hat{G}' = \hat{G}''$, где $\hat{G}', \hat{G}''$ — Σ-выражения.

5) *Удаление нестандартных уравнений вида* $\hat{G}' = \hat{G}''$.

Если система содержит уравнение

$$\sum_{i=1}^{k} \hat{E}'_i \cdot a_i + L' = \sum_{i=1}^{k} \hat{E}''_i \cdot a_i + L'' \,,$$

то нужно удалить его из системы и добавить вместо него более простые новые уравнения $\hat{E}'_i + L'/a_i = \hat{E}''_i + L''/a_i, 1 \le i \le k$. Нетрудно заметить, что в результате применения этого правила будет получена равносильная система уравнений. Согласно утверждению 3, в новых уравнениях $\hat{\Delta}$-выражения по обе стороны равенства остаются префиксными. Таким образом, все нестандартные уравнения вида $\hat{G}' = \hat{G}''$ исчезают из системы $\hat{\mathcal{E}}_t$, и мы получаем равносильную систему уравнений $\hat{\mathcal{E}}_{t+1}$ с меньшим числом активных переменных. Как и в разделе 6, для представления и обработки $\hat{\Delta}$-выражений используются multi-DFAs. Указанные соображения, вкупе с утверждениями 23 и 24 приводят к следующему результату.

Теорема 5. *Проблема эквивалентности для детерминированных конечных биавтоматов разрешима за кубическое время.*

Описанный здесь метод проверки эквивалентности применим только к детерминированным биавтоматам (DBAs). Решающая роль в нем отводится правилам 3) деактивизации переменных и 5) удаления нестандартных уравнений вида $\hat{G}' = \hat{G}''$. Эти правила требуют, чтобы трансдьюсер π_B обладал свойством префиксности, а оно является следствием детерминированности биавтомата B.

Разрешающий алгоритм можно существенно упростить для некоторых классов DBAs. Например, если биавтомат не имеет возможности сколь угодно долго считывать буквы, оставаясь при этом все время на правой стороне слова, то его можно транслировать в префиксный трансдьюсер реального времени, для которого проверку ctn-эквивалентности состояний можно проводить значительно эффективнее, чем для префиксных однородных трансдьюсеров.

Для произвольного DBA $B = \langle Q_L, Q_R, F, Left, Right\rangle$ его функцию переходов $Right : Q_R \times \Sigma \longrightarrow Q_L \cup Q_R$ можно обычным образом распространить на слова из Σ^*: для любых состояния $q \in Q_R$, слова $w \in \Sigma^*$ и буквы $x \in \Sigma$ будем полагать $Right^*(q, \varepsilon) = q$ и $Right^*(q, wx) = Right(p, x)$, если $Right^*(q, w) = p \in Q_R$. DBA B назовем *детерминированным ациклическим справа биавтоматом* (DABA), если $F \subseteq Q_L$ и $Right^*(q, w) \ne q$ для любого состояния $q \in Q_R$ и всякого непустого слова $w \ne \varepsilon$. Как видно из этого определения, каждый детерминированный ослабленный справа биавтомат (DWBA) является ациклическим справа. При этом верны строгие включения $\mathcal{L}(DWBA) \subset \mathcal{L}(DABA) \subset \mathcal{L}(DBA)$, поскольку

- язык $\{a^n b^n : n \geq 0\} \cup \{a^n c^n : n \geq 0\}$, очевидно, принадлежит классу $\mathcal{L}(DABA)$, но не принадлежит семейству языков $\mathcal{L}(DWBA)$;
- язык $\{a^n (bc^*)^n : n \geq 0\} : n \geq 0\}$, очевидно, принадлежит классу $\mathcal{L}(DBA)$, но не принадлежит семейству языков $\mathcal{L}(DABA)$.

Сопоставим DBA $B = \langle Q_L, Q_R, F, Left, Right\rangle$ трансдьюсер $\pi_B = \langle Q_L, F, \longrightarrow\rangle$, в котором отношение переходов $\longrightarrow$ определено следующим образом: для любой пары состояний $p, q \in Q_L$, буквы $x \in \Sigma$ и слова $w \in \Sigma^*$ трансдьюсер π_B имеет переход $p \xrightarrow{x|w} q$ в том и только том случае, если выполнено одно из двух условий

1. $Left(p, x) = q$ и $w = \varepsilon$, или
2. $Left(p, x) = r \in Q_R$ и $Right^*(r, w) = q$.

Для предложенной альтернативной трансляции биавтоматов в трансдьюсеры реального времени верно

Утверждение 25. *Для любого DABA B соответствующий ему трансдьюсер π_B является префиксным конечным трансдьюсером реального времени, и для любого его состояния $q \in Q_L$ верно равенство $L(B, q) = \{uv^{-1} : (u, v) \in TR(\pi_B, q)\}$.*

Доказательство. Так как биавтомат является ациклическим справа, для любого состояния $r \in Q_R$ обобщенная функция переходов $Right^*(r, w)$ определена на конечном множестве слов w. Поэтому, как следует из определения отношения переходов для трансдьюсера π_B, для всякой буквы $x \in \Sigma$ из любого состояния $p \in Q_L$ исходит конечное число переходов вида $p \xrightarrow{x|w} q$. Поскольку биавтомат B является детерминированным, для любого состояния $q \in Q_R$ множество слов $\{w : Right^*(q, w) \in Q_L\}$ обладает свойством префиксности. Поэтому, как следует из определения отношения переходов для трансдьюсера π_B, для любого состояния p и всякой буквы x множества слов $Out_{\pi_B}(p, q, x)$, где $q \in Q_L$, являются попарно вполне несовместными префиксными языками. Значит, π_B — префиксный трансдьюсер реального времени. И, наконец, как это было уже отмечено в утверждении 22, равенство $L(B, q) = \{uv^{-1} : (u, v) \in TR(\pi_B, q)\}$ непосредственно следует из определения языка $L(B, q)$ и отношения переходов для π_B. □

Таким образом, проблема эквивалентности для DABAs может быть сведена к задаче проверки ctn-эквивалентности состояний префиксных конечных трансдьюсеров реального времени, и эту задачу можно решить путем проверки разрешимости соответствующих систем линейных уравнений так же, как это было показано ранее для префиксных однородных трансдьюсеров. Отличие будет состоять в том, что теперь коэффициентами Δ-выражений будут не префиксные регулярные языки, а отдельные слова. Поэтому для проверки разрешимости таких систем уравнений можно воспользоваться модифицированной процедурой, описанной в разделе 5 и основанной на утверждении 10. Эта модификация, по существу, касается только формы уравнений и приводит к следующим результатам.

Теорема 6. *Проблема эквивалентности для детерминированных ациклических справа конечных биавтоматов (DABAs) разрешима за квадратичное время.*

Следствие 2. *Проблема эквивалентности для детерминированных ослабленных справа конечных биавтоматов (DWBAs) разрешима за квадратичное время.*

Следствие 3. *Проблема эквивалентности для детерминированных линейных к-с грамматик разрешима за квадратичное время.*

9. Проблема эквивалентности для простых грамматик

Метод Гаусса исключения переменных можно также успешно применять для проверки разрешимости систем нелинейных уравнений, которые описывают языки, порождаемые некоторыми к-с грамматиками, и, таким образом, получать эффективное решение проблемы эквивалентности для этих грамматик. Впервые алгебраическим подходом к решению проблемы эквивалентности для к-с грамматик воспользовались авторы статьи [45], доказав разрешимость указанной задачи для т.н. простых грамматик. В этом разделе мы покажем, как можно подходящим образом приспособить общую схему метода исключения переменных для проверки разрешимости систем уравнений, описывающих задачу проверки эквивалентности нетерминалов простых грамматик.

Контекстно-свободная грамматика в алфавите Σ задается парой $\Gamma = \langle \mathcal{N}, \mathcal{P} \rangle$, состоящей из конечного множества *нетерминалов* $\mathcal{N}$ и конечного множества *грамматических правил* (продукций) $\mathcal{P}$. Символы множества $\mathcal{N}$ (нетерминалы) отличны от букв алфавита Σ (терминалов). Для удобства именования условимся последовательности букв в алфавите Σ называть терминальными словами и обозначать символами u, v, w, а последовательности букв в алфавите $\Sigma \cup \mathcal{N}$ называть предложениями и обозначать символами α, β, γ. Грамматические правила — это пары вида $N \longrightarrow \alpha$, где N — нетерминал из множества $\mathcal{N}$, который называется *заголовком правила*, а α — предложение, которое называется *телом правила* . Грамматика Γ задает *отношение непосредственной выводимости* $\longrightarrow_\Gamma$ на множестве предложений: два предложения β_1, β_2 находятся в отношении непосредственной выводимости $\beta_1 \longrightarrow_\Gamma \beta_2$ в том и только том случае, если $\beta_1 = \beta' N \beta''$, $\beta_2 = \beta' \alpha \beta''$ и при этом множество $\mathcal{P}$ содержит правило $N \longrightarrow \alpha$. *Отношение грамматического вывода* $\overset{*}{\longrightarrow}_\Gamma$, порождаемое грамматикой Γ, — это рефлексивно-транзитивное замыкание отношения непосредственной выводимости $\longrightarrow_\Gamma$. *Язык предложения* α в грамматике Γ — это множество терминальных слов $L(\Gamma, \alpha) = \{w : w \in \Sigma^*, \alpha \overset{*}{\longrightarrow}_\Gamma w\}$, выводимых из предложения α в грамматике Γ.

Нетерминал N называется *несущественным* в грамматике Γ, если $L(\Gamma, N) = \emptyset$. Как показано в учебнике [67], все несущественные нетерминалы к-с грамматики можно выявить за квадратичное время. Все правила, содержащие несущественные нетерминалы, можно удалить из грамматики Γ, и в результате будет получена грамматика Γ', для любого нетерминала которой верно равенство $L(\Gamma, N) = L(\Gamma', N)$. Поэтому при изучении проблемы эквивалентности ограничиваются рассмотрением к-с грамматик, не содержащих несущественных нетерминалов.

Два предложения α_1, α_2 считаются *эквивалентными в грамматике* Γ (обозначается $\alpha_1 \sim_\Gamma \alpha_2$), если $L(\Gamma, \alpha_1) = L(\Gamma, \alpha_2)$. Нетрудно заметить, что для к-с грамматик отношение эквивалентности предложений является отношением конгруэнтности относительно операции конкатенации: соотношение $\beta \sim_\Gamma \gamma \Longrightarrow \alpha' \beta \alpha'' \sim_\Gamma \alpha' \gamma \alpha''$ выполняется для любых предложений $\alpha', \alpha'', \beta, \gamma$.

Проблема эквивалентности нетерминалов состоит в том, чтобы для любой заданной грамматики Γ и для любой пары ее нетерминалов N_1, N_2 проверить выполнимость отношения эквивалентности $N_1 \sim_\Gamma N_2$.

Проблему эквивалентности нетерминалов будем решать для класса простых грамматик. Грамматическое правило находится в *нормальной форме*, если оно имеет вид $N \longrightarrow x\alpha$, где x — терминал, который называется *ключом правила*, а $\alpha \in \mathcal{N}^*$ — цепочка нетерминалов. К-с грамматика Γ называется *простой* [45], если все правила грамматики находятся в нормальной форме, и при этом все правила с одинаковым заголовком имеют попарно различные ключи.

Как видно из определения, простая грамматика Γ обладает свойством однозначности: для любого нетерминала N и слова $w \in L(\Gamma, N)$ существует единственный левосторонний грамматический вывод w из N. Это свойство проявляет себя в следующем утверждении, которое важно для решения проблемы эквивалентности.

Утверждение 26. *Пусть* $\Gamma = \langle \mathcal{N}, \mathcal{P} \rangle$ *— простая грамматика. Тогда для любых предложений* $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathcal{N}^*$ *и терминального слова* w *верно соотношение*

$$\alpha_1 \sim_\Gamma \alpha_2 \ \wedge \ \alpha_1 \xrightarrow{*}_\Gamma w\beta_1 \ \wedge \ \alpha_2 \xrightarrow{*}_\Gamma w\beta_2 \implies \beta_1 \sim_\Gamma \beta_2.$$

В статье [45] было установлено, что для любой простой грамматики $\Gamma = \langle \mathcal{N}, \mathcal{P} \rangle$ и нетерминала $N \in \mathcal{N}$ язык $L(\Gamma, N)$ является префиксным. Поэтому, как заметили авторы указанной статьи, для решения задач анализа простых грамматик целесообразно опираться на некоторые основополагающие свойства префиксных языков. К их числу относятся законы левого и правого сокращения префиксных языков относительно операции конкатенации.

Утверждение 27 ([45])**.** *Для любых префиксных языков* L, L', L'' *выполняются следующие соотношения:*

1) $LL' = LL'' \iff L' = L''$,
2) $L'L = L''L \iff L' = L''$.

Алгоритм проверки эквивалентности двух нетерминалов M, N произвольной заданной простой грамматики $\Gamma = \langle \mathcal{N}, \mathcal{P} \rangle$ состоит из двух этапов: построение системы уравнений $\mathcal{E}(\Gamma, M, N)$, которая описывает проблему эквивалентности, и проверка разрешимости построенной системы уравнений.

Уравнения системы $\mathcal{E}(\Gamma, M, N)$ строятся из выражений, а сами выражения образуются из переменных $X_1, X_2, \ldots$ и констант при помощи операций конкатенации $\cdot$ и альтернации $+$. Каждый нетерминал X рассматривается как переменная. Константами являются $0, 1$, и терминалы из алфавита Σ. Все выражения интерпретируются в полукольце языков в алфавите Σ.

Для каждого нетерминала X формируется базовое уравнение $X = E_X$, где $E_X = \sum\limits_{X \to \alpha \in \mathcal{P}} \alpha$. Система уравнений, описывающая задачу проверки эквивалентности $M \overset{?}{\sim}_\Gamma N$, имеет вид $\mathcal{E}(\Gamma, M, N) = \mathcal{E}_\Gamma \cup \{M = N\}$, где $\mathcal{E}_\Gamma = \{X = E_X \ : \ X \in \mathcal{N}\}$.

Утверждение 28 ([45])**.** *Для любой простой грамматики* Γ*, не содержащей несущественных нетерминалов, система базовых уравнений* $\mathcal{E}_\Gamma$ *имеет единственное решение* $X = L(\Gamma, X)$ *для каждого нетерминала* X*.*

Следствие 4. *Для любой простой грамматики* Γ*, не содержащей несущественных нетерминалов, система уравнений* $\mathcal{E}(\Gamma, M, N) = \mathcal{E}_\Gamma \cup \{M = N\}$ *имеет решение тогда и только тогда, когда* $M \sim_\Gamma N$*.*

На втором этапе алгоритм проверки эквивалентности $M \sim_\Gamma N$ применяет процедуру проверки разрешимости систем уравнений вида

$$\mathcal{E}_t = \mathcal{E}_\Gamma \cup \{\alpha_1 = \beta_1, \ldots, \alpha_m = \beta_m\}, \tag{9}$$

где $\alpha_i, \beta_i \in \mathcal{N}^*$. Проверочное уравнение $\alpha_i = \beta_i$ системы (9) называется *приведенным*, если предложение α_i состоит из единственной переменной X_j, которая не имеет других вхождений в уравнения системы; переменная X_j в этом случае также называется *приведенной*. Система уравнений (9) называется *приведенной*, если все проверочные уравнения являются приведенными. Как следует из утверждения 28, всякая приведенная система уравнений имеет решение. На основании этого, процедура проверки разрешимости систем уравнений вида (9) применяет равносильные преобразования в стремлении построить приведенную систему. На каждой итерации эта процедура либо увеличивает число приведенных переменных, либо обнаруживает несовместные уравнения, и это означает, что система не имеет решения.

Процедура начинает работу с системой уравнений $\mathcal{E}_1 = \mathcal{E}(\Gamma, M, N)$, и на каждой итерации t применяет к сложившейся системе уравнений (9) следующие правила.

1) *Упрощение уравнений.* К обеим частям проверочных уравнений применяются законы левого и правого сокращения (утверждение 27) $X\alpha = X\beta \Longrightarrow \alpha = \beta$ и $\alpha X = \beta X \Longrightarrow \alpha = \beta$. Если в результате применения этих преобразований образуется тождество $1 = 1$, то оно удаляется из системы. В том случае если одно из проверочных уравнений преобразуется к виду $X\alpha = 1$ или $1 = X\beta$, то процедура проверки прекращает работу и объявляет систему уравнений $\mathcal{E}(\Gamma, M, N)$ неразрешимой. Если таких несовместных уравнений не возникнет, то после сокращений переменных все оставшиеся в системе проверочные уравнения будут иметь вид $X\alpha = Y\beta$, где X и Y — различные переменные.

2) *Проверка активности.* Если в системе $\mathcal{E}_t$ все проверочные уравнения $\alpha_i = \beta_i$ являются приведенными, то $\mathcal{E}_t$ — приведенная система уравнений. Тогда процедура проверки разрешимости завершает свое выполнение и объявляет о разрешимости исходной системы уравнений $\mathcal{E}(\Gamma, M, N)$.

3) *Построение уравнений вида $X = \alpha$.* Если в системе $\mathcal{E}_t$ содержится проверочное уравнение $X\alpha = Y\beta$, и при этом $|\alpha| > 0$ и $|\beta| > 0$, то проводится сравнение длин кратчайших терминальных слов u и v, выводимых из нетерминалов X и Y: $X \overset{*}{\longrightarrow}_\Gamma u$, $Y \overset{*}{\longrightarrow}_\Gamma v$. Далее будем полагать, что $|u| \le |v|$ (в противном случае левую и правую части уравнения можно поменять местами). Если нельзя построить левосторонний грамматический вывод вида $Y \overset{*}{\longrightarrow}_\Gamma u\gamma$, то процедура проверки прекращает работу и объявляет систему уравнений $\mathcal{E}(\Gamma, M, N)$ неразрешимой. А если $Y \overset{*}{\longrightarrow}_\Gamma u\gamma$ для некоторого предложения $\gamma \in \mathcal{N}^*$, то уравнение $X\alpha = Y\beta$ в системе $\mathcal{E}_t$ заменяется парой уравнений $\alpha = \gamma\beta$ и $Y = X\gamma$. После этого в системе $\mathcal{E}_t$ появляется хотя бы одно проверочное уравнение вида $Y = X\gamma$, левая часть которого состоит только из одной переменной и отлична от правой части.

4) *Приведение переменных.* В получившейся системе $\mathcal{E}_t$ выбирается неприведенное уравнение вида $Y = \beta$, в котором предложение β отлично от переменной Y. Если Y содержится в β, то процедура проверки прекращает работу и объявляет систему уравнений $\mathcal{E}(\Gamma, M, N)$ неразрешимой. Если предложение β не содержит переменной Y, то ко всем остальным уравнениям системы применяется подстановка $\{Y/\beta\}$. После ее применения проверочное уравнение $Y = \beta$ становится приведенным, и, таким образом, число приведенных уравнений системы увеличивается.

Однако применение подстановки $\{Y/\beta\}$ к базовым уравнениям системы имеет побочный эффект: в системе могут возникнуть либо два базовых уравнения с одинаковой левой частью, либо уравнение вида $X\beta = \sum\limits_{a\in\Sigma} a \cdot \alpha_a$, где $|\beta| > 0$. Поэтому далее необходимо применить преобразование, которое восстанавливает канонический вид системы.

5) *Устранение дубликатов.* Если в системе возникают два уравнения вида $X = \sum\limits_{a\in\Sigma} a \cdot \alpha_a$ и $X\beta = \sum\limits_{a\in\Sigma} a\cdot\alpha'_a$, левые части которых начинаются одинаковой переменной, то второе уравнение удаляется из системы, и вместо него в систему добавляются проверочные уравнения $\alpha_a\beta = \alpha'_a$, $a \in \Sigma$. В результате применения этого преобразования канонический вид получившейся системы уравнений будет восстановлен, и процедура проверки разрешимости переходит к следующей итерации $t + 1$.

Утверждение 29. *Для любой простой грамматики Γ и пары ее нетерминалов M, N описанная процедура после применения к системе уравнений $\mathcal{E}(\Gamma, M, N)$ всегда завершает работу и корректно устанавливает разрешимость этой системы.*

Доказательство. Описанная процедура всегда завершает работу, т.к. после каждой итерации t число разрешенных переменных системы $\mathcal{E}_{t+1}$ увеличивается.

Правила проверки разрешимости задают равносильные преобразования систем уравнений. Для правил 1), 4) и 5) это следует из утверждения 27 и конгруэнтности отношения $\sim_\Gamma$. Рассмотрим правило 3) и допустим, что $X \overset{*}{\longrightarrow}_\Gamma u$ и $Y \overset{*}{\longrightarrow}_\Gamma u\gamma$. Тогда $X\alpha \overset{*}{\longrightarrow}_\Gamma u\alpha$ и $Y\beta \overset{*}{\longrightarrow}_\Gamma u\gamma\beta$, и, значит, согласно утверждениям 26 и 27 из равенства $X\alpha = Y\beta$ следуют равенства $\alpha = \gamma\beta$ и $X\gamma = Y$. Таким образом, если $X \overset{*}{\longrightarrow}_\Gamma u$ и $Y \overset{*}{\longrightarrow}_\Gamma u\gamma$, то уравнение $X\alpha = Y\beta$ равносильно системе из двух уравнений

$\alpha = \gamma\beta$ и $Y = X\gamma$. Значит, правило 3) также задает равносильное преобразование. Поэтому на каждой итерации t процедура работает с системой уравнений $\mathcal{E}_t$, которая равносильна исходной системе $\mathcal{E}(\Gamma, M, N)$.

Процедура правильно проверяет разрешимость исходной системы уравнений. Мы показали, что каждая система уравнений $\mathcal{E}_t$ равносильна системе $\mathcal{E}_\Gamma \cup \{M = N\}$. Согласно утверждению 28 единственным решением системы уравнений $\mathcal{E}_\Gamma$ являются значения переменных-нетерминалов $X = L(\Gamma, X)$. Заметим, что правила вывода простых грамматик устроены так, что $\varepsilon \notin L(\Gamma, X)$ для любого нетерминала X. Поэтому значения $X = L(\Gamma, X)$ не удовлетворяют уравнениям вида $X\alpha = 1$ или $X = \beta' X \beta''$. Очевидно также, что если эти значения переменных удовлетворяют равенству $X\alpha = Y\beta$, то для любого слова $w \in L(\Gamma, X)$ язык $L(\Gamma, Y)$ содержит либо префикс, либо расширение слова w. Поэтому правила 1), 3) и 4) правильно устанавливают неразрешимость системы $\mathcal{E}_t$. Так как всякая приведенная система уравнений имеет решение, правило 2) правильно распознает разрешимость системы $\mathcal{E}_t$. □

Для простой грамматики Γ обозначим записью $\nu(\Gamma)$ величину $\max\limits_{N \in \mathcal{N}} \min\limits_{w \in L(\Gamma, N)} |w|$. Из утверждения 29 вытекает

Теорема 7. *Проблема эквивалентности нетерминалов для произвольной простой грамматики* Γ *разрешима за время* $O(n^3 \nu(\Gamma))$*, где* n *— суммарный размер правил грамматики* Γ.

Доказательство. Разрешимость проблемы эквивалентности следует из утверждения 29. Для представления уравнений используются ссылочные структуры данных. Процедура, проверяющая разрешимость системы уравнений, совершает не более n итераций и на каждой из них, имея систему проверочных уравнений размера m, может выполнить правила 1) и 4) за время $O(m)$, правила 2) и 5) — за время $O(n)$, правило 3) — за время $O(n\nu(\Gamma))$, и увеличить при этом размер системы проверочных уравнений на величину $O(n\nu(\Gamma))$. Отсюда следует указанная оценка сложности. □

Заключение

По мнению автора этой статьи, потенциальные возможности предложенного алгебраического подхода к построению быстрых алгоритмов проверки эквивалентности различных видов автоматов не исчерпываются теми результатами, которые были здесь представлены. Можно отметить несколько направлений, по которым целесообразно продолжить исследования, связанные с дальнейшим развитием разработанного метода и расширением его области применения.

Во-первых, вполне естественно попробовать применить алгебраический метод проверки эквивалентности к другим моделям вычислений, поведение которых можно описать при помощи систем алгебраических уравнений. Главной целью здесь можно наметить задачу построения полиномиального по времени алгоритма проверки эквивалентности детерминированных k-ленточных автоматов для $k \geq 3$. Однако свойства префиксных языков, благоприятствующие построению быстрого алгоритма проверки эквивалентности детерминированных двухленточных автоматов и проявляющиеся, например, в утверждении 10, уже не смогут, по-видимому, помочь решению этой более трудной задачи. Необходимо, насколько это возможно, отыскать принципиально новые приемы восстановления канонического вида системы уравнений после применения подстановок для разрешенных переменных (см. правила 4)-6) в разделе 5). Вполне вероятно, что достичь этого удастся за счет исследования проблемы эквивалентности для некоторых сравнительно узких классов многоленточных автоматов, которые можно транслировать в детерминированные конечные автоматы-преобразователи, работающие над специальными полугруппами (см. [34]), в которых возможно, проявляются свойства, полезные для эффективной проверки разрешимости систем уравнений, аналогичные указанным выше свойствам префиксных языков.

Другое направление развития алгебраического метода проверки эквивалентности автоматов — это построение с его помощью алгоритмов минимизации различных типов автоматов, и, в первую очередь, детерминированных автоматов-преобразователей (трансдьюсеров), работающих над специальными полугруппами. Успешный пример построения эффективных алгоритмов минимизации трансдьюсеров на основе алгоритмов проверки эквивалентности представлен в статьях [8, 73]. Одной из интересных задач для дальнейших исследований может быть задача разработки полиномиальных по времени алгоритмов минимизации детерминированных двухленточных автоматов или биавтоматов. Здесь можно также опираться на результаты статьи [10].

И, наконец, заслуживает внимания задача построения эффективных алгоритмов проверки эквивалентности и минимизации в классе конечных автоматов-преобразователей, работающих над полугруппами подстановок. Как показано в статье [9], именно эта автоматная модель вычислений в наибольшей мере соответствует модели последовательных императивных программ, семантика которых задается отношением логико-термальной эквивалентности [74]. Таким образом, на этом пути исследований может быть обнаружена возможность непосредственного использования алгебраического метода проверки эквивалентности автоматов для решения задач оптимизации компьютерных программ.

References

[1] A. L. Rosenberg, “A machine realization of the linear context-free languages”, *Information and Control*, vol. 10, no. 2, pp. 175–188, 1967.

[2] M. Mohri, “Finite-state transducers in language and speech processing”, *Computational linguistics*, vol. 23, no. 2, pp. 269–311, 1997.

[3] A. Roche-Lima and R. K. Thulasiram, “Bioinformatics algorithm based on a parallel implementation of a machine learning approach using transducers”, *Journal of Physics: Conference Series*, vol. 341, no. 1, p. 012 034, 2012.

[4] R. Alur and P. Černy, “Streaming transducers for algorithmic verification of single-pass list-processing programs”, in *Proceedings of 38-th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, 2011, pp. 599–610.

[5] J. Thakkar and R. Kanade A.and Alur, “Transducer-based algorithmic verification of retransmission protocols over noisy channels”, in *Proceedings of IFIP Joint International Conference on Formal Techniques for Distributed Systems*, Springer, 2013, pp. 209–224.

[6] M. Veanes, P. Hooimeijer, B. Livshits, and et al., “Symbolic finite state transducers: Algorithms and applications”, in *Proceedings of the 39-th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, 2012, pp. 137–150.

[7] D. G. Kozlova and V. A. Zakharov, “On the model checking of sequential reactive systems”, in *Proceedings of the 25-th International Workshop on Concurrency, Specification and Programming, Rostock, Germany, September 28-30*, vol. 1698, 2016, p. 233.

[8] M. Mohri, “Minimization algorithms for sequential transducers”, *Theoretical Computer Science*, vol. 234, no. 2, pp. 177–201, 2000.

[9] V. A. Zakharov and S. R. Jaylauova, “On the minimization problem for sequential programs”, *Automatic Control and Computer Sciences*, vol. 51, no. 7, pp. 689–700, 2017.

[10] H. Tamm, M. Nykänen, and E. Ukkonen, “On size reduction techniques for multitape automata”, *Theoretical computer science*, vol. 363, no. 2, pp. 234–246, 2006.

[11] D. C. Luckham, D. M. Park, and M. S. Paterson, “On formalized computer programs”, *Journal of Computer and System Sciences*, vol. 4, no. 3, pp. 220–249, 1970.

[12] R. Loukanova, "Linear context free languages", in *International Colloquium on Theoretical Aspects of Computing*, Springer, 2007, pp. 351–365.

[13] B. Bedregal, "λ-ALN: autômatos lineares não-determinısticos com λ-transições", *Tendencias em Matematica Aplicada e Computacional*, vol. 12, no. 3, pp. 171–182, 2011.

[14] G. Jiraskova and O. Klima, "Deterministic Biautomata and Subclasses of Deterministic Linear Languages", in *International Conference on Language and Automata Theory and Applications*, Springer, 2019, pp. 315–327.

[15] M. Holzer and S. Jakobi, "Nondeterministic biautomata and their descriptional complexity", in *International Workshop on Descriptional Complexity of Formal Systems*, Springer, 2013, pp. 112–123.

[16] O. Klima and L. Polak, "On biautomata*", *RAIRO-Theoretical Informatics and Applications*, vol. 46, no. 4, pp. 573–592, 2012.

[17] M. Holzer and S. Jakobi, "Minimization and characterizations for biautomata", *Fundamenta Informaticae*, vol. 136, no. 1-2, pp. 113–137, 2015.

[18] P. C. Fischer and A. L. Rosenberg, "Multitape one-way nonwriting automata", *Journal of Computer and System Sciences*, vol. 2, no. 1, pp. 88–101, 1968.

[19] T. Griffiths, "The unsolvability of the equivalence problem for Λ-free nondeterministic generalized machines", *Journal of the ACM (JACM)*, vol. 15, no. 3, pp. 409–413, 1968.

[20] O. Ibarra, "The unsolvability of the equivalence problem for ε-free NGSM's with unary input (output) alphabet and applications", *SIAM Journal on Computing*, vol. 7, no. 4, pp. 524–532, 1978.

[21] M. Blattner and T. Head, "Single-valued a-transducers", *Journal of Computer and System Sciences*, vol. 15, no. 3, pp. 310–327, 1977.

[22] M. P. Schutzenberger, "Sur les relations rationnelles", in *Proceedings of the Conference on Automata Theory and Formal Languages*, 1975, pp. 209–213.

[23] M. Blattner and T. Head, "The decidability of equivalence for deterministic finite transducers", *Journal of Computer and System Sciences*, vol. 19, no. 1, pp. 45–49, 1979.

[24] E. M. Gurari and O. Ibarra, "A note on finite-valued and finitely ambiguous transducers", *Mathematical systems theory*, vol. 16, no. 1, pp. 61–66, 1983.

[25] A. Weber, "On the valuedness of finite transducers", *Acta Informatica*, vol. 27, no. 8, pp. 749–780, 1990.

[26] K. Culik and J. Karhumäki, "The equivalence of finite valued transducers (on HDT0L languages) is decidable", *Theoretical Computer Science*, vol. 47, pp. 71–84, 1986.

[27] A. Weber, "A decomposition theorem for finite-valued transducers and an application to the equivalence problem", in *Proceedings of the 13-th International Symposium on Mathematical Foundations of Computer Science*, Springer, 1988, pp. 552–562.

[28] A. Weber, "Decomposing finite-valued transducers and deciding their equivalence", *SIAM Journal on Computing*, vol. 22, no. 2, pp. 175–202, 1993.

[29] M.-P. Béal, O. Carton, C. Prieur, and J. Sakarovitch, "Squaring transducers: an efficient procedure for deciding functionality and sequentiality", *Theoretical Computer Science*, vol. 292, no. 1, pp. 45–63, 2003.

[30] J. Sakarovitch and R. de Souza, "On the decomposition of k-valued rational relations", in *Proceedings of the 25-th International Symposium on Mathematical Foundations of Computer Science*, 2008, pp. 588–600.

[31] J. Sakarovitch and R. de Souza, "On the decidability of bounded valuedness for transducers", in *Proceedings of the 25-th International Symposium on Mathematical Foundations of Computer Science*, 2008, pp. 588–600.

[32] J. Sakarovitch and R. de Souza, "Lexicographic decomposition of *k*-valued transducers", *Theory of Computing Systems*, vol. 47, no. 3, pp. 758–785, 2010.

[33] R. de Souza, "On the decidability of the equivalence for k-valued transducers", in *Proceedings of the 12-th International Conference on Developments in Language Theory*, Springer, 2008, pp. 252–263.

[34] V. A. Zakharov, "Equivalence checking problem for finite state transducers over semigroups", in *Proceedings of the 6-th International Conference on Algebraic Informatics*, Springer, 2015, pp. 208–221.

[35] A. Weber, "On the lengths of values in a finite transducer", *Acta Informatica*, vol. 29, no. 6-7, pp. 663–687, 1992.

[36] R. de Souza, "On the Decidability of the Equivalence for a Certain Class of Transducers", in *Proceedings of the 13-th International Conference on Developments in Language Theory*, Springer, 2009, pp. 478–489.

[37] M. Bird, "The equivalence problem for deterministic two-tape automata", *Journal of Computer and System Sciences*, vol. 7, no. 2, pp. 218–236, 1973.

[38] L. G. Valiant, "The equivalence problem for deterministic finite-turn pushdown automata", *Information and Control*, vol. 25, no. 2, pp. 123–133, 1974.

[39] C. Beeri, "An improvement on Valiant's decision procedure for equivalence of deterministic finite turn pushdown machines", *Theoretical Computer Science*, vol. 3, no. 3, pp. 305–320, 1976.

[40] E. P. Friedman and S. A. Greibach, "A polynomial time algorithm for deciding the equivalence problem for 2-tape deterministic finite state acceptors", *SIAM Journal on Computing*, vol. 11, no. 1, pp. 166–183, 1982.

[41] T. Harju and J. Karhumäki, "The equivalence problem of multitape finite automata", *Theoretical Computer Science*, vol. 78, no. 2, pp. 347–355, 1991.

[42] J. Worrell, "Revisiting the equivalence problem for finite multitape automata", in *Proceedings of the 40-th International Colloquium on Automata, Languages, and Programming*, Springer, 2013, pp. 422–433.

[43] B. Bedregal, "Nondeterministic Linear Automata and a Class of Deterministic Linear Languages", *Preliminary Proceedings LSFA*, pp. 183–196, 2015.

[44] Y. Bar-Hillel, M. Perles, and E. Shamir, "On formal properties of simple phrase structure grammars", *Sprachtypologie und Universalienforschung*, vol. 14, pp. 143–172, 1961.

[45] A. J. Korenjak and J. E. Hopcroft, "Simple deterministic languages", in *7th Annual Symposium on Switching and Automata Theory (swat 1966)*, IEEE, 1966, pp. 36–46.

[46] E. P. Friedman, "The inclusion problem for simple languages", *Theoretical Computer Science*, vol. 1, no. 4, pp. 297–316, 1976.

[47] D. Caucal, "A fast algorithm to decide on simple grammars equivalence", in *International Symposium on Optimal Algorithms*, Springer, 1989, pp. 66–85.

[48] C. Bastien, J. Czyzowicz, W. Fraczak, and W. Rytter, "Prime normal form and equivalence of simple grammars", in *International Conference on Implementation and Application of Automata*, Springer, 2005, pp. 78–89.

[49] Y. Hirshfeld, M. Jerrum, and F. Moller, "A polynomial algorithm for deciding bisimilarity of normed context-free processes", *Theoretical Computer Science*, vol. 158, no. 1-2, pp. 143–159, 1996.

[50] L. G. Valiant and M. S. Paterson, "Deterministic one-counter automata", *Journal of Computer and System Sciences*, vol. 10, no. 3, pp. 340–350, 1975.

[51] S. Böhm and S. Göller, "Language equivalence of deterministic real-time one-counter automata is NL-complete", in *International Symposium on Mathematical Foundations of Computer Science*, Springer, 2011, pp. 194–205.

[52] G. Sénizergues, "The equivalence problem for t-turn dpda is co-NP", in *Proceedings of the 30-th International Colloquium on Automata, Languages, and Programming*, Springer, 2003, pp. 478–489.

[53] M. Linna, "Two decidability results for deterministic pushdown automata", *Journal of Computer and System Sciences*, vol. 18, no. 1, pp. 92–107, 1979.

[54] V. Y. Meytus, "The equivalence problem for real-time strict deterministic pushdown automata", *Cybernetics*, vol. 25, no. 2, pp. 581–594, 1989.

[55] M. Oyamaguchi, "The equivalence problem for real-time DPDAs", *Journal of the ACM*, vol. 34, no. 3, pp. 731–760, 1987.

[56] V. Y. Romanovskii, "Equivalence problem for real-time deterministic pushdown automata", *Cybernetics*, vol. 22, no. 2, pp. 162–175, 1985.

[57] D. J. Rosenkrantz and R. E. Stearns, "Properties of deterministic top-down grammars", *Information and Control*, vol. 17, no. 3, pp. 226–256, 1970.

[58] E. Tomita, "An extended direct branching algorithm for checking equivalence of deterministic pushdown automata", *Theoretical Computer Science*, vol. 32, no. 1-2, pp. 87–120, 1984.

[59] E. Ukkonen, "The equivalence problem for some non-real-time deterministic pushdown automata", *Journal of the ACM*, vol. 29, no. 4, pp. 1166–1181, 1982.

[60] G. Sénizergues, "The equivalence problem for deterministic pushdown automata is decidable", in *Proceedings of the 24-th International Colloquium on Automata, Languages, and Programming*, Springer, 1997, pp. 671–681.

[61] C. Stirling, "Deciding DPDA equivalence is primitive recursive", in *Proceedings of the 29-th International Colloquium on Automata, Languages, and Programming*, Springer, 2002, pp. 821–832.

[62] R. Madhavan, M. Mayer, S. Gulwani, and V. Kuncak, "Automating grammar comparison", in *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, 2015, pp. 183–200.

[63] V. A. Zakharov, "Program equivalence checking by two-tape automata", *Cybernetics and Systems Analysis*, vol. 46, no. 4, pp. 554–562, 2010.

[64] T. Olshansky and A. Pnueli, "A direct algorithm for checking equivalence of LL(k) grammars", *Theoretical Computer Science*, vol. 4, no. 3, pp. 321–349, 1977.

[65] J. Karhumäki, "Equations over finite sets of words and equivalence problems in automata theory", *Theoretical computer science*, vol. 108, no. 1, pp. 103–118, 1993.

[66] A. Okhotin, "Decision problems for language equations", *Journal of Computer and System Sciences*, vol. 76, no. 3-4, pp. 251–266, 2010.

[67] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation (3rd ed.)* Addison-Wesley, 2006.

[68] S. Eilenberg, *Automata, languages, and machines*. Academic press, 1974.

[69] Y. Han, K. Salomaa, and D. Wood, "State Complexity of Prefix-Free Regular Languages.", in *DCFS*, 2006, pp. 165–176.

[70] A. Martelli and U. Montanari, "An efficient unification algorithm", *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 2, pp. 258–282, 1982.

[71] D. Arden, "Delayed-logic and finite-state machines", in *Proceedings of 2-nd Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1961)*, IEEE, 1961, pp. 133–151.

[72] J. E. Hopcroft, *A linear algorithm for testing equivalence of finite automata.* Defense Technical Information Center, 1971, vol. 114.

[73] V. A. Zakharov and G. G. Temerbekova, "On the minimization of finite state transducers over semigroups", *Automatic Control and Computer Sciences*, vol. 51, no. 7, pp. 523–530, 2017.

[74] V. E. Itkin, "Logiko-termalnaya ekvivalentnost skhem programm", *Kibernetika i sistemnyj analiz*, no. 1, pp. 5–27, 1972.