

Филиал Московского Государственного Университета  
имени М.В. Ломоносова в городе Ташкенте  
Факультет прикладной математики и информатики  
Кафедра прикладной математики и информатики

---

Бутузов Игорь Владимирович

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**НА ТЕМУ: «ПОИСК МАТЕМАТИЧЕСКИХ УТВЕРЖДЕНИЙ В БАЗЕ ДАННЫХ»  
ПО НАПРАВЛЕНИЮ 01.04.02 «ПРИКЛАДНАЯ МАТЕМАТИКА И  
ИНФОРМАТИКА»**

ВКР рассмотрена и рекомендована к защите  
зав. кафедрой «ПМиИ» д.ф.-м.н., профессор \_\_\_\_\_ Кудрявцев В.Б.

Научный руководитель  
к.ф.-м.н. \_\_\_\_\_ Калачев Г.В.

«\_\_\_\_\_» \_\_\_\_\_ 2020 год

Ташкент 2020

## Аннотация

Данная работа посвящена разработке программы, способной найти само утверждение или похожее на него в базе данных с математическими теоремами. Такая программа сможет по запросам пользователя ранжировать теоремы таким образом, чтобы теоремы, максимально подходящие под запрос, получали более высокий приоритет при выдаче результата.

В первой части работы приводятся результаты ранжирования теорем из базы с математическими теоремами при использовании 6 алгоритмов. Данная база утверждений состоит из 111 утверждений, которые поделены на 50 классов.

Вторая глава работы посвящена обзору алгоритмов, которые используются для вычисления схожести между текстовыми утверждениями. Также в этой главе освещаются методы выявления схожести математических формул без учета текста на естественном языке.

В третьей главе работы описана работа реализованной программы. Рассматриваются этапы её работы.

**Ключевые слова:** синтаксический анализ, теорема, функция ранжирования, математическая формула

## Abstract

This paper is devoted to the development of a program capable of finding the statement itself or similar to it in a database with mathematical theorems. Such a program will be able to rank the theorems at the user's request so that the theorems that are most suitable for the query receive a higher priority when issuing the result.

The first part of the paper presents the results of ranking theorems from a base with mathematical theorems using 6 algorithms. This statement base consists of 111 statements, the number of types of theorems is 50.

The second chapter of the paper is devoted to a review of algorithms that are used to calculate the similarity between textual statements. Also in this chapter, methods for identifying the similarity of mathematical formulas without regard to natural language text, are highlighted.

The third chapter of the work describes the work of the implemented program. The stages of its work are described.

**Keywords:** syntax analysis, theorem, ranking function

# Содержание

<b>Введение</b>	<b>3</b>
<b>Постановка задачи</b>	<b>4</b>
<b>1 Полученные результаты</b>	<b>6</b>
<b>2 Обзор существующих алгоритмов</b>	<b>8</b>
2.1 Методы выявления схожести математических формул без учета текста на естественном языке . . . . .	8
2.2 Описание используемых алгоритмов в программе . . . . .	9
2.2.1 Алгоритм основанный на преобразовании теоремы в абстрактное дерево и использование коэффициента Жаккара для выявления схожести . . . . .	9
2.2.2 Алгоритм поиска по ключевым словам . . . . .	9
2.2.3 Алгоритм основанный на преобразовании теоремы в абстрактное дерево и использование частотности слов для выявления схожести	10
2.2.4 Алгоритм Левенштейна . . . . .	11
<b>3 Описание системы математического поиска</b>	<b>12</b>
3.1 Описание базы данных с математическими утверждениями . . . . .	12
3.2 Описание модуля перевода текста, написанного на естественном языке, на язык формул логики предикатов . . . . .	12
3.3 Описание модуля преобразования математических формул . . . . .	14
3.4 Описание графического интерфейса программы . . . . .	15
<b>Заключение</b>	<b>17</b>
<b>Список литературы</b>	<b>18</b>

# Введение

В настоящее время доказываются огромное количество теорем и утверждений. Большинство этих теорем находятся в открытом доступе в сети Интернет. Однако поиск нужного утверждения в столь огромном массиве данных представляется нетривиальной задачей. В особенности это трудно, если автор использовал свою терминологию, а существенная часть утверждения задаётся сложной формулой, поскольку по формулам поисковые системы не ищут, а ключевые слова можно не знать. Для эффективности системы поиска необходимо использовать семантическую структуру утверждения.

Одну и ту же теорему можно записать несколькими способами. Например, изменив обозначения в используемых формулах на эквивалентные или изменив конструкцию, записанную на естественном языке, на конструкцию с тем же смыслом. По этой причине поиск по ключевым словам в теоремах может не дать желаемого результата. Если рассматривать только поиск по формулам, входящим в состав математического утверждения без учета текста на естественном языке, то такой поиск имеет ряд существенных недостатков:

- Формула может встречаться в довольно большом количестве теорем;
- Формула может быть записана при помощи других обозначений или другой формулой с тем же семантическим значением;
- Формула может быть составной частью другой формулы;

Формализация структуры математического утверждения способна повысить вероятность найти искомое утверждение. Для того, чтобы формализовать текст теорем на естественном языке удобно использовать язык логики предикатов. Однако в процессе осуществления такого перевода возникает ряд трудностей: такой процесс не может быть полностью автоматизирован, так как результат работы программы может сильно измениться из-за неверно истолкованного слова.

Полученное представление текста теоремы на языке логики предикатов является аналогом формулы и оно может быть представлено в виде абстрактного дерева. Объединив такое дерево с формулами из данного утверждения, предварительно представленными в виде деревьев схожей структуры, можно получить дерево исходного математического утверждения. К такому представлению формулы удобно применять алгоритмы работы с деревьями.

Результат перевода на язык логики предикатов будет более качественным, если входной текст для такого перевода будет структурирован и в нем будет содержаться вся необходимая для перевода информация [1],[2]. Такое представление текста могут дать синтаксические анализаторы. В данной работе используется синтаксический анализатор Stanza [3], для работы которого используется модель СинТагРус [4],[5]. Для преобразования математических формул в форму дерева, используется РБНФ-грамматика (расширенная форма Бэкуса — Наура [7]) и библиотека для Python 3 TatSu [6], которая позволяет по файлу, содержащему правила построения грамматики, построить дерево формулы.

# Постановка задачи

Целью данной работы является создание программы, способной найти само утверждение или похожее на него в базе данных с математическими теоремами.

Реализуемая система по введенному пользователю утверждению проведет ранжирование теорем, содержащихся в базе утверждений. В результате ранжирования, максимально подходящие под запрос теоремы, получают более высокий приоритет при выдаче результата.

Актуальность данной темы заключается в том, что ежегодно доказывается несколько сотен тысяч теорем и других математических утверждений. Поиск нужного утверждения в столь огромном массиве данных представляется нетривиальной задачей.

В отличие от поиска по ключевым словам реализуемая система будет иметь более высокий результат, так как будет опираться не на вхождение определенных слов в запрос, а на смысловое содержание введенной теоремы. Поэтому такая система математического поиска сможет сопоставлять теорему, которую ввел пользователь и теорему из базы данных, даже если они записаны при помощи разных формулировок.

Предполагается, что как заданная теорема, так и теоремы из базы данных записаны на естественном языке (возможно, с использованием математических формул, представленных в формате системы TeX [8]). При этом в программе должно учитываться, что формулировки заданной теоремы и той же теоремы в базе данных могут несколько отличаться.

В процессе работы было решено несколько промежуточных задач:

1. Задача перевода текста, написанного на естественном языке, на язык формул логики предикатов (с помощью синтаксического и семантического анализа), и последующего представления этих формул в виде деревьев;
2. Задача представления в том же виде формул, записанных в формате системы TeX;
3. Задача ранжирования формул базы данных, записанных в виде деревьев, по степени похожести на формулу теоремы-запроса.

Кроме того, для проверки работоспособности программы необходима база теорем, а также некоторое множество теорем, выступающих в качестве запросов к программе.

В качестве входных данных для данной программы используется синтаксический анализ Stanza. Программа реализована на языке Python 3[9],[10]. Результатом работы программы является представление базы математических утверждений, в которых формулировка искомой теоремы или похожей на нее должна появиться среди лучших кандидатов, найденных программой.

Сформулируем основные понятия, используемые в данной работе:

*Токен* — слово, устойчивое выражение или знак препинания. Токен имеет словоформу, лемму и набор морфологических характеристик.

*Синтаксическое дерево зависимостей* — способ представления синтаксической структуры предложения. В узлах дерева расположены токены данного предложения [11]. Под словами «родительский» и «дочерний» по отношению к токенам в данной работе следует понимать отношения между соответствующими узлами дерева. Каждому токenu (исключение составляет корень синтаксического дерева) поставлено в соответствие положительное число — номер другого токена, который является родителем для данного. Для корня синтаксического дерева это число равно нулю. Между «родительским» и «дочерним» токенами устанавливается синтаксическое отношение, его мы будем называть типом связи. У корня синтаксического дерева родителя нет.

*Словоформа* — форма слова, полученная из основной части слова с помощью склонения и спряжения.

*Лемма* — словарная форма слова.

*Коэффициент Жаккара* [12] - мера сходства, предложенная Полем Жаккарсом в 1901 году. Для множеств  $A$  и  $B$  коэффициент Жаккара вычисляется по следующей формуле:

$$K = \frac{n(A \cap B)}{n(A \cup B)}, \text{ где } n(C) - \text{ количество элементов в множестве } C$$

Если оба множества состоят из одинаковых элементов, то коэффициент  $K$  равен 1. Чем больше различий между множествами, тем значение  $K$  ближе к 0.

*Расстояние Левенштейна* - метрика, измеряющая разность между двумя последовательностями символов. Она определяется как минимальное количество односимвольных операций (а именно вставки, удаления, замены), необходимых для превращения одной последовательности символов в другую.[13]

*Косинусное сходство* - мера сходства между двумя векторами, которая используется для измерения косинуса угла между ними. Если даны два вектора признаков,  $A$  и  $B$ , то косинусное сходство,  $\cos(\theta)$ , может быть представлено используя скалярное произведение и норму:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

Будем считать, что косинусное сходство принимает значения от 0 до 1 и чем ближе его значение к 1, тем больше сходства между рассматриваемыми объектами.

# 1 Полученные результаты

В результате данной работы была реализована программа, которая способна найти само утверждение или похожее на него в базе данных с математическими теоремами.

Данная программа может по запросам пользователя ранжировать теоремы таким образом, чтобы теоремы, максимально подходящие под запрос, получали более высокий приоритет при выдаче результата.

Работа программы тестировалась на базе утверждений, состоящей из 111 теорем. Среди данных теорем встречаются разные формулировки одних и тех же теорем. Каждой теореме соответствует свой отдельный класс. Общее количество таких классов для данной базы данных равно 50. Исследовалось качество работы 6 алгоритмов.

Качество работы алгоритма оценивалось по двум метрикам.

## Метрика 1.

Для тестирования каждой теореме присваивался свой уникальный номер. Номера формулировок одних и тех же теорем объединялись в некоторые множества номеров. Допустим, нам известно, что формулировки теоремы Больцано-Коши имеют номера 1,2. Соответственно, множество номеров для данной теоремы  $\{1, 2\}$ . Однако если номер тестируемой теоремы в базе утверждений равен 1 или 2, то он исключается из множества номеров.

1. Для каждой теоремы из файла, содержащего утверждения теорем, строилась своя формула и данная формула заносилась в базу данных. Таким образом, строилась база формул с которой и происходило сравнение заданной формулы и каждого алгоритма.

2. Формула для которой проводилось ранжирование сравнивалась Алгоритмами 1-6 с остальными формулами базы данных. Таким образом, строилась таблица ранжирования для данной формулы.

3. Каждый индекс формулы в таблице ранжирования сравнивался с множеством номеров, содержащих номер заданной теоремы. Если текущего индекса в таблице ранжирования нет в массиве номеров, то накладывался штраф, равный 1. Чем меньше штрафов было получено, тем лучше качества ранжирования.

Опишем алгоритм на примере. Имеется база данных формул в количестве 111 элементов.  $A_i$  ( $i = 1..111$ ). Для каждого номера  $i$  известны индексы той же самой теоремы, но в другой формулировке. Предположим, что для теоремы с номером  $i$  теоремой с другой формулировкой будут теоремы с индексами  $j$  и  $k$  ( $j \neq i, j \neq k, j = 1..111$ ). Для данной теоремы множество номеров равно  $\{j, k\}$ .

Выберем произвольный номер  $n$  ( $n = 1, ..., 111$ ). Сравним разбор теоремы  $A_n$  с каждой имеющейся формулой  $A_i$  в базе данных. Получим таблицу ранжирования с индексами формул  $p_1, p_2, ..., p_{111}$ .

Величина текущего штрафа будет храниться в переменной *Penalty*.

$m = 1, 2, 3, \dots, 110, 111 :$

Если  $p_m \neq j$  и  $p_m \neq k$ , то  $Penalty = Penalty + 1$

Если  $p_m = j$  или  $p_m = k$ , то штраф не начисляется.

Если найдены теоремы с индексами  $j$  и  $k$ , то алгоритм завершает свою работу.  
Итоговая оценка для данной теоремы по метрике 1 равна:

$$Result_1 = 1 - \frac{Penalty}{size(A)}$$

Пусть результат работы функции ранжирования  $[A_1, B_1, A_2, A_3, B_2, B_3, A_4, \dots, B_{111}]$ , где  $A$  формулировка нужной нам теоремы. В данном примере за неверно ранжированные теоремы  $B_1$ ,  $B_2$  и  $B_3$  будет начислен штраф, равный 3. Итоговая оценка по метрике 1 для данного примера равна:  $1 - \frac{3}{111} \approx 0.972$

## Метрика 2.

Опишем алгоритм на примере. Пусть результат работы функции ранжирования  $[A_1, B_1, A_2, A_3, B_2, B_3, A_4, \dots, B_{111}]$ . Найдем номера всех теорем  $A$ , формулировки которых мы ищем, кроме самой первой. В данном примере искомые теоремы имеют номера 3,4,7. Если бы алгоритм отработал точно, то номера искомых теорем были бы равны 2,3,4. Оценка для данного примера равна:

$$Result_2 = 1 - \frac{3 + 4 + 7 - 2 - 3 - 4}{3 \cdot size(A)} \approx 0.984$$

№ алгоритма	Метрика 1	Метрика 2	Описание алгоритма
Алгоритм 1	0.951	0.975	Мера Жаккара*.
Алгоритм 2	0.949	0.977	Поиск при помощи ключевых слов
Алгоритм 3	0.904	0.950	Ключевые слова*
Алгоритм 4	0.768	0.889	Расстояние Левенштейна (текст)
Алгоритм 5	0.847	0.927	Косинусное сходство (текст)
Алгоритм 6	0.867	0.938	Сходство по метрике Жаккара (тексту)

Таблица 1: Результаты для 111 теорем. Значения Метрика 1 и Метрика 2 получены как средние значения  $Result_1$  и  $Result_2$  к общему количеству теорем соответственно. \* - в алгоритмах 1 и 3 теоремы представлялась в виде абстрактных деревьев (список вложенных списков). Для алгоритма 1 для ранжирования использовалась мера Жаккара, для алгоритма 3 частота встречаемости слов

Для данной выборки алгоритмы 1 и 2 показали приблизительно одинаковые результаты, что говорит о том, что алгоритм 1 не уступает по качеству поиску по ключевым словам.



## 2 Обзор существующих алгоритмов

### 2.1 Методы выявления схожести математических формул без учета текста на естественном языке

Как отмечалось ранее, поиск по формулам, входящим в состав математического утверждения без учета текста, записанного на естественном языке имеет ряд существенных недостатков: например, формула может встречаться в нескольких теоремах, она может быть записана при помощи других обозначений, другой формулой с тем же семантическим значением или являться частью другой более длинной теоремы.

Однако методы, используемые при таком поиске, могут быть востребованы, если получится представить математическое утверждение вместе с формулой и текстом на естественном языке в виде некоторой обобщенной формулы.

В работе [14] исследуется структурно-синтаксическое сходство математических выражений. Основное внимание уделяется модификации алгоритма наложения деревьев зависимостей для математических выражений, представленных в формате MathML. Рассмотрено применение трех алгоритмов: Tree edit distance, Subpath set и Tree overlapping algorithm.

Tree edit distance ( Расстояние редактирования дерева ) определяет схожесть между двумя деревьями как число операций добавления, удаления и изменения для того, чтобы преобразовать одно дерево в другое.

Также в данной работе рассмотрен алгоритм Subpath set ( Множество подпутей ). Подпутем называется множество всех путей из корневой вершины к остальным вершинам, включая промежуточные пути. Коэффициент схожести в данном алгоритме определяется как число общих подпутей у рассматриваемых деревьев.

Третьим представленным алгоритмом является модификация Tree overlapping algorithm ( Алгоритм наложения деревьев ). Данный алгоритм может быть описан следующим образом: при наложении произвольной вершины  $n_1$  дерева  $T_1$  на вершину  $n_2$  дерева  $T_2$  может возникнуть одинаковое продукционное правило наложения деревьев  $T_1$  и  $T_2$ . Под схожестью в данном алгоритме понимается количество таких продукционных правил.

В работе [15] описан подход сравнения двух математических формул, представленных в формате MathML, при помощи чередования прямого и обратного обходов деревьев. Эксперимент показал, что описанный алгоритм не только определяет сходство математических выражений по одинаковой структуре, но и семантическое сходство с высокой точностью.

В работе [16] предлагается использовать язык разметки MathML и ранее описанный алгоритм Subpath set.

## 2.2 Описание используемых алгоритмов в программе

### 2.2.1 Алгоритм основанный на преобразовании теоремы в абстрактное дерево и использование коэффициента Жаккара для выявления схожести

1. На вход алгоритму подается утверждение теоремы или таблица (массив), содержащая порядковый номер токена, его словоформу, лемму, номер родительского токена, тип синтаксической связи и часть речи. Предварительно, все формулы в данном утверждении заменены на обозначение «formula\_№», где - порядковый номер в списке формул.

2. При помощи специально написанного словаря группы токенов, образующих устойчивое выражение объединялись в один. При помощи словаря с синонимами токены и группы токенов заменялись на общее обозначение.

3. Дерево синтаксического разбора перестраивалось таким образом, чтобы вершины дерева с обозначениями forall , if , then ,  $\leq$  , exists стали вершинами максимально возможных по размеру поддеревьев.

4. По дереву синтаксического разбора строится список вложенных списков (массив вложенных массивов), где вершиной списка является родительская вершина.

5. В данный список вместо обозначений «formula\_№» подставляется дерево разбора соответствующей формулы.

6. Формула для данной теоремы в виде списка вложенных списков при помощи рекурсивным образом разбивается на более мелкие подформулы, которые также представляют собой вложенные списки.

7. Для каждой полученной на предыдущем шаге подформулы строится множество  $A_i$  всех путей от корневой вершины до всех остальных, включая все промежуточные пути. Все множества  $A_i$  добавляются в множество  $A$ . При помощи коэффициента Жаккара множество  $A$  сравнивается с аналогично построенным множеством  $B$ .

8. Результат сравнения формулы  $A$  с каждой теоремой из базы данных заносится в отдельный массив. Затем полученный массив (таблица ранжирования) сортируется по убыванию.

**Недостатки алгоритма.** Необходимость использования синтаксического анализатора, зависимость от качества его работы.

### 2.2.2 Алгоритм поиска по ключевым словам

На вход алгоритма подается текстовое утверждение теоремы. Метод опирается на использование весовых характеристик слов и словосочетаний. Данная характеристика определяется исходя из частоты появления слов и словосочетаний в базе утверждений.

Два предложения будут максимально похожи, если у них имеется большое количество общих слов или групп слов с высоким весом.

1. Проводим индексирование базы данных. Для каждого слова из базы утверждений определяем его лемму.

2. Для каждого утверждения  $R$  выписываем частоту появления каждого слова. При помощи словаря синонимов, в котором слова разбиты на классы эквивалентности вместо самого слова записывается его класс эквивалентности. Для каждого слова  $A$  получим величину  $N(R, A)$ .

3. Такую же информацию сохраняем для словосочетаний из двух слов. Словосочетание  $AB$  учитывается, если  $A$  и  $B$  не предлоги, между словами  $A$  и  $B$  нет знаков препинания. Если между  $A$  и  $B$  есть  $k$  других слов без учёта предлогов, то словосочетание учитывается с весом  $\frac{1}{k}$ . Словосочетание  $BA$  также добавляется с весом  $\frac{1}{2k}$ . Для каждой пары слов  $A, B$  получим величину  $N(R, AB)$ .

4. Для каждого слова и словосочетания  $X$  суммируем величины  $N(R, X)$  по всем утверждениям  $R$  из базы данных, получим сумму  $S(X)$ . Слову или словосочетанию  $X$  присваивается вес  $W(X) = \frac{1}{S(X)}$ .

5. При поиске для данного запроса также вычисляются количество раз, которое встретилось каждое слово и словосочетание (тем же методом). После этого ищется утверждение с наиболее похожими характеристиками.

6. Схожесть  $A$  и  $B$  с весами  $A_i$  и  $B_i$  вычислим по формуле

$$J_w(A, B) = \frac{\sum_i \min(A_i, B_i)}{\sum_i \max(A_i, B_i)}$$

Элементы формул также считаются словами. Слова из английских букв и числа выделяются, как целые слова, а остальные символы считаются отдельными словами или знаками препинания.

**Недостатки алгоритма.** Низкая скорость работы, необходимость считать частоту встречаемости слов и словосочетаний по всей базе данных.

### 2.2.3 Алгоритм основанный на преобразовании теоремы в абстрактное дерево и использование частотности слов для выявления схожести

1. Повторяются шаги 1-5 алгоритма основанного на преобразовании теоремы в абстрактное дерево и использования коэффициента Жаккара для выявления схожести.

2. Так как представление теоремы в виде списка вложенных списков является деревом, то из этого списка построим для каждой формулы множество всех путей из корневой вершины ко всем остальным, включая промежуточные пути.

3. Для слов из которых состоят полученные на предыдущем шаге пути определим весовую характеристику. Она будет основана на частоте появления слов во всех множествах путей каждой теоремы из базы данных. В результате будет получен словарь в котором каждому слову будет сопоставлена весовая характеристика.

4. При помощи данного словаря каждому множеству путей будет сопоставлен свой словарь с весовыми характеристиками.

5. Пусть имеются формулы  $A$  и  $B$  с наборами весов  $A_w$  и  $B_w$  соответственно. Построим множество  $C$  в котором будут содержаться все уникальные элементы из  $A$  и  $B$ . Для всех элементов  $x$  из множества  $C$  найдем минимальное и максимальное значение весов в  $A_w$  и  $B_w$ . Отношение суммы всех минимумов к сумме всех всех максимумов и будет числом, отражающим схожесть формул  $A$  и  $B$ .

**Недостатки алгоритма.** Необходимость использования синтаксического анализатора, зависимость от качества его работы. Также более низкий по сравнению с ранее описанными результат на данный момент.

## 2.2.4 Алгоритм Левенштейна

Расстояние Левенштейна называется метрика, измеряющая разность между двумя последовательностями символов. Она определяется как минимальное количество операций вставки, удаления, замены одного символа, необходимых для превращения одной последовательности символов в другую. Элементы строк в данном алгоритме нумеруются с первого элемента.

Пусть  $s_1$  и  $s_2$  — две строки длиной  $M$  и  $N$  соответственно над некоторым алфавитом, то расстояние Левенштейна  $d(s_1, s_2)$  можно подсчитать по формуле

$$d(s_1, s_2) = D(M, N), \text{ где}$$

$$D(i, j) = \begin{cases} 0, & i = 0, j = 0 \\ i, & j = 0, i > 0 \\ j, & i = 0, j > 0 \\ \min\{ & \\ & D(i, j - 1) + 1, \\ & D(i - 1, j) + 1, \\ & D(i - 1, j - 1) + m(s_1[i], s_2[j]) \\ \} & j > 0, i > 0 \end{cases},$$

где  $m(a, b) = 0$ , если  $a = b$ ,  $m(a, b) = 1$ , если  $a \neq b$   
 $\min\{a, b, c\}$  возвращает наименьший из аргументов.

Здесь шаг по  $i$  символизирует удаление (D) из первой строки, по  $j$  — вставку (I) в первую строку, а шаг по обоим индексам символизирует замену символа (R) или отсутствие изменений (M).

**Недостатки алгоритма.** Скорость работы, самое низкое качество работы из всех рассматриваемых алгоритмов.

## 3 Описание системы математического поиска

### 3.1 Описание базы данных с математическими утверждениями

Для тестирования работы реализованной программы были выбраны 111 теорем на русском языке из википедии и различных учебных пособий. Теоремы были взяты из математического и функционального анализа.

1. Все теоремы были разделены на классы эквивалентности. Общее количество классов равно 50. В отдельный файл заносится принадлежность теорем к определенному классу эквивалентности

2. Программным способом разделялись текстовая часть утверждения и содержащиеся в них математические формулы на языке TeX. Формулы добавлялись в отдельный текстовый файл, а в самих теоремах заменялись на специальные обозначения вида `formula_№`, где № – порядковый номер формулы в данном файле.

К каждому преобразованному таким образом математическому утверждению применялся синтаксический анализатор Stanza, который создавал таблицу для каждого слова со следующими полями:

№	Слово	Лемма	Номер родителя	Тип синтаксической связи	Часть речи
---	-------	-------	----------------	--------------------------	------------

Для преобразования математических формул в абстрактное синтаксическое дерево используется РБНФ-грамматика (расширенная форма Бэкуса — Наура) и библиотека для Python 3 TatSu, которая позволяет по файлу с правилами грамматики построить дерево.

Для Алгоритмов 1 и Алгоритмов 3 по таблице с синтаксической и морфологической информацией строится представление теоремы в виде списка вложенных списков, в котором обозначения `formula_№` заменены на соответствующие разборы этих формул. Данные списки вложенных файлов заносились в текстовый файл.

### 3.2 Описание модуля перевода текста, написанного на естественном языке, на язык формул логики предикатов

Входными данными для данного модуля являются предложения, записанные на естественном языке. Математические формулы в данном предложении заменены на обозначения типа «`formula_№`», где № – номер соответствующего обозначения или формулы в файле со списком обозначений и формул. Выходными данными для данного модуля являются исходное предложение, записанное при помощи логики предикатов.

Входные данные подаются синтаксическому анализатору Stanza. Результат работы анализатора представляет собой таблицу (массив), содержащий подробную морфологическую и синтаксическую информацию, соответствующую обозначениям Национального Корпуса Русского Языка (СинТагРус).

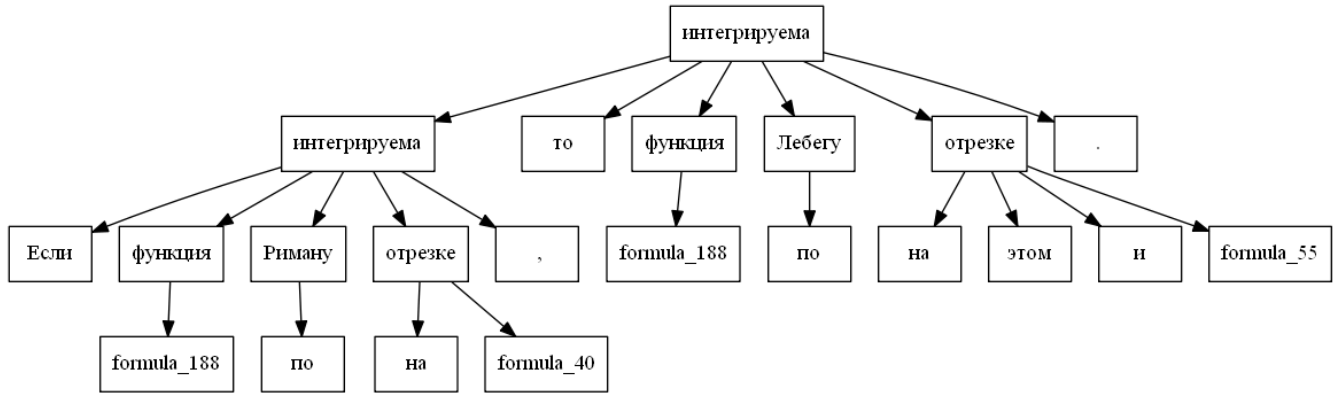


Рис. 1: Дерево синтаксического разбора теоремы

Каждый токен входного предложения представляет собой отдельный массив в данной таблице. В данном массиве содержится порядковый номер токена, его словоформа, лемма, номер родителя, тип связи между родителем и данным токеном, часть речи данного токена.

Для дальнейшей работы в полученную таблицу необходимо внести корректировки.

На первом этапе несколько отдельных токенов объединяются в один и заменяются либо на соответствующее название на английском языке, либо на стандартное математическое обозначение. Несколько подряд идущих токенов в таблице токенов. Например, токены, образующие конструкцию вида «интегрируема по Риману», будут заменены одним токеном  $R$ . Конструкция «знакопередающийся числовой ряд» будет заменена на конструкцию «alternating\_series», а конструкция «тогда и только тогда, когда» будет заменена на токен « $\Leftrightarrow$ ». При каждой такой замене происходит пересчет номеров родителей для дочерних токенов.

Для математических терминов написан словарь, где указаны соответствующая лемма термина и эквивалентное ему слово, на которое его можно заменить. Если произошло совпадение токена и значения в словаре, то будет произведена замена.

Дерево синтаксического разбора перестраивалось таким образом, чтобы вершины дерева с обозначениями forall, if, then,  $\Leftrightarrow$ , exists стали вершинами максимально возможных по размеру поддеревьев. Например, для теоремы: Если функция  $f$  интегрируема по Риману на отрезке  $[a, b]$ , то функция  $f$  интегрируема по Лебегу на этом

отрезке и  $\int_{[a,b]} f \cdot d\mu = \int_a^b f(x)dx$  синтаксическое дерево изображено на рисунке 1.

А ее представление в виде списка вложенных списков на рисунке 2.

По дереву синтаксического разбора строится список вложенных списков (массив вложенных массивов), где вершиной списка является родительская вершина.



### 3.4 Описание графического интерфейса программы

Для удобства использования всех модулей программы написан графический интерфейс. Интерфейс представляет собой 4 текстовых поля и ряд кнопок:

#### Текстовые поля

- 1.1. Поле ввода используется для ввода информации.
- 1.2. Два поля вывода данных нужны для вывода результата работы.
- 1.3. Поле вывода ошибок используется для вывода ошибок, возникших в результате работы.

#### Кнопки

- 2.1. Clean - очистка всех текстовых полей кроме поля ввода.
- 2.2. Load\_and\_run. После нажатия данной кнопки произойдет обращение к файлу конфигурации, будет найдено поле Infile. К соответствующему этому полю пути будет добавлен путь корневой директории. Данные из файла, находящегося по данному пути ( в качестве данных могут выступать текстовое утверждение или массив с синтаксической информацией) будут преобразованы в формулу данного утверждения. Результат разбора будет выведен в первое поле вывода данных. Входные данные для разбора будут выведены во второе поле вывода данных.
- 2.3. Parse - данные из поля ввода (текстовое утверждение или массив с синтаксической информацией) будут преобразованы в формулу данного утверждения. Результат разбора будет выведен в первое Поле вывода данных.
- 2.4. Rank - запуск ранжирования. Данные из поля ввода (текстовое утверждение или массив с синтаксической информацией) будут преобразованы в соответствующую формулу данного утверждения. Полученная формула будет сравниваться с каждой формулой из базы формул. Чем выше результат сравнения (от 0 до 1), тем выше формула из базы формул будет в итоговой выдаче. Отсортированные формулы из базы формул будут выведены в первое поле вывода данных. Результаты сравнения будут выведены во второе поле вывода данных.
- 2.5. Test - запуск тестирующего модуля.
- 2.6. Picture - модуль отрисовки формул. Входные данные должны быть в формате списка вложенных списков. Данный модуль отрисовывает данные из поля ввода в отдельном окне.
- 2.7. Edit\_config - редактирование файла конфигурации. Данные из данного файла будут выведены во второе поле вывода данных. Для сохранения изменений необходимо нажать кнопку Save\_config.



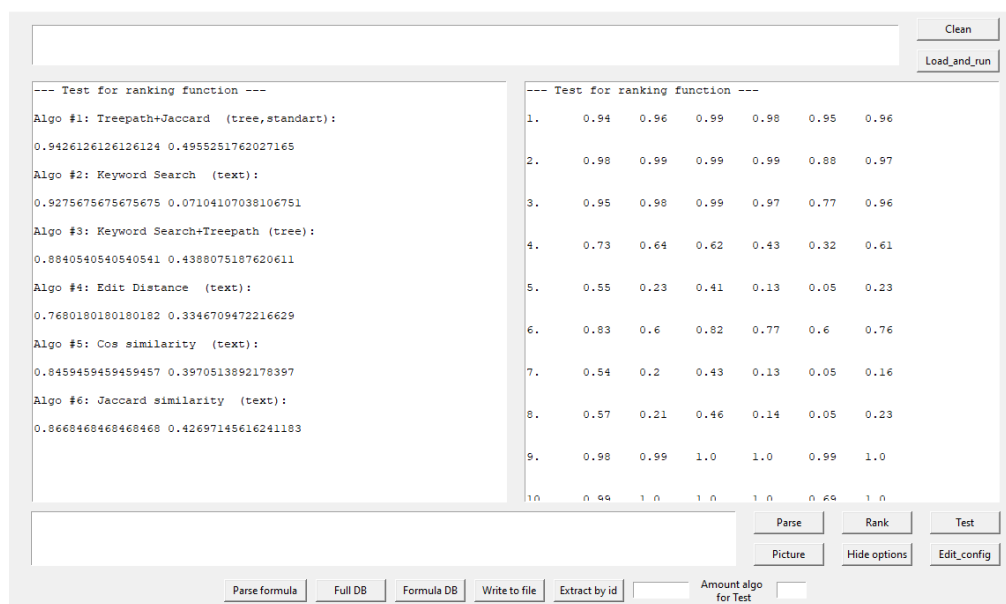


Рис. 3: Графический интерфейс программы

2.8. Parse\_formula - осуществляет преобразование формулы из текстового представления в виде дерева.

2.9. Full\_DB - создает базу данных

2.10. Formula\_DB - создает базу разбора формул.

## Заключение

Результаты, полученные в ходе выполнения проекта, можно резюмировать следующим образом:

1. Написана и протестирована программа, способная найти само утверждение или похожее на него в базе данных с математическими теоремами.

Данная работа весьма актуальна, потому что система, способная искать похожие математические утверждения в базах данных, написанные с использованием естественного языка и формул на языке TeX, сможет найти применение в любой сфере деятельности, где используется математический язык.

2. Для тестирования работы программы были выбраны 111 математических теорем из разделов математики. При тестировании результат проверялось сможет ли реализованная программа найти не только саму себя, но и другую формулировку этой же теоремы.

Несмотря на хороший результат программу нужно улучшать для разбора большего количества теорем, пополнять внутренние словари, повышать качество их разбора. Также дополнительный интерес представляет дальнейшее совершенствование алгоритма ранжирования.

## Список литературы

- [1] Hiroyuki Yamauchi. Processing of syntax and semantics of natural language by Predicate Logic. Institute of Space and Aeronautical Science, University of Tokyo. <https://www.aclweb.org/anthology/C80-1059>
- [2] Julian E. Herwitz. Natural Language to Predicate Logic [https://www.cs.rochester.edu/~brown/173/exercises/samples/ParseTranslate10\\_2.pdf](https://www.cs.rochester.edu/~brown/173/exercises/samples/ParseTranslate10_2.pdf). Страница 5.
- [3] <https://stanfordnlp.github.io/stanza/>
- [4] <http://www.ruscorpora.ru/new/instruction-syntax.html>
- [5] Апресян Ю. Д., Богуславский И. М., Иомдин Б. Л. и др. Синтаксически и семантически аннотированный корпус русского языка: современное состояние и перспективы // Национальный корпус русского языка: 2003—2005. М.:Индрик, 2005, 193—214.
- [6] <https://tatsu.readthedocs.io/en/stable/intro.html>
- [7] Информационная технология — Синтаксический мета-язык — Расширенная Форма Бэкуса-Наура (Extended BNF) <https://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf> Перевод Андрея Бушмана. ISO/IEC 14977 [https://docs.google.com/file/d/0B7H\\_2Cq9tBXdd2prejdTRlJlSDA/edit](https://docs.google.com/file/d/0B7H_2Cq9tBXdd2prejdTRlJlSDA/edit)
- [8] Кнут Д. Э. Все про TeX = The TeXbook / Пер. с англ. М. В. Лисиной. — Протвино: АО R<sup>D</sup>TeX, 1993. — 592 с.
- [9] David Beazley, Guido Van Rossum. Python: Essential Reference. — New Riders Publishing, 1999.
- [10] <https://www.python.org/>
- [11] *Большакова Е.И., Воронцов К.В., Ефремова Н.Э., Клышинский Э.С., Лукашевич Н.В., Сапин А.С.* Автоматическая обработка текстов на естественном языке и анализ данных: учеб. пособие — М.: Изд-во НИУ ВШЭ, 2017. — 269 с. Страницы 25-28.
- [12] Jaccard P. Distribution de la flore alpine dans le Bassin des Dranses et dans quelques regions voisines // Bull. Soc. Vaudoise sci. Natur. 1901. V. 37. Bd. 140. S. 241—272.
- [13] В. И. Левенштейн. Двоичные коды с исправлением выпадений, вставок и замещений символов. Доклады Академий Наук СССР, 1965. 163.4:845-848.
- [14] Evgeny Pyshkin. University of Aizu, Japan. Mathematical Equation Structural Syntactical Similarity Patterns: A TreeOverlapping Algorithm and Its Evaluation <https://pdfs.semanticscholar.org/53d4/e6043e49e42cbd90fc86185a6c7af6030bbd.pdf>
- [15] Yuping Qin, Junnan Guo, Aihua Zhang. A Mathematical Formula Matching Algorithm Based on MathML <https://www.atlantispress.com/proceedings/lemcs-15/25838389>
- [16] An Approach to Similarity Searchfor Mathematical Expressions using MathML [https://dml.cz/bitstream/handle/10338.dmlcz/702557/DML\\_002-2009-1\\_5.pdf](https://dml.cz/bitstream/handle/10338.dmlcz/702557/DML_002-2009-1_5.pdf)