Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к магистерской диссертации

«Построение конечных автоматов с применением итеративных программных средств решения задачи о выполнимости»

Автор: Закирзянов Илья Тимурович _____
Направление подготовки (специальность): 01.04.02 Прикладная математика и информатика
Квалификация: Магистр
Руководитель: Ульянцев В.И., канд. техн. наук _____

К защите допустить
Зав. кафедрой Васильев В.Н., докт. техн. наук, проф. _____

«___» _____ 20___ г.

Санкт-Петербург, 2017 г.

Студент Закирзянов И.Т. **Группа** М4238 **Кафедра** компьютерных технологий **Факультет** информационных технологий и программирования

Направленность (профиль), специализация Технологии проектирования и разработки программного обеспечения

Квалификационная работа выполнена с оценкой _____

Дата защиты «___» _____ 20___ г.

Секретарь ГЭК *Павлова О.Н.* Принято: «___» _____ 20___ г.

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

УТВЕРЖДАЮ
Зав. каф. компьютерных технологий
докт. техн. наук, проф.
_____ Васильев В.Н.
«___» _____ 20___ г.

ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

Студент Закирзянов И.Т. **Группа** М4238 **Кафедра** компьютерных технологий **Факультет** информационных технологий и программирования **Руководитель** Ульянцев В.И., канд. техн. наук, доцент кафедры компьютерных технологий Университета ИТМО

1 Наименование темы: Построение конечных автоматов с применением итеративных программных средств решения задачи о выполнимости

Направление подготовки (специальность): 01.04.02 Прикладная математика и информатика

Направленность (профиль): Технологии проектирования и разработки программного обеспечения

Квалификация: Магистр

2 Срок сдачи студентом законченной работы: «___» _____ 20___ г.

3 Техническое задание и исходные данные к работе.

В рамках исследования необходимо разработать и реализовать методы построения детерминированных конечных автоматов при помощи сведения к задаче о выполнимости и применения итеративных программных средств решения данной задачи. Показать практическую эффективность или теоретическую значимость разработанных методов в сравнении с аналогами.

4 Содержание магистерской диссертации (перечень подлежащих разработке вопросов)

а) Пояснительная записка должна содержать обзор существующих методов решения задачи построения минимального детерминированного конечного автомата;
б) должно присутствовать подробное описание разработанных методов для решения поставленной задачи
в) должны быть подробно описаны проведенные эксперименты, представлены результаты и проведен сравнительный анализ с уже существующими методами.

5 Перечень графического материала (с указанием обязательного материала)

Не предусмотрено

6 Исходные материалы и пособия

а) *Heule M., Verwer S.* Exact DFA Identification Using SAT Solvers // ICGI. — 2010. — P. 66-79;

б) *Grinchtein O., Leucker M., Piterman N.* Inferring Network Invariants Automatically // IJCAR. — 2006. — P. 483-497.

в) *Ulyantsev V., Zakirzyanov I., Shalyto A.* BFS-Based Symmetry Breaking Predicates for DFA Identification // LATA. — 2015. — P. 611-622.

7 Календарный план

№№ пп.	Наименование этапов магистерской диссертации	Срок выполнения этапов работы	Отметка о выполнении, подпись руков.
1	Постановка и формализация задачи	10.2015	
2	Проведение аналитического обзора по существующим исследованиям	04.2016	
3	Разработка метода нахождения всех неизоморфных автоматов минимального размера	05.2016	
4	Разработка более компактных кодирований различных частей задачи	10.2016	
5	Разработка метода, использующего только значимую часть входных слов	01.2017	
6	Проведение экспериментального исследования	04.2017	
7	Написание пояснительной записки	05.2017	

8 Дата выдачи задания: «01» сентября 2015 г.

Руководитель _____

Задание принял к исполнению _____ «01» сентября 2015 г.

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

АННОТАЦИЯ
МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

Студент: Закирзянов Илья Тимурович

Наименование темы работы: Построение конечных автоматов с применением итеративных программных средств решения задачи о выполнимости

Наименование организации, где выполнена работа: Университет ИТМО

ХАРАКТЕРИСТИКА МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

1 Цель исследования: Разработка методов построения детерминированных конечных автоматов при помощи сведения к задаче о выполнимости и применения итеративных программных средств решения данной задачи.

2 Задачи, решаемые в работе:

а) разработать алгоритм нахождения всех неизоморфных ДКА минимального размера;

б) разработать асимптотически более компактное кодирование как основной задачи, так и предикатов нарушения симметрии;

в) разработать алгоритм, позволяющий использовать только значимую часть префиксного автомата.

3 Число источников, использованных при составлении обзора: 26

4 Полное число источников, использованных в работе: 27

5 В том числе источников по годам

Отечественных			Иностранных		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
0	0	0	7	4	16

6 Использование информационных ресурсов Internet: нет

7 Использование современных пакетов компьютерных программ и технологий: В рамках исследования были использованы следующие программные средства: язык программирования Java и библиотека args4j для него, язык программирования Python, язык описания графов DOT и пакет GraphViz.

8 Краткая характеристика полученных результатов: Все поставленные задачи были выполнены (ну последняя пока не до конца), методы разработаны, реализованы и экспериментально протестированы. Алгоритм нахождения всех неизоморфных ДКА минимального размера является единственным известным для решения данной задачи. Более компактное кодирование основного сведения позволило представить те примеры, которые раньше не могли быть представлены; более компактное кодирование предикатов нарушения симметрии и дополнительные

подходы к сокращению пространства поиска позволило получить выигрыш по производительности.

9 Гранты, полученные при выполнении работы: При выполнении работы был получен грант:

Премия Правительства Санкт-Петербурга победителям конкурса грантов для студентов вузов, расположенных на территории Санкт-Петербурга, аспирантов вузов, отраслевых и академических институтов.

10 Наличие публикаций и выступлений на конференциях по теме работы: Исследование прошло апробацию на конференциях:

а) Всероссийский конгресс молодых ученых Университета ИТМО 2016;
б) XLVI Научная и учебно-методическая конференция университета ИТМО;
в) Всероссийский конгресс молодых ученых Университета ИТМО 2017.

Выпусник: Закирзянов И.Т. _____

Руководитель: Ульянцев В.И. _____

«___» _____ 20___ г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Задача построения *детерминированного конечного автомата* (ДКА) минимального размера по конечному множеству примеров поведения (пар входные-выходные данные) изучается уже на протяжении нескольких десятилетий. Первая известная серьезная работа в данном направлении была написана в начале 70-х [1]. Значимый вклад в развитие данного направления был сделан рядом работ в 90-х. Данные работы были связаны с такими областями, как *раскраска графов* [2], техники *программирования в ограничениях* (constraint programming) [3, 4] и *алгоритмы слияния состояний* [5, 6]. Подходы, основанные на сведении к задачам *выполнимости формул в теориях* (satisfiability modulo theories, SMT) и *выполнимости булевых формул* (Boolean satisfiability, SAT), были предложены в течение последнего десятилетия и показали многообещающие результаты [7–11]. Тем не менее, размер булевых формул, кодирующих данную задачу с помощью существующих подходов, не применим в случае, когда ДКА имеют достаточно большой размер. Использование подходов, основанных на сведении к задаче SMT не решает проблему, так как алгоритмы решения задачи SMT сводят ее к решению задачи SAT и все равно используют булево кодирование.

В данной работе изучаются подходы к решению задачи построения ДКА, основанные на сведении к задаче SAT, и предлагается более компактное сведение. Также в данной работе исследуются существующие техники нарушения симметрии [8, 12–14] и предлагаются улучшения к ним, позволяющие асимптотически уменьшить размер их кодирования в булевы формулы, а также помогающие дополнительно сократить пространство поиска. Помимо этого в данной работе предлагается способ решения задачи нахождения всех ДКА минимального размера по примерам поведения, которая не могла быть решена эффективно ранее. Также представляются новые предикаты нарушения симметрии на основе алгоритма обхода графа в глубину, однако они представляют только теоретический интерес.

Исследования, представленные в главах 4 и 5, были проведены совместно с командой профессора Жуана Маркеша-Сильвы из Лиссабонского университета — собственно, профессор Маркеш-Сильва, а также Алексей Игнатьев и Антонио Моргадо.

ГЛАВА 1. АНАЛИТИЧЕСКИЙ ОБЗОР

В данной главе приводятся постановка задачи, вводится нотация и приводятся результаты аналитического обзора.

1.1. Основные определения, нотация и постановка задачи построения детерминированного конечного автомата минимального размера

В данном разделе приводятся основные определения, необходимые далее в работе, и используемая нотация. Также приводится формальная постановка задачи построения ДКА минимального размера.

1.1.1. Основные определения и нотация

В работе предполагается, что автоматы определены над некоторым набором символов Σ — *алфавитом*. Размер алфавита (количество символов в нем) будем обозначать как $L = |\Sigma|$. Для большинства примеров в данной работе используется $\Sigma = B$, где $B = \{0,1\}$. Для ранних работ в данной области чаще всего рассматривался случай, когда $\Sigma = B$ [4, 5]. В последних же работах иногда рассматриваются и алфавиты большего размера [15].

Детерминированный конечный автомат (ДКА) — это набор из шести элементов $\mathcal{D} = (D,\Sigma,\delta,d_1,D^+,D^-)$, где D — это конечное множество состояний, Σ — входной алфавит, $\delta : D \times \Sigma \to D$ — функция переходов, d_1 — стартовое состояние, D^+ — множество принимающих состояний и $D^- = D \setminus D^+$ — множество отвергающих состояний. Для входной строки $\pi \in \Sigma^*$ определим $\hat{\delta}(d_1,\pi)$ индуктивно следующим образом [16]: (i) $\hat{\delta}(d_1,\varepsilon) = d_1$; (ii) Если $\pi = \pi' c$, тогда $\hat{\delta}(d_1,\pi) = \delta(\hat{\delta}(d_1,\pi'),c)$.

Предполагается, что используются стандартные условия построения ДКА минимального размера по множеству примеров его поведения [17, 18], то есть по обучающему словарю, каждый элемент которой представлен входной строкой, которая либо принимается, либо отвергается некоторым заранее неизвестным ДКА $\mathcal{U} = (U,\mu,u_1,U^+,U^-)$. Такая форма обучения часто называется как *пассивное обучение*, как противоположность *активному обучению* [11, 19], которое разрешает обучающему алгоритму (который пытается построить целевой ДКА) формулировать некоторые запросы к некоторому учителю (которому известен целевой ДКА).

Обучающий словарь — это множество пар $\mathbb{T} = \{(\pi_1,o_1),\ldots,(\pi_R,o_R)\}$, где каждая пара $(\pi_r,o_r) \in \Sigma^* \cup \varepsilon \times \{0,1\}$ означает, что на выходе был получен результат o_r в результате обработки автоматом входной строки π_r. Если $o_r = 1$,

тогда π_r называется *принимаемым* примером. Если $o_r = 0$, тогда π_r называется *отвергаемым* примером.

По данному обучающему словарю можно построить *расширенный префиксный автомат* (augmented prefix tree acceptor, APTA) [8, 20, 21], определяемый как $\mathcal{T} = (T,\Sigma,\tau,t_1,T^+,T^-)$. Данный автомат называется расширенным, потому что некоторые состояния не являются ни принимающими, ни отвергающими, то есть $T^+ \cup T^- \neq T$. Данный автомат называется префиксным, потому что в нем пути, соответствующие двум строкам достигают одинакового состояния v тогда и только тогда, когда данные строки имеют одинаковый префикс. Иначе говоря, для двух входных строк $\pi_1 = \pi_a\pi_{b_1}$ и $\pi_2 = \pi_a\pi_{b_2}$ общий префикс π_a будет соответствовать уникальной последовательности состояний в APTA. Для префиксного автомата $\mathcal{T}$ обозначим $N = |T|$. Будем ссылаться на состояния автомата $\mathcal{T}$ по их индексу — t_i для таких i, что $i = 1,\ldots,N$. $\lceil(i)$ используется для обозначения расстояния между корнем APTA t_1 и состоянием t_i (иными словами, для обозначения глубины состояния t_i относительно корня).

1.1.2. Задача построения детерминированного конечного автомата минимального размера

Задача *построения (нахождения, синтеза) ДКА минимального размера* (MinDFA) заключается в нахождении такого ДКА $\mathcal{S} = (S,\sigma,s_1,S^+,S^-)$ с минимальным возможным количеством состояний, что для всех пар из обучающего словаря (π_r,o_r) верно, что $\hat{\sigma}(s_1,\pi_r) \in S^+$ тогда и только тогда, когда $o_r = 1$, и $\hat{\sigma}(s_1,\pi_r) \in S^-$ тогда и только тогда, когда $o_r = 0$. Для искомого ДКА $\mathcal{S}$ определим булевы переменные $e_{v,p,q}$ так, что они принимают значение 1 тогда и только тогда, когда $\sigma(p,v) = q$. Для ДКА $\mathcal{S}$ введем обозначение $M = |S|$. Будем ссылаться на состояния автомата $\mathcal{S}$ также по их индексу — s_p для таких p, что $p = 1,\ldots,M$. В примере 1 представлены искомый автомат, обучающий словарь и префиксный автомат для него.

Пример 1. Рассмотрим автомат $\mathcal{U} = (U,\mu,u_1,U^+,U^-)$, определенный следующим образом:

$$
\begin{aligned}
&U = \{u_1,u_2,u_3\},\\
&U^+ = \{u_3\},\\
&U^- = \{u_1,u_2\},\\
&\mu \triangleq \{\mu(u_1,0) = u_1, \mu(u_1,1) = u_2,\\
&\qquad \mu(u_2,0) = u_1, \mu(u_2,1) = u_3,\\
&\qquad \mu(u_3,0) = u_1, \mu(u_3,1) = u_3\}.
\end{aligned}
\tag{1}
$$

ДКА $\mathcal{U}$ показан на рисунке 1. Более того, допустим, что обучающий словарь представлен в таблице 1. По данным примерам поведения, можно построить префиксный автомат $\mathcal{T}$, представленный на рисунке 2. Наконец, можно заметить, что не все обучающие примеры необходимы для построения автомата $\mathcal{U}$. Например, примеры 03$\sim$04 и 07$\sim$12 не нужны для построения $\mathcal{U}$ по $\mathcal{T}$.

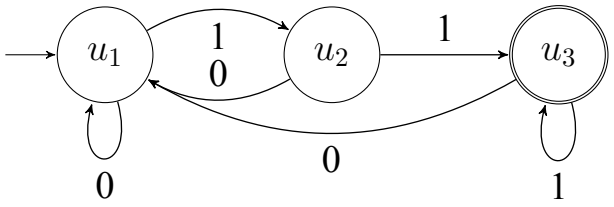

Рисунок 1 – Пример искомого (заранее неизвестного) автомата

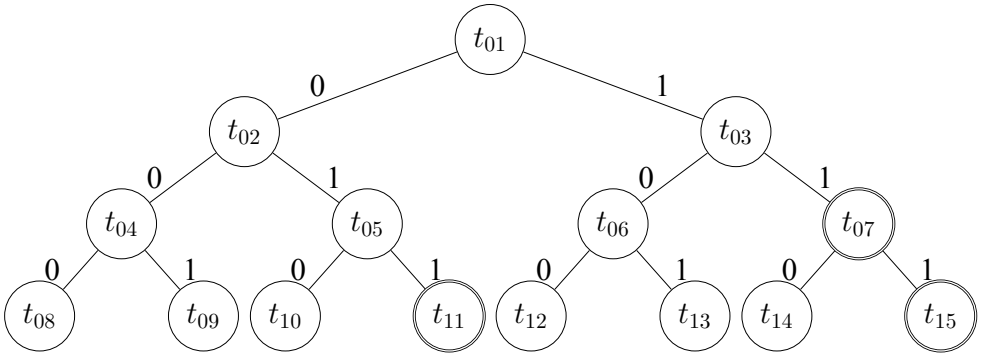

Рисунок 2 – Префиксный автомат для примеров из обучающего словаря, приведенного в таблице 1

Таблица 1 – Пример обучающего словаря

номер примера	входная строка	результат
01	0	0
02	1	0
03	00	0
04	01	0
05	10	0
06	11	1
07	000	0
08	001	0
09	010	0
10	011	1
11	100	0
12	101	0
13	110	0
14	111	1

1.1.3. Дополнительная нотация

В данной работе обозначение $[R]$, для некоторого положительного целого R, используется, чтобы обозначить множество $\{1, \ldots, R\}$. Более того, целые положительные числа будут использоваться и для того, чтобы ссылаться на символы (например, символы Σ будут связаны с множеством $[L]$), и для того, чтобы ссылаться на состояния (аналогично, множества состояний $\mathcal{T}$ и $\mathcal{S}$ будут связаны с множествами $[N]$ и $[M]$ соответственно). Связь символов некоторого алфавита с целыми числами упрощает определение лексикографического порядка на символах, которое понадобится далее.

В данной работе используются стандартные определения для задач SAT и SMT (смотри, например, [22, 23]).

1.2. Существующие подходы к решению поставленной задачи

С начала 70-х было проведено множество исследований и опубликовано множество работ по решению данной задачи. Одним из основных направлений исследований являлись так называемые алгоритмы *слияния состояний* [5, 6, 18, 24]. Такие подходы, обычно, находят лишь локальный оптимум, но не глобальный. Нет никаких гарантий, что найденный автомат, соответствующий обучающему словарю, имеет минимальное возможное количество состояний. Поэтому в данной работе основное внимание уделено точным подходам, основанным на

сведениях к различным задачам ограничений. В течение последних лет были предложены различные такие подходы: например, одна из первых и базовых работ в данной области Бьерманна и Фельдмана [25], подход, основанный на сведении к задаче раскраски графа Костэ и Николаса [2], и метод ветвей и границ Оливейры и Маркеша-Сильвы [4]. Ближе к настоящему времени были разработаны подходы основанные на сведении к задаче SAT [7, 8, 12] и к задаче SMT [10, 11]. В данной разделе приводится аналитический обзор данных точных методов.

1.2.1. Базовый алгоритм

На листинге 1 приведен обобщенный наиболее широко используемый подход для решения задачи MinDFA.

Листинг 1 – Обобщенный алгоритм уточнения нижней оценки

```
function MinimumDFA(𝒯)
    λ ← FindLowerBound(𝒯)
    while true do
        𝒮 ←FindConsistentDFA(𝒯,λ)
        if 𝒮 ≠ ∅ then
            return 𝒮
        end if
        λ ← λ + 1
    end while
end function
```

Изначально высчитывается нижняя оценка на размер искомого автомата. Обычной практикой является нахождение некоторой большой клики среди состояний префиксного автомата, которые не могут быть объединены в одно состояние в ДКА [4, 7, 8, 10–13]. Затем, начиная с найденной нижней оценки и перебирая все возможные значения количества состояний ДКА, некоторым алгоритмом решается существует ли автомат $\mathcal{S}$, соответствующий обучающим примерам, по которым построен префиксный автомат $\mathcal{T}$.

Алгоритм, приведенный на листинге 1, является линейным и часто называется как *LSUS* (linear-search, UNSAT until SAT). Могут применяться и другие алгоритм, например, бинарный поиск. В таком случае необходимо некоторым образом найти верхнюю оценку на количество состояний искомого автомата (например, с помощью уже упомянутых алгоритмов слияния состояний).

1.2.2. Сведение к задаче SAT

В данном пункте приводится краткое описание сведения задачи нахождения минимального ДКА к задаче SAT, описанное в [8]. Покажем, как закодировать задачу по заданному префиксному автомату $\mathcal{T}$ и по заданной границе M на число состояний искомого ДКА $\mathcal{S}$, Вместо формулировки с использованием терминов задачи раскраски графа, будем говорить просто о задании отображения (соответствия) N состояний АРТА $\mathcal{T}$ на M состояний искомого автомата $\mathcal{S}$. Данное кодирование устроено следующим образом. Булевы переменные:

а) $m_{i,p}$ которая истинна тогда и только тогда, когда состояние t_i в $\mathcal{T}$ отображается на состояние s_p в $\mathcal{S}$.

б) $e_{v,p,q}$ которая истинна тогда и только тогда, когда существует переход из состояния s_p в состояние s_q по символу l_v в $\mathcal{S}$.

в) a_p которая истинна тогда и только тогда, когда состояние s_p является принимающим $\mathcal{S}$.

Ограничения:

а) Существует ровно один исходящий переход из состояния s_p по символу v:

$$\sum_{q=1}^{M} e_{v,p,q} = 1, \qquad p = 1,\ldots,M \wedge v = 1,\ldots,L. \tag{2}$$

Нужно заметить, что достаточно использовать ограничение, что сумма не превышает единицы, (незаданные переходы будут выбраны рандомно). Однако ограничение, что сумма равна единице помогает программному средству найти решение быстрее.

б) Каждое состояние в $\mathcal{T}$ отображается ровно на одно состояние в $\mathcal{S}$:

$$\sum_{p=1}^{M} m_{i,p} = 1, \qquad i = 1,\ldots,N. \tag{3}$$

Нужно заметить, что достаточно использовать ограничение, что сумма не меньше единицы. Однако ограничение, что сумма равна единице сново помогает программному средству найти решение быстрее.

в) Принимающее состояние из $\mathcal{T}$ отображается на принимающее состояние в $\mathcal{S}$:

$$m_{i,p} \rightarrow a_p, \qquad i = 1,\ldots,N \wedge t_i \in T^+ \wedge p \in \{1,\ldots,M\}. \tag{4}$$

г) Отвергающее состояние из $\mathcal{T}$ отображается на отвергающее состояние в $\mathcal{S}$:

$$m_{i,p} \to \neg a_p, \qquad i = 1,\ldots,N \wedge t_i \in T^- \wedge p \in \{1,\ldots,M\}. \tag{5}$$

д) Если существует переход из состояния t_i в состояние t_k по символу v, и состояние t_i отображается в s_p, тогда если выполняется одно из условий ниже, то должно выполняться и второе:

1) существует переход из s_p в s_q по символу v;
2) состояние t_k отображается на s_q.

Данное ограничение может быть выражено следующим образом:

$$m_{i,p} \wedge e_{v,p,q} \to m_{k,q}, \qquad \tau(t_i,l_v) = t_k \wedge v \in [L] \wedge i,k \in [N] \wedge p,q \in [M], \tag{6a}$$

$$m_{i,p} \wedge m_{k,q} \to e_{v,p,q}, \qquad \tau(t_i,l_v) = t_k \wedge v \in [L] \wedge i,k \in [N] \wedge p,q \in [M]. \tag{6b}$$

Как было сказано в [8], выбор, например, только ограничения (6b) должно быть достаточно, однако рекомендуется использовать оба по тем же причинам, что и ранее.

Размер сведения, приведенного выше, для некоторого искомого ДКА размера M может быть расчитан следующим образом:

а) Ограничение (2) может быть закодировано с помощью $\mathcal{O}(M \times M)$ дизъюнктов.
б) Ограничение (3) — $\mathcal{O}(N \times M)$ дизъюнктов.
в) Ограничение (4) — $\mathcal{O}(N \times M)$ дизъюнктов.
г) Ограничение (5) — $\mathcal{O}(N \times M)$ дизъюнктов.
д) Ограничение (6) — $\mathcal{O}(N \times M^2)$ дизъюнктов. В некотором АРТА $\mathcal{T}$ существует $\mathcal{O}(N)$ переходов и для каждого из них надо рассмотреть $\mathcal{O}(M^2)$ возможных случаев.

Как можно заключить из вышеприведенного, рассматриваемое кодирование выражается через $\mathcal{O}(N \times M^2)$ дизъюнктов.

1.2.3. Сведение к задаче SMT

Элегантное кодирование, основанное на *теории неинтерпретируемых функций с равенством* (logic of equality with uninterpreted functions, EUF) было

предложено Нейдером и остальными в работах [10, 11], основанных на предыдущей работе авторов [26].

Кодирование на основе EUF (будем также называть его SMT-кодированием) может быть описано следующим образом. Пусть m_i, $i = 1,\ldots,N$, обозначает целочисленную переменную с доменом $[M]$. Переменная m_i имеет значение p тогда и только тогда, когда состояние t_i из префиксного автомата *соответствует* состоянию s_p искомого ДКА $\mathcal{S}$.

Кодирование использует две неитерпретируемые функции — A и E. Функция $A : [M] \to \{0,1\}$ является неитерпретируемой функцией, которая сообщает является ли состояние s_p искомого ДКА $\mathcal{S}$ принимающим или нет. Функция $E : [L] \times [M] \to [M]$ является неитерпретируемой функцией, такой, что $E(v,p) = q$ представляет переход $\sigma(s_p,l_v) = s_q$, где $l_v \in \Sigma$ и $s_p,s_q \in S$.

SMT-кодирование состоит из следующих ограничений:

— Если t_i является принимающим состояние в $\mathcal{T}$, тогда вызов функции $A(m_i)$ должен возвращать значение 1.

— Если t_i является отвергающим состоянием в $\mathcal{T}$, тогда вызов функции $A(m_i)$ должен возвращать значение 0.

— Если существует переход из t_i в t_k по символу l_v в $\mathcal{T}$, то есть $\tau(t_i,l_v) = t_k$, тогда $E(v, m_i) = m_k$.

Дополнительно необходимо ограничить значения переменных m_i и функций E и A. Исходя из вышеперечисленного, SMT-кодирование может быть формализовано следующим образом. Во-первых, ограничения на искомый автомат

$$\begin{aligned} &A(m_i) = 1\,, && i = 1,\ldots,N \wedge t_i \in T^+, \\ &A(m_i) = 0\,, && i = 1,\ldots,N \wedge t_i \in T^-, \\ &(E(v,m_i) = m_k)\,, && i,k = 1,\ldots,N \wedge v = 1,\ldots,L \wedge \tau(t_i,l_v) = t_k. \end{aligned} \tag{7}$$

Во-вторых, ограничения на значения переменных, функций и их параметров:

$$\begin{aligned} &1 \leqslant m_i \leqslant M\,, && i = 1,\ldots,N, \\ &1 \leqslant E(v,j) \leqslant M\,, && j = 1,\ldots,M \wedge v = 1,\ldots,L, \\ &0 \leqslant A(j) \leqslant 1\,, && j = 1,\ldots,M. \end{aligned} \tag{8}$$

Пример 2. Построим SMT-кодирование для примера 1, когда $M = 2$, и ограничим словарь только следующими примерами: 01, 02, 05, 06, 13, 14, соответствующими префиксному автомату $\mathcal{T}'$ с состояниями $t_{01}, t_{02}, t_{03}, t_{07}, t_{14}, t_{15}$.

Пусть $\Sigma = \{0,1\} = \{l_1,l_2\}$, где 0 сопоставляется с l_1 и 1 — с l_2. Переменные m_i связаны с состояниями t_i и имеют домен значений $\{1,2\}$. Как было сказано ранее, необходимо определить также функции E с областью допустимых значений $\{1,2\}$ и A с областью допустимых значений $\{0,1\}$. Наконец необходимо задать следующие ограничения:

$$\begin{aligned} &E(1,m_1) = m_2 \wedge E(2,m_1) = m_3 \wedge E(2,m_3) = m_7 \wedge \\ &E(1,m_7) = m_{14} \wedge E(1,m_7) = m_{14} \wedge E(2,m_7) = m_{15} \wedge \\ &\neg A(m_1) \wedge \neg A(m_2) \wedge \neg A(m_3) \wedge \neg A(m_{14}) \wedge A(m_7) \wedge A(m_{15}) \end{aligned} \tag{9}$$

Помимо указанных выше ограничений, необходимо задать ограничения на значения переменных, функций и их параметров, что делается очевидным образом.

SMT-кодирование использует $\mathcal{O}(N)$ целочисленных переменных и $\mathcal{O}(N)$ ограничений. Для префиксного автомата с N состояниями, существует не более чем N ограничений на неитерпретируемую функцию A и не более чем N ограничений на неитерпретируемую функцию E. Действительно, каждое состояние префиксного автомата может быть принимающим, отвергающим или неопределенным, а для каждого состояния кроме корневого существует ровно одно входящее ребро.

1.3. Предикаты нарушения симметрии

В данном разделе приводятся подходы к нарушению симметрии для сокращения пространства поиска поставленной задачи.

1.3.1. Общая идея нарушения симметрии

Два графа, которые отличаются только нумерацией вершин, но совпадают по структуре и имеют одинаковое (с поправкой на нумерацию вершин) множество принимающих состояний, называются *изоморфными* друг другу. Пример пары изоморфных автоматов приведен на рисунке Для каждого графа существует $n!$ (что, очевидно, равняется числу различных перестановок номеров вершин) изоморфных ему графов, образующих класс эквивалентности по изоморфизму.

При использовании любого точно подхода для решения задачи MinDFA, программное средство пытается подобрать автомат некоторого размера, удовлетворяющий заданным ограничениям. Однако, можно заметить, что после рассмотрения некоторого автомата, не удовлетворяющего данным ограничениям, не имеет смысла рассматривать изоморфные ему автоматы, так как они также

заведомо не будут удовлетворять данным ограничениям. Таким образом, алгоритм, приведенный на листинге 1, на каждой итерации поиска для текущего размера автомата λ из полуинтервала $\left[lowerBound, answer\right)$ для каждого автомата будет рассматривать $\lambda!$ изоморфных ему автоматов. Использование некоторой техники *нарушения симметрии* (symmetry-breaking) позволяет значительно сократить количество перебираемых изоморфных автоматов (в лучшем случае до одного представителя). В следующих подразделах будут описаны известные используемые техники.

1.3.2. Предикаты нарушения симметрии на основе больших клик

Некая *большая* или *максимальная клика* (maximal or maximum clique) графа совместимости G_I может быть использована для нарушения симметрии в задаче MinDFA. Необходимо зафиксировать отображение вершин клики в часть целевого автомата. Иными словами, каждому состоянию префиксного дерева из клики присваивается конкретный номер целевого автомата. Например, если клика состоит из вершин $\{t_{i_1}, \ldots, t_{i_K}\}$ префиксного дерева $\mathcal{T}$, тогда для каждой из вершин t_{i_j} можно зафиксировать номер j, $j \in \{1 \ldots, K\}$, в целевом автомате $\mathcal{S}$. Необходимо заметить, что при таком подходе для каждого класса эквивалентности автоматов по изоморфизму рассматривается $(\lambda - K)!$ представителей. Такой результат достаточно хорош при небольших значениях λ и при достаточно близком к λ размере клики K. Но в общем же случае точному алгоритму решения задачи MinDFA придется рассмотреть факториальное число изоморфных автоматов. Данный подход активно применялся в прошлых исследованиях, основанных как на сведении к задаче SAT, так и на сведении к задаче SMT [7, 8, 10–12].

1.3.3. Предикаты нарушения симметрии на основе алгоритма обхода графа в ширину

В данном подразделе представлены результаты последней работы по нарушениям симметрии для задачи MinDFA, основанные на фиксировании нумерации состояний автомата [13].

Данный подход может быть формализован следующим образом. Пусть $\mathcal{S} = (S, \sigma, s_1, S^+, S^-)$ — искомый ДКА. Необходимо зафиксировать нумерацию состояний автомата $\mathcal{S}$ в порядке обхода данного графа алгоритмом *обхода в ширину* (breadth-first search, BFS). При обходе графа (или, что в данном случае рав-

носильно автомата) алгоритмом BFS, строится BFS-дерево. Порядок построения данного дерева задает необходимую нумерацию. Таким образом, предикаты нарушения симметрии в данном случае зависит только от состояний и переходов искомого ДКА $\mathcal{S}$.

Для того, чтобы любому автомату соответствовал единственный возможный вариант обхода алгоритмом BFS, необходимо зафиксировать некоторый порядок на символах из Σ. Подойдет любой порядок, но для простоты будем считать, что они пронумерованы в лексикографическом порядке и, как и ранее, пронумерованы числами от 1 до L, но теперь нумерация соответствует конкретному порядку. На рисунке 3 представлено BFS-дерево для автомата, изображенного на рисунке 1.

Осталось заметить, что при фиксированном порядке на символах алфавита, для каждого автомата существует ровно один возможный обход алгоритмом BFS (иными словами, существует единственное BFS-дерево). Тогда, если закодировать такое требование к нумерации искомого ДКА, получатся «идеальные» предикаты нарушения симметрии. Они являются «идеальными», потому что для каждого класса эквивалентности автоматов по изоморфизму существует только один представитель, пронумерованный указанным образом. Таким образом, точный алгоритм решения задачи MinDFA на каждой итерации будет рассматривать только один автомат для каждого класса эквивалентности либо вместо $\lambda!$ в случае отсутствия каких-либо подходов к нарушению симметрии, либо вместо $(\lambda - K)!$ в случае применения подхода на основе больших клик.

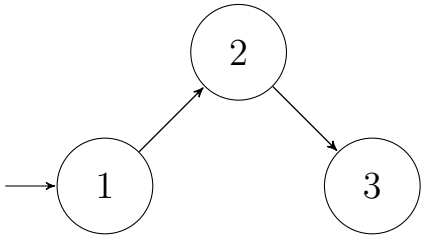

Рисунок 3 – BFS-дерево для ДКА, изображенного на рисунке 1

1.3.3.1. Предикаты нарушения симметрии для сведения к задаче SAT

Будем для краткости вместо «состояние с номером i» говорить просто «состояние i». Для задания предикатов нарушения симметрии на основе алгоритма BFS необходимы следующие переменные:

а) $p_{q,r}$, где $1 \leqslant r < q \leqslant M$. $p_{q,r} = 1$ тогда и только тогда, когда состояние r является родителем состояния q в BFS-дереве.

б) $t_{p,q}$, где $1 \leqslant p < q \leqslant M$. $t_{p,q} = 1$ тогда и только тогда, когда существует переход из состояния p в состояние q.

в) $m_{v,p,q}$, где $v \in \Sigma$ и $1 \leqslant p < q \leqslant M$. $m_{v,p,q} = 1$ тогда и только тогда, когда существует переход из состояния p в состояние q по символу l_v и не существует перехода между данными состояними по лексикографически меньшему символу.

Используя данные переменные, можно задать следующие ограничения:

а) За исключением стартового состояния (имеющего номер 1), каждое состояние должно иметь родителя с меньшим номером:

$$\bigwedge_{2 \leqslant q \leqslant M} (p_{q,1} \vee p_{q,2} \vee \cdots \vee p_{q,q-1}) . \tag{10}$$

б) Состояние q должно быть рассмотрено (в порядке обхода, согласно алгоритму BFS) раньше, чем состояние $q+1$. Тогда родитель r состояния $q+1$ не может быть меньше, чем родитель s состояния q:

$$\bigwedge_{1 \leqslant r < s < q < M} (p_{q,s} \to \neg p_{q+1,r}) . \tag{11}$$

в) Переменные $p_{q,r}$ задаются при помощи переменных $t_{q,r}$ следующим образом:

$$\bigwedge_{1 \leqslant r < q \leqslant M} (t_{r,q} \leftrightarrow e_{1,r,q} \vee \cdots \vee e_{L,r,q}) . \tag{12}$$

г) Переменные $t_{q,r}$ в свою очередь задаются при помощи переменных $e_{v,q,r}$:

$$\bigwedge_{1 \leqslant r < q \leqslant M} (p_{q,r} \leftrightarrow t_{r,q} \wedge \neg t_{r-1,q} \wedge \cdots \wedge \neg t_{1,q}) . \tag{13}$$

д) Переменные $m_{v,p,q}$ связываются с переходами в ДКА следующим образом:

$$\bigwedge_{1 \leqslant r < q \leqslant M} \bigwedge_{1 \leqslant v \leqslant L} (m_{v,r,q} \leftrightarrow e_{v,r,q} \wedge \neg e_{v-1,r,q} \wedge \cdots \wedge \neg e_{1,r,q}) . \tag{14}$$

е) Последовательные состояния q и $q+1$, имеющие одного родителя r должны быть пронумерованы согласно символам на переходах, ведущих из ро-

дителя в них:

$$\bigwedge_{1\leqslant r<q<M}\ \bigwedge_{1\leqslant u<v\leqslant L}\left(p_{q,r}\wedge p_{q+1,r}\wedge m_{v,r,q}\rightarrow\neg m_{u,r,q+1}\right). \tag{15}$$

Проанализировав, указанные выше ограничения, можно заключить, что количество размер формулы, кодирующей данные предикаты нарушения симметрии — $\mathcal{O}(M^3+M^2L^2+M^2L)$ дизъюнктов. Можно заметить, что обычно $L \ll M$, поэтому основной вклад в размер формулы вносит слагаемое M^3, получающееся из формул (11) и (13).

Также необходимо заметить, что при $|\Sigma| = 2$, авторы [13] предлагают заменить формулы (14) и (15) на новые формулы,

$$\bigwedge_{1\leqslant r<q<M}\left(p_{q,r}\wedge p_{q+1,r}\rightarrow e_{1,r,q}\right), \tag{16}$$

и

$$\bigwedge_{1\leqslant r<q<M}\left(p_{q,r}\wedge p_{q+1,r}\rightarrow e_{2,r,q+1}\right). \tag{17}$$

Достаточно использовать только одну из формул (16) и (17), но использование обеих может дополнительно помочь программному средству.

1.3.3.2. Предикаты нарушения симметрии для сведения к задаче SMT

Предикаты нарушения симметрии для булева кодирования могут быть обобщены и для SMT-кодирования с использования равенств и неинтерпретируемых функций. Достаточно определить следующие неинтерпретируемые функции:

а) $P : [M] \rightarrow [M]$, где $P(q)$, для $1 \leqslant q \leqslant M$ и $P(q) \leqslant q - 1$, представляет родителя состояния q в BFS-дереве.

б) $T : [M]\times[M] \rightarrow \{0,1\}$, где $T(r,q)$, для $1 \leqslant r,q \leqslant M$, обозначает существование перехода из состояния r в состояние q. Дополнительно потребуем выполнение неравенства $r < q$.

в) $W : [M]\times[M] \rightarrow [L]$, где $W(r,q) = v$ обозначает, что наименьший символ (в лексикографическом порядке над алфавитом) на переходах из состояния r в q — это v.

Соответственно, ограничения, выраженные на языке SMT, принимают следующий вид:

а) $P(q) < q$.

б) $P(q) \leqslant P(q+1)$.

в) $P(q) = r \leftrightarrow T(r,q) \wedge \bigwedge_{s=1}^{q-1} \neg T(s,q)$.

г) $T(r,q) = 1$ тогда и только тогда, когда $\vee_{v \in [M]} E(v,r) = q$.

д) $W(r,q) = v \leftrightarrow E(v,r) = q \wedge \bigwedge_{s=1}^{v-1} E(s,r) \neq q$.

Проанализировав вышеуказанные ограничения, можно придти к выводу, что размер SMT-кодирования данных предикатов нарушения симметрии — $\mathcal{O}(M^2)$ ограничений, ввиду квадратичного числа аргументов для неинтерпретируемых функций T и W.

Выводы по главе 1

В данной главе были даны основные определения и нотация, используемая в данной работе. Также была сформулирована задача построения минимального ДКА по заданному словарю.

ГЛАВА 2. ПРЕДИКАТЫ НАРУШЕНИЯ СИММЕТРИИ НА ОСНОВЕ АЛГОРИТМА DFS

В этой главе предлагается способ, как зафиксировать нумерацию состояний автомата, чтобы избежать рассмотрения изоморфных ДКА во время решения задачи SAT. Данная идея не оригинальна, но основана на на подходе к нарушению симметрии на основе алгоритма BFS. Основной идеей предлагаемого подхода к нарушению симметрии является фиксирование номеров состояний в порядке обхода в глубину (depth-first search, DFS). Таким образом, снова для каждого класса эквивалентности по изоморфизму будет рассмотрен только один представитель.

В ходе работы алгоритма DFS необходимо найти все смежные непосещенные состояния для каждого еще непосещенного состояния ДКА. Изначально алгоритм рассматривает стартовое состояние. Затем алгоритм рассматривает детей этого состояния и рекурсивно повторяет вычисления на них. Будем использовать дополнительную структуру — массив с переходами, которые соединяют его элементы. Данные элементы являются номерами состояний пронумерованных в порядке обхода ДКА алгоритмом DFS. Каждый переход, соединяющий элементы массива, является копией перехода в ДКА, использованного во время работы алгоритма обхода в глубину. Так как переходы помечены символами алфавита Σ, будем рассматривать детей в лексикографическом порядке символов l на переходах $i \xrightarrow{l} j$. Будем называть ДКА *DFS-пронумерованным*, если его дополнительный массив заполнен последовательными числами в возрастающем порядке, начиная с 1. Пример DFS-пронумерованного ДКА с шестью состояниями показан на рисунке 4 (жирным выделены ребра, которые использовались во время обхода в глубину — так называемое DFS-дерево).

Все переменные, которые использовались для нумерации в порядке BFS используются и для DFS-нумерации, но некоторые ограничения должны быть изменены. Определение переменных $p_{j,i}$ ($p_{j,i}$ истинна тогда и только тогда, когда состояние i является родителем (**p**arent) состояния j в DFS-дереве) изменяется на следующее: при DFS-нумерации состояние j было добавлено в массив, когда рассматривалось состояние с максимальны номером i среди состояний, которые имеют переход в состояние j:

$$\bigwedge_{1 \leqslant i < j \leqslant C} (p_{j,i} \Leftrightarrow t_{i,j} \wedge \neg t_{i+1,j} \wedge \ldots \wedge \neg t_{j-1,j}), \tag{18}$$

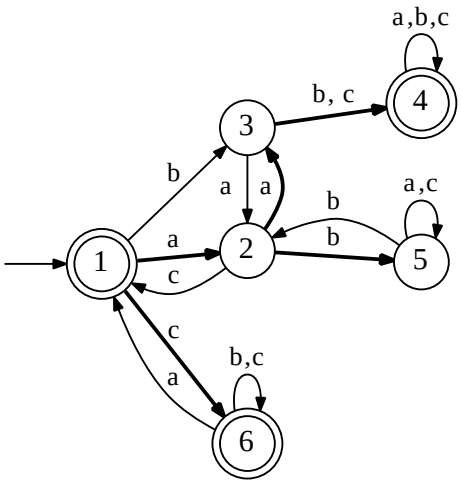

Рисунок 4 – DFS-пронумерованный ДКА, где выделенные ребра образуют DFS-дерево

где переменная $t_{i,j} \equiv 1$ тогда и только тогда, когда существует переход (**t**ransition) между состояниями i и j (данные переменные в свою очередь определяются с помощью переменных $e_{v,r,q}$).

Более того, в DFS-нумерации родители состояний также должны быть упорядочены. Если состояние i является родителем состояния j, а состояние k — состояние между i и j ($i < k < j$) тогда не должно существовать перехода из состояния k в состояние q, которое больше чем j:

$$\bigwedge_{1 \leqslant i<k<j<q<C} (p_{j,i} \Rightarrow \neg t_{k,q}). \tag{19}$$

Далее, чтобы обеспечить DFS-пронумерованность, необходимо упорядочить детей состояний в лексикографическом порядке символов на переходах. Рассмотрим два случая: алфавит Σ состоит из двух символов $\{a,b\}$ и алфавит состоит из более чем двух символов $\{l_1, \ldots, l_L\}$. В случае двух символов только два состояния j и k могут иметь одного родителя i (где без потери общности можно считать, что $j < k$). В этом случае, убедимся, что переход, который начинается в состоянии i, помеченный символом a идет в состояние j, а не k:

$$\bigwedge_{1 \leqslant i<j<k<C} (p_{j,i} \wedge p_{k,i} \Rightarrow y_{a,i,j}). \tag{20}$$

Во втором случае необходимо снова использовать переменные $m_{l,i,j}$, которые определяются следующим образом: $m_{l,i,j}$ истинна тогда и только тогда,

когда существует переход $i \xrightarrow{l} j$ и не существует перехода между данными состояниями по лексикографически меньшему символу. Теперь осталось упорядочить состояния j и k, имеющие общего родителя i, в лексикографическом порядке минимальных символов на переходам между состоянием i и ними:

$$\bigwedge_{1 \leqslant i < j < k \leqslant C} \bigwedge_{1 \leqslant m < n \leqslant L} (p_{j,i} \wedge p_{k,i} \wedge m_{l_n,i,j} \Rightarrow \neg m_{l_m,i,k}). \tag{21}$$

Выше были описаны предикаты нарушения симметрии, которые выражаются перечисленными условиями. Предикаты (для случая с более чем двухсимвольным алфавитом) сводятся к $\mathcal{O}(C^3 + C^2L^2)$ КНФ дизъюнктов.

Выводы по главе 2

В данном разделе были предложены предикаты нарушения симметрии, основанные на другом алгоритме обхода графа, нежели алгоритм обхода в ширину. Конкретно, были разработаны предикаты на основе алгоритма обхода в глубину (DFS).

ГЛАВА 3. МЕТОДЫ ПОСТРОЕНИЯ НЕСКОЛЬКИХ ДКА

В настоящей главе описывается задача нахождения нескольких (или, что эквивалентно, всех возможных) неизоморфных ДКА с минимальным количеством состояний, соответствующих заданным словарям. Предлагается модификация метода, основанного на сведении к задаче о выполнимости, решающая данную задачу. Рассматривается два варианта использования программных средств решения SAT: перезапуск неинкрементального средства после нахождения каждого нового автомата и использование инкрементального средства. Инкрементальный SAT-решатель после нахождения решения задачи о выполнимости сохраняет свое состояние и готов принять новые ограничения и продолжить свою работу. Основные техники и алгоритмы, связанные с инкрементальными программными средствами были предложены в работе [27]. Также в разделе описывается эвристический метод поиска с возвратом как основной и единственный, с которым можно сравнить основанные на задаче SAT методы.

3.1. Методы, основанные на сведении к SAT

Основная идея методов, основанных на сведении к SAT, заключается в том, чтобы запрещать удовлетворяющие подстановки, которые уже были найдены. Очевидно, что если не использовать предлагаемые в [13] предикаты нарушения симметрии, то данный подход будет находить множество изоморфных автоматов (а именно, $M!$ для каждого класса эквивалентности по изоморфизму, где M - размер автомата). Так как подход, основанный на клике, описанный в [8], фиксирует только K цветов (где K - размер клики графа совместимости), то предлагаемый подход с данными предикатами будет находить $(M-K)!$ изоморфных автоматов, что при больших M также плохо. Предикаты, основанные на алгоритме обхода графа в ширину, позволяют запрещать все изоморфные ДКА из одного класса эквивалентности путем запрещения соответственно пронумерованного представителя. Данный подход легко реализуется добавлением одного дополнительного ограничения в булеву формулу. Так как известно, что переменные e_v,i,j полностью задают искомый автомат (подробнее можно узнать в статьях [13, 27]), то достаточно запретить значения этих переменных из найденной подстановки:

$$\neg e_1 \vee \neg e_2 \vee \ldots \vee \neg e_{M|\Sigma|},$$

где e_s это какое-то истинное $e_{v,p,q}$ из найденной удовлетворяющей подстановки для всех $1 < s < M|\Sigma|$.

Как было указано выше, существует два варианта использования программных средств решения SAT. Во-первых, можно перезапускать неинкрементальную версию программы с новой булевой формулой в качестве входных данных с дополнительным ограничением после нахождения каждого нового автомата. Во-вторых, можно использовать инкрементальную версию программы: после каждого найденного автомата, нужно добавить дополнительное ограничение на вход решателю и продолжить его выполнение.

Необходимо дополнительно рассмотреть случай, когда некоторые переходы найденного автомата не определяются перехода префиксного автомата. Это означает, что существуют некоторые *свободные* переходы, которые не используются в процессе обработки слов из входных словарей. Такие переходы могут заканчиваться в произвольном состоянии автомата, так как это никак не влияет на соответствие автомата обучающим данным. Далее предлагается способ, как принудительно заставить все такие переходы заканчиваться в том состоянии, из которого они выходят. Иными словами, предлагается способ фиксирования всех свободных переходов в виде петель. Для решения поставленной задачи необходимо добавить дополнительные переменные *использования* (англ. **u**sed): $u_{v,p}$ истинна тогда и только тогда, когда существует переход в префиксном автомате, исходящий из состояния, покрашенного в p-ый цвет, и помеченный символом v:

$$\bigwedge_{v\in\Sigma}\bigwedge_{1\leqslant p\leqslant M} u_{v,p} \leftrightarrow m_{1,p} \vee \ldots \vee m_{|V_v|,p}, \tag{22}$$

где V_v — это множество всех состояний префиксного автомата, которые имеют исходящее ребро, помеченное символом v. Для того, чтобы заставить все свободные переходы иметь вид петель, необходимо добавить следующие ограничения:

$$\bigwedge_{v\in\Sigma}\bigwedge_{1\leqslant p\leqslant M} \neg u_{v,p} \rightarrow e_{v,p,p}. \tag{23}$$

Эти дополнительные ограничения задаются с помощью $\mathcal{O}(C|V|)$ дизъюнктов. На рисунке 5 представлен пример префиксного автомата для $S_+ = \{ab, b, ba, bbb\}$ и $S_- = \{abbb\}$, а на рисунке 6 представлен соответствующий ему автомат со свободным переходом. Если добавить предлагаемые ограничения, то этот переход будет зафиксирован как петля, что изображено пунктиром на рисунке 6.

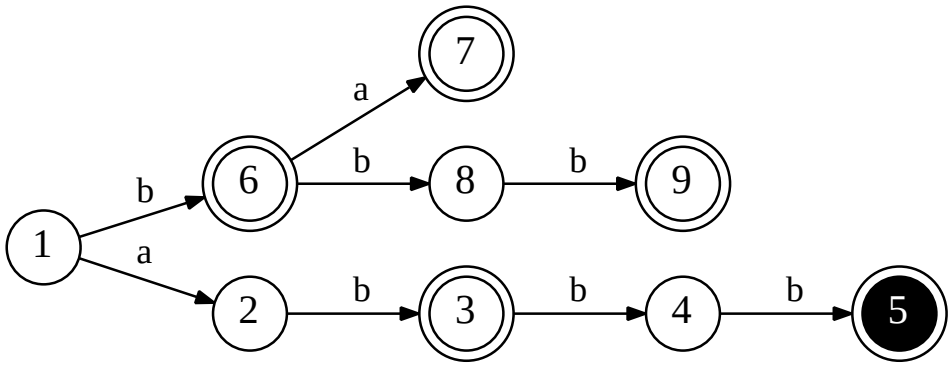

Рисунок 5 – Пример префиксного автомата для $S_+ = \{ab, b, ba, bbb\}$ и $S_- = \{abbb\}$

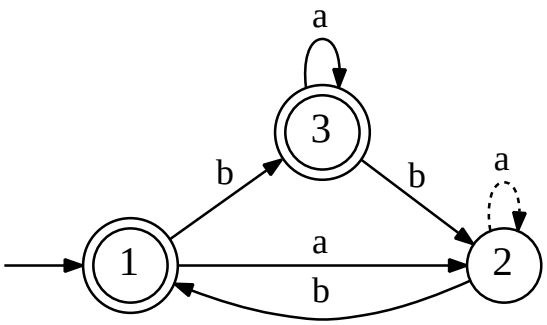

Рисунок 6 – ДКА построенный по префиксному автомату, изображенному на рисунке 5 со свободным переходом, исходящем из состояния 2 и помеченный символом a

3.2. Метод поиска с возвратом

Подход, основанный на поиске с возвратом не использует какие-либо сторонние программные средства вроде SAT-решателей. Данный алгоритм работает следующим образом. Изначально есть пустой ДКА размера n (то есть имеется только M вершин без переходов). Также будем хранить *фронт* - множество ребер префиксного автомата, которые еще не представлены в итоговом ДКА. Изначально фронт содержит все ребра, исходящие из корня префиксного автомата. Рекурсивная функция `Backtracking` поддерживает фронт в правильном состоянии. Если фронт не пуст, тогда данная функция пытается достроить ДКА, добавив один из переходов фронта. Каждый такой достроенный ДКА проверяется на соответствие с префиксным автоматом, и если противоречия не найдено, то фронт обновляется. Если фронт пуст, то ДКА проверяется на полноту (ДКА считается полным, если существуют исходящие переходы из всех состояний по всем символам алфавита). В случае неполноты найденного автомата, недостающие ребра добавляются как петли с помощью функции `MakeComplete`. Псев-

докод алгоритма представлен в листинге 2. Функция `FindNewFrontier` возвращает новый фронт для достроенного ДКА или `null`, если автомат несовместим с префиксным деревом.

Листинг 2 – Поиск с возвратом

```
procedure Backtracking(APTA, DFA, frontier)
    DFAset ← new Set()
    edge ← ребро из frontier
    for all destination ∈ 1..|S| do
        source ← состояние DFA из которого исходит edge
        DFA′ ← DFA ∪ transition(source, destination, edge.label)
        frontier′ ← FindNewFrontier(APTA, DFA′, frontier)
        if frontier′ ≠ null then
            if frontier′ = ∅ then
                DFAset.add(MakeComplete(DFA′))
            else
                DFAset.add(Backtracking(APTA, DFA′, frontier′))
            end if
        end if
    end for
return DFAset
end procedure
```

Выводы по главе 3

В данной главе был представлен новый метод решения задачи построения всех неизоморфных ДКА минимального размера по заданным обучающим словарям. Ранее данная задача не могла быть решена эффективно, так как не существовало подходов к нарушению симметрии, позволяющих не рассматривать изоморфные графы. Также был предложен эвристический метод поиска с возвратом как единственный, с которым можно проводить сравнение разработанного метода по нахождению всех ДКА.

ГЛАВА 4. ПРЕДИКАТЫ НАРУШЕНИЯ СИММЕТРИИ

В данной главе предлагаются усовершенствования предикатов нарушения симметрии на основе алгоритма обхода в ширину, описанных в разделе 1.3.3.1.

4.1. Компактное кодирование предикатов нарушения симметрии, основанных на алгоритме BFS

Булево кодирование предикатов нарушения симметрии, представленное в разделе 1.3.3.1 и представляемое в виде $\mathcal{O}(M^3 + M^2L^2 + M^2L)$ дизъюнктов неприменимо на практике для задач поиска больших ДКА [5, 15]. В данном разделе показано как модифицировать данные предикаты нарушения симметрии, чтобы размер кодирования составил лишь $\mathcal{O}(M^2L)$ дизъюнктов. Заметим, что приводятся модификации только для предикатов нарушения симметрии, основанных на алгоритме BFS. Схожие идеи могут быть применены и для предикатов на основе алгоритма DFS.

В новом кодировании предлагается изменить способ задания ограничений, задаваемых формулами (10), (11) и (13), а также формулами (14) и (15). Продублируем данные ограничения еще раз для упрощения навигации по работе. Ниже представлены дубликаты формул (10), (11) и (13):

$$\bigwedge_{2 \leqslant q \leqslant M} (p_{q,1} \vee p_{q,2} \vee \cdots \vee p_{q,q-1}), \tag{24}$$

$$\bigwedge_{1 \leqslant r < s < q < M} (p_{q,s} \to \neg p_{q+1,r}), \tag{25}$$

$$\bigwedge_{1 \leqslant r < q \leqslant M} (p_{q,r} \leftrightarrow t_{r,q} \wedge \neg t_{r-1,q} \wedge \cdots \wedge \neg t_{1,q}). \tag{26}$$

Напомним смысл данных переменных:

а) $p_{q,r}$, где $1 \leqslant r < q \leqslant M$. $p_{q,r} = 1$ тогда и только тогда, когда состояние r является родителем состояния q в BFS-дереве.

б) $t_{p,q}$, где $1 \leqslant p < q \leqslant M$. $t_{p,q} = 1$ тогда и только тогда, когда существует переход из состояния p в состояние q.

в) $m_{v,p,q}$, где $v \in \Sigma$ и $1 \leqslant p < q \leqslant M$. $m_{v,p,q} = 1$ тогда и только тогда, когда существует переход из состояния p в состояние q по символу l_v и не существует перехода между данными состояними по лексикографически меньшему символу.

Также далее будем считать, что фраза «ограничение (n)» эквивалентна фразе «ограничение, задаваемое формулой (n)».

Итак, для начала потребуем, чтобы вместо ограничения (24) выполнялось следующее ограничение:

$$\sum_{r=1}^{q-1} p_{q,r} = 1, \quad 1 < q \leqslant M. \tag{27}$$

Данное ограничение является более строгой версией предыдущего. Ограничение (24) задает свойство, что $\sum_{r=1}^{q-1} p_{q,r} \geqslant 1$, а новое ограничение помимо этого требует, чтобы выполнялось также $\sum_{r=1}^{q-1} p_{q,r} \leqslant 1$. Более того, использование этих двух ограничений — это единственный способ задать само ограничение (27). Таким образом, на самом деле, мы не заменяем ограничение (24), а дополняем его. Дополнительное ограничение может быть выраженно с помощью $\mathcal{O}(M^2)$ дизъюнктов. Смысл же данного изменения заключается в том, что вместо требования «у каждого состояния кроме стартового есть родитель в BFS-дереве, имеющий меньший номер» мы хотим, чтобы выполнялось новое требование «у каждого состояния кроме стартового есть *ровно один* родитель в BFS-дереве, имеющий меньший номер». Данное усиление необходимо для возможности задать более компактно следующие ограничения.

Далее рассмотрим кодирование пары ограничений (25) и (26). Будем использовать формулировки на языке SMT из пункта 1.3.3.2. Напомним введенные там функции:

а) $P : [M] \to [M]$, где $P(q)$, для $1 \leqslant q \leqslant M$ и $P(q) \leqslant q - 1$, представляет родителя состояния q в BFS-дереве.

б) $T : [M] \times [M] \to \{0,1\}$, где $T(r,q)$, для $1 \leqslant r,q \leqslant M$, обозначает существование перехода из состояния r в состояние q. Дополнительно потребуем выполнение неравенства $r < q$.

в) $W : [M] \times [M] \to [L]$, где $W(r,q) = v$ обозначает, что наименьший символ (в лексикографическом порядке над алфавитом) на переходах из состояния r в q — это v.

Теперь если провести аналогии между SAT-кодированием и SMT-кодированием, то становится ясно, что ограничение (25) кодирует следующее свойство $P(q) \leqslant P(q+1)$, а ограничение (13) соответственно — $P(q) = r \leftrightarrow T(r,q) \wedge \bigwedge_{s=1}^{q-1} \neg T(s,q)$.

Для начала разберемся, как более компактно закодировать свойство $P(q) \leqslant P(q+1)$. При фиксированном q мы можем рассматривать значения набора переменных $p_{q,r}$ при $1 \leqslant r \leqslant q-1$ как бинарную строку длиной $q-1$ бит. Тогда, соответственно, значение набора переменных $p_{q+1,r}$ при $1 \leqslant r \leqslant q$ будем рассматривать как бинарную строку длины q. Для простоты сравнения, добавим фиктивный 0 в конец строки, представляющей значения $p_{q,r}$ ($p_{q,q} = 0$), тогда строки для наборов значений $p_{q,r}$ и $p_{q+1,r}$ будут одинаковой длины.

Из (27) следует, что данные строки будут состоять из нулей и ровно одной единицы. Теперь для задания свойства $P(q) \leqslant P(q+1)$ необходимо сформулировать на языке SAT ограничение, что в строке для значений $p_{q,r}$ единственная единица находится не правее, чем единственная единица в строке для значений $p_{q+1,r}$. Введем новые булевы переменные $ng_{q,r}$ такие, что $ng_{q,r} = 1$ тогда и только тогда, когда единица в последних $q-r+1$ битах строки, связанной со значениями $p_{q,r}$, находится в позиции не большей (**n**o **g**reater) чем позиция единицы в последних $q-r+1$ битах строки, связанной со значениями $p_{q+1,r}$. Если единицы среди данных битов нет, то считаем, что позиция единицы равна -1. Иными словами, значение переменной $ng_{q,r}$ истинно, если в суффиксе строки для $p_{q,r}$ единица находится левее, чем в суффиксе строки для $p_{q+1,r}$. Тогда понятно, что неравенство $P(q) \leqslant P(q+1)$ выполняется тогда и только тогда, когда $ng_{q,1} = 1$. Заметим также, что раз мы добавляли фиктивный 0 ($p_{q,q} = 0$), то, очевидно, выполняется, что $ng_{q,q} = 1$.

Далее, можно находить значения переменных $ng_{q,r}$ по индукции: если r-ые биты в обеих строках одинаковы ($eq_{q,r} = 1$, где $eq_{q,r} = 1 \Leftrightarrow p_{q,r} \leftrightarrow p_{q+1,r}$), тогда значение переменной $ng_{q,r}$ равно значению для меньшего на единицу суффикса — $ng_{q,r+1}$; если же r-ые биты не одинаковы, то если единица стоит в первой строке ($p_{q,r} \wedge \neg p_{q+1,r}$), то требуемое свойство выполняется, иначе - не выполняется. Формально данное определение может быть записано следующим образом:

$$\begin{aligned} &ng_{q,q} \leftrightarrow 1, \\ &ng_{q,r} \leftrightarrow ng_{q,r+1} \wedge eq_{q,r} \vee p_{q,r} \wedge \neg p_{q+1,r}, \qquad 1 \leqslant r < q. \end{aligned} \tag{28}$$

Дополнительно потребуем выполнение $ng_{q,1} \leftrightarrow 1$, что, как уже было сказано, равносильно требуемому свойству $P(q) \leqslant P(q+1)$. Нетрудно заметить, что данные ограничения выражаются с помощью $\mathcal{O}(M^2)$ дизъюнктов.

Похожая идея может быть применена для компактного кодирования ограничения (26). Заметим, что $p_{q,r} = 1$ тогда и только тогда, когда $t_{r,q} = 1$ и ни для какого $1 \leqslant s < r$ переменная $t_{s,q}$ не истина. Введем новые булевы переменные $nt_{r,q}$, кодирующие данное свойство, то есть которые истинны тогда и только тогда, когда ни для какого $1 \leqslant s < r$ не верно, что $t_{s,q} = 1$ (**n**o such **t**). Очевидно по определению, что $nt_{0,q} = 1$ для любого q. Значения переменных $nt_{r,q}$ задаются индуктивно следующим образом: $nt_{r,q}$ истинно тогда и только тогда, когда истинно предыдущее значение $nt_{r-1,q}$ и ложно $t_{r,q}$. Формально данные ограничения задаются следующим образом:

$$\begin{aligned} &nt_{0,q} \leftrightarrow 1, \\ &nt_{r,q} \leftrightarrow nt_{r-1,q} \wedge \neg t_{r,q}, \qquad 1 \leqslant r < q. \end{aligned} \tag{29}$$

Используя новые переменные $nt_{r,q}$ можно заменить ограничение (26) на следующее эквивалентное ему:

$$\bigwedge_{1 \leqslant r < q \leqslant M} \left(p_{q,r} \leftrightarrow t_{r,q} \wedge nt_{r-1,q}\right). \tag{30}$$

Ограничения (29) и (30) также выражаются через $\mathcal{O}(M^2)$ дизъюнктов.

Таким образом, мы заменили два ограничения (11) и (13), которые кодировались с помощью $\mathcal{O}(M^3)$ дизъюнктов на более компактные. Итоговое количество дизъюнктов, требуемых для кодирования предикатов нарушения симметрии, на данный момент составляет $\mathcal{O}(M^2L^2 + M^2L)$. Покажем теперь, как можно оптимизировать ограничения (14) и (15), которые и дают слагаемое $\mathcal{O}(M^2L^2)$. Как и ранее, продублируем данные ограничения для простоты навигации по работе:

$$\bigwedge_{1 \leqslant r < q \leqslant M} \bigwedge_{1 \leqslant v \leqslant L} \left(m_{v,r,q} \leftrightarrow e_{v,r,q} \wedge \neg e_{v-1,r,q} \wedge \cdots \wedge \neg e_{1,r,q}\right), \tag{31}$$

$$\bigwedge_{1 \leqslant r < q < M} \bigwedge_{1 \leqslant u < v \leqslant L} \left(p_{q,r} \wedge p_{q+1,r} \wedge m_{v,r,q} \rightarrow \neg m_{u,r,q+1}\right). \tag{32}$$

Применим аналогичную описанной ранее стратегию для модификации ограничения (31). Добавим новые булевы переменные $ne_{v,r,q}$ такие, что $ne_{v,r,q} = 1$ тогда и только тогда, когда когда переменные $e_{u,r,q}$ ложны (**n**o such **e**) для любых $u \leqslant v$. Иными словами, $ne_{v,r,q} = 1$ в том случае, если не существует перехода из состояния r в состояние q по символу v и по лексикографически

меньшим v символам. Значения переменных ne можно задать рекурсивно аналогично заданию значений переменных nt: очевидно, что $ne_{0,r,q}$ всегда истинно; $ne_{v,r,q}$ же истинно, если истинно $ne_{v-2,r,q}$ и ложно $e_{v-1,r,q}$. Формально это можно записать следующим образом:

$$\begin{aligned} &ne_{0,r,q} \leftrightarrow 1, \\ &ne_{v,r,q} \leftrightarrow \neg e_{v,r,q} \wedge ne_{v-1,r,q}, \qquad 1 \leqslant v < L. \end{aligned} \tag{33}$$

Тогда, используя дополнительные переменные $ne_{v,r,q}$, можно заменить ограничение (31) на следующее ограничение:

$$\bigwedge_{1 \leqslant r < q \leqslant M} \bigwedge_{1 \leqslant v \leqslant L} \left(m_{v,r,q} \leftrightarrow e_{v,r,q} \wedge ne_{v-1,r,q}\right). \tag{34}$$

Осталось заметить, что данное ограничение вкупе с формулой (33) выражаются с помощью $\mathcal{O}(M^2 L)$ дизъюнктов.

Аналогичная идея может быть применена и для того, чтобы модифицировать ограничение (32). Добавим новые булевы переменные $zm_{v,r,q}$ такие, что $zm_{v,r,q} = 1$ тогда и только тогда, когда все переменные $m_{u,r,q}$ ложны (**z**ero-valued **m**) для всех $u \leqslant v$. Переменные $zm_{v,r,q}$ можно рекурсивно определить идентично определению переменных $ne_{v,r,q}$ и формально выразить следующим образом:

$$\begin{aligned} &zm_{0,r,q} \leftrightarrow 1, \\ &zm_{v,r,q} \leftrightarrow \neg m_{v,r,q} \wedge zm_{v-1,r,q}, \qquad 1 \leqslant v < L. \end{aligned} \tag{35}$$

Теперь, используя дополнительные переменные $zm_{v,r,q}$, модифицируем ограничение (32):

$$\bigwedge_{1 \leqslant r < q \leqslant M} \bigwedge_{1 < v \leqslant L} \left(p_{q,r} \wedge p_{j+1,r} \wedge m_{v,r,q} \rightarrow zm_{v-1,r,q+1}\right) \tag{36}$$

Ограничения (35) и (36) выражаются также с помощью $\mathcal{O}(M^2 L)$ дизъюнктов.

Таким образом, заменив ограничения (10), (11), (13), (14), (15) новыми ограничениями (27), (28), (29), (30), (33), (34), (35) и (36) можно сократить общий размер формулы, задающей предикаты нарушения симметрии на основе алгоритма обхода графа в ширину, с $\mathcal{O}(M^3 + M^2 L^2 + M^2 L)$ до $\mathcal{O}(M^2 L)$ дизъюнктов.

4.2. Дополнительные методы ограничения пространства поиска, использующие BFS-нумерацию автомата

В данном разделе приводятся дополнительные разработанные методы и техники, позволяющие дополнительно ограничивать пространство поиска для задачи MinDFA. Данные методы и техники активно используют дополнительную информацию, которую можно получить из того факта, что с помощью предикатов нарушения симметрии фиксируется нумерация искомого автомата в порядке алгоритма BFS.

Заметим, что в BFS-дереве для некоторого автомата у каждого состояния может быть не более чем L дочерних состояний, где, напомним, L — размер алфавита. Назовем такое дерево L-деревом. L-дерево, у которого у каждого внутреннего состояния ровно L детей называется полным. Так как нумерация состояний автомата однозначно задается L-BFS-деревом, то проанализировав возможные структуры таких деревьев, можно получить дополнительные ограничения на нумерацию искомого ДКА. На рисунке7 изображено полное BFS-дерево, с помощью которого можно отследить такие границы значений. Большая часть предлагаемых идей непосредственно связана с данным рисунком.

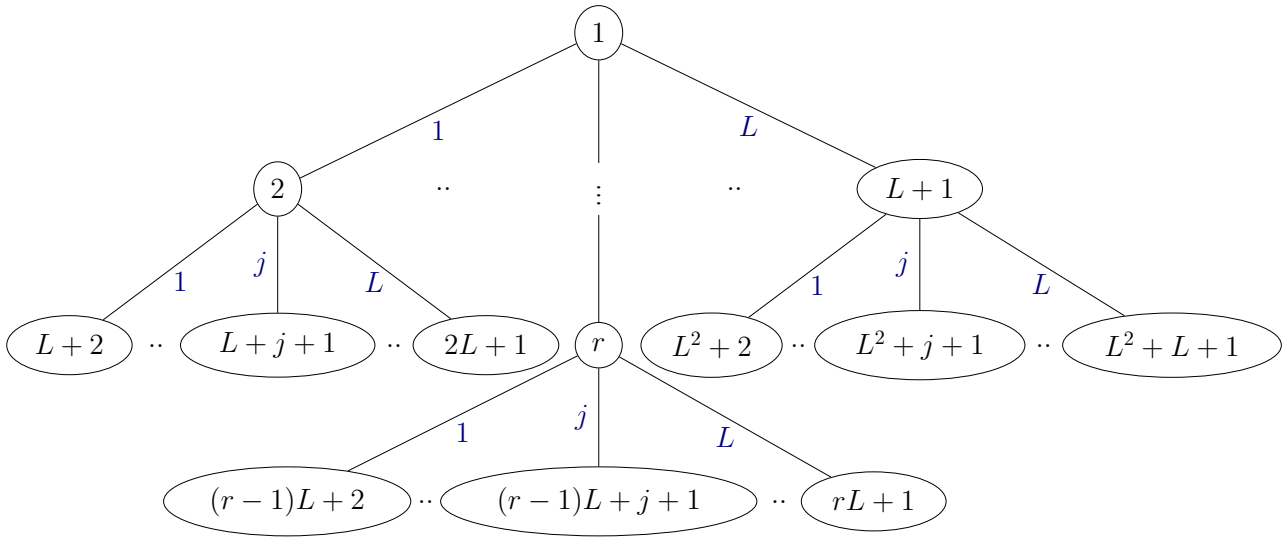

Рисунок 7 – Полное L-BFS-дерево, $1 \leqslant j \leqslant L$

Ранее мы задавали родительские переменные $p_{q,r}$ для таких r и q, чтобы выполнялось неравенство $1 \leqslant r < q \leqslant M$. Однако, проанализировав рисунок 7, можно получить более точные границы для значений q при фиксированном r. Или, говоря предметно, можно определить для фиксированной вершины с номе-

ром r границы возможных значений номеров дочерних состояний. Для начала докажем лемму, которая потребуется нам в дальнейшем.

Лемма 1. Количество состояний в полном L-BFS-дереве на глубине k равно L^k (если считать, что корень дерева находится на нулевой глубине).

Доказательство. Докажем по индукции. Для корня, который находится на нулевой глубине, очевидно, выполняется лемма: $L^0 = 1$. Теперь, пусть лемма выполняется для некоторой глубины дерева k, и количество вершин на данной глубине равно L^k. Тогда, так как дерево полное и каждое состояние имеет по L детей, то количество вершин на следующем уровне будет равно $L^k \times L = L^{k+1}$.

Теперь, докажем интересное свойство L-BFS-дерева.

Теорема 1. Состояние r L-BFS-дерева, где $1 \leqslant r \leqslant M$, может быть родителем только состояний с номерами от $r + 1$ до $rL + 1$.

Доказательство. Нижняя граница очевидна — она достигается, например, для корневого состояния BFS-дерева. Верхняя же граница достигается в полном L-BFS-дерево, когда у вершины L имеется максимальное возможное число родных и двоюродных братьев справа, а также максимальное количество племянников слева. Пусть состояние r находится на глубине k. Из леммы 1 следует, что суммарное количество вершин на глубине не больше k равно $S(k) = \sum_{i=0}^{k} L^i$. А так это BFS-дерево, то и нумерация состояний производится по глубине (или, иными словами, по уровням). Тогда число родных и двоюродных братьев r справа в полном L-BFS-дереве равно $S(k) - r$. Число же родных и двоюродных братьев слева — $r - S(k - 1) - 1$. Тогда количество «левых» племянников — $(r - S(k - 1) - 1)L$. Ну и, наконец, у самого состояния r имеется L детей. Тогда номер самого правого ребенка в полном L-BFS-дереве — $r + S(k) - r + (r - S(k - 1) - 1)L + L = rL + S(k) - S(k - 1)L = rL + 1$.

Верхняя граница из теоремы 1 приведена на рисунке 7. Из этой же теоремы следует, что значение переменных $p_{q,r}$ могут быть ненулевыми только при выполнении неравенства $r + 1 \leqslant q \leqslant rL + 1$. Так как $q > r$ по определению переменных $r_{q,r}$, то верна следующая теорема.

Теорема 2. $p_{q,r} = 0$ for $q > rL + 1$.

Доказательство данной теоремы очевидно следует из доказательства теоремы 1. Из теоремы 2 следует, что значения переменных $e_{v,r,q}$ также ограничены.

Теорема 3. $e_{v,r,q} = 0$ for $q > rL + 1$ and $v \in [L]$.

Доказательство. Докажем от противного. Пусть $e_{v,r,q} = 1$, то есть существует переход из состояния r в состояние q по некоторому символу, при некоторых значениях r и q таких, что $q > rL + 1$. Из теоремы 1 следует, что r не может быть родителем состояния q (то есть $p_{q,r} = 0$). Пусть тогда родитель состояния q в BFS-дереве имеет номер r' (то есть $p_{q,r'} = 1$). Тогда по теореме 1 выполняется свойство $q < r'L + 1$. Из этого следует, что $rL + 1 < q < r'L + 1 \Rightarrow r < r'$. Но согласно алгоритму BFS мы должны были посетить состояние q, находясь в состоянии r, так как существует переход из r в q. А значит, должно выполняться $p_{q,r} = 1$. Противоречие.
Из определения переменных $t_{r,q}$ также следует, что если $e_{v,r,q} = 0$, то и $t_{r,q} = 0$.

Так как в нашей формулировке BFS-дерево имеет фиксированный порядок на символах алфавита (по договоренности, лексикографический, однако можно использовать любой), то также верно следующее утверждение.

Утверждение 1. $e_{v,r,rL+2-j} = 0$ for $j \in [L-1]$ and $v \in \{1, \ldots, L-j\}$.

Оставим данное утверждение без доказательства, так как оно схоже с доказательством теоремы 1. Необходимо аккуратно рассмотреть максимально возможные номера детей состояния r справа налево и рассмотреть максимально возможный символ на ребре, идущем в конкретного ребенка.

Свойство непрерывности. Заметим теперь, что значения переменных $p_{q,r}$ обладают *свойством непрерывности*, продиктованным порядком обхода алгоритма BFS. Суть данного свойства заключается в том, что дети состояния r имеют последовательные номера и их не более чем L.

Утверждение 2. Родительские переменные должны обладать свойством непрерывности, то есть для некоторого фиксированного r верно следующее:

$$\begin{aligned} &p_{q,r} = 1, && a \leqslant q \leqslant b, \\ &p_{q,r} = 0, && q < a, \\ &p_{q,r} = 0, && q > b, \\ &b - a + 1 \leqslant L. \end{aligned} \tag{37}$$

Свойство непрерывности может быть эффективно и компактно закодировано с использованием дополнительных переменных и ограничений, которые представлены далее.

Определим новые булевы переменные $lnp_{q,r}$ (**l**eft-**n**o-**p**arent), которые истинны, когда состояние r является родителем некоторого состояния $q+1$, но не является родителем состояния q. Задать данные переменные можно следующим образом:

$$\neg p_{q,r} \wedge p_{q+1,r} \to lnp_{q,r}. \tag{38}$$

Дополнительно, потребуем выполнения ограничений:

$$\begin{aligned} &lnp_{q,r} \to lnp_{q-1,r}, \quad r+1 < q \leqslant M, \\ &lnp_{q,r} \to \neg p_{q,r}. \end{aligned} \tag{39}$$

Таким образом переменные $lnp_{q,r}$ для фиксированного r истинны начиная с $q = r+1$ и до q' такого, что $p_{q'+1,r} = 1$.

Определим аналогичным образом новые переменные $rnp_{q,r}$ (**r**ight-**n**o-**p**arent), которые истинны, когда состояние r является родителем состояния $q-1$, но не является родителем состояния q. Зададим их схожим образом:

$$p_{q-1,r} \wedge \neg p_{q,r} \to rnp_{q,r}. \tag{40}$$

Снова дополнительно потребуем выполнения ограничений, использующих переменные $rnp_{q,r}$:

$$\begin{aligned} &rnp_{q,r} \to rnp_{q+1,r}, \quad r+1 \leqslant q < M, \\ &rnp_{q,r} \to \neg p_{q,r}, \\ &rnp_{q,r} \to \neg e_{v,q,r}. \end{aligned} \tag{41}$$

Таким образом переменные $rnp_{q,r}$ для фиксированного r истинны начиная с $q = M$ и в порядке уменьшения до q' такого, что $p_{q'-1,r} = 1$. Переменные $lnp_{q,r}$ и $rnp_{q,r}$ задают границы a и b из утверждения 2, формулы (37).

Далее заметим, что длина отрезка $[a; b]$ не превышает значения L, так как у каждого состояния не более чем L дочерних. Из данного факта следует следующее ограничение:

$$\begin{aligned} &p_{q,r} \to rnp_{q+L,r}, \quad \text{если } q+L \leqslant M, \\ &p_{q,r} \to lnp_{q-L,r}, \quad \text{если } q-L \geqslant r+1. \end{aligned} \tag{42}$$

Основная цель переменных $lnp_{q,r}$ и $rnp_{q,r}$ — это заставить переменные $p_{q,r}$ быть ложны вне отрезка $[a; b]$. Однако, в некоторых ситуациях, зная значения этих переменных, можно вывести, что значение некоторых переменных $p_{q,r}$ наоборот должны быть истинны. Например, для всех таких промежутков значений q, что одновременно $lnp_{q_1,r}$ и $rnp_{q_2,r}$ ложны, верно, что $p_{q,r}$, где $q \in [q_1; q_2]$, должно быть истинно:

$$\neg lnp_{q_1,r} \wedge \neg rnp_{q_2,r} \to p_{q',r}, \qquad \begin{array}{c} q_1 < q' < q_2, \\ q_1 < q_2 \leqslant \min(q_1 + L - 1, rL + 1, M), \\ r + 1 \leqslant q_1 < \min(rL + 1, M). \end{array} \tag{43}$$

Здесь ограничения на q_1 и q_2 продиктованы теоремой 1 и свойством непрерывности.

Аналогично можно задать ограничения и на сами переменные $r_{q,r}$:

$$p_{q,r} \wedge p_{s,r} \to p_{s-1,r}, \qquad \begin{array}{c} q < s \leqslant \min(q + L - 1, rL + 1, M), \\ r + 1 \leqslant q < \min(rL + 1, M). \end{array} \tag{44}$$

Ограничения на q и s продиктованы теми же соображениями, что и ранее ограничения на q_1 и q_2.

Теперь, зная про свойство непрерывности и пользуясь доказанными ранее теоремы, можно добавить еще дополнительные ограничения на переменные $p_{q,r}$ и $e_{v,r,q}$. Если состояние r является родителем состояния q, то оно никак не может быть родителем состояний больших чем $q + L - 1$:

$$p_{q,r} \to \neg p_{q+L,r}, \qquad q \in \{l \,|\, l \geqslant r + 1 \wedge l + L \leqslant M \wedge l + L \leqslant rL + 1\}. \tag{45}$$

Более того, из аналогичных рассуждений можно получить более строгое ограничение:

$$p_{q,r} \to \neg e_{v,r,q+L}, \qquad \begin{array}{c} q \in \{l \,|\, l \geqslant r + 1 \wedge l + L \leqslant M \wedge l + L \leqslant rL + 1\}, \\ v \in [L]. \end{array} \tag{46}$$

Если же воспользоваться утверждением 1, то можно сделать ограничение (46) еще более строгим:

$$p_{q,r} \to \neg e_{v,r,q+j}, \qquad \begin{array}{c} j \in \{l \,|\, 1 \leqslant l \leqslant L-1 \wedge q+l \leqslant M \wedge q+l \leqslant rL+1\}, \\ r+1 \leqslant q \leqslant \min(rL+1, M), \\ v \in \{1, \ldots, j\}. \end{array} \tag{47}$$

4.3. Использование информации из префиксного автомата

В данном разделе приводятся дополнительные ограничения, которые могут быть добавлены к предикатам нарушения симметрии. Данные ограничения основаны на информации, получаемое из структуры префиксного автомата (APTA), а так же том факте, что алгоритм BFS по своей сути находит кратчайшее расстояние от начального состояния до всех остальных.

Пусть функция $D : [S] \to [M]$ для каждого состояния q искомого автомата возвращает длину пути от корня BFS-дерева до состояния q. Данное расстояние будет кратчайшим путем между состояниями 1 (корень) и q просто по определению алгоритма обхода в ширину. Функцию D можно индуктивно определить следующим образом:

$$\begin{array}{ll} D(q) = 0, & \text{если } q = 1, \\ D(q) = D(P(q)) + 1, & \text{иначе.} \end{array} \tag{48}$$

Пусть $\delta(i)$ для некоторого состояния i обозначает глубину данного состояния в APTA $\mathcal{T}$, и при этом $\delta(1) = 0$ (корень находится на нулевой глубине).

Тогда, очевидно, что всегда должно выполняться следующее свойство:

$$D(q) > \delta(i) \to m_i \neq q \tag{49}$$

Напомним, что $m_i = q$ обозначает, что состояние i префиксного автомата $\mathcal{T}$ соответствует состоянию q искомого ДКА.

Далее можно заметить, что исходя из того, что нумерация ДКА задается обходом алгоритмом BFS, то если для некоторых q и r известно, что $q > r$, тогда должно выполняться неравенство $D(q) \geqslant D(r)$. Пользуясь этим фактом, можно получить более строгую версию свойства (49):

$$D(q) > \delta(i) \to m_i < q \tag{50}$$

Действительно, пусть $m_i = r$. Тогда из свойства (49) следует, что $D(r) \leqslant \delta(i)$. Но тогда верна следующая цепочка неравенств $D(q) > \delta(i) \geqslant D(r) \Rightarrow D(q) > D(r) \Rightarrow q > r \Rightarrow m_i < q$.

Данное свойство означает, что если состояние i находится на некоторой глубине K в АРТА и оно соответствует состоянию q в ДКА ($m_i = q$), тогда в ДКА *должен* быть путь от стартового состояния 1 до состояния q, состоящий из K или менее переходов.

Пример 3. Рассмотрим снова пример искомого (изначально неизвестного) ДКА (автомат $\mathcal{S}$ с $M = 3$) на рисунке 1, и префиксный автомат, изображенный на рисунке 2, по которому был построен данный ДКА. Так как $m_1 = 1$ выполняется всегда, то рассмотрим другие пары состояний префиксного автомата и искомого ДКА. Заметим, что $\delta(2) = 1$ и $\delta(3) = 1$. Тогда можно добавить следующие ограничения:

$$(D(3) > 1 \rightarrow m_2 \neq 3) \wedge (D(3) > 1 \rightarrow m_3 \neq 3)$$

Или же, используя более строгое свойство (50), получим:

$$(D(3) > 1 \rightarrow m_2 < 3) \wedge (D(3) > 1 \rightarrow m_3 < 3)$$

Далее будет предложен способ задания свойств (48) и (49) на языке SAT. Сначала закодируем первое из этих свойств. Определим дополнительные булевы переменные $d_{q,j}$, где $q \in [M]$ и $1 \leqslant j < q$, такие, что $d_{q,j} = 1$ тогда и только тогда, когда длина пути в BFS-дереве от корня (имеющего номер 1) до состояния q есть j. То есть данные переменные кодируют выполнение равенства $D(q) = j$. Более того, определим еще одни дополнительные переменные $se_{q,j}$, где $q \in [M]$ и $1 \leqslant j < q$, такие, что $se_{q,j} = 1$ тогда и только тогда, когда длина пути в BFS-дереве от корня до состояния q меньше или равна (**s**maller or **e**qual) чем j. Данные переменные можно задать индуктивно, используя переменные $d_{q,j}$ следующим образом:

$$\begin{aligned} &se_{q,0} \leftrightarrow 0, \\ &se_{q,j} \leftrightarrow se_{q,j-1} \vee d_{q,j}. \end{aligned} \tag{51}$$

Используя переменные $se_{q,j}$, теперь можно определить переменные $d_{q,j}$:

$$d_{q,j} \leftrightarrow \neg se_{q,j-1} \wedge \left(\bigvee_{r=j}^{q-1} p_{q,r} \wedge d_{r,j-1} \right), \tag{52}$$

где $q \in [M]$ и $j \in [q-1]$. Так как r принимает значения в отрезке от 1 до $j-1$, то размер кодирования ограничений (51) и (52) — $\mathcal{O}(M^3)$. Можно заметить, что определения переменных $se_{q,j}$ и $d_{q,j}$ используют друг друга, однако если «считать» их значения по очереди, то взаимного зацикливания не получается.

Используя теперь переменные $d_{q,j}$, можно получить дополнительные ограничения на переменные $m_{i,p}$. Пусть t_i — состояние префиксного автомата такое, что его глубина $\delta(i) = I$. Тогда для любого состояния в искомом ДКА должно выполняться следующее ограничение:

$$d_{q,q-1} \vee d_{q,q-2} \vee \cdots \vee d_{q,I+1} \rightarrow \neg m_{i,q}. \tag{53}$$

Пример 4. Продолжая пример 3, и имея $I = 1$ и $q = 3$, можно создать следующие ограничения:

$$(\neg d_{3,2} \vee \neg m_{2,3}) \wedge (\neg d_{3,2} \vee \neg m_{3,3}).$$

Выводы по главе 4

В данной главе были представлены модифицированные предикаты нарушения симметрии. В первом разделе был представлен способ компактного кодирования, который позволяет сократить размер получаемой формулы $\mathcal{O}(M^3 + M^2L^2 + M^2L)$ до $\mathcal{O}(M^2L)$ дизъюнктов. Во втором разделе были представлены способы дополнительного отсечения областей пространства поиска на основе анализа структуры автомата и других особенностей.

ГЛАВА 5. КОМПАКТНОЕ СВЕДЕНИЕ НА ОСНОВЕ SMT-ФОРМУЛИРОВОК

В данной главе описывается новый подход к сведению задачи нахождения минимального ДКА к задаче SAT, основанный на использовании SMT-формулировок.

5.1. Кодирование сведения на языке SAT на основе SMT-формулировок

В данном разделе описывается как SMT-формулировки из раздела 1.2.3, а конкретно ограничения (7), могут быть использованы, чтобы получить новое сведение на языке SAT.

Чтобы закодировать некоторый терм Z, будем использовать оператор $\mathsf{Enc}(Z)$, который представляется парой $(\mathsf{Var}(Z),\mathsf{Cls}(Z))$, где $\mathsf{Var}(Z)$ это набор булевых констант или переменных $\langle v_1,\ldots,v_k\rangle$, необходимых для кодирования Z, и $\mathsf{Cls}(Z)$ — набор ограничений на данные булевы константы и переменные. Будем считать, что $k = 1$, в случае кодирования булевых констант и переменных; $k = M$, в случае унарного кодирования целочисленных констант или переменных; $k = \max(1, \lceil \log M \rceil)$, в случае бинарного кодирования целочисленных констант или переменных. При унарном кодировании переменных необходимо задать ограничение, что ровно одна булева переменная из k равна 1. При бинарном кодировании переменных необходимо задать ограничение, запрещающее значения между M и $2^{\lceil \log M \rceil}$.

Для целочисленной константы C такой, что $C \in [M]$, значение $\mathsf{Enc}(C) = (\mathsf{Var}(C),\mathsf{Cls}(C))$ определяет булево представление константы C. То есть $\mathsf{Var}(C) = \langle b_1,\ldots,b_k\rangle$, где $b_j \in \{0,1\}$, и $\mathsf{Cls}(C) = \emptyset$. Для целочисленной переменной I такой, что $C \in [M]$, верно, что $\mathsf{Enc}(I) = (\mathsf{Var}(I),\mathsf{Cls}(I))$, где $\mathsf{Var}(I)$, $\mathsf{Cls}(I)$ определяет ограничения для унарного или бинарного представления целых чисел.

Помимо этого, необходимо закодировать следующие SMT-термы: предикат равенства, записываемый как $X = Y$, где X и Y одновременно либо булевы значения, либо целочисленные с областью допустимых значений $[M]$; неинтерпретируемые функции $F(X)$, где область определения — $[M]$, а значения либо булевы, либо целочисленные из отрезка $[M]$.

Для предиката равенства $X = Y$ кодирование задается следующим образом:

$$\mathsf{Enc}(X = Y) \triangleq (\emptyset,\mathsf{Var}(X) \Leftrightarrow \mathsf{Var}(Y) \wedge \mathsf{Cls}(X) \wedge \mathsf{Cls}(Y)), \qquad (54)$$

где $\mathsf{Var}(X) \Leftrightarrow \mathsf{Var}(Y)$ кодируется как $(x_1 \leftrightarrow y_1) \wedge \ldots \wedge (x_k \leftrightarrow y_k)$ при условии, что представление термов X и Y — это $\langle x_1, \ldots, x_k \rangle$ и $\langle y_1, \ldots, y_k \rangle$, соответственно.

Для терма неинтерпретируемой функции $F(X)$ заметим, что $F(X)$ может быть записано как $\mathsf{Select}(F_1, \ldots, F_M;\, X)$, где $F_i \triangleq F(i)$. Значение функции F выбрается по X среди всех возможных значений F. Так как мы не заранее не знаем точно значения $F(i)$, то они должны быть закодированы как целочисленные переменные.

Пусть представление переменной F_i будет следующим: $\mathsf{Var}(F_i) = \langle f_{i,1}, \ldots, f_{i,k} \rangle$, и тогда пусть представление $F(X)$ будет: $\mathsf{Var}(F(X)) = \langle g_1, \ldots, g_k \rangle$. Тогда

$$\mathsf{Cls}(F(X)) \triangleq \bigwedge_{j=1}^{k} \Gamma_j(F(X)) \wedge \mathsf{Cls}(X) \wedge \bigwedge_{i=1}^{M} \mathsf{Cls}(F_i). \tag{55}$$

где,

$$\Gamma_j(F(X)) = \left(g_j \leftrightarrow \bigvee_{i=1}^{M} (\mathsf{Var}(X) \Leftrightarrow \mathsf{Var}(i)) \wedge f_{i,j} \right). \tag{56}$$

Данная функция кодирует факт, что $j^{\text{ый}}$ бит $\mathsf{Var}(F(X))$ должен быть закодирован с помощью $f_{i,j}$, когда X принимает значение i.

Более того, можно использовать дополнительные булевы переменные $p_{X=i}$, чтобы кодировать значение $\mathsf{Var}(X) \Leftrightarrow \mathsf{Var}(i)$. С использованием переменных $p_{X=i}$, кодирование функции $\Gamma_j(F(X))$ может быть переписанно следующим образом:

$$\Gamma_j(F(X)) = \left(g_j \leftrightarrow \bigvee_{i=1}^{M} p_{X=i} \wedge f_{i,j} \right). \tag{57}$$

В результате, $F(X)$ кодируется как $\mathsf{Enc}(F(X) \triangleq (\langle g_1, \ldots, g_k \rangle, \mathsf{Cls}(F(X)))$.

Таким образом, можно закодировать любое из ограничений вида $A(m_i) = v \in \{0,1\}$ и ограничений вида $E^v(m_i) = m_k$, где $v \in \Sigma$, используя представленный выше подход, конкретно ограничения (54), (55), (56), и (57).

Покажем как кодировать формулы (7). Пусть мы используем некоторое представление целочисленных переменных и констант. Тогда некоторая переменная m_i будет закодирована как пара $\mathsf{Enc}(m_i) = (\mathsf{Var}(m_i), \mathsf{Cls}(m_i))$. Аналогично, некоторые вызовы функций $A(m_i)$ и $E^v(m_i)$ будут закодированы с

помощью (55), (56) и (57). Наконец, равенства вида $A(m_i) = v \in \{0,1\}$ и $E^v(m_i) = m_k$ будут закодированы как (54). На каждом шаге будут добавляться необходимые дополнительные переменные.

Пусть $\mathbb{E}$ обозначает количество булевых переменных, использованные для кодирования целочисленных значений. Общее количество неинтерпретируемых функции равно $|\Sigma|+1$ — одна функция A и $|\Sigma|$ функций E^v. Как следует из формулы 7, существует ровно $\mathcal{O}(N)$ предикатов равенств в SMT-формулировке, где не более чем N равенств для неинтерпретируемой функции A, и где не более чем N равенств для неинтерпретируемых функций E^v. Кодирование каждой неинтерпретируемой функции требует кодирования целочисленных переменных в выбранном представлении, то есть $\mathcal{O}(\mathbb{E})$ переменных, и подсчет всех M возможных значений. Суммарно имеем, что число дизъюнктов в кодировании равняется $\mathcal{O}(N \times M \times \mathbb{E})$.

5.2. Кодирование целочисленных переменных

Два различных варианта кодирования целочисленных переменных может быть рассмотрено, конкретно, *унарное* и *бинарное*. По определению унарное кодирование требует $\mathbb{E} = \mathcal{O}(M)$ переменных, а для бинарного — $\mathbb{E} = \mathcal{O}(\log M)$. Таким образом, размер формул для представленного метода равняется $\mathcal{O}(N \times M^2)$ дизъюнктов при унарном кодировании целых чисел, и $\mathcal{O}(N \times M \times \log M)$ при бинарном кодировании. Насколько известно, данный подход является самым компактным на данный момент.

Необходимо отметить несколько нюансов по кодированию целочисленных переменных. При использовании унарного кодирования, для каждой целочисленной переменной необходимо добавить ограничение Equals1, которое указывает, что ровно одна булевая переменная в представлении истинна. Как упоминалось раннее, такое ограничение может быть закодировано с помощью $\mathcal{O}(M)$ дизъюнктов, что не влияет на асимптотику всего кодирования. При использованию бинарного кодирования, будем говорить, что значения начинаются с 0. Для каждого значения большего максимально возможного необходимо добавить блокирующее ограничение. Данные ограничения также не влияют на размер формулы.

Выводы по главе 5

В данной главе было представленно новое сведение задачи нахождения минимального ДКА к задаче SAT. Данное сведение основано на использовании

формулировок на языке SMT. При использовании бинарного кодирования целочисленных переменных размер формулы равен $\mathcal{O}(N \times M \times \log M)$ дизъюнктов, что является наиболее компактным известным сведением на данный момент.

ГЛАВА 6. ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ

Экспериментальные исследования проводились на персональном компьютере с процессором *AMD Opteron 6378* @ 2,4 ГГц и операционной системой *Ubuntu 14.04*. Все методы и алгоритмы были реализованы на языке `Java`. Был использован собственный алгоритм для генерации экземпляров задачи. Данный алгоритм генерирует входные словари по следующим параметрам: M - размер автомата, L - размер алфавита, S - количество слов, которые необходимо сгенерировать.

6.1. Методы нахождения всех неизоморфных ДКА

Первый эксперимент был связан с задачей построения всех неизоморфных ДКА минимального размера, подробно описанный в главе 3. Для экспериментального сравнения предложенных методов использовались следующие параметры для генерации входных экземпляров: $M \in [5; 12]$, $L = 2$, $S \in \{5N, 10N, 25N\}$. Сравнивались основанный на SAT метод со стратегией перезапуска, основанный на SAT метод с инкрементальной стратегией и метод поиска с возвратом. Решалась задача нахождения всех ДКА минимального размера. Каждый эксперимент был повторен 100 раз. Ограничение по времени работы было установлено на 3600 секунд. В таблицах 2, 3, 4 приведены результаты экспериментальных запусков для различных размеров словарей. Значения, выделенные *курсивом*, означают, что не все 100 запусков успели успешно завершиться в течении указанного ограничения по времени. Минимальное время работы для каждого размера автомата выделено серым.

В данных таблицах используются следующие обозначения:

- >1 — количество запусков, которые имели более одного ДКА в решении;
- M — размер автоматов;
- rest — среднее время работы основанного на SAT метода со стратегией перезапуска;
- inc — среднее время работы основанного на SAT метода с инкрементальной стратегией;
- btr — среднее время работы метода поиска с возвратом;
- TL — все запуски не успели завершиться в указанное ограничение по времени.

Из таблиц видно, что методы, основанные на SAT, работают значительно быстрее чем метод поиска с возвратом. Также можно видеть, что инкрементальной

Таблица 2 – Результаты экспериментальных запусков по задаче нахождения всех ДКА. Приведено время работы в секундах, усредненное по 100 запускам. $S = 5M$

M	>1	rest	inc	btr
5	79	1,1	0,6	0,3
6	77	1,9	0,9	0,9
7	87	5,1	1,4	5,8
8	91	22,7	2,3	95,1
9	98	45,9	3,7	*879,7*
10	99	206,1	7,8	TL
11	99	697,9	20,6	TL
12	100	2051,2	62,9	TL

ная стратегия полностью выигрывает у стратегии перезапуска. Данный факт может быть объяснен тем, что инкрементальное программное средство сохраняет свое состояние между запусками, а неинкрементальное делает повторяющиеся действия при каждом выполнении.

6.2. Предикаты нарушения симметрии на основе алгоритма DFS

Второй эксперимент проводился для сравнения предложенных в главе 2 предикатов нарушения симметрии на основе алгоритма DFS. Использовались также случайно сгенерированные данные. Для их генерации использовались следующие параметры: $M \in [10; 20]$, $L = 2$, $S = 50N$. Сравнивались между собой три подхода к нарушению симметрии: на основе больших клик, описанный в разделе 1.3.2, на основе алгоритма BFS (лучший на данные момент), описанный в разделе 1.3.3 и предложенный в данной работе в главе 2 подход на основе алгоритма DFS. Каждый эксперимент был также повторен 100 раз. Ограничение по времени работы было установлено на 3600 секунд. Результаты представлены в таблице 5. Снова значения, выделенные *курсивом* обозначают, что не все 100 примеров были решены в рамках ограничения по времени. Если же было решено менее 50 примеров, то в таблице вместо значения указано TL.

Из таблицы видно, что подход, основанный на DFS предикатах нарушения симметрии значительно выигрывает у подхода, основанного на больших кликах, однако при размере искомого автомата больше чем 14, уступает подходу, основанному на алгоритме BFS. Таким образом, DFS-подход на данный мо-

Таблица 3 – Результаты экспериментальных запусков по задаче нахождения всех ДКА. Приведено время работы в секундах, усредненное по 100 запускам. $S = 10M$

M	>1	rest	inc	btr
5	31	1,0	0,8	0,4
6	35	1,4	1,1	0,6
7	25	2,3	1,6	2,6
8	36	4,1	2,4	39,5
9	40	6,7	3,4	*399,7*
10	43	15,2	5,2	*1800,3*
11	47	23,5	8,3	TL
12	52	39,2	13,3	TL

мент не представляет практического интереса, но может служить основой для дальнейших исследований.

6.3. Модифицированные предикаты нарушения симметрии на основе алгоритма BFS

Третий эксперимент был посвящен сравнению базовых предикатов нарушения симметрии на основе алгоритма BFS, описанных в разделе 1.3.3, и модифицированной версии, которой посвящена вся глава 4. Данные предикаты используются совместно с классическим кодированием задачи нахождения минимального ДКА на языке SAT, описанном в разделе 1.2.2. Также данные подходы сравниваются с новым разработанным сведением на основе SMT-формулировок, описанным в главе 5 вкупе с модифицированными предикатами нарушения симметрии на основе алгоритма BFS. Снова использовались случайно сгенерированные данные. Использовались следующие параметры: $M \in \{20; 25; 30; 35\}$, $L = 2$, $S = 50M$. Каждый эксперимент был также повторен 100 раз. Ограничение по времени работы было установлено на 3600 секунд. Результаты представлены в таблице 6. Базовый подход с базовыми предикатами расположен в столбце с заголовком `Old`, с модифицированными предикатами — `New`, новое сведение с модифицированными предикатами — `SMT`.

Из таблицы видно, что базовое кодирование в паре с модифицированными предикатами нарушения симметрии работает примерно на 15% быстрее, чем

Таблица 4 – Результаты экспериментальных запусков по задаче нахождения всех ДКА. Приведено время работы в секундах, усредненное по 100 запускам. $S = 25M$

M	>1	rest	int	btr
5	14	2,3	2,0	1,1
6	20	4,6	3,4	2,0
7	5	5,8	5,2	4,7
8	11	8,3	7,3	*25,4*
9	6	11,6	10,2	*71,6*
10	12	15,4	13,7	*4455*
11	9	21,5	18,9	TL
12	8	29,2	26,2	TL

базовое кодирование с базовыми предикатами. Следует заметить, что хоть и кодирование, основанное на SMT-формулировках, проигрывает остальным двум методам, оно все работает быстрее, чем базовый подход с предикатами на основе клики (данный подход не способен решить в течение часа даже задачи, где $M = 15$). Как же будет показано в следующем эксперименте, данный подход позволяет значительно сократить размер формулы, что очень важно для б*о*льших задач.

6.4. Компактное сведение задачи построения ДКА на основе SAT-формулировок

В последнем эксперименте сравнивались размеры SAT-формул, получаемых при базовом кодировании из раздела 1.2.2 в паре с базовыми предикатами нарушения симметрии на основе алгоритма BFS из раздела 1.3.3 (то есть лучший известный способ решения задачи MinDFA к моменту начала работы над данной диссертацией) и при новом разработанном кодировании на основе SMT-формулировок из главы 5 в паре с модифицированными предикатами нарушения симметрии из главы 4. Сравнение проводилось на задачах с известного соревнования «`Abbadingo One: DFA Learning Competition`». Префиксные автоматы и искомые ДКА в данных задачах слишком велики для решения существующими точными методами нахождения ДКА минимального размера. Более того, раннее не существовало эффективного способа даже представить некоторые особо крупные из этих задач в ви-

Таблица 5 – Результаты экспериментальных запусков по предикатам нарушения симметрии на основе алгоритма DFS. Приведено время работы в секундах, усредненное по 100 запускам.

M	DFS	BFS	Клика
10	80.1	80.3	158.2
11	109.7	109.1	*337.3*
12	159.9	151.6	*684.4*
13	200.5	196.3	*1146.3*
14	301.1	254.4	TL
15	406.4	332.6	TL
16	824.7	560.4	TL
17	*1217.6*	631.9	TL
18	*1722.4*	685.7	TL
19	*2294.1*	778.9	TL
20	TL	903.1	TL

Таблица 6 – Результаты экспериментальных запусков по модифицированным предикатам нарушения симметрии. Приведено время работы в секундах, усредненное по 100 запускам.

M	New	Old	SMT
20	43,57	51,8	77,44
25	112,81	130,79	214,17
30	237,57	276,9	476,17
35	551,07	598,1	1806,05

де SAT-формул. Новое же разработанное сведение, которое выражается через $\mathcal{O}(NMlogM + M^2)$ (в то время как старое — через $\mathcal{O}(NM^2 + M^3)$), должно позволить закодировать их и представить на вычислительном устройстве. Результаты экспериментальных исследований представлены на таблице 7. В первом столбце указано название задачи согласно ее названию на соревновании. Во втором столбце (M) указан размер искомого ДКА, в третьем (N) — размер префиксного автомата, в четвертом (`Base + BFS`) — количество дизъюнктов в формуле для базового метода, в пятом (SMT + BFS*) — количество дизъюнктов

в формуле для нового метода, и, наконец, в последнем столбце указано отношение последних двух показателей.

Таблица 7 – Результаты экспериментальных запусков по компактному сведению на основе SMT-формулировок и модифицированных предикатов нарушения симметрии на основе алгоритма BFS.

Задача	M	N	Base + BFS	SMT + BFS*	Улучшение
A	61	15.006	61.224.354	30.766.270	1,99
B	119	51.273	768.529.707	234.755.006	3,27
C	247	155.253	9.782.126.761	1.662.006.041	5,89
D	498	517.873	130.803.043.574	12.454.394.622	10,50
1	63	12.824	55.698.107	26.277.248	2,12
2	138	42.621	852.569.590	253.770.360	3,36
3	260	128.208	8.931.320.477	1.601.169.918	5,58
R	499	425.869	107.724.894.309	9.975.751.227	10,80
4	68	10.242	51.529.659	26.726.228	1,93
5	130	33.508	596.583.032	188.127.242	3,17
6	262	98.784	6.987.426.170	1.241.195.023	5,63
S	506	322.067	83.760.950.600	7.637.347.376	10,97
7	65	7.242	33.430.813	18.081.131	1,85
8	125	22.586	369.830.980	104.828.929	3,53
9	267	64.896	4.766.845.970	829.298.531	5,75
T	519	201.041	55.107.908.559	5.506.817.691	10,01

Из таблицы видно, что размер SAT-формулы становится меньше в 2-10 раз. Для самых сложных задач D, R, S и T выигрыш в размере составляет порядок. Да, данные формулы все еще слишком сложны, чтобы решать их современными программными средствами, но новое сведение позволяет хотя бы закодировать их и представить на компьютере. Программные средства для решения задачи SAT активно развиваются, поэтому данный результат в перспективе может помочь решать даже такие сложные задачи.

ЗАКЛЮЧЕНИЕ

В данной работе были предложен ряд новых методов для решения задач построения конечных автоматов, таких как построение минимального детерминированного конечного автомата по заданному словарю и построение всех неизоморфных минимальных детерминированных конечных автоматов по заданному словарю.

Для задачи нахождения всех неизоморфных ДКА минимального размера был разработан метод, основанный на использовании того факта, что предикаты нарушения симметрии на основе алгоритма BFS для каждого класса эквивалентности по изоморфизму рассматривают только одного представителя как кандидата в процессе решения. Без данных предикатов нарушения симметрии данная задача не могла быть никак решена эффективно. Использование итеративных программных средств для решения задачи SAT помогает дополнительно заметно увеличить производительность данного метода.

Также были разработаны предикаты нарушения симметрии на основе алгоритма DFS, но в ходе экспериментального исследования было показано, что они уступают в эффективности предикатам на основе BFS. Так что данная разработка несет скорее теоретический характер.

В ходе работы было придумано компактное кодирование предикатов нарушения симметрии на основе BFS. Удалось сократить размер формулы асимптотически путем анализа структуры ограничений и добавления дополнительных переменных. Также было разработанно несколько дополнительных ограничений, которые не влияют на асимптотику, но помогают программному средству эффективнее перебирать пространство поиска. Данные улучшения позволили улучшить и без того лучший известный подход к решению задачи нахождения минимального ДКА на 15%.

Также было разработано принципиально новое кодирование самой задачи построения ДКА минимального размера по заданным словарям. Данный метод использует в качестве промежуточного шага сведения SMT-формулировки. В ходе экспериментальных исследований было показано, что данный метод несколько уступает традиционному методу, но не критично. Однако, если сравнивать размер получаемых формул, то на некоторых крупных примерах задач размер формулы сокращается на порядок.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Biermann A. W.*, *Feldman J. A.* On the Synthesis of Finite-State Machines from Samples of Their Behavior // IEEE Trans. Computers. — 1972. — Vol. 21, no. 6. — P. 592–597.

2 *Coste F.*, *Nicolas J.* Regular inference as a graph coloring problem // IWGI. — 1997.

3 *Coste F.*, *Nicolas J.* How Considering Incompatible State Mergings May Reduce the DFA Induction Search Tree // ICGI. — 1998. — P. 199–210.

4 *Oliveira A. L.*, *Marques-Silva J.* Efficient Algorithms for the Inference of Minimum Size DFAs // Machine Learning. — 2001. — Vol. 44, no. 1/2. — P. 93–119.

5 *Lang K. J.*, *Pearlmutter B. A.*, *Price R. A.* Results of the Abbadingo One DFA Learning Competition and a New Evidence-Driven State Merging Algorithm // Grammatical Inference, 4th International Colloquium, ICGI-98, Ames, Iowa, USA, July 12-14, 1998, Proceedings. — 1998. — P. 1–12.

6 *Lang K. J.* Faster algorithms for finding minimal consistent DFAs: tech. rep. / NEC Research Institute. — 1999.

7 *Grinchtein O.*, *Leucker M.*, *Piterman N.* Inferring Network Invariants Automatically // IJCAR. — 2006. — P. 483–497.

8 *Heule M.*, *Verwer S.* Exact DFA Identification Using SAT Solvers // ICGI. — 2010. — P. 66–79.

9 *Ulyantsev V.*, *Tsarev F.* Extended Finite-State Machine Induction Using SAT-Solver // ICMLA. — 2011. — P. 346–349.

10 *Neider D.*, *Jansen N.* Regular Model Checking Using Solver Technologies and Automata Learning // NFM. — 2013. — P. 16–31.

11 *Neider D.* Applications of automata learning in verification and synthesis : PhD thesis / Neider Daniel. — RWTH Aachen University, 2014. — URL: `http://darwin.bth.rwth-aachen.de/opus3/volltexte/2014/5169`.

12 *Heule M.*, *Verwer S.* Software model synthesis using satisfiability solvers // Empirical Software Engineering. — 2013. — Vol. 18, no. 4. — P. 825–856.

13 *Ulyantsev V.*, *Zakirzyanov I.*, *Shalyto A.* BFS-Based Symmetry Breaking Predicates for DFA Identification // LATA. — 2015. — P. 611–622.

14 *Ulyantsev V.*, *Zakirzyanov I.*, *Shalyto A.* Symmetry Breaking Predicates for SAT-based DFA Identification // CoRR. — 2016. — Vol. abs/1602.05028.

15 STAMINA: a competition to encourage the development and assessment of software model inference techniques / N. Walkinshaw [et al.] // Empirical Software Engineering. — 2013. — Vol. 18, no. 4. — P. 791–824.

16 *Hopcroft J. E.*, *Motwani R.*, *Ullman J. D.* Introduction to automata theory, languages, and computation - international edition (2. ed). — Addison-Wesley, 2003. — ISBN 978-0-321-21029-6.

17 *Higuera C. de la*. A bibliographical study of grammatical inference // Pattern Recognition. — 2005. — Vol. 38, no. 9. — P. 1332–1348. — DOI: 10.1016/j.patcog.2005.01.003. — URL: http://dx.doi.org/10.1016/j.patcog.2005.01.003.

18 *Bugalho M. M. F.*, *Oliveira A. L.* Inference of regular languages using state merging algorithms with search // Pattern Recognition. — 2005. — Vol. 38, no. 9. — P. 1457–1467. — DOI: 10.1016/j.patcog.2004.03.027. — URL: http://dx.doi.org/10.1016/j.patcog.2004.03.027.

19 *Angluin D.* Learning Regular Sets from Queries and Counterexamples // Inf. Comput. — 1987. — Vol. 75, no. 2. — P. 87–106.

20 *Trakhtenbrot B. A.*, *Barzdin Y. M.* Finite Automata: Behavior and Synthesis. — North-Holland, 1973.

21 *Abela J.*, *Coste F.*, *Spina S.* Mutually Compatible and Incompatible Merges for the Search of the Smallest Consistent DFA // ICGI. — 2004. — P. 28–39.

22 Handbook of Satisfiability. Vol. 185 / ed. by A. Biere [et al.]. — IOS Press, 2009. — (Frontiers in Artificial Intelligence and Applications). — ISBN 978-1-58603-929-5.

23 *Kroening D.*, *Strichman O.* Decision Procedures - An Algorithmic Point of View. — Springer, 2008. — (Texts in Theoretical Computer Science. An EATCS Series). — ISBN 978-3-540-74104-6.

24 *Gold E. M.* Complexity of Automaton Identification from Given Data // Information and Control. — 1978. — Vol. 37, no. 3. — P. 302–320. — DOI: 10.1016/S0019-9958(78)90562-4. — URL: http://dx.doi.org/10.1016/S0019-9958(78)90562-4.

25 *Biermann A. W.*, *Baum R. I.*, *Petry F. E.* Speeding up the Synthesis of Programs from Traces // IEEE Trans. Computers. — 1975. — Vol. 24, no. 2. — P. 122–136.

26 *Neider D.* Computing Minimal Separating DFAs and Regular Invariants Using SAT and SMT Solvers // ATVA. — 2012. — P. 354–369.

27 *Eén N.*, *Sörensson N.* An extensible SAT-solver // Theory and applications of satisfiability testing. — Springer. 2004. — P. 502–518.