Университет ИТМО

УНИВЕРСИТЕТ ИТМО

На правах рукописи

Закирзянов Илья Тимурович

Методы генерации детерминированных конечных автоматов с использованием сокращения пространства поиска при решении задачи выполнимости

Специальность 05.13.17 —
Теоретические основы информатики

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
канд. техн. наук
Ульянцев Владимир Игоревич

Санкт-Петербург — 2020

ITMO University

ITMO UNIVERSITY

As a manuscript

Zakirzyanov Ilya Timurovich

Deterministic finite automata inference methods using space search pruning when solving Boolean satisfiability problem

Specialty 05.13.17 —
Theoretical foundations of computer science

A thesis submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy

Scientific advisor:
Doctor of Philosophy
Ulyantsev Vladimir Igorevich

Saint Petersburg — 2020

Содержание

Реферат

Общая характеристика работы

Актуальность темы.

Степень разработанности темы.

Целью исследования является разработка точных методов генерации детерминированных конечных автоматов по имеющимся примерам поведения с использованием сокращения пространства поиска при решении задачи выполнимости.

Для достижения указанной цели определены следующие **задачи**:

а) Разработать точный метод построения ДКА по имеющимся примерам поведения с использованием «компактных» предикатов нарушения симметрии на основе алгоритма обхода графа в ширину.

б) Разработать методы сокращения пространства поиска при решении задачи выполнимости для построения ДКА по имеющимся примерам поведения, основанные на структурных особенностях искомого автомата.

в) Разработать точный метод построения всех возможных ДКА по имеющимся примерам поведения с использованием программных средств решения задачи выполнимости.

г) Разработать комбинированный точный метод построения ДКА по имеющимся примерам поведения на основе сведения к задаче выполнимости и с использованием подхода уточнения абстракции по контрпримерам.

д) Разработать программный комплекс, реализующий все вышеуказанные методы.

е) Провести экспериментальные исследования для оценки применимости и времени работы разработанных и реализованных методов, изложенных выше.

Объект исследования — методы генерации конечных автоматных моделей.

Предмет исследования — точные методы генерации детерминированных конечных автоматов по имеющимся примерам поведения.

Соответствие паспорту специальности. Данная диссертация соответствует пункту 5 «**Разработка и исследование моделей и алгоритмов анализа данных, обнаружения закономерностей в данных и их извлечениях разработка и исследование методов и алгоритмов анализа текста**, устной речи и изображений.».

*??? пункт 10 «**Разработка основ** математической теории языков и грамматик, **теории конечных автоматов** и теории графов.»*

*??? пункт 7 «**Разработка методов** распознавания образов, фильтрации, **распознавания и синтеза** изображений, **решающих правил**. Моделирование формирования эмпирического знания.»*

*??? пункт 3 «Исследование методов и разработка средств кодирования информации в виде данных. Принципы создания языков описания данных, языков манипулирования данными, языков запросов. **Разработка и исследование моделей данных и новых принципов их проектирования**».*

Основные положения, выносимые на защиту:

а) Разработан точный метод построения ДКА по имеющимся примерам поведения с использованием «компактных» предикатов нарушения симметрии на основе алгоритма обхода графа в ширину.

б) Разработаны методы сокращения пространства поиска при решении задачи выполнимости для построения ДКА по имеющимся примерам поведения, основанные на структурных особенностях искомого автомата.

в) Разработан точный метод построения всех возможных ДКА по имеющимся примерам поведения с использованием программных средств решения задачи выполнимости.

г) Разработан комбинированный точный метод построения ДКА по имеющимся примерам поведения на основе сведения к задаче выполнимости и с использованием подхода уточнения абстракции по контрпримерам.

Научная новизна заключается в следующем:

а) Впервые точный метод построения ДКА по имеющимся примерам поведения с использованием предикатов нарушения симметрии на основе алгоритма обхода графа в ширину был предложен научным руководителем В.И. Ульянцевым при участии автора диссертации. Новизна предложенного в данной работе метода заключается в использовании «компактных» предикатов нарушения симметрии — пропозициональное кодирование данных предикатов на языке задачи выполнимости содержит квадратичное относительно размера автомата количество дизъюнктов вместо кубического из оригинального метода. Разработанный метод помимо уменьшения размера булевой формулы обеспечивает статистически значимое преимущество скорости работы по сравнению с существующими точными методами.

б) Впервые были предложены методы сокращения пространства поиска при решении задачи выполнимости для построения ДКА по имеющимся примерам поведения, основанные на структурных особенностях искомого автомата. Использование данных методов обеспечивает статистически значимое преимущество скорости работы по сравнению с существующими точными методами.

в) Впервые был предложен точный метод построения всех возможных ДКА по имеющимся примерам поведения с использованием программных средств решения задачи выполнимости. Новизна предложенного метода заключается в использовании предикатов нарушения симметрии на основе алгоритма обхода графа в ширину, что позволяет оставить для рассмотрения единственного представителя для каждого класса эквивалентности по изоморфизму.

г) Впервые был предложен комбинированный точный метод построения ДКА по имеющимся примерам поведения на основе сведения к задаче выполнимости и с использованием подхода уточнения абстракции по контрпримерам. Новизна предложенного метода заключается в использовании только части имеющихся примеров поведения при построении ДКА, что обеспечивает статистически значимое преимущество скорости работы по сравнению с существующими точными методами в случаях, когда число имеющихся примеров поведения излишне велико относительно размера искомого автомата.

Методология и методы исследования. В работе используются методы теории автоматов, теории сложности, дискретной математики, объектно-ориентированное программирование, методы проведения и анализа экспериментальных исследований

Достоверность полученных результатов, подтверждается корректным обоснованием постановок задач, точной формулировкой критериев, а также результатами проведенных экспериментальных исследований по использованию предложенных в диссертации методов.

Теоретическая значимость работы состоит в том, что для задачи генерации ДКА по имеющимся примерам поведения было предложено пропозициональное кодирование предикатов нарушения симметрии на основе алгоритма обхода графа в ширину, содержащее асимптотически меньшее количество дизъюнктов. Также для задачи генерации ДКА по имеющимся примерам поведения были предложены методы по сокращению пространства поиска при решении задачи выполнимости. Помимо этого, был предложен метод для решения задачи построения всех различных неизоморфных ДКА по имеющимся примерам поведения, для решения которой ранее эффективных методов не существовало. Также был предложен метод, объединяющий в себе два различных подхода: сведение к задаче выполнимости и уточнение абстракции по контрпримерам.

Практическая значимость работы состоит в том, что разработанные методы и реализованные в рамках программного комплекса на основе данных ме-

тодов алгоритмы позволяют ускорить и улучшить генерацию ДКА по имеющимся примерам поведения, что позволяет решать задачи более высокой сложности, выраженной в размере искомого автомата, в размере алфавита, в недостаточном или избыточном покрытии автомата примерами поведения.

Участие в научно-исследовательских работах. Результаты диссертации использовались при проведении НИР «Разработка эффективных методов машинного обучения для построения детерминированных конечных автоматов на основе решения задачи выполнимости» (грант РФФИ «Мой первый грант» 18-37-00425, 2018–2020 гг.) под руководством автора диссертации.

Полученные результаты также использовались при проведении следующих НИР:

- «Биоинформатика, машинное обучение, технологии программирования, теория кодирования, проактивные системы» (субсидия 074-U01 в рамках государственной финансовой поддержки ведущих университетов Российской Федерации, 2013–2017 гг.);
- «Методы, модели и технологии искусственного интеллекта в биоинформатике, социальных медиа, киберфизических, биометрических и речевых системах», (субсидия 08-08 в рамках государственной финансовой поддержки ведущих университетов Российской Федерации, 2018–2020 гг.).

Апробация результатов работы. Основные результаты работы докладывались на следующих конференциях и семинарах:

а) 9th International Conference on Language and Automata Theory and Applications (LATA 2015). 2015, Ницца, Франция.

б) 6th International Symposium “From Data to Models and Back (DataMod)”. 2017, Тренто, Италия.

в) 13th International Conference on Language and Automata Theory and Applications (LATA 2019). 2015, Санкт-Петербург.

г) IV-VII Всероссийский Конгресс молодых ученых. 2015-2018, Санкт-Петербург.

д) IX Конгресс молодых ученых. 2020, Санкт-Петербург.
е) XLVI Научная и учебно-методическая Конференция Университета ИТМО. 2017, Санкт-Петербург.
ж) XLVIII Научная и учебно-методическая Конференция Университета ИТМО. 2019, Санкт-Петербург.

Личный вклад автора.. Идея точного метода построения ДКА по имеющимся примерам поведения с использованием «компактных» предикатов нарушения симметрии на основе алгоритма обхода графа в ширину принадлежит совместно автору диссертации и Ж. Маркешу-Сильве, реализация алгоритма на базе предложенного метода принадлежит лично автору диссертации, проведение вычислительных экспериментов принадлежит совместно автору диссертации и А.И. Игнатьеву. Идея методов сокращения пространства поиска при решении задачи выполнимости для построения ДКА по имеющимся примерам поведения, основанные на структурных особенностях искомого автомата принадлежит совместно автору диссертации и Ж. Маркешу-Сильве, реализация алгоритмов на базе предложенных методов принадлежит лично автору диссертации, проведение вычислительных экспериментов принадлежит совместно автору диссертации и А.И. Игнатьеву. Идея точного метода построения всех возможных ДКА по имеющимся примерам поведения с использованием программных средств решения задачи выполнимости принадлежит совместно автору диссертации и научному руководителю В.И. Ульянцеву, реализация алгоритма на базе предложенного метода и проведение вычислительных экспериментов принадлежит лично автору. Идея комбинированного точного метода построения ДКА по имеющимся примерам поведения на основе сведения к задаче выполнимости и с использованием подхода уточнения абстракции по контрпримерам принадлежит совместно автору диссертации и научному руководителю В.И. Ульянцеву, реализация алгоритма на базе предложенного метода и проведение вычислительных экспериментов принадлежит лично автору.

Публикации. Основные результаты по теме диссертации изложены в девяти публикациях, из них три опубликованы в изданиях, индексируемых в базе

цитирования Scopus, одна публикация издана в журнале, рекомендованном ВАК. Также у автора диссертации имеются две публикации по другим темам, из которых одна связана с машинным обучением, другая с построением автоматных моделей для кибер-физических систем, обе опубликованы в изданиях, индексируемых в базе цитирования Scopus.

Содержание работы

Во **введении** обосновывается актуальность исследований, проводимых в рамках данной диссертационной работы, приводится краткий обзор научной литературы по изучаемой проблеме, формулируется цель, ставятся задачи работы, излагается её научная новизна, теоретическая и практическая значимость.

В **заключении** приведены основные результаты работы, которые заключаются в следующем:

а) .

б) .

...

Публикации автора по теме диссертации

Публикации в зарубежных изданиях, индексируемых в базах цитирования Web of Science или Scopus

1. *Ulyantsev*, *V.*, *Zakirzyanov*, *I.*, *Shalyto*, *A.* BFS-Based Symmetry Breaking Predicates for DFA Identification // Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Nice, France, March 2-6, 2015, Proceedings. Vol. 8977. — Springer, 2015. — P. 611–622. — (Lecture Notes in Computer Science).
2. *Zakirzyanov*, *I.*, *Shalyto*, *A.*, *Ulyantsev*, *V.* Finding All Minimum-Size DFA Consistent with Given Examples: SAT-Based Approach // Software Engineering and Formal Methods - SEFM 2017 Collocated Workshops: DataMod, FAACS, MSE, CoSim-CPS, and FOCLASA, Trento, Italy, September 4-5, 2017, Revised Selected Papers. Vol. 10729. — Springer, 2017. — P. 117–131. — (Lecture Notes in Computer Science).
3. *Zakirzyanov*, *I.*, *Morgado*, *A.*, *Ignatiev*, *A.*, *Ulyantsev*, *V.*, *Marques-Silva*, *J.* Efficient Symmetry Breaking for SAT-Based Minimum DFA Inference // Language and Automata Theory and Applications - 13th International Conference, LATA 2019, St. Petersburg, Russia, March 26-29, 2019, Proceedings. Vol. 11417. — Springer, 2019. — P. 159–173. — (Lecture Notes in Computer Science).

Публикации в журналах из перечня ВАК

4. *Закирзянов*, *И. Т.* Построение детерминированных конечных автоматов по примерам поведения с использованием подхода уточнения абстракции по

контрпримерам // Научно-технический вестник информационных технологий, механики и оптики. — 2020. — Т. 20, № 3.

Прочие публикации

5. *Закирзянов*, *И. Т.* Разработка предикатов нарушения симметрии на основе поиска в ширину для построения детерминированных конечных автоматов // Сборник тезисов докладов Всероссийского конгресса молодых ученых. — 2015.

6. *Закирзянов*, *И. Т.* Эмпирическая оценка производительности программных средств решения задачи SAT для синтеза ДКА // Сборник тезисов докладов Всероссийского конгресса молодых ученых. — 2016.

7. *Закирзянов*, *И. Т.*, *Ульянцев*, *В. И.* Сравнительный анализ методов задания ограничений типа at-most-one на примере задачи построения ДКА с использованием программных средств решения SAT // Сборник тезисов докладов Всероссийского конгресса молодых ученых. — 2017.

8. *Закирзянов*, *И. Т.* Разработка предикатов нарушения симметрии для построения автоматных моделей программного обеспечения по заданной спецификации в виде темпоральных формул // Сборник тезисов докладов Всероссийского конгресса молодых ученых. — 2018.

9. *Закирзянов*, *И. Т.* Построение минимального ДКА на основе сведения к SAT с использованием предположений // Сборник тезисов докладов конгресса молодых ученых. — 2020.

Публикации автора по другим темам

Публикации в зарубежных изданиях, индексируемых в базах цитирования Web of Science или Scopus

10. *Kachalsky*, *I.*, *Zakirzyanov*, *I.*, *Ulyantsev*, *V.* Applying Reinforcement Learning and Supervised Learning Techniques to Play Hearthstone // 16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017, Cancun, Mexico, December 18-21, 2017. — IEEE, 2017. — P. 1145–1148.
11. *Ovsiannikova*, *P.*, *Chivilikhin*, *D.*, *Ulyantsev*, *V.*, *Stankevich*, *A.*, *Zakirzyanov*, *I.*, *Vyatkin*, *V.*, *Shalyto*, *A.* Active Learning of Formal Plant Models For Cyber-Physical Systems // 16th IEEE International Conference on Industrial Informatics, INDIN 2018, Porto, Portugal, July 18-20, 2018. — IEEE, 2018. — P. 719–724.

Synopsis

Thesis overview

Thesis contents

To **conclude**, the results of this study are:

Author's publications on the topic of the thesis

Publications indexed in Web of Science or Scopus

1. *Ulyantsev*, *V.*, *Zakirzyanov*, *I.*, *Shalyto*, *A.* BFS-Based Symmetry Breaking Predicates for DFA Identification // Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Nice, France, March 2-6, 2015, Proceedings. Vol. 8977. — Springer, 2015. — P. 611–622. — (Lecture Notes in Computer Science).
2. *Zakirzyanov*, *I.*, *Shalyto*, *A.*, *Ulyantsev*, *V.* Finding All Minimum-Size DFA Consistent with Given Examples: SAT-Based Approach // Software Engineering and Formal Methods - SEFM 2017 Collocated Workshops: DataMod, FAACS, MSE, CoSim-CPS, and FOCLASA, Trento, Italy, September 4-5, 2017, Revised Selected Papers. Vol. 10729. — Springer, 2017. — P. 117–131. — (Lecture Notes in Computer Science).
3. *Zakirzyanov*, *I.*, *Morgado*, *A.*, *Ignatiev*, *A.*, *Ulyantsev*, *V.*, *Marques-Silva*, *J.* Efficient Symmetry Breaking for SAT-Based Minimum DFA Inference // Language and Automata Theory and Applications - 13th International Conference, LATA 2019, St. Petersburg, Russia, March 26-29, 2019, Proceedings. Vol. 11417. — Springer, 2019. — P. 159–173. — (Lecture Notes in Computer Science).

Publications indexed in Russian journal included in the List of the Higher Attestation Commission

4. *Закирзянов*, *И. Т.* Построение детерминированных конечных автоматов по примерам поведения с использованием подхода уточнения абстракции по

контрпримерам // Научно-технический вестник информационных технологий, механики и оптики. — 2020. — Т. 20, № 3.

Other publications (in Russian)

5. *Закирзянов*, *И. Т.* Разработка предикатов нарушения симметрии на основе поиска в ширину для построения детерминированных конечных автоматов // Сборник тезисов докладов Всероссийского конгресса молодых ученых. — 2015.
6. *Закирзянов*, *И. Т.* Эмпирическая оценка производительности программных средств решения задачи SAT для синтеза ДКА // Сборник тезисов докладов Всероссийского конгресса молодых ученых. — 2016.
7. *Закирзянов*, *И. Т.*, *Ульянцев*, *В. И.* Сравнительный анализ методов задания ограничений типа at-most-one на примере задачи построения ДКА с использованием программных средств решения SAT // Сборник тезисов докладов Всероссийского конгресса молодых ученых. — 2017.
8. *Закирзянов*, *И. Т.* Разработка предикатов нарушения симметрии для построения автоматных моделей программного обеспечения по заданной спецификации в виде темпоральных формул // Сборник тезисов докладов Всероссийского конгресса молодых ученых. — 2018.
9. *Закирзянов*, *И. Т.* Построение минимального ДКА на основе сведения к SAT с использованием предположений // Сборник тезисов докладов конгресса молодых ученых. — 2020.

Author's publications on other topics

Publications indexed in Web of Science or Scopus

10. *Kachalsky*, *I.*, *Zakirzyanov*, *I.*, *Ulyantsev*, *V.* Applying Reinforcement Learning and Supervised Learning Techniques to Play Hearthstone // 16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017, Cancun, Mexico, December 18-21, 2017. — IEEE, 2017. — P. 1145–1148.
11. *Ovsiannikova*, *P.*, *Chivilikhin*, *D.*, *Ulyantsev*, *V.*, *Stankevich*, *A.*, *Zakirzyanov*, *I.*, *Vyatkin*, *V.*, *Shalyto*, *A.* Active Learning of Formal Plant Models For Cyber-Physical Systems // 16th IEEE International Conference on Industrial Informatics, INDIN 2018, Porto, Portugal, July 18-20, 2018. — IEEE, 2018. — P. 719–724.

Глава 1 Обзор предметной области

1.1 Задача выполнимости

1.2 Детерминированные конечные автоматы

Алфавитом Σ называется некоторое конечное непустое множество символов. *Размером алфавита* Σ называется число его символов — $L = |\Sigma|$. В данной диссертации в основном будет рассматривать бинарный алфавит $\Sigma = \mathbb{B} = \{0, 1\}$. *Словом* (*строкой*, *цепочкой*) ω называется конечная последовательность символов некоторого алфавита. *Длиной слова* называется число символов в нем, обозначается как $|\omega|$. Множество слов длины k над алфавитом Σ обозначается как Σ^k. *Пустой строкой* называется слово, не содержащее ни одного символа. Такая строка, обозначаемая как ε, имеет нулевую длиной и может рассматриваться как слово над любым алфавитом — $\Sigma^0 = \{\varepsilon\}$. Множество всех возможных слов, составленных из символов некоторого алфавита Σ, является его замыканием:

$$\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k.$$

Подмножество множества всех слов над алфавитом Σ называется *языком* — $\mathcal{L} \subset \Sigma^*$.

Детерминированным конечным автоматом (ДКА) называется пятерка $\mathcal{D} = (D, \Sigma, \delta, d_1, D^+)$, где D — конечное множество состояний, Σ — алфавит входных символов, $\delta : D \times \Sigma \to D$ — *функция переходов*, d_1 — *стартовое* (*начальное*) состояние, $D^+ \subset D$ — множество *допускающих* (*принимающих*) состояний. В неявном виде также задано множество *недопускающих* (*отвергающих*) состояний $D^- = D \setminus D^+$.

Индуктивно определим вспомогательную *расширенную функцию переходов* $\hat{\delta} : D \times \Sigma^* \to D$:

а) для любого состояния d_i верно, что переход по пустой строке не осуществляется — $\hat{\delta}(d_i, \varepsilon) = d_i$;

б) для любого состояния d_i верно, что переход по строке $\pi = \pi' c$, где $\pi, \pi' \in \Sigma^*$, $c \in \Sigma$, может быть определен следующим образом $\hat{\delta}(d_i, \pi) = \delta\left(\hat{\delta}(d_i, \pi'), c\right)$.

Говорят, что ДКА $\mathcal{D}$ *допускает* (*принимает*) слово ω, если $\hat{\delta}(d_1, \omega) \in D^+$. Иначе, если $\hat{\delta}(d_1, \omega) \in D^-$, говорят, что ДКА $\mathcal{D}$ *не допускает* (*отвергает*) слово ω. Множество всех слов, допускаемых автоматом $\mathcal{D}$, называется языком автомата $\mathcal{D}$: $\mathcal{L}(D) = \{\omega \mid \hat{\delta}(d_1, \omega)\}$.

[Илья: *рассказать про полноту и детерминированность*]

1.3 Задача генерации детерминированных конечных автоматов

[Илья: *Добавить вступление про то, зачем и почему нужно их генерировать. И прочую мишуру*]

[Илья: *Тут описать задачу — про примеры поведения, соответствие и прочее*]

1.4 Генерация детерминированных конечных автоматов при помощи сведения к задаче выполнимости

[Илья: *Здесь описание метода Сикко-Вервера.*]

1.4.1 Расширенное префиксное дерево

Первым шагом рассматриваемого подхода является построение *расширенного префиксного дерева* (augmented prefix tree acceptor — APTA) по имеющимся множествам примеров поведения S_+ и S_-. Расширенное префиксное дерево — это древовидная структура данных, основанная на обычном префиксном дереве (бор, нагруженное дерево, prefix tree, trie), отличающаяся от нее метками вершин. Более формально, префиксным деревом называется шестерка $\mathcal{T} = (T, \Sigma, \tau, t_1, T^+, T^-)$, где T — конечное множество вершин, Σ — алфавит входных символов, $\tau : T \times \Sigma \twoheadrightarrow T$ — *функция переходов*, t_1 — *корневая* вершина (*корень*), $T^+ \subset T$ — множество *допускающих* (*принимающих*) вершин, $T^- \subset T$ — множество *не допускающих* (*отвергающих*) вершин. В отличие от функции переходов в ДКА, функция переходов τ является частичной. Также, в префиксном дереве $T^+ \cup T^- \subset T$, то есть некоторые вершины могут не являться ни принимающими, ни отвергающими.

Расширенное префиксное дерево для множеств примеров поведения S_+ и S_- строится как обычное префиксное дерево, но терминальные вершины для каждого слова помечаются соответствующей меткой. Вершины, в которых не заканчивается ни один из примеров поведения, остаются непомеченными. Пример расширенного префиксного дерева приведен на [Илья: *рисунке*].

Также далее в диссертации с помощью $\Delta(v)$ будет обозначаться расстояния от корневой вершины t_1 до вершины t_v.

1.4.2 Пропозициональное кодирование

[Илья: *вступление какое-то про кодирование*]

Для того, чтобы закодировать задачу генерации ДКА минимального размера на языке SAT, авторами [1] было предложено ввести три типа булевых переменных [Илья: *где-то указать, что* $[N] = 1, 2, \ldots, N$]:

а) переменные соответствия $\{x_{v,i}\}_{v \in T, i \in [M]}$, которые истинны тогда и только тогда, когда вершина t_v в расширенном префиксном дереве $\mathcal{T}$ соответствует состоянию d_i в автомате $\mathcal{D}$;

б) переменные перехода $\{y_{i,l,j}\}_{i,j \in [M], l \in \Sigma}$, которые истинны тогда и только тогда, когда в автомате $\mathcal{D}$ существует переход из состояния d_i в состояние d_j по символу l;

в) переменные допуска $\{z_i\}_{i \in [M]}$, которые истинны тогда и только тогда, когда в автомате $\mathcal{D}$ состояние d_i является допускающим.

Используя вышеопределенные переменные, сведение, предложенное авторами [1], можно представить с помощью следующих множеств дизъюнктов.

[Илья: *возможно надо сформулировать на языке логики первого порядка, а потом уже привести к дизъюнктам*]

[Илья: *ОПА. Только что осознал, что так как половина из этих дизъюнктов является избыточными, необязательными, то они тоже служат для сокращения пространства поиска, а значит их надо, видимо, отдельно рассматривать. У Сикко и Хойла это называлось прямое сведение и какое-то там еще.*]

а) $(x_{v,1} \vee x_{v,2} \vee \ldots \vee x_{v,M})$ для $v \in [N]$ — каждой вершине t_v расширенного префиксного дерева $\mathcal{T}$ в соответствие ставится как минимум одно состояние автомата $\mathcal{D}$.

б) $(\neg x_{v,i} \vee \neg x_{v,j})$ для $v \in [N]; i, j \in [M]; i < j$ — каждой вершине t_v расширенного префиксного дерева $\mathcal{T}$ в соответствие ставится не более одного состояния автомата $\mathcal{D}$.

в) $(y_{i,l,1} \vee y_{i,l,2} \vee \ldots \vee y_{i,l,M})$ для $i \in [M]; l \in \Sigma$ — из каждого состояния d_i автомата $\mathcal{D}$ существует как минимум один переход по каждому символу l алфавита Σ, иными словами, ДКА $\mathcal{D}$ полон.

г) $(\neg y_{i,l,j} \vee \neg y_{i,l,h})$ для $i, j, h \in [M]; j < h; l \in \Sigma$ — из каждого состояния d_i автомата $\mathcal{D}$ существует не более одного перехода по каждому символу l алфавита Σ, иными словами, ДКА $\mathcal{D}$ детерминирован.

д) $(\neg x_{v,i} \vee z_i)$ для $t_v \in T^+; i \in [M]$ — если принимающей вершине t_v расширенного префиксного дерева $\mathcal{T}$ в соответствие ставится состояние d_i автомата $\mathcal{D}$, то это состояние также должно быть принимающим.

е) $(\neg x_{v,i} \vee \neg z_i)$ для $t_v \in T^-; i \in [M]$ — если отвергающей вершине t_v расширенного префиксного дерева $\mathcal{T}$ в соответствие ставится состояние d_i автомата $\mathcal{D}$, то это состояние также должно быть отвергающим.

ж) $(\neg x_{v,i} \vee \neg x_{w,j} \vee y_{i,l,j})$ для $v, w \in [N]; i, j \in [M]; l \in \Sigma; \tau(t_v, l) = t_w)$ — если вершине t_v расширенного префиксного дерева $\mathcal{T}$ в соответствие ставится состояние d_i автомата $\mathcal{D}$, вершине t_w — состояние d_j и в префиксном дереве $\mathcal{T}$ существует переход из вершины t_v в вершину t_w по символу l, то в автомате $\mathcal{D}$ должен быть переход из состояния d_i в состояние d_j по символу l.

и) $(\neg x_{v,i} \vee \neg y_{i,l,j} \vee x_{w,j})$ для $v, w \in [N]; i, j \in [M]; l \in \Sigma; \tau(t_v, l) = t_w)$ — если вершине t_v расширенного префиксного дерева $\mathcal{T}$ в соответствие ставится состояние d_i автомата $\mathcal{D}$, и в префиксном дереве $\mathcal{T}$ существует переход из вершины t_v в вершину t_w по символу l и в автомате $\mathcal{D}$ существует переход из состояния d_i в состояние d_j по символу l, вершине t_w дерева $\mathcal{T}$ должно ставиться в соответствие состояние d_j.

к) $x_{1,1}$ — корню t_1 расширенного префиксного дерева $\mathcal{T}$ в соответствие ставится начальное состояние d_1 автомата $\mathcal{D}$.

Все представленные выше наборы дизъюнктов могут быть объединены с помощью конкатенации в одну большую булеву формулу. Всего в такой формуле будет $\mathcal{O}(M^3 + N \times M^2)$ дизъюнктов и для их кодирования будет использовано $\mathcal{O}(M^2 + N \times M)$ переменных.

1.5 Подходы к сокращению пространства поиска при генерации детерминированных конечных автоматов

[Илья: *рассказать про пространство поиска в принципе*]

1.5.1 Граф несовместимости

Помимо пропозиционального кодирования, описанного в разделе 1.4.2, авторы [1] предложили использовать вспомогательную структуру данных, названную ими *графом совместимости* (*consistency graph*). Несмотря на оригинальное название, как будет видно далее, данная структура данных должна скорее называться *графом несовместимости* (*inconsistency graph*). В данной диссертации предлагается использовать последнее название.

В основе данной идеи лежит подход со слияниями состояний. Две вершины t_v и t_w расширенного префиксного дерева можно слить (объединить, merge) в одну вершину $t_{v'}$, объединив множества исходящих из них переходов. Если в результате данной операции в префиксном дереве возникла недетерминированность, то есть из вершины $t_{v'}$ теперь исходят два различных перехода по одному и тому же символу в различные вершины t_q и t_r, то от нее можно избавиться объединив вершины t_q и t_r в одну вершину $t_{q'}$. Рекурсивно продолжая данный процесс, можно избавиться от всех случаев недетерминированности в префиксном дереве. Важным фактом является то, что если в процессе избавления от недетерминированности в какой-то момент приходится объединить принимающую вершину с отвергающей, то изначальное слияние вершин t_v и t_w невозможно. В таком случае говорят, что вершины t_v и t_w *несовместимы* (*inconsistent*). Данную информацию можно использовать для помощи программному средству для решения задачи SAT.

Более формально, по имеющемуся расширенному префиксному дереву $\mathcal{T} = (T, \Sigma, \tau, t_1, T^+, T^-)$ предлагается построить граф $\mathcal{I} = (V, E)$ такой, что его множество вершин V совпадает со множеством вершин $\mathcal{T}$ префиксного дерева $\mathcal{T}$, а множество ребер E определяется следующим образом. Две вершины в графе $\mathcal{I}$ соединены ребром тогда и только тогда, когда их объединение и последующее избавление от недетерминированности приводит к несовместимости.

Так как искомый ДКА $\mathcal{D}$ соответствует префиксному дереву $\mathcal{T}$, то если две вершины t_v и t_w смежны в графе несовместимости $\mathcal{I}$, то они не могут соответствовать одному и тому же состоянию d_i автомата $\mathcal{D}$. Данное свойство может быть выражено с помощью переменных $x_{v,i}$ в виде следующего множества дизъюнктов — $(\neg x_{v,i} \vee \neg x_{w,i})$ для $v, w \in [N]\,; (v, w) \in E; i \in [M]$. Такие дизъюнкты необязательны, но их добавление к основной булевой формуле помогает значительно сократить пространство поиска программного средства для решения SAT. Однако, надо заметить, что в общем случае таких дизъюнктов будет $\mathcal{O}\left(N^2 \times M\right)$, что на порядок относительно N увеличивает размер формул. Использование графа несовместимости в таком виде для построения больших автоматов по большому множеству примеров поведения нецелесообразно. В разделе [Илья: *ДОБАВИТЬ ССЫЛКУ*] будут предложены несколько способов использования только части графа несовместимости, что не увеличивает размер формулы, но позволяет дополнительно сократить пространство поиска.

1.5.2 Изоморфные автоматы

[Илья: *рассказать про изоморфные автоматы и симметрию. Отметить, что их факториал*]

1.5.3 Большая клика

Как было сказано в предыдущем разделе, при отсутствии каких-либо ограничений на нумерацию состояний автомата, существует $M!$ изоморфных автоматов. Авторы [1] предложили свой способ нарушения симметрии, позволяющий сократить число рассматриваемых изоморфных автоматов. Так как в графе несовместимости смежные вершины по определению не могут соответствовать одному состоянию в искомом ДКА, то все вершины в некоторой клике (клика — полный граф, граф в котором каждая вершина соединена ребром со всеми остальными) [Илья: *ссылку на статью про клики*] такого графа будут соответствовать различным состояниям автомата. Учитывая, что итоговая нумерация состояний автомата не важна, можно, не уменьшая общности, зафиксировать номера вершин такой клики, что и было предложено в рассматриваемой статье.

Однако, задача поиска клики максимального размера в некотором графе является NP-полной [Илья: *ссылку*]. Поэтому авторами было предложено найти клику большого, но не обязательно максимального размера. Сделать это можно с помощью следующего эвристического подхода. В графе ищется вершина максимальной степени, которая будет входить в искомую большую клику. Затем ищется смежная ей вершина максимальной степени и добавляется в клику. Затем ищется вершина максимальной степени смежная обеим предыдущим и также добавляется в клику. Повторяя данный процесс, пока есть возможность существуют вершины смежные всем уже добавленным в клику.

Таким образом, если размер найденной клики равен C, то вместо $M!$ изоморфных автоматов будут рассмотрены $(M - C)!$ изоморфных автомата. Учитывая скорость роста факториала, разница между $M!$ и $(M - C)!$ может быть значительной, однако в общем случае число рассматриваемых изоморфных автоматов все еще остается факториальным относительно размера автомата. Помимо вышеуказанного недостатка, данный подход требует обязательного построения полного графа несовместимости, что, как было/будет [Илья: *???*] рассмотрено

ранее/далее, далеко не всегда является возможным.

1.5.4 Предикаты нарушения симметрии на основе алгоритма обхода графа в ширину

Идеальным нарушением симметрии является ситуация, когда для каждого класса эквивалентности относительно симметрии остается единственный представитель. Для задачи генерации ДКА — когда для каждого класса эквивалентности по изоморфизму остается единственный представитель. Автором данной диссертации совместно с научным руководитем [Илья: *тут понять как статью с первой латы обыгрывать*] в [2] были разработаны предикаты нарушения симметрии на основе алгоритма обхода графа в ширину, которые позволяют добиться идеального нарушения симметрии

Идея предложенного подхода состоит в том, чтобы зафиксировать нумерацию состояний в порядке *обхода в ширину* (*breadth-first search* — BFS) данного автомата. [Илья: *наверное, нужно рассказать, что за BFS такой. описание+псевдокод*] Для того, чтобы сделать обход в ширину уникальным, необходимо зафиксировать некоторый порядок на входных символах переходов, например, лексикографический. Будем называть ДКА *BFS пронумерованным*, если нумерация его состояний соответствует порядку обхода его состояний в ширину. Тогда в каждом классе эквивалентности по изоморфизму будет ровно один BFS пронумерованный представитель.

Другими словами, если рассмотреть дерево BFS, построенное для некоторого ДКА, расположив при этом детей в соответствии с выбранным порядком на символах переходов, тогда номера состояний:

– должны быть различными числами от 1 до M;
– на одной глубине должны увеличиваться слева направо (*порядок по уровню*);

– должны увеличиваться сверху вниз (*порядок по глубине*).

Пример BFS пронумерованного автомата и построенного для него дерева BFS приведен на рисунке [Илья: *рисунок*].

Если закодировать требование к автомату, чтобы он был BFS пронумерованным, в виде булевых предикатов и добавить к имеющейся формуле, предложенной в [1], то программное средство для решения SAT будет искать автоматы, удовлетворяющие заданным примерам поведения, пронумерованные в порядке BFS. Для того, чтобы закодировать данное требование, было предложено ввести три новых типа булевых переменных:

а) переменные родителей $\{p_{j,i}\}_{1 \leqslant i < j \leqslant M}$, которые истинны тогда и только тогда, когда состояние d_i является родителем состояния d_j в BFS дереве автомата $\mathcal{D}$;

б) переменные наличия переходов $\{t_{i,j}\}_{1 \leqslant i < j \leqslant M}$, которые истинны тогда и только тогда, когда в автомате $\mathcal{D}$ существует переход из состояния d_i в состояние d_j;

в) переменные минимального символа $\{m_{i,l,j}\}_{1 \leqslant i < j \leqslant M; l \in \Sigma}$, которые истинны тогда и только тогда, когда в автомате $\mathcal{D}$ из состояния d_i в состояние d_j существует переход по символу l, но не существует переходов по меньшим (согласно выбранному порядку на символах) символам. Данные переменные используются только в случае небинарного алфавита.

Используя данные переменные, можно закодировать свойство BFS пронумерованности автомата с помощью следующих множеств дизъюнктов.

а) $(t_{i,j} \leftrightarrow y_{i,l_1,j} \vee y_{i,l_2,j} \vee \ldots \vee y_{i,l_L,j})$ для $1 \leqslant i < j \leqslant M; l_k \in \Sigma$ — в автомате $\mathcal{D}$ переход из состояния d_i в состояние d_j существует тогда и только тогда, когду существует переход из состояния d_i в состояние d_j хотя бы по одному из символов алфавита Σ.

б) $(p_{j,i} \leftrightarrow t_{i,j} \wedge \neg t_{i-1,j} \wedge \neg t_{i-2,j} \wedge \ldots \wedge \neg t_{1,j})$ для $1 \leqslant i < j \leqslant M$ — состояние d_i автомата $\mathcal{D}$ является родителем состояния d_j в BFS дереве, если из состояния d_i существует переход в состояние d_j, а из состояний с меньшим номером такого перехода не существует.

в) $(p_{j,1} \vee p_{j,2} \vee \ldots \vee p_{j,j-1})$ для $2 \leqslant j \leqslant M$ — у каждого состояния d_j автомата $\mathcal{D}$, кроме стартового, родитель в BFS дереве должен иметь меньший номер. Данные дизъюнкты позволяют закодировать порядок по глубине в поддереве.

г) $(p_{j,i} \to \neg p_{j+1,k})$ для $1 \leqslant k < i < j \leqslant M$ — если состояние d_i является родителем в дереве BFS состояния d_j, то у состояния с бо́льшим номером d_{j+1} родитель не может иметь номер меньший, чем i. Данные дизъюнкты позволяют задать порядок по уровню для детей различных родителей, а также порядок по глубине в разных поддеревьях.

д) $(p_{j,i} \wedge p_{j+1,i} \to y_{i,l_1,j})$ для $1 \leqslant i < j \leqslant M; \Sigma = \{l_1, l_2\}$ — если состояние d_i является родителем в дереве BFS двух состояний d_j и d_{j+1} и алфавит бинарный, то переход из состояния d_i в d_j должен быть по меньшему символу. В случае бинарного алфавита данного множества дизъюнктов достаточно, чтобы упорядочить двух детей d_j и d_{j+1} одного состояния d_i. Дополнительно, для сокращения пространства поиска, можно добавить множество дизъюнктов $(p_{j,i} \wedge p_{j+1,i} \to y_{i,l_2,j+1})$ для $1 \leqslant i < j \leqslant M$. Данные дизъюнкты задают порядок по уровню для детей одного родителя в случае бинарного алфавита.

е) $\left(m_{i,l_n,j} \leftrightarrow y_{i,l_n,j} \wedge \neg y_{i,l_{n-1},j} \wedge \neg y_{i,l_{n-2},j} \wedge \ldots \wedge \neg y_{i,l_1,j}\right)$ для $1 \leqslant i < j \leqslant M; 1 \leqslant n \leqslant L; l_n \in \Sigma$ — в автомате $\mathcal{D}$ символ l_n является минимальным символом на переходах из состояния d_i в состояние d_j тогда и только тогда, когда в автомате существует переход из состояния d_i в состояние d_j по символу l_n, но не существует переходов по меньшим (относительно выбранного порядка) символам. Данное множество дизъюнктов используется в случае небинарного алфавита.

ж) $(p_{j,i} \wedge p_{j+1,i} \wedge m_{i,l_n,j} \to \neg m_{i,l_k,j+1})$ для $1 \leqslant i < j \leqslant M; 1 \leqslant k < n \leqslant L; l_n, l_k \in \Sigma$ — если состояние d_i является родителем в дереве BFS двух состояний d_j и d_{j+1} и алфавит состоит из более чем двух символов, то из состояния d_i переход в состояние d_j должен быть по меньшему (относительно выбранного порядка) символу чем в состояние d_{j+1}. Дан-

ные дизъюнкты задают порядок по уровню для детей одного родителя в случае небинарного алфавита.

Все представленные выше наборы дизъюнктов могут быть объединены с помощью конкатенации в одну большую булеву формулу. Всего в такой формуле будет $\mathcal{O}(M^3 + M^2 \times L^2)$ дизъюнктов и для их кодирования будет использовано $\mathcal{O}(M^2 \times L)$ переменных.

Глава 2 Компактные предикаты нарушения симметрии

Кодирование свойства BFS пронумерованности автомата, описанное в разделе 1.5.4, состоящее из $\mathcal{O}\left(M^3 + M^2 \times L^2\right)$ дизъюнктов, слабо применимо на практике для ДКА большого размера, то есть при большом M. В данной главе описывается как модифицировать предикаты нарушения симметрии, рассмотренные ранее, таким образом, что для их булевого кодирования понадобится только $\mathcal{O}\left(M^2 \times L\right)$ дизъюнктов.

[Илья: *подумать о подразделах*]

2.1 Ревизия существующего кодирования

Повторно кратко приведем наборы дизъюнктов, с помощью которых кодируются предикатов нарушения симметрии на основе алгоритма BFS, приведенные в разделе 1.5.4.

$$\bigwedge_{1 \leqslant i < j \leqslant M} \left(t_{i,j} \leftrightarrow y_{i,l_1,j} \vee y_{i,l_2,j} \vee \ldots \vee y_{i,l_L,j}\right) \tag{1}$$

$$\bigwedge_{1 \leqslant i < j \leqslant M} \left(p_{j,i} \leftrightarrow t_{i,j} \wedge \neg t_{i-1,j} \wedge \neg t_{i-2,j} \wedge \ldots \wedge \neg t_{1,j}\right) \tag{2}$$

$$\bigwedge_{2 \leqslant j \leqslant M} \left(p_{j,1} \vee p_{j,2} \vee \ldots \vee p_{j,j-1}\right) \tag{3}$$

$$\bigwedge_{1 \leqslant k < i < j \leqslant M} \left(p_{j,i} \rightarrow \neg p_{j+1,k}\right) \tag{4}$$

$$\bigwedge_{1 \leqslant i < j \leqslant M} \bigwedge_{1 \leqslant n \leqslant L} \left(m_{i,l_n,j} \leftrightarrow y_{i,l_n,j} \wedge \neg y_{i,l_{n-1},j} \wedge \neg y_{i,l_{n-2},j} \wedge \ldots \wedge \neg y_{i,l_1,j}\right) \tag{5}$$

$$\bigwedge_{1 \leqslant i < j \leqslant M} \bigwedge_{1 \leqslant k < n \leqslant L} \left(p_{j,i} \wedge p_{j+1,i} \wedge m_{i,l_n,j} \rightarrow \neg m_{i,l_k,j+1}\right) \tag{6}$$

Изучив формулы, приведенные выше, можно заключить, что:

а) формула (1) содержит $\mathcal{O}\left(M^2 \times L\right)$ дизъюнктов;

б) формула (2) содержит $\mathcal{O}\left(M^3\right)$ дизъюнктов;

в) формула (3) содержит $\mathcal{O}\left(M\right)$ дизъюнктов;

г) формула (4) содержит $\mathcal{O}\left(M^3\right)$ дизъюнктов;

д) формула (5) содержит $\mathcal{O}\left(M^2 \times L^2\right)$ дизъюнктов;

е) формула (6) содержит $\mathcal{O}\left(M^2 \times L^2\right)$ дизъюнктов.

Таким образом, будет предложено новое кодирование для свойств, ранее выраженных формулами (2), (3), (5) и (6).

Прежде всего, заметим, что формула (3) задает свойство, что у каждого состояния d_j (кроме начального) автомата $\mathcal{D}$ существует как минимум один родитель в дереве BFS, при этом с меньшим номером. Однако, надо заметить, что по определению в любом дереве у любой вершины кроме корня существует ровно один родитель. Тогда, можно добавить ограничение, задающее свойство, что у каждого состояния d_j (кроме начального) автомата $\mathcal{D}$ существует не более одного родителя в дереве BFS, при этом с меньшим номером. В совокупности два данных ограничения зададут вышеупомянутое свойство об единственности родителя. Заметим, что ограничение, задающее свойство, что не более чем одна из M переменных истинна, может быть выражено через $\mathcal{O}\left(M^2\right)$ или $\mathcal{O}\left(M \times \log M\right)$ дизъюнктов, что укладывается в целевой размер формулы $\mathcal{O}\left(M^2 \times L\right)$ дизъюнктов. [Илья: *либо написать как закодировать прямо тут, либо сослаться на раздел, где про это напишу*].

Фактически, новое ограничение можно записать следующим образом.

$$\bigwedge_{1<j\leqslant M} \sum_{i=1}^{j-1} p_{j,i} = 1 \tag{7}$$

2.2 Определение родительских переменных

Формула

$$\bigwedge_{1 \leqslant i < j \leqslant M} (p_{j,i} \leftrightarrow t_{i,j} \wedge \neg t_{i-1,j} \wedge \neg t_{i-2,j} \wedge \ldots \wedge \neg t_{1,j})$$

при преобразовании в КНФ выражается через $\mathcal{O}\left(M^3\right)$ дизъюнктов, так как обе переменные i и j имеют область допустимых значений размера M, а также правая часть формулы имеет длину $O\left(M\right)$. Так как переменные i и j независимы,то, чтобы сократить количество дизъюнктов, нужно сократить правую часть формулы. Для этого предлагается ввести новые булевы переменные $\{ft_{i,j}\}_{0 \leqslant i < j \leqslant M}$. Переменная $ft_{i,j}$ истинна тогда и только тогда, когда все переменные $t_{k,j}$, где $1 \leqslant k \leqslant i$, ложны (**f**alse ***t*** variables — ложные переменные t). Иными словами, $ft_{i,j} \leftrightarrow \neg t_{i,j} \wedge \neg t_{i-1,j} \wedge \ldots \neg t_{1,j}$. Для $i = 0$ в явном виде определим, что $ft_{0,j} = 1$ для любого j. Стоит, однако, заметить, что определение переменных $ft_{i,j}$, приведенное выше, требует также $\mathcal{O}\left(M^3\right)$ дизъюнктов, что не решает изначальную проблему.

При этом, можно заметить, что значение переменной $ft_{i,j}$ зависит от значения всех переменных $ft_{k,j}$, где $k < i$. Тогда, можно определить переменные $ft_{i,j}$ рекурсивно:

$$ft_{i,j} \leftrightarrow \begin{cases} 1 & i = 0, 1 \leqslant j \leqslant M \\ ft_{i-1,j} \wedge \neg t_{i,j} & 1 \leqslant i < j \leqslant M \end{cases} \tag{8}$$

При преобразовании в КНФ формула (8) будет состоять из $\mathcal{O}\left(M^2\right)$ дизъюнктов.

Используя новые переменные $ft_{i,j}$ формулу (2) можно переписать следующим образом.

$$\bigwedge_{1 \leqslant i < j \leqslant M} \left(p_{j,i} \leftrightarrow t_{i,j} \wedge ft_{i-1,j}\right) \tag{9}$$

В данной формуле решена проблема длинной правой части и общее число дизъюнктов, требуемых для кодирования переменных $p_{j,i}$, также $\mathcal{O}\left(M^2\right)$.

2.3 Порядок детей с помощью родительных переменных

[Илья: *БЛЯ! Мы по поводу этого раздела много выясняли, спорили с Жуаном, как правильно тут все устроено и т.д. Потому что выглядит все реально сложно. И только сейчас я понял, что это можно сделать ровно также как в разделе 2.5*]

[Илья: *Переписать этот раздел*] [Илья: *И, вероятно, всю главу тогда можно единообразно переписать*]

Формула

$$\bigwedge_{1 \leqslant k < i < j \leqslant M} (p_{j,i} \to \neg p_{j+1,k})$$

при преобразовании в КНФ выражается через $\mathcal{O}\left(M^3\right)$ дизъюнктов, так как все три переменные i, j и k имеют домен размера M. Переменные i и j являются независимыми, поэтому сократить размер всей формулы можно только избавившись от переменной k.

Фактически, данная формула задает ограничение, что родитель состояния d_{j+1} автомата $\mathcal{D}$ имеет номер не меньший, чем номер родителя состояния d_j. Также можно заметить, что учитывая ограничение, заданное формулой (7), двоичное число $\mathbf{p_j} = \overline{p_{j,1}p_{j,2}\dots p_{j,j-1}}$ состоит из $j-2$ нулей и одной единицы. Тогда рассматриваемая формула говорит, что в числе $\mathbf{p_{j+1}}$ единственная единица стоит не левее чем в векторе $\mathbf{p_j}$ (и наоборот, что в числе $\mathbf{p_j}$ единственная единица стоит не правее чем в векторе $\mathbf{p_{j+1}}$). Для удобства сравнения данных чисел, рассматривается расширенное число $\tilde{\mathbf{p}}_\mathbf{j} = \overline{p_{j,1}p_{j,2}\dots p_{j,j-1}0}$. В контексте имеющейся формулы расширенное число не отличается от обычного, так как в нем все еще содержится ровно одна единица на том же месте, что и раньше (считая, слева), но теперь двоичные числа $\tilde{\mathbf{p}}_\mathbf{j}$ и $\mathbf{p_{j+1}}$ имеют одинаковое число цифр. Тогда исходная формула фактически задает ограничение $\tilde{\mathbf{p}}_\mathbf{j} \geqslant \mathbf{p_{j+1}}$. С помощью $\mathbf{p_j}^\mathbf{i}$ далее в данном разделе будет обозначаться двоичное число, являющееся суффиксом числа $\mathbf{p_j}$, начинающимся с i-ой цифры и заканчивая последней: $\mathbf{p_j}^\mathbf{i} = \overline{p_{j,i}p_{j,i+1}\dots p_{j,j-1}}$. Аналогично, $\tilde{\mathbf{p}}_\mathbf{j}^\mathbf{i} = \overline{p_{j,i}p_{j,i+1}\dots p_{j,j}}$.

Для сравнения необходимо ввести новые булевы переменные $\{geq_{j,i}\}_{1<j\leqslant M,1\leqslant i\leqslant j+1}$. Переменная $\{geq_{j,i}\}$ истинна тогда и только тогда, когда число $\tilde{\mathbf{p}_\mathbf{j}}^{\mathbf{i}}$ больше или равно (**g**reater or **eq**ual) чем число $\mathbf{p_{j+1}}^{\mathbf{i}}$, а значит и единица во втором числе находится не левее чем в первом. Определить переменные $geq_{j,i}$ можно рекурсивно следующим образом:

$$geq_{j,i} \leftrightarrow \begin{cases} 1 & i = j+1, 1 \leqslant j \leqslant M \\ geq_{j,i+1} \wedge (p_{j,i} \leftrightarrow p_{j+1,i}) \vee p_{j,i} \wedge \neg p_{j+1,i} & 1 \leqslant i \leqslant j \leqslant M \end{cases} \tag{10}$$

Ограничение $geq_{j,j+1} = 1$ задает исходное значение для рекурсивного определения переменных $geq_{j,i}$. Далее, число $\tilde{\mathbf{p}_\mathbf{j}}^{\mathbf{i}}$ больше или равно числа $\mathbf{p_{j+1}}^{\mathbf{i}}$, то есть $geq_{j,i} = 1$, если i-ый бит чисел $\tilde{\mathbf{p}_\mathbf{j}}$ и $\mathbf{p_{j+1}}$ совпадает, а для суффиксов $\tilde{\mathbf{p}_\mathbf{j}}^{\mathbf{i+1}}$ и $\mathbf{p_{j+1}}^{\mathbf{i+1}}$ верно, что первый больше либо равен второго, или если i-ый бит числа $\tilde{\mathbf{p}_\mathbf{j}}$ равен единице, а числа $\mathbf{p_{j+1}}$ — нулю. Последнее верно, так как оба числа содержат по одной единице и вне зависимости от того, где находится единица во втором числе, правее или левее, число $\tilde{\mathbf{p}_\mathbf{j}}^{\mathbf{i}}$ строго больше числа $\mathbf{p_{j+1}}^{\mathbf{i}}$.

Для упрощения записи и уменьшения размера дизъюнктов можно ввести еще одни вспомогательные переменные $\{peq_{j,i}\}_{1\leqslant i\leqslant j\leqslant M}$. Переменная $peq_{j,i}$ истинна тогда и только тогда, когда $p_{j,i} = p_{j+1,i}$.

$$peq_{j,i} \leftrightarrow (p_{j,i} \leftrightarrow p_{j+1,i}) \tag{11}$$

Тогда формула (10) примет окончательный вид:

$$geq_{j,i} \leftrightarrow \begin{cases} 1 & i = j+1, 1 \leqslant j \leqslant M \\ geq_{j,i+1} \wedge peq_{j,i} \vee p_{j,i} \wedge \neg p_{j+1,i} & 1 \leqslant i \leqslant j \leqslant M \end{cases} \tag{12}$$

При преобразовании в КНФ формула (12) будет состоять из $\mathcal{O}\left(M^2\right)$ дизъюнктов.

Используя новые переменные $geq_{i,j}$ формулу (4) можно переписать следующим образом.

$$\bigwedge_{1<j\leqslant M} geq_{j,1} \tag{13}$$

Действительно, $geq_{j,1} = 1 \Leftrightarrow \tilde{\mathbf{p}_\mathbf{j}} = \tilde{\mathbf{p}_\mathbf{j}}^{\mathbf{1}} \geqslant \mathbf{p_{j+1}}^{\mathbf{1}} = \mathbf{p_{j+1}}$.

Таким образом, формула (9) выражается через $\mathcal{O}(M)$ дизъюнктов, а формулы (11) и (12) — через $\mathcal{O}\left(M^2\right)$ дизъюнктов.

2.4 Определение переменных минимального символа

Формула

$$\bigwedge_{1 \leqslant i < j \leqslant M} \bigwedge_{1 \leqslant n \leqslant L} \left(m_{i,l_n,j} \leftrightarrow y_{i,l_n,j} \wedge \neg y_{i,l_{n-1},j} \wedge \neg y_{i,l_{n-2},j} \wedge \ldots \wedge \neg y_{i,l_1,j}\right)$$

при преобразовании в КНФ выражается через $\mathcal{O}\left(M^2 \times L^2\right)$ дизъюнктов, так как обе переменные i и j имеют область допустимых значений размера M, переменная n — размера L, а также правая часть формулы имеет длину $O(L)$. Так как переменные i, j и n независимы,то, чтобы сократить количество дизъюнктов, нужно сократить правую часть формулы. Можно заметить, что данная формула по своей структуре аналогична той, что рассматривалась в разделе 2.2. Тогда аналогично можно ввести новые булевы переменные $\{fy_{i,l_n,j}\}_{0 \leqslant i < j \leqslant M, 0 \leqslant n \leqslant M}$. Переменная $fy_{i,l_n,j}$ истинна тогда и только тогда, когда все переменные $y_{i,l_k,j}$, где $1 \leqslant k \leqslant n$, ложны (**f**alse $\boldsymbol{y}$ variables — ложные переменные y). Иными словами, $fy_{i,l_n,j} \leftrightarrow \neg y_{i,l_n,j} \wedge \neg y_{i,l_{n-1},j} \wedge \ldots \neg y_{i,l_1,j}$. Для $n = 0$ в явном виде определим, что $fy_{i,l_0,j} = 1$ для любых i, j. Далее, аналогично тому, как это было сделано в разделе 2.2, определим переменные $fy_{i,l_n,j}$ рекурсивно.

$$fy_{i,l_n,j} \leftrightarrow \begin{cases} 1 & 1 \leqslant i < j \leqslant M, n = 0 \\ fy_{i,l_{n-1},j} \wedge \neg y_{i,l_n,j} & 1 \leqslant i < j \leqslant M, 1 \leqslant n \leqslant L \end{cases} \tag{14}$$

При преобразовании в КНФ формула (14) будет состоять из $\mathcal{O}\left(M^2 \times L\right)$ дизъюнктов.

Используя новые переменные $fy_{i,l_n,j}$ формулу (5) можно переписать следующим образом.

$$\bigwedge_{1\leqslant i<j\leqslant M}\bigwedge_{1\leqslant n\leqslant L}\left(m_{i,l_n,j}\leftrightarrow y_{i,l_n,j}\wedge fy_{i,l_{n-1},j}\right) \tag{15}$$

В данной формуле решена проблема длинной правой части и общее число дизъюнктов, требуемых для кодирования переменных $m_{i,l_n,j}$, также $\mathcal{O}\left(M^2\times L\right)$.

2.5 Порядок детей одного родителя

Формула

$$\bigwedge_{1\leqslant i<j\leqslant M}\bigwedge_{1\leqslant k<n\leqslant L}(p_{j,i}\wedge p_{j+1,i}\wedge m_{i,l_n,j}\rightarrow\neg m_{i,l_k,j+1})$$

при преобразовании в КНФ выражается также через $\mathcal{O}\left(M^2\times L^2\right)$ дизъюнктов, так как обе переменные i и j имеют область допустимых значений размера M, а обе переменные n и k — размера L. Переменные i,j и n являются независимыми, поэтому сократить размер всей формулы можно только избавившись от переменной k. Можно заметить, что данная формула по своей структуре аналогична той, что рассматривалась в разделе 2.3. Тогда аналогично можно ввести новые булевы переменные $\{fm_{i,l_n,j}\}_{0\leqslant i<j\leqslant M,0\leqslant n\leqslant M}$. Переменная $fm_{i,l_n,j}$ истинна тогда и только тогда, когда все переменные $m_{i,l_k,j}$, где $1\leqslant k\leqslant n$, ложны (**f**alse $\boldsymbol{m}$ variables — ложные переменные m). Иными словами, $fm_{i,l_n,j}\leftrightarrow\neg m_{i,l_n,j}\wedge\neg m_{i,l_{n-1},j}\wedge\ldots\neg m_{i,l_1,j}$. Для $n=0$ в явном виде определим, что $fm_{i,l_0,j}=1$ для любых i,j. Далее, аналогично тому, как это было сделано в разделе 2.3, определим переменные $fm_{i,l_n,j}$ рекурсивно.

$$fm_{i,l_n,j}\leftrightarrow\begin{cases}1 & 1\leqslant i<j\leqslant M, n=0\\ fm_{i,l_{n-1},j}\wedge\neg m_{i,l_n,j} & 1\leqslant i<j\leqslant M, 1\leqslant n\leqslant L\end{cases} \tag{16}$$

При преобразовании в КНФ формула (16) будет состоять из $\mathcal{O}\left(M^2 \times L\right)$ дизъюнктов.

Используя новые переменные $fm_{i,l_n,j}$ формулу (6) можно переписать следующим образом.

$$\bigwedge_{1 \leqslant i < j < M} \bigwedge_{1 \leqslant n \leqslant L} \left(p_{j,i} \wedge p_{j+1,i} \wedge m_{i,l_n,j} \rightarrow \neg fm_{i,l_{n-1},j+1} \right) \tag{17}$$

Таким образом общее число дизъюнктов, требуемых для ограничения порядка детей одного состояния, равняется $\mathcal{O}\left(M^2 \times L\right)$.

Глава 3 Подходы к сокращению пространства поиска, основанные на структурных особенностях автомата

В данной главе предлагаются новые методы по сокращению пространства поиска в задаче генерации ДКА минимального размера по заданным словарям. Данные методы не являются необходимыми для нахождения соответствующего автомата, но помогают сделать это быстрее. В основе предлагаемых методов лежит использование структурных особенностей BFS пронумерованного ДКА, а также связь между расширенным префиксным деревом и ДКА.

3.1 Полное дерево обхода в ширину

[Илья: *рисунок дерева*]

На рисунке [Илья: *ссылка*] показано полное BFS дерево, построенное по некоторому автомату. Данное дерево является полным, так как у каждой его внутренней вершины имеется по L детей. Тогда данное дерево показывает максимально возможные номера, которые могут быть у детей некоторого состояния d_i. Действительно, нельзя добавить в данное дерево новые вершины, которые будут иметь номер между i и $i \cdot L + 1$, так как все возможные позиции заняты. В то же время, если удалить какие-то из вершин правее или ниже вершины d_i, то номера детей могут только уменьшиться. Далее будут представлены дополнительные ограничения, которые следуют из рисунка [Илья: *ссылка*].

Сокращение области определения родительских переменных. У некоторого состояния d_i, где $1 \leqslant i < M$, детьми в BFS дереве могут быть только состояния с номерами от $i + 1$ до $\min(i \cdot L + 1, M)$. Так как в BFS дереве номер

ребенка всегда больше номера родителя, то нижняя граница тривиальна. Рисунок [Илья: *ссылка*] иллюстрирует обоснование верхней границы. Действитель-

но, можно доказать по индукции, что состояния на k-ом уровне имеют номера от $\sum_{r=0}^{k-1} L^r + 1$ до $\sum_{r=0}^{k} L^r$. База индукции при $k = 0$, очевидно, верна. Если для некоторого слоя k утверждение выше верно, то для слоя $k+1$ верно, что нумерация состояний на нем начинается с $\sum_{r=0}^{k-1} L^r + 1$, а всего вершин $\left(\sum_{r=0}^{k} L^r - \left(\sum_{r=0}^{k-1} L^r + 1\right) + 1\right) \cdot L = L^k * L = L^{k+1}$, из чего следует, что последнее состояние имеет номер $\sum_{r=0}^{k} L^r + L^{k+1} = \sum_{r=0}^{k+1} L^r$.

[Илья: *возможно, оформить в виде полноценной теоремы и доказательства.*]

Выразить данное свойство можно, либо определив переменные для соответствующей области определения — $\{p_{j,i}\}_{1 \leqslant i < j \leqslant min(i \cdot L+1, M)}$, либо в явном виде указав, что $p_{j,i} = 0$ при $j > i \cdot L + 1$.

Сокращение области определения переменных перехода и переменных наличия переходов. Помимо закономерностей между номерами родителей и детей в BFS пронумерованном автомате, можно заметить более общую закономерность относительно переходов. Из состояния d_i в BFS пронумерованном автомате не может в принципе существовать перехода в состояние d_j если $j > i \cdot L + 1$. Действительно, из доказанного в предыдущей секции следует, что у состояния d_j родителем должно быть состояние d_k, где $k > i$. Но, если существует из состояния d_i существует переход в состояние d_j, то по принципу BFS обхода родителем состояния d_j должно быть состояние d_k, где $k \leqslant i$. Получившееся противоречие доказывает исходное утверждение. Таким образом, можно сделать заключение, что $y_{i,l,j} = 0$ при $j > i \cdot L + 1; l \in \Sigma$.

Как следствие, по определению переменных наличия переходов верно, что $t_{i,j} = 0$ при $j > i \cdot L + 1$.

[Илья: $y_{i,l,iL+2-j}$ *— пока скипнул, может добавить про них*]

3.2 Свойство непрерывности родительских переменных

Помимо того, что у каждого состояния d_i автомата $\mathcal{D}$ детьми могут быть состояния с номерами от $i+1$ до $i \cdot L + 1$, можно утверждать, что состояние d_i может быть родителем не более чем L состояний, которые при этом пронумерованны последовательно. Количество детей ограничено размером алфавита, так как рассматриваемый автомат является детерминированным. Последовательная нумерация следует из структуры алгоритма BFS — дети некоторого состояния поочередно добавляются в очередь и им присваиваются последовательные номера. Данное свойство можно назвать *свойством непрерывности*. Для булевого кодирования предикатов нарушения симметрии данное свойство означает, что для фиксированного i переменные $p_{j,i}$ ложны для всех j, кроме некоторого отрезка $[j_0, \ldots, j_s]$, где $1 \leqslant j_0 \leqslant j_s \leqslant M, s \leqslant L$.

Можно добавить дополнительные ограничения, задающие данное свойство, которые дополнительно ограничат пространство поиска. Для этого необходимо ввести два дополнительных множества булевых переменных — $\{lnp_{j,i}\}_{1 \leqslant i < j \leqslant M}$ и $\{rnp_{j,i}\}_{1 \leqslant i < j \leqslant M}$.

Переменная $lnp_{j,i}$ истинна тогда, когда переменная $p_{j,i} = 0$ и $j < j_0$. Иными словами, данная переменная истинна в случае, когда j находится левее отрезка истинных родительских переменных (**l**eft **n**o **p**arent). [Илья: *рисунок*]. Определе-

лить на языке выполнимости булевых формул данные переменные можно следующим образом. Формула

$$\bigwedge_{1 \leqslant i < j \leqslant M} \neg p_{j,i} \wedge p_{j+1,i} \to lnp_{j,i}$$

задает пограничное истинное значение переменных $lnp_{j,i}$. Далее, необходимо добавить формулу

$$\bigwedge_{1 \leqslant i < M, i+1 < j \leqslant M} lnp_{j,i} \to lnp_{j-1,i},$$

которая задает значения переменных $lnp_{j,i}$ левее пограничного. Как следствие из определения переменных $lnp_{j,i}$, можно добавить следующую формулу:

$$\bigwedge_{1 \leqslant i < j \leqslant M} lnp_{j,i} \to \neg p_{j,i}.$$

Таким образом, переменные $lnp_{j,i}$ для каждого i истинны начиная с $j = 1$ и до тех пор, пока $p_{j+1,i}$ не будет истинно. Можно заметить, что начиная с момента, когда $p_{j,i}$ истинно, значение переменных $lnp_{j,i}$ не определено, что, как будет показано далее, не играет никакой роли.

Аналогичным образом определяются переменные $rnp_{j,i}$. Переменная $rnp_{j,i}$ истина тогда, когда переменная $p_{j,i} = 0$ и $j > j_s$, то есть когда j находится правее отрезка истинных родительских переменных (**r**ight **n**o **p**arent). Пограничное истинное значение переменных $rnp_{j,i}$ задается с помощью формулы

$$\bigwedge_{1 \leqslant i < j \leqslant M} p_{j-1,i} \wedge \neg p_{j,i} \to rnp_{j,i}.$$

Значение переменных правее пограничного задаются аналогично предыдущему случаю:

$$\bigwedge_{1 \leqslant i < j < M} rnp_{j,i} \to rnp_{j+1,i}.$$

Как и в случае с переменными $lnp_{j,i}$, можно добавить формулу

$$\bigwedge_{1 \leqslant i < j \leqslant M} rnp_{j,i} \to \neg p_{j,i}.$$

Переменные $rnp_{j,i}$ для каждого i истинны начиная с $j = M$ и в порядке убывания истинны до тех пор, пока $p_{j-1,i}$ не будет истинно. Можно заметить, что, аналогично, начиная с $j = 1$, и до тех пор, пока $p_{j,i}$ не станет ложной после серии истинных значений, значение переменных $rnp_{j,i}$ не определено. Помимо этого, можно добавить следующую формулу:

$$\bigwedge_{1 \leqslant i < j \leqslant M, l \in \Sigma} rnp_{j,i} \to \neg y_{i,l,j}.$$

Действительно, если состояние d_i имеет детей с номерами $j_0, \ldots, j_s$, то из состояния d_i не может быть переходов состояния с номерами бо́льшими чем j_s, иначе данные состояния были бы также детьми состояния d_i.

Дополнительно, из того, что d_i может иметь не более чем L детей, следует, что

$$\bigwedge_{1 \leqslant i < M; i+L < j \leqslant M-L} p_{j,i} \to \mathit{lnp}_{j-L,i}$$

и что

$$\bigwedge_{1 \leqslant i < j \leqslant M-L} p_{j,i} \to \mathit{rnp}_{j+L,i}.$$

Переменные $\mathit{lnp}_{j,i}$ и $\mathit{rnp}_{j,i}$ помогают задать некоторым переменным $p_{j,i}$ ложное значение. Однако, исходя из их значения, можно некоторым переменным $p_{j,i}$ задать истинное значение. Так, если для некоторых $j_1 < j_2$ верно, что $\mathit{lnp}_{j_1,i}$ и $\mathit{rnp}_{j_2,i}$ ложны, то для всех j' таких, что $j_1 \leqslant j' \leqslant j_2$ верно, что $p_{j',i}$ истинна. Формально,

$$\bigwedge_{1 \leqslant i < M; i < j_1 \leqslant j' \leqslant j_2 \leqslant \min(j_1+L-1, M)} \neg \mathit{lnp}_{j_1,i} \wedge \neg \mathit{rnp}_{j_2,i} \to p_{j',i}.$$

Также, учитывая, что дети некоторого состояния d_i пронумерованны последовательно, можно добавить следующее ограничение:

$$\bigwedge_{1 \leqslant i < j < k \leqslant \min(j+L-1, M)} p_{j,i} \wedge p_{k,i} \to p_{k-1,i}.$$

3.3 Минимальное расстояние в дереве обхода автомата в ширину

Еще одним следствием анализа полного BFS дерева [Илья: *рисунок*], является ограничение минимального расстояния от стартового состояния автомата $\mathcal{D}$ до всех других. Не сложно заметить, что в полном BFS дереве, представленном на [Илья: *рисунке*], глубина некоторого состояния d_j минимальна. Действительно, в неполном BFS дереве на каждой глубине состояний не больше чем в полном дереве, а значит состояние может находится или на том же уровне, или глубже. Тогда глубина состояния в полном BFS дереве будет являться оценкой снизу для глубины состояния в случайном дереве.

Для доказательства можно воспользоваться ранее доказанным фактом, что на уровне k в полном BFS дереве находятся состояния с номерами от $\left(\sum_{i=0}^{k-1}+1\right)$ до $\left(\sum_{i=0}^{k}\right)$. Иными словами, номера состояний на уровне k находятся в полуоткрытом интервале $\left(\sum_{i=0}^{k-1} L^i; \sum_{i=0}^{k} L^i\right]$. Если домножить левую и правую границы интервала на $(L-1)$ и прибавить единицу, то получится интервал $\left(L^k; L^{k+1}\right]$. Теперь, если взять логарифм по основанию L от обеих границ и вычесть единицу, то получится, интервал $(k-1; k]$. Из этого можно заключить, что минимальная глубина состояния с номером j, а значит и минимальное расстояние от стартового состояния до него, равняется $D_{\min}(j) = \lceil \log_L (j \cdot (L-1)+1) - 1 \rceil$.

Таким образом, для любого состояния d_j автомата $\mathcal{D}$ минимальное расстояние от стартового состояния d_1 до d_j не меньше, чем $D_{\min}(j)$. Тогда, если расстояние от корня t_1 префиксного дерева $\mathcal{T}$ до некоторой вершины t_v меньше, чем минимально возможное расстояние до состояния d_j автомата $\mathcal{D}$: $\Delta(v) < D_{\min}(j)$, то можно утверждать, что вершина t_v не может соответствовать состоянию d_j, то есть $x_{v,j} = 0$.

3.4 Дополнительные ограничения, основанные на связи между префиксным деревом и автоматом

Глава 4 Генерация всех детерминированных конечных автоматов по примерам поведения

В данной главе рассматривается задача генерации всех неизоморфных детерминированных конечных автоматов минимального размера, соответствующих заданным примерам поведения, которая ранее не имела эффективного решения. Разрабатывается метод, основанный на подходе к решению задачи генерации одного ДКА с использованием программных средстве решения SAT, подробно описанный в разделе 1.4. Рассматриваются два варианта использования программных средств решения SAT: перезапуск неинкрементального средства после нахождения каждого автомата и использование инкрементального средства — если такое средство находит некоторое решение, то оно сохраняет свое текущее состояние и может принимать новые дизъюнкты. Подробнее про инкрементальное решение SAT можно прочитать вТакже в данной главе предлагается переборный метод, который служит базовым для сравнения с методом, основанным на сведении к задаче выполнимости.

4.1 Метод, основанный на сведении к задаче выполнимости

Как было описано в разделе 1.5.2, для каждого ДКА с M состояниями существует $\mathcal{O}(M!)$ изоморфных ему автоматов. Основной идеей предлагаемого метода по генерации всех ДКА является запрет (блокирование) найденных удовлетворяющих подстановок. Очевидно, что без использования предикатов нарушения симметрии, придется блокировать $\mathcal{O}(M)$ изоморфных автоматов для каждого найденного уникального ДКА, что даже при $M = 10$ требует значительных вычислительных затрат. Так как подход к нарушению симметрии с помощью большой клики, описанный в разделе 1.5.3, позволяет зафиксировать только ну-

мерацию k состояний, то метод, предложенный в [1] находит $(C-k)!$ изоморфных автоматов, что в общем случае все еще вычислительно сложно.

Предикаты же нарушения симметрии на основе обхода в ширину, как было описано в разделе 1.5.4, допускают единственного представителя каждого класса эквивалентности по изоморфизму. Конкретно, BFS предикаты допускают BFS пронумерованный автомат. Тогда, заблокировав некоторый найденный автомат, автоматически блокируется весь класс эквивалентности, а значит программное средство может найти другой автомат неизоморфный ранее найденным. Необходимо заметить, что несмотря на то, что идея блокировать найденные решения с целью найти остальные достаточно проста и известна, применять ее на практике невозможно без эффективных предикатов нарушения симметрии. Так, ранее не было описано способов бороться с факториальным числом изоморфных автоматов, а значит и рассматриваемая задача не имела эффективного решения.

Список литературы

1. *Heule*, *M.*, *Verwer*, *S.* Exact DFA Identification Using SAT Solvers // Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium, ICGI 2010, Valencia, Spain, September 13-16, 2010. Proceedings. Vol. 6339. — Springer, 2010. — P. 66–79. — (Lecture Notes in Computer Science). — URL: https://doi.org/10.1007/978-3-642-15488-1%5C_7.
2. *Ulyantsev*, *V.*, *Zakirzyanov*, *I.*, *Shalyto*, *A.* BFS-Based Symmetry Breaking Predicates for DFA Identification // Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Nice, France, March 2-6, 2015, Proceedings. Vol. 8977. — Springer, 2015. — P. 611–622. — (Lecture Notes in Computer Science).