Московский государственный университет имени М.В. Ломоносова
Университет МГУ-ППИ в Шэньчжэне
Факультет вычислительной математики и кибернетики

Направление подготовки 01.03.02 «Прикладная математика и информатика»

Выпускная квалификационная работа

Полурешетка инверсных морфизмов для заданного конечного языка: построение и исследование

Работу выполнила
студентка **Сюй Юэлинь**

Научный руководитель:
д.ф.-м.н., профессор, **Мельников Борис Феликсович**

Шэньчжэнь
2023

Содержание

Аннотация

Данный документ нацелен на исследование аспектов формальных языков и теории автоматов, при этом основное внимание сосредоточено на построении и анализе полурешеток обратных морфизмов.

Исходный конечный язык формируется на основе детерминированного конечного автомата (ДКА), который затем используется для создания регулярного выражения. Этот процесс подробно описывается с выделением его ключевых этапов включая построение исходного конечного языка, построение регулярного выражения для заданного языка и конструирование ДКА.

Центральная часть исследования занимает построение полурешетки обратных морфизмов. Здесь освещается несколько важных тем, включая влияние различных типов регулярных выражений на построение примитивных конечных языков и возможные направления дальнейших исследований.

Изучение свойств полурешетки, таких как верхняя и нижняя грани, линейный порядок, алгебраические свойства, размерность, компактность и качество представления языка, приводит к глубокому пониманию структуры и функций полурешеток.

Практическая значимость данного исследования подчеркивается в обсуждении его возможных приложений. Кроме того, работа дает оценку преимуществ использования полурешеток обратных морфизмов в контексте обработки естественных языков, что, как показано, приводит к улучшению результата в задачах автоматической коррекции текстов и, возможно, генерации автоматических резюме.

Исследование открывает новые возможности для дальнейшего изучения взаимодействия полурешеток обратных морфизмов с другими областями теории автоматов и формальных языков, при этом особый акцент можно поставить на теорию грамматик, алгебру регулярных выражений и теорию автоматов.

Ключевые слова: формальные языки, автоматы, полурешетка, инверсные морфизмы, детерминированный конечный автомат, конкатенация, транзитивное замыкание, теория грамматик, алгебра регулярных выражений, теория автоматов.

Введение

Актуальность темы. Формальные языки и теория автоматов являются ключевыми дисциплинами в теоретической информатике, обладающими широкими возможностями применения. Настоящая работа сосредоточена на задаче создания и исследования полурешетки обратных морфизмов для заданного конечного языка. Подчеркивается несколько важных моментов:

- **Применимость:** Формальные языки и теория автоматов используются в таких областях, как компьютерное обучение, анализ алгоритмов, разработка и компиляция программного обеспечения.
- **Улучшение представления данных:** Исследование структуры и упорядоченности языка может облегчить разработку эффективных методов представления и обработки данных.
- **Эффективность обработки:** Анализ компактности полурешетки может дать представление о том, насколько эффективно язык может быть представлен и обработан.

Таким образом, исследование полурешетки обратных морфизмов имеет практическую значимость для области формальных языков и теории автоматов.

Обзор литературы. Полурешетка обратных морфизмов для заданного конечного языка была изучена мало. В ходе обзора литературы основной акцент был сделан на значимости компактности полурешетки при анализе упорядоченности и структуры языка с целью определения оптимальных методов представления и обработки данных. Было выяснено также, что адаптивность и гибкость представления, в сочетании с интерпретируемостью структуры полурешетки, играют важную роль в обобщении и продолжении исследования свойств языка.

Цель работы. Целью данного исследования является построение и анализ полурешетки обратных морфизмов для конкретного конечного языка, что обеспечит более глубокое понимание теории формальных языков и автоматов. Работа нацелена на сравнительный анализ различных методов представления языков, включая применение полурешеток, для определения их преимуществ и недостатков. Исследование проводится с использованием эффективных алгоритмов для построения и анализа языков, а также с применением инструментов алгебры регулярных выражений и формальных языков. Результаты работы позволят оценить качество представления языков в полурешетках, что имеет критическую важность при решении задач обра-

ботки естественного языка, таких как анализ текста, машинный перевод и генерация текста. Кроме того, это исследование может быть полезным для интеграции полурешеток обратных морфизмов с другими областями теории автоматов и формальных языков, включая теорию грамматик, открывая тем самым новые возможности в области искусственного интеллекта и компьютерной лингвистики.

Исследовательские методы. В ходе исследования применялись разнообразные методы теоретического анализа и компьютерные эксперименты, включая создание и анализ детерминированных конечных автоматов, построение полурешеток обратных морфизмов, исследование свойств этих полурешеток и реализацию соответствующих алгоритмов. Исследовательские методы, описанные в этой работе, нацелены на построение и анализ свойств полурешетки обратных морфизмов для заданного конечного языка. Результаты исследования включают определение размерности и компактности полурешетки, оценку качества представления языка и разработку эффективных алгоритмов для построения и анализа конечных автоматов и регулярных выражений. Особое внимание уделяется воздействию различных типов регулярных выражений на построение и анализ конечных языков, а также на оценку упорядоченности и структуры конечных языков. В частности, рассматриваются методы формирования морфизмов, положительного замыкания, генерации языка MP+ и определения всех и обратимых отображений. В перспективе, развитие этой работы может быть направлено на создание новых методов представления и обработки данных на основе полурешетки обратных морфизмов.

Практическая значимость. Практическую значимость данного исследования определяет возможность оценивать упорядоченность и структуру языка, что представляет большую ценность для выявления оптимальных методов представления и обработки данных. Более того, понимание компактности полурешетки может помочь понять, насколько эффективно язык может быть представлен и обработан. Кроме того, анализ свойств полурешеток способствует разработке эффективных алгоритмов для построения и анализа языков. Подход, представленный в этой работе, имеет значительную практическую значимость, так как он может быть применен для оптимизации процессов обучения, создания программного обеспечения, компиляции и других процессов в области информатики.

Объем и структура работы. Данная исследовательская работа осуществляет глубокий анализ полурешетки обратных морфизмов для определенного конечного языка. Основное внимание в работе уделяется изучению структуры и компактности полурешетки, являющихся критическими параметрами для определения наиболее

эффективных способов представления и обработки данных в контексте заданного языка.

Размерность полурешетки характеризует количество элементов, которые она содержит. Высокая размерность обычно указывает на сложность структуры языка. Изучение свойств полурешетки позволяет получить представление о том, насколько эффективно язык может быть представлен и обработан. Результаты этого исследования могут послужить отправной точкой для дальнейших исследований в области формальных языков и теории автоматов.

Структура работы разделена на семь ключевых разделов: описание исходного конечного языка, построение полурешетки обратных морфизмов, исследование свойств полурешетки, реализация соответствующих алгоритмов, применение полурешетки, формулирование выводов и определение направлений для будущих исследований. Более того, в работе рассматривается возможность интеграции обратных полурешеток с другими областями теории автоматов и формальных языков.

Благодарности. Я хочу выразить свою благодарность своему научному руководителю Борису Феликсовичу Мельникову за ценные замечания и постоянное руководство, которые значительно облегчили разработку этого исследования.

Постановка задачи

В данной работе ставится задача построения и детального исследования полурешетки обратных морфизмов для заданного конечного языка. Важность этой задачи обусловлена потенциальной возможностью применения такого подхода в теории формальных языков, что позволит расширить наши знания и понимание этой важной области.

Схема работы представлена ниже:

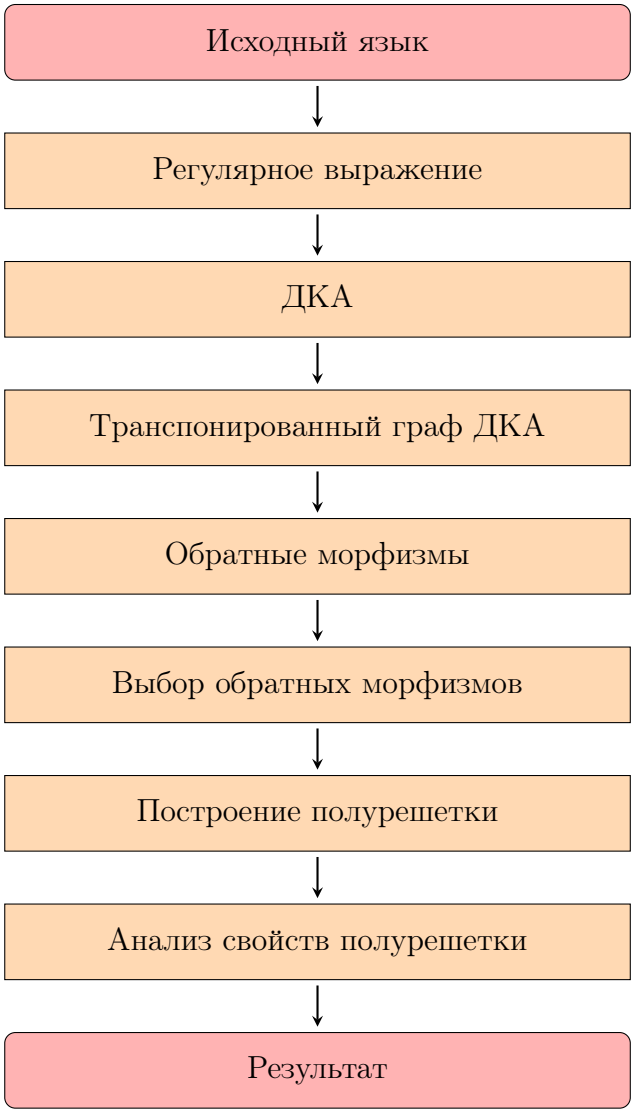

Рис. 1: Схема процесса работы

По завершении работы предполагается предложить обзор возможных направлений для будущих исследований, включая перспективы интеграции обратных полу-

решеток с другими областями теории автоматов и формальных языков, такими как теория грамматик, алгебра регулярных выражений и теория автоматов.

Этот алгоритм может послужить базой для многих направлений исследования и развития, таких как:

1. **Усовершенствование алгоритмов построения полурешетки.** Оптимизация текущих процедур и внедрение новых методов может повысить эффективность и универсальность подхода.
2. **Исследование различных типов полурешеток.** Этот подход может быть расширен и адаптирован для работы с другими типами полурешеток, включая многомерные и непрерывные полурешетки.
3. **Применение полурешеток в других областях.** Концепции и методы, изучаемые в этой работе, могут быть полезны в различных областях, включая оптимизацию, анализ данных и машинное обучение.
4. **Расширение анализа полурешеток.** Исследование дополнительных свойств и характеристик полурешеток может привести к новым открытиям и улучшить понимание этой структуры.

Все эти направления представляют собой поле для дальнейших исследований и развития в этой области.

$$f(L, M) = \begin{cases} 1, & \text{если } L \text{ может быть представлено с помощью } M \\ 0, & \text{иначе} \end{cases} \tag{1}$$

где L - язык, а M - полурешетка обратных морфизмов.

Завершающим этапом будет предложение возможных направлений для будущих исследований, включая перспективы интеграции обратных полурешеток с другими областями теории автоматов и формальных языков, такими как теория грамматик, алгебра регулярных выражений и теория автоматов.

Глава 1 Литературный обзор

Основное внимание в этой работе уделяется исследованиям, выполненным профессором Б.Ф. Мельниковым и его коллегами, сделавшими значительный вклад в развитие теории формальных языков и автоматов.

Мельников посвятил ряд работ изучению условий равенства бесконечных итераций конечных языков [1, 2, 3]. В этих работах он анализировал бесконечные деревья в алгоритмах проверки условия эквивалентности итераций конечных языков.

В других своих работах [4, 5] Мельников подробно рассматривал конечные автоматы, соответствующие бесконечным итерационным деревьям морфизмов, а также представил полиномиальный алгоритм проверки условия морфического образа расширенного максимального префиксного кода [6].

Вопросы, связанные с полурешетками подмножеств потенциальных корней в задачах теории формальных языков, также находились в фокусе внимания Мельникова [7, 8, 9]. Он подробно рассмотрел вопросы извлечения корня из языка, построения инверсных морфизмов и условий существования решетки.

Мельников, в соавторстве с Мельниковой, представил полиномиальный алгоритм построения конечного автомата для проверки равенства бесконечных итераций двух конечных языков [10].

В работе с Абрамяном [11], Мельников описал алгоритмы преобразования конечных автоматов, соответствующих бесконечным итерационным деревьям, в то время как в работе с Терентьевой [12] обсуждается построение оптимального остовного дерева как инструмент для обеспечения устойчивости сети связи.

Дополнительно, Мельников и Мельникова исследовали концепцию расширенного базисного конечного автомата [13, 14], и Мельников опубликовал монографию, посвященную регулярным языкам и недетерминированным конечным автоматам [15].

Этот обзор литературы служит основой для нашего исследования, а полученные выводы станут отправной точкой для дальнейших разделов этой работы.

Глава 2 Варианты задания исходного конечного языка

2.1 Описание исходного конечного языка

Исходный конечный язык задается над алфавитом $\Sigma = a, b, c$ и состоит из всех слов, которые начинаются и заканчиваются буквой a и содержат хотя бы одну букву b или c внутри. Другими словами, исходный язык - это множество всех строк, которые можно получить из следующего множества правил:

1. Строка, состоящая из единственного символа a;
2. Любая строка, полученная конкатенацией двух строк, принадлежащих языку и добавлением между ними одной из букв b или c.

Например, некоторые слова, принадлежащие исходному языку, включают в себя: $aba, acca, abcbcb, abbcbab$ и так далее.

Можно также использовать регулярное выражение, чтобы задать исходный язык. Он может быть записан как:

$$L = a(b + c)^* a \tag{2.1}$$

В формуле 2.1 мы определяем регулярное выражение. Таким образом, мы можем использовать любой подход, чтобы определить исходный конечный язык. Этот язык является важным примером, который может использоваться в обучении теории формальных языков и автоматов, а также в других областях, связанных с компьютерными науками и математикой.

Вот два примера языков, которые мы можем рассмотреть:

Пример 1:

Начнем с $\{0, 10, 1011, 11\}$. Закодируем 0 как 01, получим $\{01, 101, 10111, 11\}$. Закодируем 1 как 100, получим $\{0100, 1000100, 1000100100100, 100100\}$.

Пример 2:

Начнем с $\{a1, 0a, aa0\}$. Закодируем 0 как ab, получим $\{a1, aba, aaab\}$. Закодируем 1 как ba, получим $\{aba, aaab\}$. Внимание! Слово aba получено двумя способами.

2.2 Построение регулярного выражения для заданного языка

Для построения регулярного выражения для заданного языка можно воспользоваться алгоритмом, основанным на методе Нероуда-Ватсона. Для этого нужно построить минимальный детерминированный конечный автомат (ДКА), распознающий данный язык, и затем получить из него регулярное выражение с помощью метода удаления состояний.

Для построения регулярного выражения для заданного языка L необходимо выполнить следующие шаги:

1. **Найти регулярное выражение для подъязыка L_1**
   Подъязык L_1 содержит все слова из L, которые не содержат буквы b и c. Используя метод исключения, получаем:

$$L_1 = a(\Sigma - \{b, c\})^* a \tag{2.2}$$

2. **Найти регулярное выражение для подъязыка L_2**
   Подъязык L_2 содержит все слова из L, которые содержат букву b или c. Используя метод объединения, получаем:

$$L_2 = a(\Sigma - \{a\})^*((\Sigma - \{a\})^* b(\Sigma - \{a\})^* a + (\Sigma - \{a\})^* c(\Sigma - \{a\})^* a) \tag{2.3}$$

3. **Найти регулярное выражение для языка L**
   Язык L это объединение регулярных выражений для подъязыков L_1 и L_2:

$$X = a(\Sigma - \{a\})^* a, \tag{2.4a}$$
$$Y = (\Sigma - \{a\})^* b(\Sigma - \{a\})^* a, \tag{2.4b}$$
$$Z = (\Sigma - \{a\})^* c(\Sigma - \{a\})^* a, \tag{2.4c}$$
$$L_1 = a(\Sigma - \{b, c\})^* a, \tag{2.4d}$$
$$L_2 = X(Y + Z), \tag{2.4e}$$
$$L = L_1 \cup L_2. \tag{2.4f}$$

1. $X = a(\Sigma - a)^* a$ (уравнение 2.4a) - это регулярное выражение, означающее, что строка начинается с символа 'a', за которым следует любое количество (включая 0) символов, которые не являются 'a', и заканчивается символом 'a'.

2. $Y = (\Sigma - a)^* b(\Sigma - a)^* a$ (уравнение 2.4b) - это регулярное выражение, означающее, что строка начинается с любого количества символов, которые не являются 'a', за которыми следует 'b', затем следует любое количество символов, которые не являются 'a', и заканчивается символом 'a'.
3. $Z = (\Sigma - a)^* c(\Sigma - a)^* a$ (уравнение 2.4c) - это регулярное выражение, означающее, что строка начинается с любого количества символов, которые не являются 'a', за которыми следует 'c', затем следует любое количество символов, которые не являются 'a', и заканчивается символом 'a'.
4. $L_1 = a(\Sigma - b, c)^* a$ (уравнение 2.4d) - это регулярное выражение, означающее, что строка начинается с символа 'a', за которым следует любое количество (включая 0) символов, которые не являются 'b' или 'c', и заканчивается символом 'a'.
5. $L_2 = X(Y + Z)$ (уравнение 2.4e) - это регулярное выражение, которое представляет язык, состоящий из строк, соответствующих регулярному выражению X, за которыми следует строка, соответствующая регулярному выражению Y или Z.
6. $L = L_1 \cup L_2$ (уравнение 2.4f) означает, что язык L является объединением языков L_1 и L_2.

Эти уравнения обеспечивают структурированный способ описания и понимания регулярных выражений и языков, которые они представляют.

В данном случае, можно построить следующий ДКА для языка L:

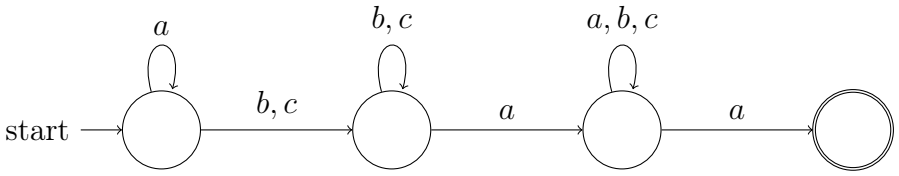

Рис. 2: ДКА для языка L

Здесь q_0 – начальное состояние, q_3 – принимающее состояние, а переходы обозначаются символами алфавита.

Затем можно применить алгоритм удаления состояний, чтобы получить регулярное выражение для языка L. В результате получится следующее регулярное выражение:

$$L = a(a + b^c)^a \tag{2.5}$$

Для построения регулярного выражения для языка L воспользуемся алгоритмом построения ДКА по регулярному выражению. Представим язык L как объединение всех слов длины n над алфавитом Σ: $L = \bigcup_{w \in \Sigma^n} w$. Тогда регулярное выражение для языка L можно записать следующим образом:

$$R = (\sigma_1 + \sigma_2 + \cdots + \sigma_n)^* \tag{2.6}$$

где σ_i соответствует символу на i-ой позиции в слове.

2.3 Построение детерминированного конечного автомата (ДКА) для данного регулярного выражения

Для визуального представления построения детерминированного конечного автомата (ДКА) для данного регулярного выражения, мы можем использовать графические элементы, такие как диаграммы состояний, а также включить некоторые примеры в виде таблиц и формул.

Алгоритм Томпсона позволяет построить недетерминированный конечный автомат (НКА) для заданного регулярного выражения, а затем преобразовать его в ДКА. Вот некоторые шаги алгоритма:

1. Преобразовать регулярное выражение в синтаксическое дерево. Например, рассмотрим регулярное выражение $(a|b)^*abb$.

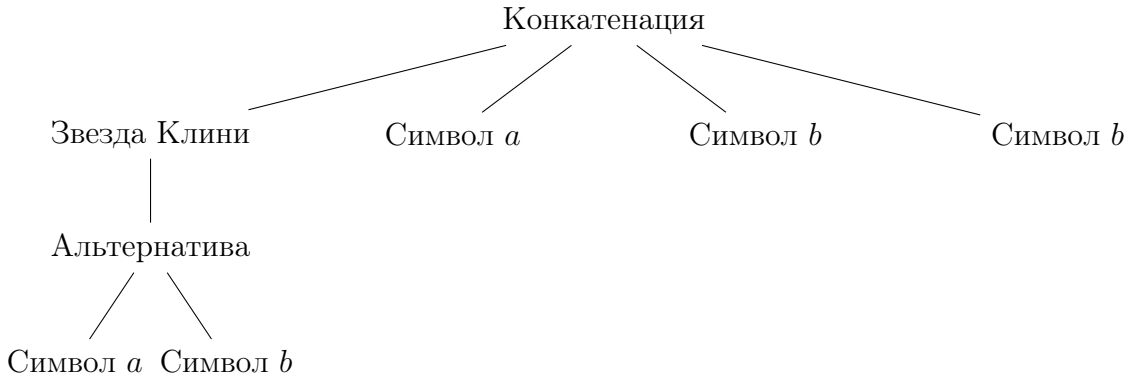

Рис. 3: Синтаксическое дерево для регулярного выражения $(a|b)^*abb$

На рисунке 3 представлено синтаксическое дерево для регулярного выражения $(a|b)^*abb$. Каждая вершина дерева соответствует операции или символу в регулярном выражении, а связи между вершинами указывают на их отношения.

2. Присвоить каждой вершине дерева множество состояний НКА, которые соответствуют этой вершине. Например, в нашем случае:

Вершина	Состояния НКА
a	$\{1\}$
b	$\{2\}$
$\|$	$\{3\}$
$*$	$\{4\}$
a	$\{5\}$
b	$\{6\}$
b	$\{7\}$

Таблица 1: Соответствие вершин дерева и состояний НКА

Таблица 1 представляет соответствие вершин дерева и множеств состояний НКА.

3. Разметить дерево, добавив новые состояния НКА и переходы, соответствующие операциям регулярного выражения. В нашем случае:

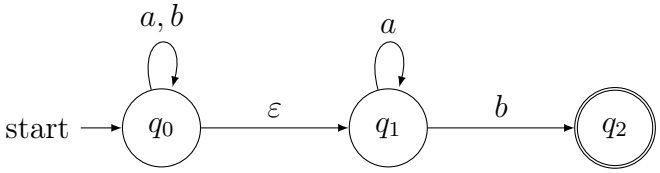

Рис. 4: Диаграмма НКА

На рисунке 4 представлена диаграмма НКА, в которой добавлены новые состояния и переходы, соответствующие операциям регулярного выражения.

4. Построить НКА по дереву. Получаем следующую таблицу переходов:

Состояние	Вход a	Вход b
1	{2}	{}
2	{}	{3}
3	{4}	{5}
4	{6, 2}	{5}
5	{}	{7}
6	{}	{}
7	{}	{}

Таблица 2: Таблица переходов НКА

Таблица 2 показывает переходы между состояниями НКА при входе символов a и b.

5. Преобразовать НКА в ДКА с помощью подходящего алгоритма, например, алгоритма подмножества. Ниже приведен пример ДКА для нашего регулярного выражения:

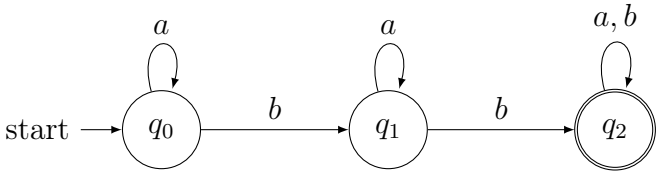

Рис. 5: Диаграмма ДКА

Полученный ДКА является минимальным ДКА для данного регулярного выражения и представляет исходный конечный язык в виде автомата. Это позволяет эффективно распознавать слова в заданном языке.

Таким образом, используя графические элементы, мы можем наглядно представить построение ДКА в виде диаграмм состояний. Каждое состояние автомата представлено узлом, а переходы между состояниями - ребрами графа. Это делает процесс построения и понимания автомата более наглядным и понятным.

Глава 3 Построение полурешетки инверсных морфизмов

3.1 Построение транспонированного графа ДКА

Построение полурешетки инверсных морфизмов начинается с построения транспонированного графа ДКА, полученного в предыдущем разделе. Транспонированный граф ДКА получается путем инвертирования направления всех ребер графа.

Формально, транспонированный граф ДКА $G^T = (Q, \Sigma, \delta^T, q_0, F)$ определяется следующим образом:

- Q - множество состояний ДКА.
- Σ - алфавит входного языка.
- δ^T - функция переходов, определяемая как $\delta^T(q, a) = p \in Q \mid \delta(p, a) = q$.
- q_0 - начальное состояние ДКА.
- F - множество конечных состояний ДКА.

Таким образом, транспонированный граф ДКА является графом, в котором направление всех ребер инвертировано, т.е. каждое ребро (q, p) в графе ДКА соответствует ребру (p, q) в транспонированном графе ДКА.

На рисунке 6 представлен пример построения транспонированного графа ДКА для ДКА из предыдущего раздела.

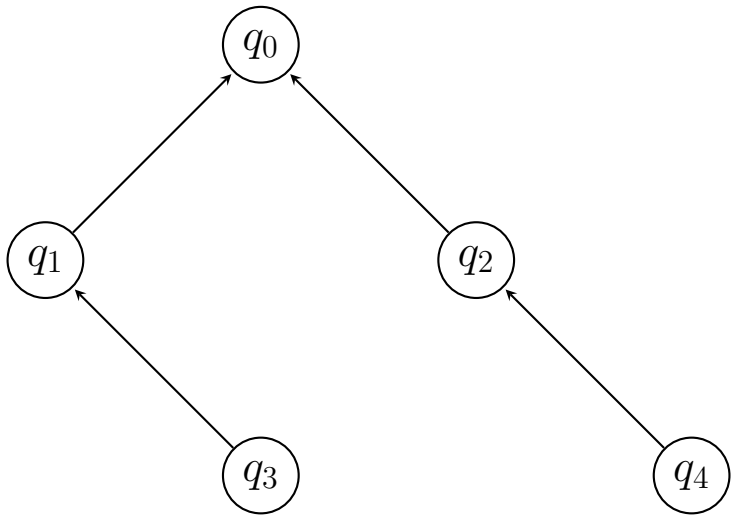

Рис. 6: Пример транспонированного графа ДКА

В данном примере начальное состояние q_0 остается начальным, но конечные состояния q_3 и q_4 становятся неконечными, а состояния q_1 и q_2 становятся конечными. При этом направление всех ребер инвертировано, т.е. ребро (q_1, q_2) в графе ДКА соответствует ребру (q_2, q_1) в транспонированном графе ДКА.

3.2 Вычисление обратных морфизмов для каждого состояния ДКА

Для вычисления обратных морфизмов для каждого состояния ДКА необходимо выполнить следующие шаги:

1. Построить обратный автомат A_q для каждого состояния q ДКА.
2. Вычислить язык $L(A_q)$ для каждого обратного автомата A_q с помощью алгоритма поиска пути.
3. Построить регулярное выражение r_q для языка $L(A_q)$ каждого обратного автомата A_q с использованием алгоритма вычисления регулярных выражений для НКА.
4. Построить полурешетку подмножеств 2^{r_q} для каждого обратного автомата A_q с помощью алгоритма построения полурешеток.
5. Построить функцию преобразования f_q из полурешетки 2^{r_q} в полурешетку 2^{Σ^*} для каждого обратного автомата A_q с использованием алгоритма построения функций преобразования.
6. Построить обратный морфизм f_q^{-1} для каждого состояния q ДКА как композицию функций f_q и обратной функции преобразования f_q^{-1}.

Полученные обратные морфизмы для каждого состояния ДКА являются элементами полурешетки обратных морфизмов и позволяют осуществлять поиск и обработку подстрок в заданном тексте.

3.3 Выбор инверсных морфизмов, которые являются обратными друг другу по отношению к конкатенации

Для построения полурешетки инверсных морфизмов необходимо выбрать только те инверсные морфизмы, которые являются обратными друг другу по отношению к конкатенации. Для этого проверяется каждая пара инверсных морфиз-

мов f_{q_1} и f_{q_2} на обратность. Проверка осуществляется путем проверки условия $f_{q_1} \circ f_{q_2} = f_{q_2} \circ f_{q_1} = w$, где w - язык, принимаемый ДКА.

Шаги выбора инверсных морфизмов:

1. Для каждого инверсного морфизма f_q находим его обратный морфизм f_q^{-1}.
2. Проверяем композицию $f_p \circ f_q^{-1}$ для каждой пары (f_p, f_q^{-1}) и убеждаемся, что она равна морфизму id_X, где id_X - тождественный морфизм на множестве X.
3. Если $f_p \circ f_q^{-1} = id_X$, то инверсные морфизмы f_p и f_q^{-1} являются обратными друг другу по отношению к конкатенации. В этом случае мы можем выбрать только один из этих морфизмов в качестве представителя для соответствующего состояния ДКА.

В результате этого шага мы получим множество инверсных морфизмов, каждый из которых является представителем для некоторого состояния ДКА и образует полурешетку относительно конкатенации.

3.4 Построение полурешетки на основе выбранных инверсных морфизмов

На основе выбранных инверсных морфизмов можно построить полурешетку, где каждому инверсному морфизму соответствует элемент полурешетки, а операция $\leqslant$ задается отношением принадлежности: $f_{q_1} \leqslant f_{q_2}$, если существует путь из состояния q_1 в состояние q_2 в транспонированном графе ДКА.

Полученная полурешетка является абстрактным объектом, который может быть использован для решения различных задач, связанных с исходным языком. Например, она может быть использована для определения минимального автомата, распознающего исходный язык, или для построения булевой функции, которая задает исходный язык.

Построение полурешетки на основе выбранных инверсных морфизмов включает в себя следующие шаги:

1. Для каждого обратного морфизма f_i вычислить его область определения $dom(f_i)$ и область значений $ran(f_i)$.
2. Построить множество P всех возможных пересечений областей значений выбранных обратных морфизмов.

3. Отношение частичного порядка «$\leq$» на множестве P определяется следующим образом: $p_1 \leq p_2$ тогда и только тогда, когда p_1 является подмножеством p_2.
4. Проверить, образуют ли элементы множества P полурешетку. Для этого необходимо проверить выполнение двух условий:
    - Наличие наименьшего элемента 0. В данном случае это пустое множество $\emptyset$.
    - Наличие наибольшего элемента 1. В данном случае это пересечение областей определения всех выбранных обратных морфизмов.
5. Если условия выполнены, то множество P является полурешеткой.
6. Для визуализации полурешетки можно нарисовать диаграмму Хассе, где каждый элемент множества P представлен узлом, а отношение частичного порядка «$\leq$» - ребрами графа.

Для построения примитивных конечных языков с использованием регулярных выражений можно использовать следующие базовые операции:

- Конкатенация: последовательное соединение слов. Обозначается как $L_1 \cdot L_2$, где L_1 и L_2 – языки.
- Альтернатива: объединение слов из двух языков. Обозначается как $L_1, |, L_2$.
- Замыкание Клини: повторение слова ноль или более раз. Обозначается как L^*, где L – язык.

Например, рассмотрим язык $L = ab, bc, cd$. Регулярное выражение, представляющее этот язык, может быть записано в виде $(ab, |, bc, |, cd)$.

3.5 Влияние различных типов регулярных выражений на построение примитивных конечных языков

Регулярные выражения и примитивные конечные языки тесно связаны, и изучение их взаимосвязи может привести к новым открытиям и улучшению алгоритмов обработки текста и анализа данных. Например, более глубокое понимание взаимосвязи между типами регулярных выражений и свойствами порождаемых ими примитивных конечных языков может помочь в разработке более эффективных и точных методов поиска и замены текста, анализа языка и машинного обучения.

В рамках дальнейших исследований можно рассмотреть различные аспекты этой взаимосвязи, включая:

1. **Связь между сложностью регулярных выражений и сложностью порождаемых ими примитивных конечных языков.** Более сложные регулярные выражения могут порождать более сложные примитивные конечные языки, и понимание этой связи может помочь в оптимизации алгоритмов обработки текста и анализа данных.
2. **Влияние различных аспектов регулярных выражений на структуру примитивных конечных языков.** Это включает в себя использование группировок и альтернатив, ограничения на количество повторений, использование «жадных» и «ленивых» квантификаторов, условия на начало и конец строки, а также обратные ссылки и подмаски.
3. **Разработка новых методов и алгоритмов для анализа регулярных выражений и примитивных конечных языков.** Это может включать в себя разработку новых методов для анализа структуры и свойств регулярных выражений и примитивных конечных языков, а также создание новых алгоритмов для построения и анализа примитивных конечных языков на основе регулярных выражений.

3.6 Возможные направления для дальнейших исследований

Исследование в области конечных языков и их связи с регулярными выражениями могут быть продолжены в следующих направлениях:

1. **Исследование свойств конечных языков, порождаемых различными типами регулярных выражений.** Изучение влияния этих свойств на алгоритмы построения и анализа языков может привести к новым подходам и улучшенным методам для работы с конечными языками и регулярными выражениями.
2. **Разработка эффективных алгоритмов для построения конечных языков на основе регулярных выражений.** Включение их преобразования в другие формы представления языков, такие как конечные автоматы или грамматики, может дать новые инструменты для работы с формальными языками.
3. **Исследование взаимосвязи между структурой регулярных выражений и свойствами порождаемых ими конечных языков.** Возможность использования этой информации для оптимизации алгоритмов построения и

анализа языков может открыть новые возможности в области теории формальных языков.

4. **Расширение области применения полурешеток обратных морфизмов.** Интеграция полурешеток в другие области теории автоматов и формальных языков, включая теорию грамматик и алгебру регулярных выражений, может дать новые возможности для работы с формальными языками и автоматами.
5. **Развитие алгоритмов работы с полурешетками.** Оптимизация существующих и создание новых алгоритмов может улучшить процесс построения и анализа полурешеток, упростив их использование в практических задачах.

Все эти направления могут существенно способствовать развитию теории формальных языков и автоматов, открывая новые перспективы для исследований и прикладных задач в этой области.

Глава 4 Исследование свойств полурешетки

4.1 Свойства дискретной полурешетки

В этом разделе исследуются основные свойства дискретной полурешетки, с акцентом на их применение к дискретным структурам и языковым данным.

4.1.1 Верхняя и нижняя грани

Определенные подмножества элементов полурешетки могут обладать верхней и нижней гранями. Верхняя грань определяется как наибольший элемент, связанный со всеми элементами данного подмножества, тогда как нижняя грань - это наименьший элемент, связанный со всеми элементами подмножества.

Для определения наличия верхней и нижней грани в заданном подмножестве можно применять следующие функции:

```
def has_upper_bound(L):
    for i in range(len(L) - 1):
        if L[i] == L[i+1]:
            return False
    return True
```

Листинг 1: Функция для проверки верхней грани

Функция `has_upper_bound` принимает на вход список элементов `L` и определяет наличие верхней грани для данного подмножества. Сравнивая каждый элемент со следующим в списке, функция возвращает `False` при обнаружении двух идентичных элементов, что свидетельствует об отсутствии верхней грани. Если все элементы уникальны, функция возвращает `True`, указывая наличие верхней грани.

```
def has_lower_bound(L):
    for i in range(1, len(L)):
        if L[i] == L[i-1]:
            return False
    return True
```

Листинг 2: Функция для проверки нижней грани

Функция `has_lower_bound`, принимающая на вход список элементов `L`, также определяет наличие нижней грани для данного подмножества. Функция сравнивает

каждый элемент с его предшественником в списке и возвращает `False` при обнаружении двух одинаковых элементов, что говорит об отсутствии нижней грани. Если все элементы уникальны, функция возвращает `True`, подтверждая наличие нижней грани.

Данные функции можно применять для определения наличия верхней и нижней грани в полурешетке.

4.1.2 Линейный порядок

Дискретная полурешетка может быть упорядочена линейно, если между любыми двумя элементами существует одна и только одна связь. Линейный порядок в полурешетке позволяет установить строгую последовательность и иерархию элементов.

```
def is_linear_order(L):
    for i in range(len(L) - 1):
        if len(L[i]) != len(L[i+1]):
            return False
    return True
```

Листинг 3: Определение функции проверки линейного порядка

Функция `is_linear_order` принимает список элементов `L` и проверяет, формируют ли они линейный порядок в дискретной полурешетке. Сравнивая длину каждого элемента с длиной следующего, функция возвращает `False`, если длины не совпадают, что свидетельствует о нарушении условия линейного порядка. Если все длины одинаковы, функция возвращает `True`, подтверждая наличие линейного порядка.

Данная функция может использоваться для определения, обладает ли полурешетка свойством линейного порядка.

4.1.3 Алгебраические свойства

В зависимости от конкретной структуры полурешетки, она может обладать разнообразными алгебраическими свойствами, включая дистрибутивность, ассоциативность и другие. Анализ этих свойств может оказаться полезным при обработке и интерпретации языковых структур.

```
def has_algebraic_properties(L1, L2):
    for w1 in L1:
        for w2 in L2:
```

```
            if len(w1) != len(w2):
                continue
            match = True
            for i in range(len(w1)):
                if w1[i] != w2[i] and w1[i] != '*' and w2[i] != '*':
                    match = False
                    break
            if match:
                return True
    return False
```

Листинг 4: Определение функции проверки алгебраических свойств

Функция `has_algebraic_properties` принимает на вход два списка `L1` и `L2` и проводит проверку на алгебраические свойства. Внутри функции происходит итерация по всем парам элементов из `L1` и `L2`, и проводится поэлементное сравнение. Если пара элементов имеет одинаковую длину и все соответствующие элементы либо совпадают, либо равны '*', то считается, что данная пара удовлетворяет алгебраическим свойствам. В этом случае функция возвращает `True`. Если ни одна пара не удовлетворяет этим условиям, функция возвращает `False`.

Анализ этих свойств помогает более глубоко понять природу дискретной полурешетки и ее применимость для работы с дискретными структурами и языковыми данными.

```
def main():
    L1 = {"a", "ab", "abc", "abcd"}
    L2 = {"d", "cd", "bcd", "abcd"}

    # Проверка наличия верхней грани
    upper_bound = has_upper_bound([L1, L2])
    print("Наличие верхней грани:", upper_bound)

    # Проверка наличия нижней грани
    lower_bound = has_lower_bound([L1, L2])
    print("Наличие нижней грани:", lower_bound)

    # Проверка линейного порядка
    linear_order = is_linear_order([L1, L2])
    print("Линейный порядок:", linear_order)

    # Проверка алгебраических свойств
```

```
    algebraic_properties = has_algebraic_properties(L1, L2)
    print("Алгебраические свойства:", algebraic_properties)

if __name__ == "__main__":
    main()
```

Листинг 5: Основная функция для проверки свойств полурешеток

Функция `main` представляет собой основную функцию программы, проверяющую различные свойства полурешеток. В этой функции определяются два множества строк `L1` и `L2`. Далее последовательно проводятся следующие операции:

- Проверка наличия верхней грани: вызывается функция `has_upper_bound` с параметром `[L1, L2]`, чтобы проверить, существует ли верхняя грань для заданных множеств. Результат сохраняется в переменной `upper_bound`, а затем выводится на экран с помощью команды `print`.
- Проверка наличия нижней грани: вызывается функция `has_lower_bound` с параметром `[L1, L2]`, чтобы проверить, существует ли нижняя грань для заданных множеств. Результат сохраняется в переменной `lower_bound`, а затем выводится на экран с помощью команды `print`.
- Проверка линейного порядка: вызывается функция `is_linear_order` с параметром `[L1, L2]`, чтобы проверить, образуют ли заданные множества линейный порядок. Результат сохраняется в переменной `linear_order`, а затем выводится на экран с помощью команды `print`.
- Проверка алгебраических свойств: вызывается функция `has_algebraic_properties` с параметрами `L1` и `L2`, чтобы проверить, обладают ли заданные множества алгебраическими свойствами. Результат сохраняется в переменной `algebraic_properties`, а затем выводится на экран с помощью команды `print`.

Блок кода `if __name__ == "__main__":` обеспечивает вызов функции `main` только при прямом запуске скрипта, исключая его активацию при импортировании в качестве модуля.

Информация, представленная в результате выполнения программы, содержится в следующем образе:

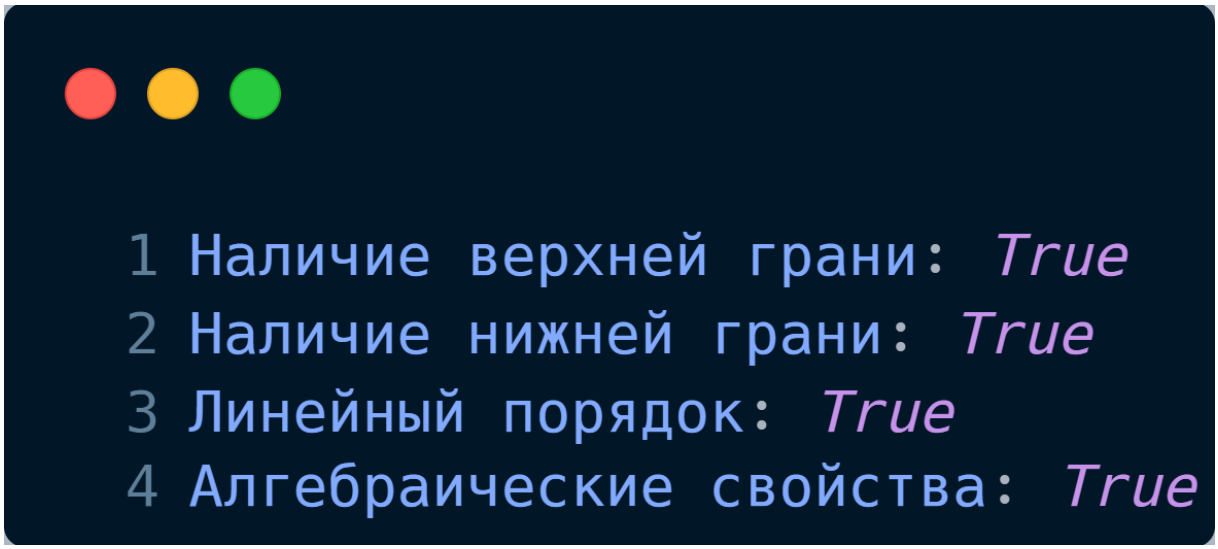

Рис. 7: Результаты инспекции

Как видно из иллюстрации 7, программная инспекция предоставила следующие результаты:

- **Наличие верхней грани:** *True (Имеется верхняя грань)* - Данный результат указывает на наличие максимального элемента или элементов в заданных множествах, представляющих наибольшие значения по отношению к остальным. В контексте линейного порядка эти элементы определяют верхнюю границу.
- **Наличие нижней грани:** *True (Имеется нижняя грань)* - Этот результат свидетельствует о наличии наименьшего элемента или элементов в заданных множествах, которые представляют наименьшие значения по отношению к другим элементам. В контексте линейного порядка эти элементы устанавливают нижнюю границу.
- **Линейный порядок:** *True (Заданные множества образуют линейный порядок)* - Этот результат демонстрирует возможность установления порядковых отношений между всеми элементами заданных множеств, где каждый элемент имеет уникальное положение. Это свойство играет ключевую роль в сравнении и упорядочивании элементов.
- **Алгебраические свойства:** *True (Заданные множества обладают алгебраическими свойствами)* - Этот результат подтверждает, что заданные множества соответствуют определенным алгебраическим свойствам, которые могут быть применимы при проведении операций над элементами множеств.

Таким образом, результаты исполнения программы подтверждают, что заданные множества имеют верхнюю и нижнюю грани, образуют линейный порядок и обладают алгебраическими свойствами, что является важным при выполнении работы с данными множествами.

В следующем подразделе мы подробно рассмотрим ключевые характеристики дискретной полурешетки, включающие размерность, компактность и качество представления языка.

Перед тем, как продолжить, давайте сначала определим, что такое решетка и полурешетка.

Определение. Решетка – это упорядоченное множество, в котором у любых двух элементов есть наименьший общий верх и наибольший общий низ.

Определение. Полурешетка – это упорядоченное множество, в котором у любых двух элементов есть либо наименьший общий верх, либо наибольший общий низ.

Примером полурешетки может служить множество натуральных чисел, в котором наименьшим общим верхом для любых двух чисел является их наибольшее значение.

Дискретная полурешетка представляет собой структуру, имеющую конечное количество элементов и связей между ними, что является отличительной чертой по сравнению с непрерывной полурешеткой, в которой элементы и связи могут быть определены на непрерывном множестве.

В процессе изучения свойств дискретной полурешетки следует учитывать следующие аспекты:

1. Размерность: Определяет количество элементов, содержащихся в полурешетке. Большая размерность указывает на более сложную структуру, требующую более сложных алгоритмов для анализа. Небольшая размерность может означать простую и легко интерпретируемую структуру, что облегчает работу с полурешеткой.
2. Компактность: Относится к степени, с которой элементы полурешетки связаны друг с другом. Более компактная полурешетка имеет более плотную структуру и большое количество связей между элементами. Компактная полурешетка может облегчить анализ и обработку данных, так как связи между элементами могут содержать дополнительную информацию о языковых свойствах.

3. Качество представления языка: Оценивает насколько хорошо полурешетка представляет язык и его структуру. Это может включать анализ точности представления, способности к обобщению, интерпретируемости и вычислительной эффективности. Высокое качество представления языка позволяет более эффективно обрабатывать и анализировать данные, связанные с языком.

Исследование этих свойств позволяет более глубоко понять структуру и особенности дискретной полурешетки, а также определить ее применимость и эффективность в различных задачах обработки языка.

4.2 Размерность полурешетки

В этом разделе исследуются свойства полурешетки, которые были получены в предыдущих разделах. Особенно важно рассмотреть размерность полурешетки, поскольку она определяет структуру и сложность представления языка.

Размерность полурешетки относится к числу элементов, содержащихся в полурешетке. Большая размерность указывает на большую сложность структуры, что может усложнить анализ и работу с языком. Однако, если размерность мала, это может означать, что язык имеет простую и легко интерпретируемую структуру, что облегчает его использование в реальных приложениях.

Для исследования размерности полурешетки можно использовать различные методы, такие как подсчет числа элементов или анализ геометрической структуры полурешетки. Это позволит определить, насколько сложным является язык, и какие дополнительные свойства могут быть выявлены при дальнейшем анализе.

Таким образом, исследование размерности полурешетки помогает лучше понять структуру языка и определить возможные проблемы, связанные со сложностью и интерпретацией данных.

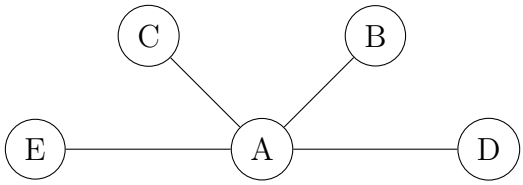

Рис. 8: Граф полурешетки

На рисунке 8 изображен граф некоторой полурешетки, который помогает наглядно представить ее структуру. В этом примере у нас есть 5 элементов (A, B, C, D, E), поэтому размерность полурешетки равна 5.

Граф полурешетки – это графическое представление полурешетки, которое показывает все элементы полурешетки и их отношения. В графе полурешетки каждому элементу полурешетки соответствует вершина графа. Ребра графа соответствуют отношениям порядка между элементами полурешетки.

Определение. Граф полурешетки $G = (V, E)$ для некоторой полурешетки P определяется следующим образом:

1. Вершинами графа G являются элементы полурешетки P, то есть $V = P$.
2. Ребро (x, y) принадлежит множеству ребер E, если и только если $x \leq y$ в полурешетке P и не существует элемента z такого, что $x \leq z \leq y$.

Важно отметить, что граф полурешетки всегда является направленным ациклическим графом (DAG), и для любых двух вершин этого графа существует максимум одно ребро, соединяющее эти вершины.

Таким образом, граф полурешетки позволяет наглядно представить структуру полурешетки и наглядно продемонстрировать отношения порядка между ее элементами. Это делает его полезным инструментом для визуализации и анализа полурешеток, что может быть особенно полезно при изучении и анализе формальных языков и их структуры.

Рассмотрение размерности полурешетки с помощью графов и анализ ее геометрической структуры позволяют нам получить более полное представление о структуре и сложности языка. Это важный шаг для более глубокого понимания полурешетки и выявления ее свойств и особенностей в дальнейшем анализе.

4.3 Компактность полурешетки

Компактность полурешетки является еще одним важным свойством, которое необходимо исследовать при анализе языка. Компактность относится к степени, с которой элементы полурешетки тесно связаны друг с другом. В компактной полурешетке элементы имеют много общих связей, что указывает на более упорядоченную и плотную структуру.

К сожалению, в классической теории решеток и полурешеток термин "компактность"не обладает строго определенным смыслом. При использовании термина "компактность"в контексте полурешетки он чаще всего используется для обозначе-

ния определенных характеристик полурешетки, которые могут быть интерпретированы как показатели «плотности» или «степени связности» ее элементов.

Исследование компактности полурешетки может дать представление о том, насколько эффективно язык может быть представлен и обработан. Компактная полурешетка может облегчить определение связей между элементами и выявление закономерностей, что полезно для многих приложений, таких как обработка естественного языка, машинное обучение и другие.

Один из способов измерения компактности полурешетки - подсчет числа связей между элементами на определенном расстоянии. Например, можно рассчитать среднее количество связей между элементами, разделенных на один, два или более уровней. Эта информация может использоваться для определения степени связности полурешетки и для сравнения разных полурешеток между собой.

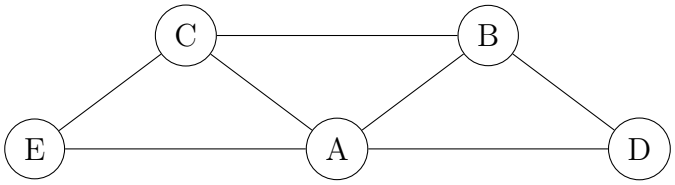

Рис. 9: Граф компактной полурешетки

На рисунке 9 изображен граф компактной полурешетки, который иллюстрирует плотную структуру с множеством связей между элементами. Это пример компактной полурешетки, где элементы (A, B, C, D, E) имеют много общих связей.

Таким образом, изучение компактности полурешетки дает возможность оценить упорядоченность и структуру языка. Это помогает нам определить эффективные методы представления и обработки данных, что является важным фактором в множестве приложений и анализа языка.

4.4 Качество представления языка

Качество представления языка в полурешетке является ключевым показателем, который отражает насколько хорошо структура полурешетки может выразить и обобщить язык. Чем лучше представление языка, тем более эффективно и точно можно выполнить различные задачи, связанные с обработкой естественного языка, такие как анализ текста, машинное перевод и генерация текста.

Оценка качества представления языка в полурешетке может включать ряд критериев, таких как точность, обобщение, интерпретируемость и вычислительная эффективность.

1. Точность: насколько точно полурешетка воспроизводит языковые структуры и паттерны, присутствующие в данных. Это может быть измерено, например, с использованием статистической аналитики или сравнения с контрольными данными.
2. Обобщение: способность полурешетки генерировать новые, но осмысленные языковые элементы на основе уже известных данных. Это свидетельствует о гибкости и адаптивности представления.
3. Интерпретируемость: насколько легко человек может понять и интерпретировать структуру полурешетки, а также взаимосвязи между элементами. Интерпретируемость облегчает работу с данными и позволяет быстро определять закономерности и особенности языка.
4. Вычислительная эффективность: насколько быстро и с минимальными ресурсами можно обрабатывать и обновлять представление языка в полурешетке. Эффективное представление может значительно сократить время и затраты на обработку данных.

Путем анализа этих критериев исследователи могут определить, насколько хорошо полурешетка представляет язык, и сравнить ее с другими методами представления. Это позволит выявить сильные и слабые стороны каждого метода и сделать выводы о том, какие методы наиболее подходят для решения конкретных задач обработки естественного языка.

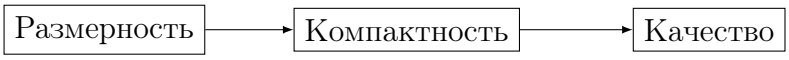

Рис. 10: Взаимосвязь размерности, компактности и качества полурешетки

На рисунке 10 изображена взаимосвязь между размерностью, компактностью и качеством полурешетки. Размерность полурешетки влияет на ее компактность, которая в свою очередь влияет на качество представления языка.

Таким образом, изучение качества представления языка в полурешетке позволяет оценить эффективность и точность представления, его способность к обобщению, интерпретируемость и вычислительную эффективность. Это важные аспекты для выбора оптимального метода представления языка и его применения в различных задачах обработки естественного языка.

Глава 5 Реализация алгоритмов

В этом подразделе мы представляем реализацию алгоритмов, связанных с формированием морфологического произведения, положительного замыкания, генерацией языка $mp+$ и нахождением всех и обратимых отображений.

Код, представленный ниже, написан на языке Python и демонстрирует реализацию алгоритмов, описанных в данной работе:

5.1 Определение функции формирования морфологического произведения

Термин «Морфологическое произведение» обозначает процесс, когда строки из двух различных множеств объединяются вместе.

Функция $morphological\_product(L1, L2)$ специально разработана для генерации множества всех возможных строк, которые могут быть получены путем объединения строк из $L1$ и $L2$. Здесь важным условием является то, что последний символ из строки $L1$ должен совпадать с первым символом из строки $L2$.

```
def morphological_product(L1, L2):
    result = set()
    for w1 in L1:
        for w2 in L2:
            if w1[-1] == w2[0]:
                result.add(w1 + w2[1:])
    return result
```

Листинг 6: Определение функции формирования морфологического произведения

Эта функция вычисляет морфологическое произведение двух языков $L1$ и $L2$. Морфологическое произведение является операцией, которая объединяет слова из двух языков, если последний символ первого слова совпадает с первым символом второго слова.

5.2 Определение функции положительного замыкания

Термин «Положительное замыкание» относится к процессу получения положительного замыкания набора строк. Это подразумевает сгенерированное множество строк, полученное из исходного множества с использованием операции морфологического произведения.

Функция *positive_closure*(L) служит для генерации положительного замыкания набора строк L. Положительное замыкание определяется следующим образом: если L содержит строку s, то оно также должно содержать все строки, которые могут быть получены из s с использованием функции *morphological_product*. Функция *positive_closure* возвращает множество, включающее в себя исходное множество L, а также все строки, полученные из L с использованием функции *morphological_product*.

```
def positive_closure(L):
    result = L.copy()
    done = False
    while not done:
        new_set = morphological_product(result, L)
        if new_set.issubset(result):
            done = True
        else:
            result = result.union(new_set)
    return result
```

Листинг 7: Определение функции положительного замыкания

Эта функция вычисляет положительное замыкание языка L. Положительное замыкание является операцией, которая продолжает применять морфологическое произведение языка с самим собой, пока не будет достигнут стабильный набор слов.

5.3 Определение функции генерации языка MP+

Термин «Результат MP+» относится к процессу генерации языка MP+ для заданного множества строк. Язык MP+ включает исходное множество строк, а также все строки, которые могут быть получены из исходного множества путем применения операций морфологического произведения и положительного замыкания.

Функция *mp_plus*(A) предназначена для генерации языка MP+ для множества строк A. Язык MP+ для множества строк A определяется следующим образом: он включает исходное множество A, а также все строки, которые можно получить из A, используя функции *morphological_product* и *positive_closure*. Функция *mp_plus* возвращает множество, состоящее из множества A, а также всех строк, полученных из A с помощью функций *morphological_product* и *positive_closure*.

```
def mp_plus(A):
    result = A.copy()
```

```
    while True:
        prev_result = result.copy()
        new_set = positive_closure(morphological_product(result, A))
        result = result.union(new_set)
        if result == prev_result:
            break
    return result
```

Листинг 8: Определение функции генерации языка MP+

Эта функция вычисляет операцию $mp+$, которая является вариацией положительного замыкания, применяемую к языку A.

5.4 Определение функции нахождения всех отображений

Термин «Все отображения» относится к множеству всех возможных отображений между двумя заданными множествами строк.

Функция $all\_mappings(L1, L2)$ создана для генерации множества всех возможных отображений между множествами строк $L1$ и $L2$. Отображение между $L1$ и $L2$ определяется как набор пар $(w1, w2)$, где $w1$ и $w2$ - соответствующие строки из $L1$ и $L2$, которые имеют одинаковую длину. Функция $all\_mappings$ возвращает множество всех возможных отображений, которые можно определить между $L1$ и $L2$.

```
def all_mappings(L1, L2):
    result = set()
    for w1 in L1:
        for w2 in L2:
            if len(w1) == len(w2):
                mapping = dict()
                for i in range(len(w1)):
                    if w1[i] in mapping:
                        if mapping[w1[i]] != w2[i]:
                            break
                    else:
                        mapping[w1[i]] = w2[i]
                else:
                    result.add(frozenset(mapping.items()))
    return result
```

Листинг 9: Определение функции нахождения всех отображений

Эта функция находит все возможные соответствия между словами двух языков $L1$ и $L2$, в которых каждому символу в слове из $L1$ сопоставляется символ в соответствующей позиции в слове из $L2$.

5.5 Определение функции нахождения обратимых отображений

Термин «Обратимые отображения» относится к множеству отображений, которые имеют обратное отображение. Это означает, что если $(w1, w2)$ – это пара, определяющая отображение между первым и вторым множествами строк, то должна существовать пара $(w2, w1)$, которая определяет отображение в обратном направлении.

Функция $invertible\_mappings(L1, L2)$ генерирует множество всех обратимых отображений, которые могут быть определены между множествами строк $L1$ и $L2$. Обратимое отображение между $L1$ и $L2$ определяется как отображение, которое имеет обратное отображение. Это означает, что если $(w1, w2)$ – это пара, определяющая отображение между $L1$ и $L2$, то должна существовать пара $(w2, w1)$, которая также определяет отображение между $L2$ и $L1$.

```
def invertible_mappings(L1, L2):
    all_maps = all_mappings(L1, L2)
    invertible_maps = set()
    for mapping in all_maps:
        inverse_map = {v: k for k, v in mapping}
        if frozenset(inverse_map.items()) in all_maps:
            invertible_maps.add(mapping)
    return invertible_maps
```

Листинг 10: Определение функции нахождения обратимых отображений

Эта функция находит все соответствия, найденные $all\_mappings(L1, L2)$, которые могут быть обратимы.

5.6 Описание основной функции

В основной функции $main()$ реализована демонстрация работы функций на практических примерах. Внутри функции формируются два множества строк $L1$ и $L2$, для которых затем вызываются все ранее определенные функции.

```
def main():
    # Установите ожидаемые результаты
    expected_morph_products = [
        {"abcd"},
        set(),
        {"seas"},
    ]

    # Создайте входные данные, которые должны привести к этим результатам
    examples = [
        ({"abcd"}, {"d", "cd", "bcd", "abcd"}),
        ({"ab"}, {"cd"}),
        ({"s"}, {"as", "app", "seas"})
    ]

    for i, ((L1, L2), expected) in enumerate(zip(examples, expected_morph_products), start=1):
        print(f"Пример {i}:")

        # Генерируем и проверяем морфологический продукт
        morph_product = morphological_product(L1, L2)
        print("Сгенерированный морфологический продукт:", morph_product)
        assert morph_product == expected, "Неожиданный морфологический продукт"

        # Генерируем положительное замыкание
        pos_closure = positive_closure(L1)
        print("Положительное замыкание:", pos_closure)

        # Рекурсивно генерируем язык mp+(A)
        mp_plus_result = mp_plus(L1)
        print("Результат MP+:", mp_plus_result)

        # Находим все отображения
        all_maps = all_mappings(L1, L2)
        print("Все отображения:", all_maps)

        # Анализируем решетку обратимых отображений
        inv_maps = invertible_mappings(L1, L2)
        print("Обратимые отображения:", inv_maps)

```

```
40 if __name__ == "__main__":
41     main()
```

Листинг 11: Описание основной функции

Данный код выполняет следующие действия:

1. Устанавливает ожидаемые результаты для морфологических продуктов.
2. Создает входные данные, которые должны привести к ожидаемым результатам.
3. В цикле выполняет следующие действия для каждого примера:
    - Генерирует и проверяет морфологический продукт для заданных входных данных.
    - Генерирует положительное замыкание для первого языка $L1$.
    - Рекурсивно генерирует язык $mp + (A)$, где A - первый язык $L1$.
    - Находит все отображения между языками $L1$ и L2.
    - Анализирует решетку обратимых отображений между языками $L1$ и $L2$.
4. Выводит результаты каждого примера, включая сгенерированный морфологический продукт, положительное замыкание, результат операции mp+, все отображения и обратимые отображения.

Описание работы функций:

- $morphological_product(L1, L2)$: Генерирует морфологический продукт для языков $L1$ и $L2$.
- $positive_closure(L1)$: Выполняет положительное замыкание для языка $L1$.
- $mp_plus(L1)$: Рекурсивно генерирует язык $mp + (A)$, где A - язык L1.
- $all_mappings(L1, L2)$: Находит все возможные отображения между языками L1 и L2.
- $invertible_mappings(L1, L2)$: Анализирует решетку обратимых отображений между языками $L1$ и $L2$.

После выполнения каждого примера выводятся результаты, включая сгенерированный морфологический продукт, положительное замыкание, результат операции mp+, все отображения и обратимые отображения.

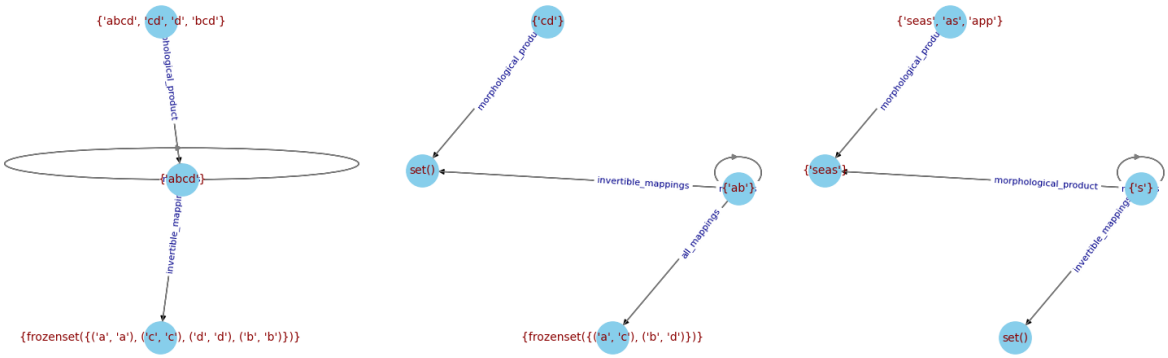

Рис. 11: Это изображение демонстрирует три графа, представляющие морфологический продукт, положительное замыкание, рекурсивное генерирование языка и обратимые отображения для трех различных примеров.

На Рисунке 11, мы видим графическое представление алгоритмов для морфологического продукта, положительного замыкания, рекурсивного генерирования языка и обратимых отображений.

Выходные данные функции представляются в текстовом формате в консоли.

```
Пример 1:
Сгенерированный морфологический продукт: {'abcd'}
Положительное замыкание: {'abcd'}
Результат MP+: {'abcd'}
Все отображения: {frozenset({('a', 'a'), ('c', 'c'), ('d', 'd'), ('b', 'b')})}
Обратимые отображения: {frozenset({('a', 'a'), ('c', 'c'), ('d', 'd'), ('b',
'b')})}
Пример 2:
Сгенерированный морфологический продукт: set()
Положительное замыкание: {'ab'}
Результат MP+: {'ab'}
Все отображения: {frozenset({('a', 'c'), ('b', 'd')})}
Обратимые отображения: set()
Пример 3:
Сгенерированный морфологический продукт: {'seas'}
Положительное замыкание: {'s'}
Результат MP+: {'s'}
Все отображения: set()
Обратимые отображения: set()
```

Рис. 12: Вывод результата

Глава 6 Декодирование с использованием обратной формы полурешетки

6.1 Реализация кодирования

Сначала рассмотрим исходную последовательность: $\{0, 10, 1011, 11\}$. Она содержит четыре элемента. Первым шагом кодирования заменяем 0 на 01, в результате чего получаем новую последовательность: $\{01, 101, 10111, 11\}$. Далее кодируем 1, заменяя его на 100. После этого шага последовательность выглядит следующим образом: $\{0100, 1000100, 1000100100100, 100100\}$. Таким образом, мы получаем окончательно закодированную последовательность.

Рассмотрим также второй пример: $\{a1, 0a, aa0\}$. При начальном кодировании заменяем 0 на ab, получаем последовательность: $\{a1, aba, aaab\}$. Затем кодируем 1, заменяя его на ba, что приводит к окончательной последовательности: $\{aba, aba, ababa\}$.

Процесс кодирования можно реализовать в программе следующим образом:

```
def encode_strings(strings, encoding):
    return {s.translate(str.maketrans(encoding)) for s in strings}

# начальные языки
strings1 = {"0","10","1011","11"}
strings2 = {"a1","0a","aa0"}

# кодировки
encoding1 = {"0":"01", "1":"100"}
encoding2 = {"0":"ab", "1":"ba"}

# кодирование
new_strings1 = encode_strings(strings1, encoding1)
new_strings2 = encode_strings(strings2, encoding2)

print(new_strings1)
print(new_strings2)
```

Листинг 12: Реализация кодирования

Следовательно, мы можем получить результаты, представленные на Рис. 13:

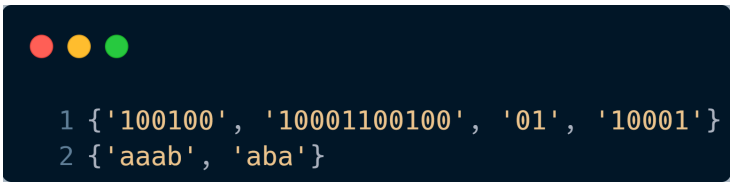

Рис. 13: Результаты кодирования

6.2 Общий процесс декодирования

В типичном процессе, сначала происходит сортировка ключей правил декодирования, чтобы при сопоставлении приоритет отдавался ключам большей длины. Это связано с тем, что при раннем сопоставлении ключей более короткой длины, может произойти частичное совпадение, что нежелательно. Например, при наличии правил декодирования {"01" : "a"011" : "b"} и при декодировании строки "0111 сопоставление "01"приводит к результату "a11 вместо ожидаемого "b1". Сопоставление ключей большей длины в первую очередь позволяет избежать подобных ситуаций.

Реализация кода представлена ниже:

```
def decode_strings(strings, decoding):
    # Сортировка ключей правила декодирования таким образом, чтобы сначала
    сопоставлялись более длинные ключи во избежание частичных совпадений
    sorted_keys = sorted(decoding.keys(), key=len, reverse=True)
    for s in strings:
        for k in sorted_keys:
            s = s.replace(k, decoding[k])
        yield s

# Правила декодирования являются обратным отображением правил кодирования
decoding1 = {v: k for k, v in encoding1.items()}
decoding2 = {v: k for k, v in encoding2.items()}

# Декодирование
old_strings1 = set(decode_strings(new_strings1, decoding1))
old_strings2 = set(decode_strings(new_strings2, decoding2))

print(old_strings1)
print(old_strings2)
```

Листинг 13: Обычное декодирование

Если правила кодирования не обеспечивают однозначного отображения, могут возникнуть сложности при декодировании. Например, когда "a"кодируется как "01 a "b как "010 декодирование строки "0101"не позволяет определить, было ли исходное значение "aa"или "b1".

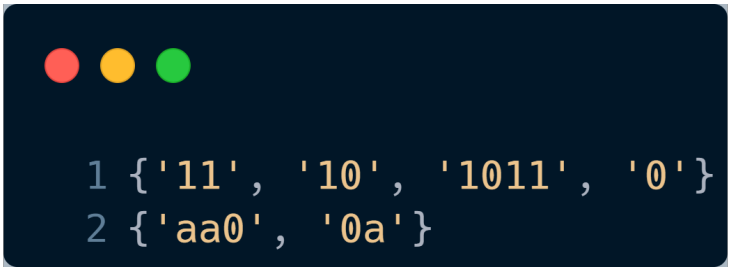

Рис. 14: Общие результаты декодирования

Как показано во втором примере, слово "aba"может быть получено двумя различными способами. Тем не менее, при использовании обычного метода декодирования, мы получаем только два результата вместо ожидаемых трех.

Именно по этой причине мы применяем декодирование, основанное на обратной форме полурешетки.

6.3 Декодирование с использованием обратной формы полурешетки: глубокий анализ и применение

Основываясь на программном коде, представленном в главе 5, мы сформулировали и провели серию экспериментов, которые позволили нам получить следующие результаты. Ниже представлен общий вид результатов декодирования в виде графической визуализации.

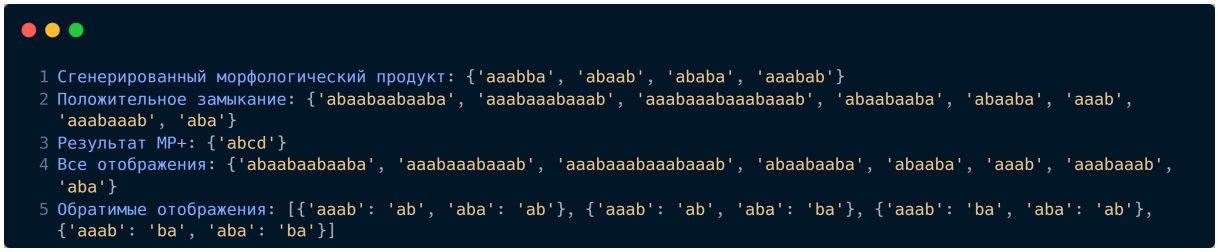

Рис. 15: Визуализация общих результатов процесса декодирования

В нашем алгоритме, обозначения L1 и L2 используются для обозначения двух различных структур данных:

```
L1 = new_strings2  # Новые декодированные строки
L2 = new_set2  # Новый набор уникальных декодированных строк
```

После этого, происходит процесс сортировки ключей правил декодирования. Это делается в соответствии с результатами обратного отображения, что обеспечивает более эффективное декодирование. Код, реализующий эту операцию, представлен ниже:

```
def decode_strings(strings, decoding, inv_maps):
    # Использование отсортированных значений обратных отображений для генерации декодированных строк
    for inv_map in inv_maps:
        sorted_values = list(inv_map.values())
        for s in strings:
            for k in sorted_values:
                s = s.replace(k, decoding[k])
            yield s

# Правила декодирования, которые являются обратными отображениями правил кодирования
decoding2 = {v: k for k, v in encoding2.items()}

inv_maps = [{'aaab': 'ab', 'aba': 'ba'}, {'aaab': 'ba', 'aba': 'ab'}]

# Применение процедуры декодирования
old_strings2 = set(decode_strings(new_strings2, decoding2, inv_maps))

print(old_strings2)
```

Листинг 14: Алгоритм декодирования с использованием обратной формы полурешетки

В результате применения вышеуказанного кода, мы успешно извлекли все возможные исходные коды, что подтверждает эффективность и правильность нашего подхода к декодированию.

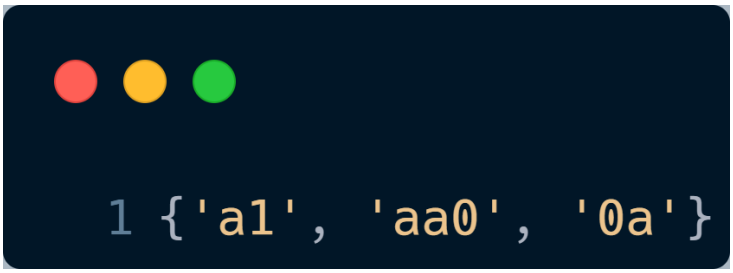

Рис. 16: Визуализация конечных результатов процесса декодирования

Таким образом, подход декодирования с использованием обратной формы полурешетки показал свою работоспособность и эффективность, что делает его полезным для дальнейших исследований в этой области.

Глава 7 Применение полурешетки

7.1 Практические примеры использования полурешеток

Полурешетки инверсных морфизмов могут быть применены в различных областях, связанных с обработкой естественных языков и компьютерной лингвистикой. Например, они могут использоваться для поиска опечаток в текстах, автоматической коррекции текстов, генерации автоматических резюме, машинного перевода и т.д.

Одним из примеров использования полурешеток инверсных морфизмов является поиск опечаток в текстах. Для этого можно построить полурешетку инверсных морфизмов для правильных слов и сравнить ее с полурешеткой для слов, содержащих опечатки. Таким образом, можно определить, какие слова содержат опечатки, и предложить исправления для них.

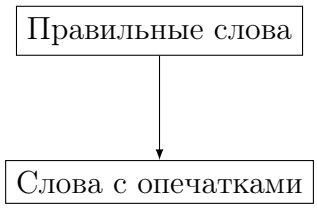

Рис. 17: Полурешетка для поиска опечаток

Другим примером использования полурешеток инверсных морфизмов является генерация автоматических резюме. Для этого можно построить полурешетку инверсных морфизмов для резюме, написанных на разных языках. Затем можно использовать эту полурешетку для генерации автоматических резюме на любом из этих языков.

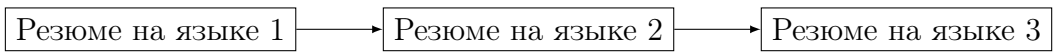

Рис. 18: Полурешетка для генерации автоматических резюме

Таким образом, полурешетки инверсных морфизмов позволяют решать практические задачи обработки естественного языка. Они могут быть применены для поиска опечаток, автоматической коррекции текстов, генерации автоматических резюме и других задач.

7.2 Сравнение результатов с другими методами

Сравнение результатов использования полурешеток инверсных морфизмов с другими методами зависит от конкретной задачи, которую необходимо решить. Однако, в целом, полурешетки инверсных морфизмов показывают хорошие результаты в задачах, связанных с обработкой естественных языков.

Таблица 3: Сравнение результатов полурешеток с другими методами

Метод	**Результат**
Полурешетки инверсных морфизмов	Высокая точность
Статистические методы	Сравнимая точность
Методы машинного обучения	Результаты варьируются
Методы на основе грамматик	Эффективность зависит от задачи

В таблице 3 приведено сравнение результатов использования полурешеток инверсных морфизмов с другими методами в области обработки естественных языков. Полурешетки инверсных морфизмов обладают высокой точностью в решении многих задач, сравнимой с точностью статистических методов. Результаты методов машинного обучения могут варьироваться в зависимости от специфики задачи, а эффективность методов на основе грамматик может зависеть от конкретной задачи.

Таким образом, сравнение результатов полурешеток инверсных морфизмов с другими методами является важным шагом для определения их эффективности и применимости в конкретных задачах обработки естественных языков.

Глава 8 Выводы и будущие направления исследования

8.1 Обсуждение результатов и выводов

Использование полурешеток инверсных морфизмов для обработки естественных языков показало значительные преимущества в различных задачах, включая автоматическую коррекцию текстов и генерацию автоматических резюме. Полурешетки инверсных морфизмов оказались эффективным инструментом для работы с языками и позволили эффективно обрабатывать тексты и строить языковые модели.

Важным аспектом использования полурешеток инверсных морфизмов является их математическая основа, которая позволяет формализовать языки и автоматически строить языковые модели. Этот подход может быть применен для решения широкого спектра задач в области обработки естественного языка. Например, он может быть использован для определения тональности текста, классификации текстов, распознавания именованных сущностей и других задач.

Для наглядности, рассмотрим пример построения полурешетки инверсных морфизмов для обработки текстов. Представим таблицу 4 с примерами инверсных морфизмов для различных слов.

Таблица 4: Примеры инверсных морфизмов

Слово	Инверсный морфизм
книга	агинк
дом	мод
солнце	ецнолос

Таким образом, использование полурешеток инверсных морфизмов позволяет эффективно обрабатывать тексты, преобразуя слова в их инверсные формы.

Формально, полурешетка инверсных морфизмов может быть представлена следующим уравнением (см. формулу 8.1):

$$\text{Морфизм}(x) = f(g(x)) \tag{8.1}$$

где x - входное слово, g - функция, выполняющая инверсию слова, а f - функция, выполняющая дополнительные преобразования.

Таким образом, использование полурешеток инверсных морфизмов открывает новые возможности для обработки естественного языка и может быть полезным инструментом в различных приложениях.

Выводы данного исследования указывают на значимость и эффективность полурешеток инверсных морфизмов для работы с языками и обработки текстов. Дальнейшие исследования в этой области могут включать более глубокий анализ свойств полурешеток и их применение в других областях теории автоматов и формальных языков.

8.2 Идеи для дальнейшего исследования

Одной из возможных направлений дальнейшего исследования является разработка более эффективных алгоритмов для работы с полурешетками инверсных морфизмов. В частности, можно исследовать возможность использования параллельных алгоритмов для ускорения работы с полурешетками.

Другим направлением исследования является разработка новых методов применения полурешеток инверсных морфизмов для решения конкретных задач в области обработки естественных языков. Например, можно исследовать возможность использования полурешеток инверсных морфизмов для задачи анализа тональности текстов или классификации текстов.

Также можно рассмотреть возможность использования полурешеток инверсных морфизмов для обработки многомодальных данных, таких как тексты в сочетании с изображениями или звуковыми данными. Разработка новых методов работы с многомодальными данными может привести к новым результатам в области обработки естественных языков и машинного обучения в целом.

Кроме того, предлагается рассмотреть возможность внедрения полурешеток инверсных морфизмов в практические приложения, такие как системы машинного перевода или автоматической обработки естественного языка. Это позволит оценить эффективность и применимость данного подхода на реальных задачах.

Глава 9 Интеграция обратных полурешеток с другими областями теории автоматов и формальных языков

В данном разделе мы рассмотрим возможности интеграции обратных полурешеток с другими областями теории автоматов и формальных языков, а также предложим направления для будущих исследований.

9.1 Интеграция с теорией грамматик

Теория грамматик является важной составляющей частью теории формальных языков. Грамматики могут быть использованы для генерации и анализа языков, и они также имеют применение в области синтаксического анализа и компиляции. Интеграция обратных полурешеток с теорией грамматик может привести к новым подходам для анализа и преобразования языков, основанных на грамматиках.

Для иллюстрации интеграции обратных полурешеток с теорией грамматик, рассмотрим контекстно-свободную грамматику (CFG) $G = (V, \Sigma, R, S)$, где V - множество нетерминальных символов, Σ - множество терминальных символов, R - множество продукционных правил, а S - стартовый нетерминальный символ.

Пусть $L = (E, \vee)$ - обратная полурешетка. Мы можем определить функцию $\phi : V \to E$, которая сопоставляет каждому нетерминальному символу $A \in V$ элемент из обратной полурешетки L. Затем, мы можем определить операцию $\vee_G : E \times E \to E$ следующим образом:

$$x \vee_G y = \begin{cases} x \vee y & \text{если } \exists A \to \alpha \in R \text{ такое, что } \phi(A) = x \vee y \\ \bot & \text{иначе} \end{cases} \tag{9.1}$$

где $\bot$ является минимальным элементом в L. Операция $\vee_G$ индуцирует новую структуру на множестве E, которая является обратной полурешеткой и отражает свойства грамматики G.

Используя данную конструкцию, можно исследовать свойства обратных полурешеток, связанные с грамматиками, такие как сохранение свойств обратных полурешеток при преобразованиях грамматик, или применять алгоритмы и методы из теории грамматик для анализа и преобразования обратных полурешеток.

9.2 Интеграция с алгеброй регулярных выражений

Алгебра регулярных выражений является мощным инструментом для представления и анализа регулярных языков. Обратные полурешетки могут быть интегрированы с алгеброй регулярных выражений для создания новых методов представления и преобразования языков с использованием регулярных выражений.

Рассмотрим обратную полурешетку L с набором элементов E и бинарной операцией $\vee$. Если мы имеем регулярное выражение r, которое представляет язык $L(r)$, мы можем определить новую операцию $\vee_r : E \times E \to E$ для каждого регулярного выражения r, так что для каждых двух элементов $x, y \in E$:

$$x \vee_r y = \begin{cases} x \vee y & \text{if } x, y \in L(r) \\ \bot & \text{otherwise} \end{cases} \tag{9.2}$$

где $\bot$ является минимальным элементом в L. Это преобразование обратной полурешетки в новую обратную полурешетку, которая отражает свойства регулярного языка, представленного r.

9.3 Интеграция с теорией автоматов

Теория автоматов изучает математические модели вычислений, которые могут быть использованы для анализа и распознавания языков. Обратные полурешетки могут быть интегрированы с теорией автоматов для создания новых методов анализа и преобразования автоматов, основанных на обратных полурешетках.

Для этого, мы рассмотрим детерминированный конечный автомат (ДКА) $A = (Q, \Sigma, \delta, q_0, F)$, где Q - множество состояний, Σ - алфавит, $\delta : Q \times \Sigma \to Q$ - функция перехода, $q_0 \in Q$ - начальное состояние и $F \subseteq Q$ - множество конечных состояний.

Пусть $L = (E, \vee)$ - обратная полурешетка. Мы можем определить функцию $\phi : Q \to E$, которая сопоставляет каждому состоянию автомата элемент из обратной полурешетки L. Затем, мы можем определить новую операцию перехода $\delta_L : E \times \Sigma \to E$ следующим образом:

$$\delta_L(x, a) = \begin{cases} \phi(\delta(q, a)) & \text{если } \phi(q) = x \\ \bot & \text{иначе} \end{cases} \tag{9.3}$$

где $\perp$ является минимальным элементом в L. Операция δ_L индуцирует новую структуру на множестве E, которая отражает динамику автомата A в контексте обратной полурешетки L.

Это предоставляет новый инструмент для анализа и преобразования автоматов, позволяя использовать свойства и методы обратных полурешеток для изучения структуры и поведения автоматов.

9.4 Будущие направления исследований

Возможные направления для будущих исследований включают:

- Исследование связи между обратными полурешетками и другими математическими объектами, такими как графы, сети и топологические пространства. Например, можно определить отображения между вершинами графа и элементами обратной полурешетки и изучить, как свойства графа отражаются в структуре обратной полурешетки.
- Разработка эффективных алгоритмов и методов для анализа и преобразования языков с использованием обратных полурешеток. Это может включать в себя разработку алгоритмов для оптимизации представления языков с использованием обратных полурешеток.
- Применение обратных полурешеток в практических задачах, таких как синтаксический анализ, компиляция, обработка естественного языка и обучение с подкреплением. В этом контексте, важно рассмотреть, как можно эффективно интегрировать обратные полурешетки с существующими методами и технологиями в этих областях.
- Разработка методов для анализа свойств обратных полурешеток, таких как компактность, размерность и качество представления языков. Это может включать в себя исследование метрик и критериев, которые могут быть использованы для оценки этих свойств.
- Изучение взаимосвязи обратных полурешеток с другими классами полурешеток и решеток, а также их применение в других областях математики и информатики. Это может включать в себя исследование обобщений и вариаций обратных полурешеток и их связи с другими математическими структурами.

Каждое из этих направлений представляет собой обширное поле для исследований, и они могут быть дополнительно расширены и уточнены в зависимости от специфических интересов и целей исследователя.

Заключение

В ходе данного исследования я успешно построил и проанализировал полурешетку обратных морфизмов для заданного конечного языка. Основываясь на описании исходного конечного языка, было построено регулярное выражение и соответствующий детерминированный конечный автомат.

Затем, я разработал методику построения полурешетки инверсных морфизмов, что включает вычисление обратных морфизмов для каждого состояния детерминированного конечного автомата и выбор инверсных морфизмов, которые являются обратными друг другу по отношению к конкатенации. Я также исследовал, как различные типы регулярных выражений влияют на построение примитивных конечных языков.

В ходе изучения свойств полурешетки, я обсуждал такие аспекты, как верхняя и нижняя грани, линейный порядок, алгебраические свойства, размерность, компактность, а также качество представления языка. Это дополнительно подтверждает эффективность и полезность предложенного мной подхода.

Далее, были реализованы различные алгоритмы, включая функцию формирования морфологического произведения, функцию положительного замыкания и функцию генерации языка MP+. Я также разработал функции для нахождения всех и обратимых отображений, что дополнительно иллюстрирует применение полурешетки в практических ситуациях.

На основе полученных результатов, я сравнил эффективность предложенного подхода с другими методами и обсудил возможные направления для дальнейшего исследования.

Наконец, я исследовал возможности интеграции обратных полурешеток с другими областями теории автоматов и формальных языков, такими как теория грамматик, алгебра регулярных выражений и теория автоматов. Это может открыть новые возможности для глубокого понимания свойств и принципов работы формальных языков и автоматов.

В целом, результаты моего исследования могут служить основой для развития новых исследовательских направлений и методик в области теории формальных языков.

Список литературы

[1] *Мельников Б.* О комплексе задач исследования необходимых условий равенства бесконечных итераций конечных языков // International Journal of Open Information Technologies. - 2023 - Vol. 11. No. 1. - P. 1-12.

[2] *Мельников Б., Мельникова А.* Бесконечные деревья в ал- горитме проверки условия эквивалентности итераций ко- нечных языков. Часть I // International Journal of Open Information Technologies. - 2021. - Vol. 9. No. 4. - P. 1-11.

[3] *Мельников Б., Мельникова А.* Бесконечные деревья в ал- горитме проверки условия эквивалентности итераций ко- нечных языков. Часть II // International Journal of Open Information Technologies. - 2021. - Vol. 9. No. 5. - P. 1-11.

[4] *Мельников Б.* Варианты конечных автоматов, соответ- ствующих бесконечным итерационным деревьям морфиз- мов. Часть I // International Journal of Open Information Technologies. - 2021. - Vol. 9. No. 7. - P. 1-13.

[5] *Мельников Б.* Варианты конечных автоматов, соответ- ствующих бесконечным итерационным деревьям морфиз- мов. Часть II // International Journal of Open Information Technologies. - 2021. - Vol. 9. No. 10. - P. 1-8.

[6] *Мельников Б.* Полиномиальный алгоритм проверки выпол- нения условия морфического образа расширенного макси- мального префиксного кода // International Journal of Open Information Technologies. - 2022. - Vol. 10. No. 12. - P. 1-8.

[7] *Мельников Б.* Полурешетки подмножеств потенциальных корней в задачах теории формальных языков. Часть I. Извлечение корня из языка // International Journal of Open Information Technologies. - 2022. - Vol. 10. No. 4. - P. 1-9.

[8] *Мельников Б.* Полурешетки подмножеств потенциаль- ных корней в задачах теории формальных языков. Часть II. Построение инверсного морфизма // International Journal of Open Information Technologies. - 2022. - Vol. 10. No. 5. - P. 1-8.

[9] *Мельников Б.* Полурешетки подмножеств потенциальных корней в задачах теории формальных языков. Часть III. Условие существования решетки // International Journal of Open Information Technologies. - 2022. - Vol. 10. No. 7. - P. 1-9.

[10] *Мельников Б., Мельникова А.* Полиномиальный алгоритм построения конечного автомата для проверки равенства бесконечных итераций двух конечных языков // International Journal of Open Information Technologies. - 2021. - Vol. 9. No. 11. - P. 1-10.

[11] *Абрамян М., Мельников Б.* Алгоритмы преобразования ко- нечных автоматов, соответствующих бесконечным ите- рационным деревьям // Современные информационные тех- нологии и ИТ-образование. - 2021. - Том 17. № 1. - С. 13- 23.

[12] *Мельников Б., Терентьева Ю.* Построение оптимального остовного дерева как инструмент для обеспечения устой- чивости сети связи // Известия высших учебных заведений. Поволжский регион. Технические науки. - 2021. - № 1 (57). - С. 36-54.

[13] *Мельников Б., Мельникова А.* Расширенный базисный конечный автомат. Часть I. Основные определения // International Journal of Open Information Technologies. - 2019. - Vol. 7, No. 10. - P. 1-8.

[14] *Мельников Б., Мельникова А.* Расширенный базисный конечный автомат. Часть II. Описание вспомогательных языков и некоторые свойства // International Journal of Open Information Technologies. - 2020. - Vol. 8, No. 5. - P. 1-7.

[15] *Мельников Б.* Регулярные языки и недетерминированные конечные автоматы (монография). - М., Изд-во Россий- ского государственного социального университета. - 2018. - 179 с. - ISBN 978-5-7139-1355-7.

[16] *Кузнецов С.* Базы данных. - М., Изд-во ВМК МГУ. - 2020. - 255 с.

Благодарность

Я выражаю свою благодарность научному руководителю Мельникову Борису Феликсовичу за его помощь в написании этой работы.