Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В. Келдыша
Российской академии наук

Ю.С. Корухова Р.В. Широкий

Система поиска научных текстов по формулам

Москва
2014

Оглавление

Аннотация

Человечество за свое существование успело накопить огромное количество научных знаний. Настоящий пик накопления научных знаний произошел в новейшее время, а с появлением интернета научные знания стали систематизировано храниться в электронных хранилищах. Учитывая обширный объем подобных хранилищ, само собой встает задача поиска по документам, хранящихся в них. Но научные тексты содержат большое количество элементов, плохо подходящих для обычного текстового поиска: графики, таблицы, изображения и формулы. В данной работе речь пойдет именно о поиске по формулам, когда поисковый запрос будет являться некой формулой, а ожидаемый результат - это список документов, где встречаются такая формула или подобные ей. В работе предложена метод поиска, позволяющий находить формулы, имеющие различный вид, но при этом тождественные. Приведено описание программной системы, в которой реализован предложенный метод.

Работа выполнена при поддержке Российского Фонда Фундаментальных Исследований, проект №14-01-00679.

1 Введение

В настоящее время есть огромные архивы научных статей, относящихся к самым разным научным областям. Для таких архивов актуально наличие поисковых систем. Известные поисковые системы умеют находить статьи по названиям, авторам или ключевым словам. Для статей в математических и близких к ней областях возникает необходимость поиска текстов, содержащих заданную формулу. Например, при поиске уже вычисленных оценок той или иной величины в области математической статистики или при поиске значений некоторого интеграла нам проще задать исходными данными для поиска формулу, чем сформулировать текстовый запрос. Тем самым идея поиска по формулам или математическим выражениям[1] достаточно очевидна, но реализация системы, выполняющей такую функцию, сопряжена с рядом проблем: распознавание, извлечение и обработка математических выражений, определение релевантности элемента поисковой выборки к запросу, ранжирование. Причем релевантностью в контексте математического поиска логично считать тождественность одного МВ другому. Тогда для определения степени релевантности элемента выдачи поисковому запросу или другими словами для определения степени тождественности одной формулы к другой стоит произвести попытку доказать тождественность двух формул, используя способы автоматического доказательства теорем. Важно отметить, что вообще говоря два МВ очевидно могут быть либо тождественны, либо не тождественны друг другу, но для целей математического поиска такая бинарная оценка мало применима, поэтому необходимы определение и способ оценки промежуточных значений тождественности.

Форматы машинного представления формул

Форматы математических выражений можно разделить на два вида: представления, передающие лишь внешний вид и *семантические* представления, передающие структуру выражения и назначение его элементов, что можно назвать *смыслом* выражения.

Краткий обзор видов машинного представления МВ

1. **Растровое изображение**
   Использование растровых изображений для машинного представления МВ имеет долгую историю. На интернет страницах и в электронных документах до сих пор распространено использование растровых изображений для отображения формул. Такой вид машинного представления МВ неудобен для обычных пользователей: сложность копирования и передачи, сложность модификации, некачественное масштабирование - ведь изменение размеров растрового изображения приводит к потере качества, что особенно заметно

[1] Далее будет использоваться сокращение МВ

на тексте. Также изображение формулы сложно обрабатывать и для машины. Но существуют работы, призванные решить и эту задачу. Конкурс CHROHME[23], посвященный соревнованию решений для автоматического распознавания МВ, выявляет достаточно хороший процент распознавания среди участников. Используемые методы, как например в работе[9], могут быть расширены до перевода из растрового изображения формулы в ее семантическое представление, понятное машине. Использование такой системы, обеспечивающей сравнительно высокий процент корректно конвертированных МВ, позволило бы проиндексировать практически любые в том числе бумажные хранилища научных текстов.

2. **TeX**
   Старый и широко используемый типографский формат с крупным объемом созданных документов. Формат нашел новое применение и для отображения формул на веб-страницах, для чего используются специальные инструменты, как например MathJax[15].

3. **MathML**
   Вышеописанное разделение форматов выражено в основанном на XML стандарте MathML, имеющем две версии: Presentation MathML и Content MathML, которые используют разные наборы тегов. Presentation MathML также как и TeX несут в себе информацию о том, каким образом должно выглядеть МВ и являются наиболее распространенными форматами. Presentation MathML изначально создавался для отображения МВ на веб-страницах и содержит информацию о том, как нужно отобразить МВ. Content MathML содержит в удобном для машины виде информацию о смысле МВ, другими словами формат позволяет записать дерево выражения и значение его термов. Формат получил узкое распространение в специализированных системах, системах компьютерной алгебры и автоматического доказательства, как составная часть для платформонезависимых форматов данных[17].

4. **OpenMath**
   Стандарт OpenMath[16] похож на Content MathML, но в отличие от него не имеет фиксированного набора токенов для математических элементов, но вместо этого использует Content Dictionaries - словари токенов и их значений. Такие словари записываются в удобной для машинной обработки форме в формате XML.

Для определения тождественности двух выражений вообще говоря необходимо знать деревья обоих выражения. Таким образом наиболее удобным представлением МВ для данной задачи является формат, передающий семантическое представление формулы, поэтому в данной работе основным форматом выбран Content

MathML, так как он имеет ограниченный набор тегов и несколько проще для обработки, нежели похожий формат OpenMath.

2 Постановка задачи

В данной работе решается задача разработки метода математического поиска с использованием волновых правил(rippling)[4][8, стр. 38] для определения тождественности двух выражений, а также реализация такого метода. Для задачи должны быть адаптированы алгоритмы для эффективной реализации метода автоматического доказательства тождественности. Процесс доказательства тождественности двух выражений может занимать длительное время даже на современных компьютерах. Для системы поиска же необходимо за небольшое время в пределах нескольких секунд совершить сотни попыток доказательства, поэтому основным требованием к системе должна быть высокая скорость процесса доказательства. Другим требованием к системе может быть адекватность итоговой оценки степени тождественности двух МВ, причем адекватность в смысле понятности и очевидности для конечного пользователя. Также немаловажен удобный ввода МВ, что может обеспечиваться сторонними средствами.

3 Обзор предметной области

Предыдущие попытки реализации систем поиска по формулам условно можно разделить на следующие синтаксическому, семантическому или смешанному подходам.

1. Синтаксический подход заключается в использовании текстового представления МВ, которые обрабатываются, используя методы текстового поиска и обработки информации. Такой подход позволяет строить сравнительно менее требовательные к ресурсам и быстрые системы поиска чем в других подходах. Системы, следующие синтаксическому подходу, обычно позволяют вводить комбинированные поисковые запросы, содержащие не только формулы, но и ключевые слова или идентификаторы статей. В то же время системы такого рода имеют ограниченные возможности по определению релевантности выражений. Обычно такие системы не способны определить α - эквивалентные выражения. Пример: проект LaTeXSearch[12].

2. Семантический подход заключается в преимущественном использовании структуры МВ, смысла. Для такого подхода характерно использование методов автоматического доказательства, а потому обладают более широкими возможностями для определения релевантных результатов. Обратной стороной является использование алгоритмов с сравнительно большой вычислительной сложностью и необходимость хранения относительно большого

1. Растровое изображение
   $\sin^2\theta + \cos^2\theta \equiv 1$

2. TeX
   \sin^2{\theta}+\cos^2{\theta}\equiv 1

3. Presentation MathML

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
 <msup>
  <mi>sin</mi>
  <mn>2</mn>
 </msup>
 <mi>&#952;</mi>
 <mo>+</mo>
 <msup>
  <mi>cos</mi>
  <mn>2</mn>
 </msup>
 <mi>&#952;</mi>
 <mo>&#8801;</mo>
 <mn>1</mn>
</math>
```

4. Content MathML

```
<math xmlns="http://www.w3.org/1998/Math/MathML" display="display">
 <apply>
 <equivalent/>
  <apply>
  <plus/>
   <apply>
   <power/>
    <apply>
    <sin/>
     <ci>&#952;</ci>
    </apply>
    <cn>2</cn>
   </apply>
   <apply>
   <power/>
    <apply>
    <cos/>
     <ci>&#952;</ci>
    </apply>
    <cn>2</cn>
   </apply>
  </apply>
  <cn>1</cn>
 </apply>
</math>
```

Примеры представлений MB

количества информации в индексе, что накладывает требования к ресурсам. Пример: проект MathWebSearch[1], использующий метод индексации термов.

3. Смешанный подход может использовать методы, характерные для предыдущих двух, или же некие другие. Пример: проект MIaS[2].

LaTeXSearch

Система осуществляет поиск по библиотеке научных текстов Springer. Поисковым запросом могут являться: формула в формате TeX, название документа, уникальный идентификатор документа. В случае поискового запроса, представленного как формула, релевантность определяется путем сравнения строк TeX кода, нормализованных особым образом так, чтобы отличающиеся по записи выражения тоже попадали в поисковую выдачу. Непосредственных описаний используемых в данной системе алгоритмов и стратегий поиска широкой публике представлено не было.

MathWebSearch

Одним из самых успешных проектов, посвященных математическому поиску, является проект MathWebSearch[1] в результате которого была реализована поисковая система, нашедшая применение в обеспечении поиска для некоторых хранилищ научных текстов, например, на сайте проекта Zentralblatt MATH[2]. В подходе, используемом системой MathWebSearch, формулы индексируются в виде дерева подстановок, что позволяет экономить память за счет эффективной записи повторяющихся данных. Система позволяет работать с формулами, записанными в виде Presentation и Content MathML, а также OpenMath. Поисковый запрос может быть представлен в вышеперечисленных форматах, а также в формате TeX и многих других, используя автоматический перевод в Presentation MathML. В рамках проекта MathWebSearch была проиндексирована большая часть онлайн библиотеки arXiv.org.

Math Indexer and Searcher

Система MIaS позволяет вводить комбинированный поисковый запрос, включающий как текст так и формулы. Система работает с формулами формата MathML, индексируя их в текстовом виде таким образом, чтобы увеличить возможность определить сходство между двумя равными выражениями, записанными в различных нотациях, или же между двумя неравными выражениями. Для определения релевантности MIaS использует эвристику, основанную на определении весов индексированных термов, которые соответствующим образом влияют

[2]http://zbmath.org/formulae/

на итоговую оценку релевантности документов, а значит и на порядок выдачи результатов[2].

4 Разработка и реализация метода поиска по формулам

4.1 Формирование базы выражений

Для системы поиска размер базы данных - поискового индекса является важным параметром, влияющим на скорость работы системы. Дизайн таких систем как MathWebSearch и MiAS позволяет заносить в индекс не само MB в той форме, в которой оно найдено, но лишь компактный набор необходимых данных, извлеченных из MB. Для описываемой в данной работе системы необходимо знать как структуру выражения, так и назначение его элементов - будь то переменная, константа или функция. Поэтому для простоты в данной работе в качестве поискового индекса используется таблица, где в каждой строке должны быть записаны MB в семантическом формате и URI документа, содержащего данное MB.

Возможные источники формул

Математический поиск может быть полезен для поиска по библиотекам научных статей, одна из которых arXiv.org может быть достаточно легко проиндексирована, благодаря наличию TeX исходников большого количества статей, но перевод MB в формат Content MathML из TeX достаточно нетривиальная задача. Распространенность Content MathML в интернете достаточно низкая. Стандарт, в отличие от Presentation MathML, не предназначен для представления формул на веб-страницах и используется в нишевых областях, закрытых от общего доступа, или же на узко специализированных сайтах, например на ресурсе, посвященном описанию стандарта MathML[14]. Также удобным источником формул может стать сайт проекта Wolfram[22], являющийся каталогом более 300 тысяч функций, записанных в нескольких форматах, включая Content MathML.

Проблема перевода

Для задачи математического поиска являлся бы полезным инструмент, позволяющий переводить MB из внешнего представления в семантическое, что по сути означает синтаксический разбор выражения и построение его дерева. Существующие инструменты[18], имеющие такую функциональность, не обеспечивают должного качества конвертирования на реальных данных. Общепринятые математическая нотация и соглашения позволяют записывать выражения, которые могут иметь несколько трактовок в зависимости от контекста.

1. $U(x)$ - U*(x) или функция U с аргументом x?

2. $\bar{x}_i^1$ - $\bar{x}_i$ в степени 1 или $\bar{x}_i$ с верхним индексом 1?

3. $else$ - $e * l * s * e$ или слово «иначе»?

Для человека более очевидны вторые варианты интерпретации строк, но для определенных инструментов автоматического перевода[18] - первые. Основные проблемы, встречающиеся на пути к разбору выражений из научных текстов: неоднозначность общепринятой математической нотации, опускание знака умножения, одинаковые формы записи различных математических сущностей, зависимость смысла выражения от научной области, обильное использование индексов и спецсимволов, использование слов естественного языка вперемешку с МВ.

Сбор формул

Попытка проиндексировать подмножество статей раздела Computer Science сайта arXiv.org выявила недостатки существующих решений. Инструмент LaTeXML, имеющий экспериментальную возможность для перевода МВ из TeX в Content MathML, показал слабое качество конвертирования. Иной доступный инструмент, SnuggleTeX по всей видимости имеет такие же недостатки. Несколько лучше показал себя веб-сервис, являющийся частью проекта Wiris[21]. Но из 240253 выражений были успешно переведены лишь чуть менее 30 тысяч, причем из них 27.5 - это тривиальные выражения, например состоящие из одной переменной. В итоге было получено 1319 нетривиальных МВ, которые зачастую содержат грубые ошибки перевода, в частности некорректно обрабатывается операция сложения. Однако стоит иметь ввиду, что веб-сервис Wiris позволял перевод из Presentation MathML в Content MathML. Для перевода из TeX в Presentation MathML использовался инструмент LaTeXML, который использовался для такой задачи в проекте MathWebSearch и показал эффективность перевода(причем из внешнего представления во внешнее) около 60%[1].
Полученная связкой LaTeXML и веб-сервиса Wiris база индексированных МВ небольшой части TeX исходников статей проекта arXiv.org стала основной для прототипной системы поиска.

4.2 Метод волновых правил

Общее описание

Метод волновых правил в простом виде применяется для приведения одной формулы к другой. Метод применим к формулам, которые являются синтаксически похожими. Изначально он применялся для доказательства методом математической индукции, где гипотеза и заключение индукции похожи. При выполнении поиска также приходится работать с похожими выражениями. Если эти выражения приводятся одно к другому с помощью тождественных преобразований, то выражения нужно считать одинаковыми. Однако процесс преобразования

нужно сделать направленным: на каждом шаге должно уменьшаться количество различий между формулами. Этот процесс преобразования выполняется с помощью волновых правил. Синтаксически общая часть формул, называется *основой*. Синтаксически различные части помечаются как *волновой фронт*. Часть основы, попавшая внутрь волнового фронта, называется *волновой дырой*. В примере разметки (1) функция s входит в волновой фронт, $x+b$ является основой, а x и $x+b$ - волновые дыры, отмеченные подчеркиванием.

$$\boxed{s(\underline{x})} \ + b = \boxed{s(\underline{x+b})} \tag{1}$$

Важным ограничением на разметку является принцип *сохранения основы*, который заключается в недопустимости изменения основы в ходе доказательства[3]. Сам ход доказательства сводится к недетерминированному применению к заключению индукции специальных правил переписывания, называемых *волновыми правилами*, в которых обе части также размечены вышеозначенными аннотациями.

Мера

Чтобы направить процесс доказательства вводится понятие меры выражения, отражающей в себе положение волновых фронтов в выражении. Мера параметризует количество различий между выражениями, а значит с уменьшением меры выражения становятся более похожими. Учитывая ограниченность множества мер снизу нулем, а также накладывая ограничение на волновые правила, которое заключается в обязательном уменьшении меры[4], получаем не только направленность процесса доказательства, но и гарантированную его завершаемость[4, стр. 10]. Мера представляет собой список неотрицательных чисел. Каждое такое число - это вес волнового фронта на данной глубине дерева основы. Сравнение двух мер проводится в лексикографическом порядке[8, стр. 46]. Для пары выражений длина мер будет заведомо равна за счет принципа сохранения основы. Тогда как в методе волновых правил обычно используется мера, основанная на *ширине* [5] волнового фронта, иногда удобнее использовать меру, основанную на *размере* волнового фронта. Например выражение

$$\boxed{f((\underline{m} - s(n)), s(n))} \tag{2}$$

будет иметь меру основанную на ширине равную [0,1,0], тогда как мера основанная на размере волнового фронта будет равна [0,3,0], потому что весом в такой мере является количество термов в волновом фронте на данном уровне основы.

В конце доказательства должна получиться копия гипотезы индукции или

[3]Впрочем, в разновидности метода волновых правил, используемой в этой работе, данный принцип ослаблен.

[4]Мера правой части должна быть строго меньше меры левой части

[5]Метод подсчета такой меры(основанной на ширине волнового фронта) можно прочитать в работе[8, стр. 45]

ее части в теле заключения индукции, которые затем заменяются либо на true, что должно привести к обращению в true всего выражения, либо на другую часть гипотезы, что приводит к равенству и к успешному завершению доказательства.

Пример

Предположим даны следующие гипотеза

$$(x + b) + c = x + (b + c) \tag{3}$$

и заключение индукции

$$(\boxed{s(\underline{x})} + b) + c = \boxed{s(\underline{x})} + (b + c) \tag{4}$$

где $s(x) = x + 1$. Предположим наличие следующего волнового правила.

$$\boxed{s(\underline{T_1})} + T_2 = \boxed{s(\underline{T_1 + T_2})} \tag{5}$$

Здесь стоит обратить внимание на тот факт, что мера левой части волнового правила равна [1,0], а правой [0,1] т.е. соблюдается принцип уменьшения меры. Длина меры при этом равна 2, как и глубина основы $x + b$. Далее ход доказательства состоит в преобразовании равенства (4). Для этого волновое правило (5) применяется к левой части

$$\boxed{s(\underline{x + b})} + c = \boxed{s(\underline{x})} + (b + c) \tag{6}$$

$$\boxed{s(\underline{(x + b) + c})} = \boxed{s(\underline{x})} + (b + c) \tag{7}$$

и к правой

$$\boxed{s(\underline{(x + b) + c})} = \boxed{s(\underline{x + (b + c)})} \tag{8}$$

Как можно заметить, мы получили копию левой части гипотезы индукции в левой части заключения индукции. Теперь, исходя из равенства (3), вместо выражения $(x + b) + c$ в левой части можно подставить $x + (b + c)$

$$\boxed{s(\underline{x + (b + c)})} = \boxed{s(\underline{x + (b + c)})} \tag{9}$$

тем самым завершив доказательство.

Модификация метода

Вообще говоря, метод волновых правил применяется для доказательства некоего выражения, называемое *целью*, с помощью структурно похожего *данного(-ых)*. Для доказательства по индукции используются соответственно заключение и гипотеза индукции. Для разметки в таком случае используется известный

алгоритм[5], так как аннотациями помечается только одно из выражений. В контексте же данной задачи оба МВ равнозначны, а *целью* является выражение вида $MB_1 = MB_2$. Решение данной проблемы заключается во взятии за *данное* некой общей части выражений, а по сути их *основы*[4, стр. 48]. Чтобы ее получить, нужно одновременно разметить оба МВ, что достигается использованием алгоритма *унификации различий*[6]. После унификации процесс доказательства состоит из недетерминированного применения волновых правил к обоим выражениям и увеличения основы за счет волнового фронта. Постепенно увеличивая основу и убирая различия, процесс в результате может привести к одинаковым выражениям[4, стр. 48]. Для данной задачи также удобнее использовать меру, основанную на размере волнового фронта, так как такая мера лучше отражает прогресс доказательства, в частности сокращение волнового фронта и увеличение основы.[7].

5 Описание реализации математической поисковой системы

Разработанный метод поиска реализован в поисковой системе(размещена по адресу 195.209.147.206:8080), выполняющей последовательность шагов:

1. Получение поискового запроса, в данном случае являющимся МВ в семантическом представлении.

2. Последовательное получение оценки тождественности поискового запроса с выражениями из базы данных. При этом конечно же совсем необязательно сверять запрос с каждым выражением, так как многие из них можно заведомо отфильтровать.

3. Формирование результатов поисковой выдачи и предоставление их пользователю.

Получение поискового запроса

Ручной ввод МВ в текстовом семантическом представлении, понятном для машины, неудобен для человека, поэтому необходим интерфейс для упрощения такой задачи. Существуют инструменты[19][20][21], представляющие собой удобную веб-среду для ввода МВ и получения его семантического представления в формате Content MathML. Также для облегчения ввода МВ являлась бы полезной возможность выполнить ввод в некотором, передающим лишь внешний вид выражения, формате, например в TeX, с последующим автоматическим переводом в семантическое представление, но реализация подобной функциональности осложнена проблемами описанными выше в разделе 4.1.

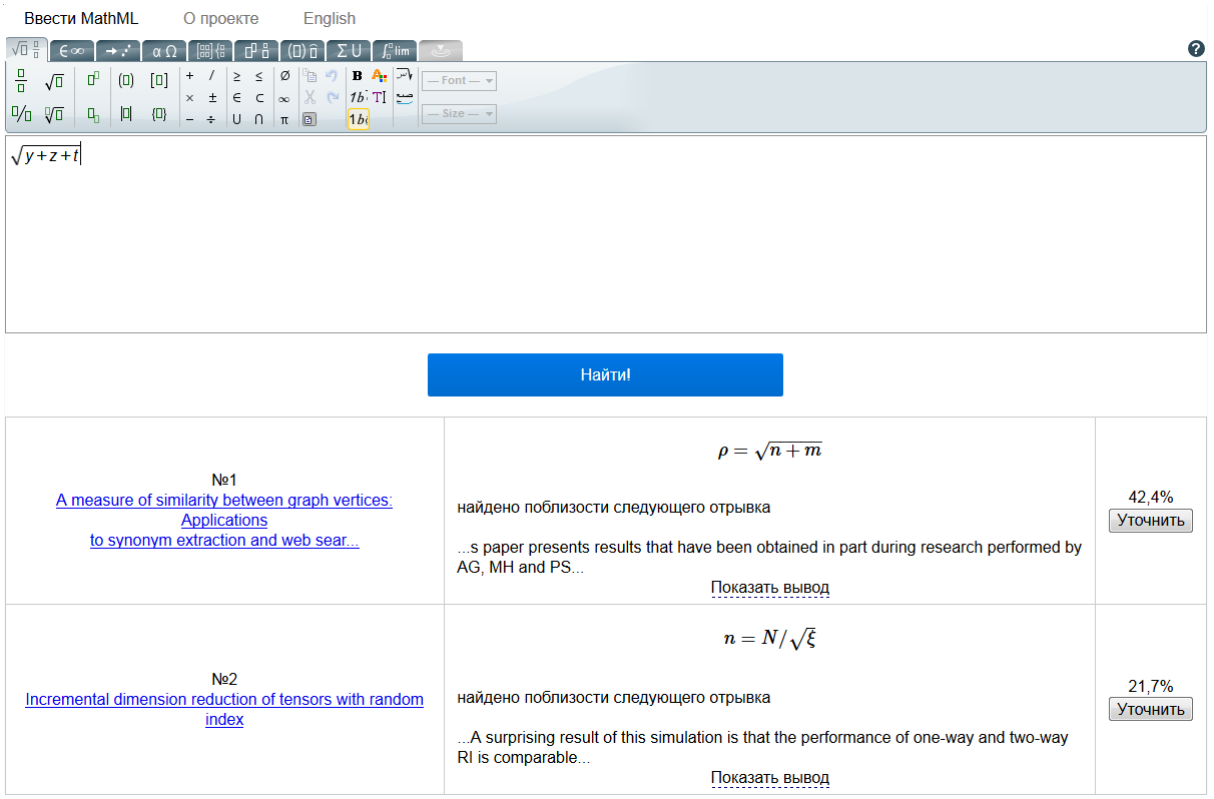

Рис. 1. Интерфейс поисковой системы

Внутреннее представление

Для простоты манипулирования MB удобно иметь внутреннее представление в виде двоичного дерева, где листьям соответствуют идентификаторы или константы выражения, а внутренним узлам - функции. Такое представление похоже на предложенный в статье[3] формат apply-free Content MathML, который может являться платформонезависимым способом передачи внутреннего формата системы, который по сути является деревом MB. Отличие apply-free Content MathML от Content MathML заключается в отказе от использования тега apply, который оказался наиболее часто встречающимся тегом[3]. Каждая вершина двоичного дерева также должна содержать информацию о ее маркировке - будь то основа или волновой фронт. Минусом такого представления является появление артефактов записи n-арных, $n > 2$ функций, когда приходится дублировать запись функции несколько раз для записи всех аргументов.

Поиск

Для простоты положим, что база проиндексированных MB представляет собой множество $E = \{E_1, E_2, ..., E_n\}, n \in N$, каждый элемент которого E_k - это MB, записанное во внутреннем семантическом представлении. Поисковой выдачей в таком случае будет являться множество

$$S = \{S_1, S_2, ..., S_n | S_i = \{E_k, w_i\}, 0 \leq w_i \leq 1\} \tag{10}$$

То есть это множество пар выражений и весов, при этом удовлетворяющих условию

$$(\forall i, j \in [1, .., n] : i \leq j) w_i \geq w_j \tag{11}$$

Проще говоря, поисковая выдача - это отсортированный по убыванию весов массив выражений. Назовем $SearchQuery$ МВ заданное как поисковый запрос. Вес w_i для каждого выражения должен быть получен в результате попытки проведения доказательства утверждения

$$SearchQuery \equiv E_k \tag{12}$$

Само доказательство состоит из нескольких шагов.

Унификация различий

На этом шаге выражения размечаются аннотациями для метода волновых правил. Для этого оба выражения $SearchQuery$ и E_k, где предварительно обеспечена уникальность имен переменных, подаются на вход алгоритма унификации различий. Так как выражения можно разметить разными способами, то в результате работы алгоритма может быть получено слишком большое количество различных вариантов разметки, далеко не все из которых полезны для доказательства. Однако получать всевозможные варианты унификации совсем необязательно, и существует возможность применять алгоритм таким образом, чтобы заведомо получать только наиболее подходящие для доказательства варианты унификации[6]. Также исходя из общих рассуждений, что в процессе доказательства требуется увеличивать основу, а значит уменьшать волновой фронт, то можно предположить, что вариант унификации будет тем более подходящий, чем меньше в нем термов, помеченных как волновой фронт[4]. Для каждой подходящей унифицированной пары затем проводится попытка доказательства методом волновых правил.

Применение метода волновых правил

Волновые правила могут применяться к обоим выражениям. Прогресс доказательства выражается с помощью описанной выше меры, основанной на размере волнового фронта. Используемая мера получается сложением мер первого и второго выражений, причем их длины обязательно равны за счет равенства основ выражений. В общем случае процесс применения метода волновых правил можно представить как итерацию двух последовательных шагов[4, стр. 49]. Задача первого шага - это применение волновых правил. Вторым шагом должно быть увеличение основы, обеспеченное изменениями в структуре и аннотациях выражений на первом шаге.

$$s(f(\boxed{g(\underline{x})}\ )) = s(\boxed{p(g(\underline{f(x)}))}) \tag{13}$$

Увеличение основ левой и правой части за счет терма g приведет к их неравенству.

Увеличение основы

Увеличение основы можно производить разными путями, но по определению увеличение основы означает, что $OldSkeleton \subseteq NewSkeleton$[7]. Один из способов - это на каждой итерации заново проводить унификацию пары $SearchQuery^i, E_k^i$, где i - номер итерации. Такой способ показал свою эффективность[6][7], но его производительность зависит от эффективности реализации алгоритма унификации различий, который при этом требует экспоненциальное время для получения всех решений[6]. Быстрота процесса доказательства критически важна для реализации поиска, так как доказательство должно выполняться сотни раз за короткий, разумный период времени между поисковым запросом и выдачей результата. Достаточно простым и эффективным показал себя способ увеличения основы путем перебора вершин, принадлежащих волновому фронту и потенциально способных увеличить основу. Как показывает пример (13) увеличение основы даже за счет соответствующих термов может привести к несовпадению основ, поэтому процесс увеличения основы проходит следующим образом. Положим WF_1, WF_2 - множества вершин, принадлежащих волновому фронту для первого и второго выражения. Тогда нужно составить множество $CommonWF = WF_1 \cap WF_2$. Добавить вершину из $CommonWF$ в основу каждого исходного выражения и проверить сохранение основы. Если основа сохраняется, то полученная пара выражений является кандидатом на дальнейшее проведение процесса доказательства. Повторить для каждой вершины из $CommonWF$, при этом каждый раз основу нужно увеличивать для исходной пары выражений. Для удобства, после увеличения основы выражения можно проверять не на совпадение основы, но на совпадение глубины основы т.е. чтобы длина мер первого и второго выражения имела одинаковую длину и их можно было просуммировать. Затем меру новой пары требуется сравнить с мерой исходной пары, причем она должна быть строго меньше, иначе новая пара отбрасывается и более не используется для доказательства. Возникает проблема, так как основы увеличены, а значит длина совокупной меры исходной пары выражений равна или меньше длины меры каждой из новых пар.

Сравнение мер разной длины

Тривиальным способом было бы добавление нулей или отбрасывание «лишних» элементов, чтобы привести обе меры к одинаковой длине. Но цель, которую должен преследовать способ сравнения мер разных длин, заключается в том, чтобы позволять отбросить не приводящие к результату ветви доказательства, при

этом не задев те, которые могут привести к успешному завершению. Чтобы решить эту проблему, в работе[7] предложена функция Shrunk, позволяющая сократить длину меры новой пары выражений и тем самым обеспечить возможность сравнения с старой мерой. Функция заключается в «наложении» особым способом друг на друга старого и нового выражения, таким образом можно найти какие элементы меры сложить между собой и тем самым сократить длину.

Оценка

Вес w_j для ранжирования поисковой выдачи должен являться оценкой семантического сходства пары выражений, другими словами вес w_j должен отражать насколько одно выражение тождественно другому. В случае, если доказательство прошло успешно, то проблем не возникает - оценка в таком случае равна 1 т.е. выражения тождественно равны. Однако успешность доказательства это исключительная ситуация, поэтому нужен метод для определения оценки семантической похожести для любой пары выражений. Учитывая тот факт, что под смыслом MB можно понимать его дерево, то для оценки необходимо определить величину структурного сходства двух деревьев, не упуская из виду совпадение или несовпадение имен соответствующих вершин. При этом важно заметить, что в процессе доказательства производится большое количество пар видоизмененных входных MB, причем для нахождения максимальной оценки требуется вычислить оценку для каждой такой пары, а это в свою очередь накладывает требования к производительности метода оценки. Существует алгоритм, описанный в работе[10], предназначенный для оценки сходства деревьев с помеченными листьями. Данный алгоритм удобен своей небольшой вычислительной сложностью, по сравнению с другими алгоритмами, выполняющими схожую задачу. Алгоритм требует время, равное $O(m^2)$, где m - мощность множества меток. Для сравнениия алгоритм подсчета tree edit distance[6] в эффективной реализациии требует $O(n^3)$ в худшем случае, где n - размер обоих деревьев[11]. Минусом метода, представленного в [10], является то, что он предназначен для деревьев с помеченными листьями, но не учитывает меток внутренних узлов. Однако алгоритм может быть приспособлен для этого. Предположим X и Y - множество меток листьев для первого и второго дерева. Тогда множество $C = X \cap Y$ содержит все встречающиеся метки. Назовем $D = C \times C$, тогда $D_i = (X_i, Y_i)$, а итоговая оценка сходства может рассчитываться по нижеследующей формуле, где a_i и b_i - это заранее подсчитанные числовые значения, соответствующие ближайшим общим предкам листьев с метками X_i, Y_i, для первого и второго дерева соответственно.

$$w_j = \frac{\sum min(a_i, b_i)}{\sum max(a_i, b_i)} \tag{14}$$

[6]Tree edit distance - распространенная мера для оценки сходства двух деревьев, обозначающая минимальное количество элементарных операций, необходимых для превращения одного дерева в другое

Простой идеей является коррекция значений $min(a_i, b_i)$ и $max(a_i, b_i)$ в зависимости от равенства ближайших общих предков для данных меток. Положим, что $P_1(A, B), P_2(A, B) \in [0, 1]$ принимают пару вершин деревьев на вход и выдают некий числовой коэффициент. Тогда формулу (14) можно видоизменить:

$$w_j = \frac{\sum P_1(A_i, B_i) min(a_i, b_i)}{\sum P_2(A_i, B_i) max(a_i, b_i)} \tag{15}$$

A_i и B_i - наименьшие общие предки вершин пары вершин X_i, Y_i с метками первого и второго дерева. Причем на одинаковых входных значениях A, B функции должны удовлетворять $P(A, B) \leq P(A, B)$, чтобы обеспечить значение w_j в пределах от [0..1]. Функции P_1 и P_2 в простейшем случае могут проверять на равенство метки вершин и выдавать соответствующий коэффициент, однако это достаточно тривиальное решение.

5.1 Пошаговое описание получения оценки тождественности двух выражений

На входе имеем два MB, которые обозначим как *given* и *goal*. Назовем унифицированной такую пару given и goal, для которой выполняется принцип сохранения основы. Тогда процесс состоит из следующих шагов:

1. Применение к *given* и *goal* алгоритма унификации различий(difference unification).

2. Для каждой унифицированной пары $given_i$ и $goal_i$:

   a) Функция checkStopConditions. Высчитывается оценка похожести деревьев $given_i$ и $goal_i$, затем проводится проверка на достижение условий завершения процесса, т.е. проверка на получение равных выражений, достижение меры вида [0,0,...,n] или достижение порогового значения оценки. Если ни одно из условий не выполняется, то процесс переходит к пункту 2b.

   b) Процедура ripplingStep1. Применение волновых правил[7] с целью обеспечить возможность увеличения основы.

   c) Процедура ripplingStep2. Получение возможных вариантов увеличения основ, по сути получение новых унифицированных пар. Мера каждой новой пары с увеличенной основой должна быть строго меньше меры предыдущей пары. С полученными парами переходим к пункту 2.

[7]Так как мета-аннотации, применяемые в данном методе, не позволяют выразить правила коммутативности в виде волновых правил[13], то их применение должно обеспечиваться программно.

```
bool checkStopConditions(given, goal, measure)
{
 double weight = Similarity(given,goal);
 if(weight>maxWeight)
  maxWeight = weight;
 if (given==goal || isEnded(measure) || maxWeight > limit)
  return false;
 else
 {
  ripplingStep1(given, goal, measure);
  return true;
 }
}
```

Псевдокод функции проверки условий завершения

Вкратце процесс получения оценки тождественности заключается в направленном переборе возможных преобразований исходных выражений, т.е. в направленном получении видоизмененных деревьев этих выражений и в нахождении максимальной оценки похожести среди них. Успешное доказательство является скорее исключительной ситуацией. В сущности, для получения оценки тождественности используется комбинация из трех алгоритмов: алгоритма получения оценки, алгоритма унификации различий и метода волновых правил. Важно отметить роль каждого алгоритма для реализации поиска. Используя лишь алгоритм получения оценки структурной похожести двух деревьев можно было бы реализовать систему поиска, способную выдавать выражения, записанные в форме похожей на поисковый запрос вплоть до полного совпадения. Такая система смогла бы определить равенство поискового запроса $2(a+b)$ выражению из базы $2(a+b)$, но при этом выражение $2(x+y)$ получило бы низкую оценку. Добавив к такой системе алгоритм унификации различий, можно было бы обеспечить, чтобы α - эквивалентные выражения распознавались как тождественно равные, то есть определилось бы и равенство $2(x+y)$ тому же самому поисковому запросу, но выражение $2y+2x$ получило бы низкую оценку. Если добавить использование метода волновых правил, система приобретает возможность направленного перебора тождественно равных форм выражения, позволяя найти наиболее похожие формы(вплоть до полного равенства) обоих выражений, т.е. такая система способна распознать тождественность выражения $2y+2x$ поисковому запросу $2(a+b)$.

5.2 Конвейер

Наивным подходом к организации взаимодействия алгоритма унификации различий и метода волновых правил было бы получение набора всевозможных

```
void ripplingStep1(given, goal, measure)
{
  foreach(rw in GetPossibleRewritings(given, goal))
   {
    ripplingStep2(rw.given, rw.goal, measure);//Важно, что для последующег
    //сравнения  используется мера пары до ее переписывания.
   }
}
```

Псевдокод процедуры ripplingStep1

```
void ripplingStep2(given, goal, measure)
{
 foreach(sk in GetSkeletonEnlargements(given, goal))
 {
  if (sk.newMeasure.Shrunk < measure)
   NewPairs.Add(sk.given. sk.goal);
 }
}
```

Псевдокод процедуры ripplingStep2

унификаций, а затем их обработка методом волновых правил. Однако такой подход имеет недостатки, заключающиеся в высокой вычислительной сложности алгоритма унификации различий, из-за чего на определенных входных данных работа алгоритма может занимать непозволительно длительное время работы. Также стоит учитывать факт, что алгоритм унификации различий в общем случае производит всевозможные унификации, большая часть из которых не приводит к успешному доказательству даже в случае тождественности унифицируемых выражений. Процесс унификации состоит в всевозможном применении правил унификации[6], поэтому с целью оптимизировать процесс доказательства разумно организовать конвейер, когда при получении новой унифицированной пары выражений, она незамедлительно попадает на обработку методом волновых правил с целью доказательства. Работа алгоритма унификации различий и работа метода волновых правил может вестись параллельно. Время работы процесса доказательства для утверждения $(x+y)z+xw+yw \equiv xz+yz+xw+yw$ с использованием конвейера занимает менее 100 мс на ПК с процессором, имеющим тактовую частоту 3.3 ГГц. В процессе перебора вариантов унификаций, к левой части найденной унифицированной пары $\boxed{(x+\underline{y})}\ z + xw + yw \equiv \boxed{xz+\underline{yz}} + xw + yw$ применяется волновое правило $\boxed{(a+\underline{b})}\ c = \boxed{ac+\underline{bc}}$, что приводит к совпадению левой и правой частей, а значит к успешному доказательству. Без использования конвейера доказательство занимает $>$1 секунды, причем большая часть времени уходит на получение сотен вариантов унификации, а на непосредственно приме-

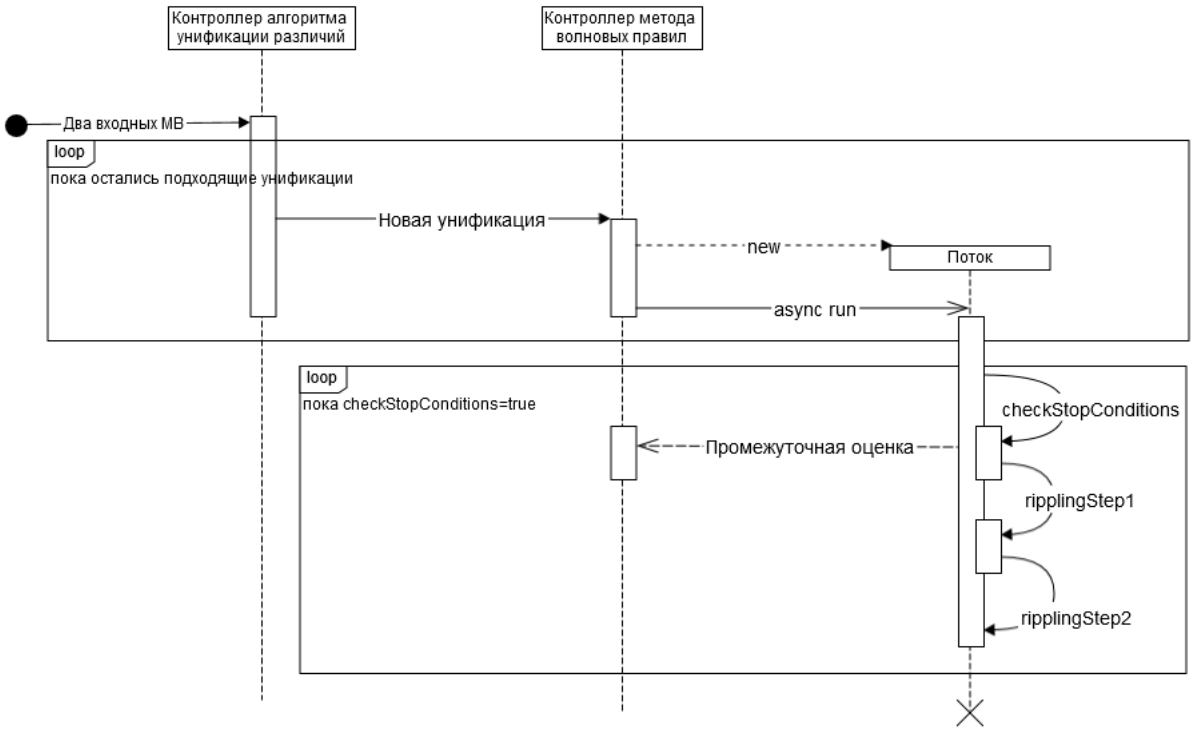

Рис 2. Диаграмма последовательности работы конвейера

нение волновых правил уходит всего несколько десятков миллисекунд. Очевидно использование конвейера дает тем больший выигрыш по времени, чем раньше относительно других пар будет найдена унифицированная пара, приводящая к успешному доказательству.

5.3 Средства реализации

Прототипная система была реализована на платформе .NET 4.5 с использованием технологий WCF и ASP.NET. На языке C# была написана динамически подключаемая библиотека(.dll), реализующая метод доказательства и всю необходимую программную среду, например классы, отвечающие за внутреннее представление формул, создание волновых правил. Основное предназначение библиотеки заключается в получении на вход двух выражений и выдаче числового значения, обозначающего их степень тождественности. Основная логика, относящаяся непосредственно к поиску, реализована внутри веб-сервиса WCF. Веб-сервис отвечает за взаимодействие с базой выражений, производит оценку и ранжирование результатов с помощью подключаемой библиотеки. Сайт - фронтэнд написан на ASP.NET и отвечает за получение поискового запроса, отправку его на веб-сервис и получение результата. Использование веб-сервиса позволяет стороннее автоматизированное использование системы через API.

Для обеспечения приемлемой скорости ответа на поисковый запрос приходится жертвовать глубиной и качеством доказательства, но для прототипной системы

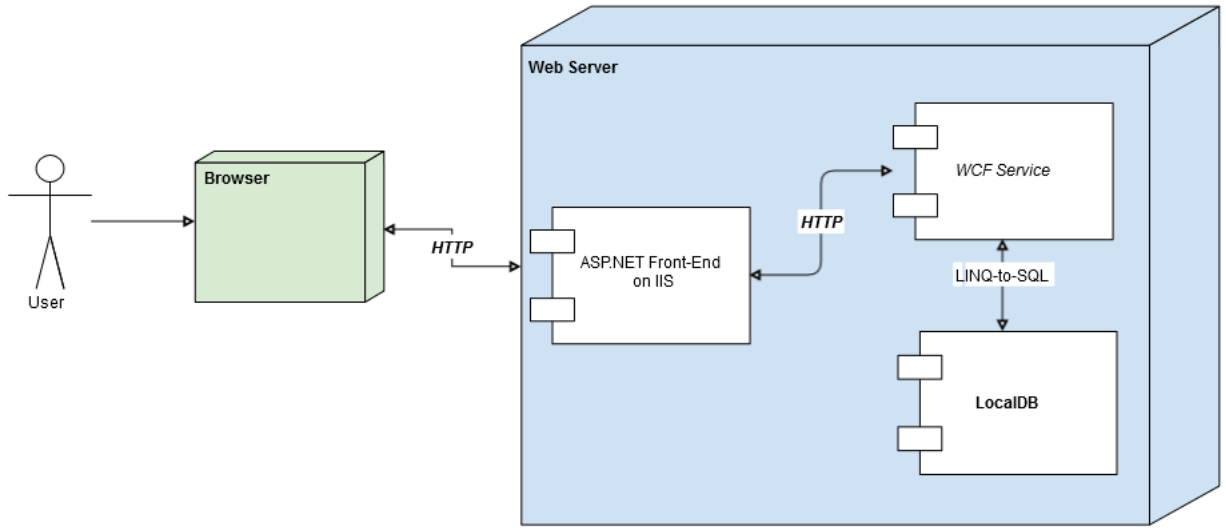

Рис 3. Диаграмма размещения прототипной системы

эта проблема была частично компенсирована возможностью проводить более глубокое доказательство для конкретного элемента поисковой выдачи. В идеальном случае пользователь, заметив сходство элемента выдачи и поискового запроса, делает запрос на проведение более глубокого доказательство и получает успешный вывод. Но обычно эта функция позволяет лишь несколько увеличить вес данного элемента поисковой выдачи, однако она также может быть полезна для использования автоматизированными системами через API.

В общей сложности для проекта были проиндексированы около 100 тысяч Content MathML выражений каталога функций Wolfram, но получившаяся база оказалась довольно большой по объему занимаемого дискового пространства, а выражения в Content MathML иногда были составлены не совсем корректно с точки зрения передачи смысла MB. Отдельно был проиндексирован раздел Information Retrieval и Symbolic Computation сайта arXiv.org. Исходные статьи и MB были представлены в формате TeX. Для перевода MB в Presentation MathML исходники сначала были обработаны инструментов LaTeXML. Затем корректно конвертированные MB были поданы на вход веб-сервису Wiris для перевода в Content MathML. Подавляющее большинство MB не было успешно переведено, но получившаяся база данных удобнее для работы чем индекс каталога функций Wolfram, поэтому примеры приведены для поиска по индексу разделов arXiv.org.

Первые три результата на запрос $x = \sqrt{y + z}$

1. $\rho = \sqrt{n + m}$, вес 90%
2. $w_{s,i} = w$, вес 80%
3. $a_{i,j} = a$, вес 80%

Первые три результата на запрос $\sqrt{y+z+t}$

1. $\rho = \sqrt{n+m}$, вес 42,4%
2. $n = N/\sqrt{\xi}$, вес 21,7%
3. $e^{\frac{-a}{\tau}}$, вес 21,7%

Первые три результата на запрос $x = 2/(1-b)$

1. $m = 2^b + 1$, вес 47%
2. $G_l = 2^l - 1$, вес 40,4%
3. $\epsilon = N^{-1/2}$, вес 36,9%

6 Заключение

В данной работе предложен метод математического поиска, и на его основе реализована прототипная система с веб-интерфейсом по адресу 195.209.147.206:8080. Для этого были частично проиндексированы онлайн библиотеки научных текстов arXiv.org и functions.wolfram.com. Система включает в себя модуль доказательства эквивалентности МВ с помощью метода волновых правил, адаптированного для задачи поиска. В целом подтверждена возможность использования метода волновых правил для проведения быстрого, экономного по ресурсам процесса доказательства. Разработанная система математического поиска на основе метода волновых правил, получив некую формулу как запрос, способна находить релевантные результаты, которые близки к запросу именно с точки зрения математического смысла формулы. Планы работ на будущее включают в себя оптимизацию процесса доказательства, для чего возможно как оптимизация конкретной реализации, например за счет распараллеливания, так и теоретические изыскания для определения оптимальных стратегий доказательства. На производительность системы доказательства может серьезно повлиять интеграция нынешнего метода с одним из методов *удаления различий*[7]. Для улучшения качества поисковой выдачи также необходима разработка механизма для корректной обработки МВ из различных областей науки и поддержка соглашений в математической нотации, используемой в них. Улучшение качества базы выражений также напрямую повлияет на качество самого поиска, а разработка способа более компактного хранения семантического представления МВ гарантированно повлияет на производительность системы в лучшую сторону. Помимо всего прочего несомненна польза от реализации в будущем инструмента для конвертирования выражений из различных представлений в семантическое с достаточно высоким процентом корректных результатов.

Литература

[1] Michael Kohlhase, Bogdan A. Matican, Corneliu-Claudiu Prodescu *MathWebSearch 0.5: Scaling an Open Formula Search Engine* // Lecture Notes in Computer Science Volume 7362, Springer Berlin Heidelberg, 2012. P. 342-357

[2] Petr Sojka, Martin Líška *Indexing and Searching Mathematics in Digital Libraries* // Lecture Notes in Computer Science Volume 6824, Springer Berlin Heidelberg, 2011. P. 228-243

[3] Yokoi, Keisuke; Aizawa, Akiko *An Approach to Similarity Search for Mathematical Expressions using MathML* // Grand Bend, Ontario, Canada, July 8-9th, Masaryk University Press, Brno, 2009. P. 27-35

[4] Alan Bundy, David Basin, Dieter Hutter, Andrew Ireland *Rippling: Meta-level Guidance for Mathematical Reasoning* // Cambridge Tracts in Theoretical Computer Science (No. 56), 2005.

[5] David Basin, Toby Walsh *Difference Matching.* // Lecture Notes in Computer Science Volume 607, Springer Berlin Heidelberg, 1992. P. 295-309.

[6] David Basin, Im Stadtwald, Toby Walsh *Difference Unification.* // In Proceedings of the 13th IJCAI. International Joint Conference on Artificial Intelligence, 1993.

[7] Alan Bundy *Draft. Towards the Integration of Difference Removal and Difference Moving.* [PDF](`http://homepages.inf.ed.ac.uk/bundy/drafts/Strategies_Workshop_Diff_Removal.pdf`).

[8] Корухова Ю.С. *Система автоматического синтеза функциональных программ.* Диссертация на соискание ученой степени кандидата физико-математических наук, 2005.

[9] Francisco Alvaro, Joan-Andreu Sanchez, Jose-Miguel Benedi *Recognition of Printed Mathematical Expressions Using Two-dimensional Stochastic Context-Free Grammars* // International Conference on Document Analysis and Recognition (ICDAR), IEEE, 2011. P. 1225-1229

[10] H. De Meyer, B. De Baets, S. Janssens *Similarity Measurement on Leaf-labelled Trees* // Proc. 2nd EUSFLAT Conference (Leicester, U.K.), 2001. P. 253-256.

[11] Demaine, E. D., Mozes, S., Rossman, B., and Weimann O. *An Optimal Decomposition Algorithm for Tree Edit Distance* // ACM Transactions on Algorithms (TALG) Volume 6 Issue 1, Article No. 2, 2009.

[12] LaTeX Search - Mathematical Equations in Scientific Publications. `http://latexsearch.com/`

[13] Frequently asked questions about rippling `http://dream.inf.ed.ac.uk/projects/ripple-faq.html`

[14] W3C Math Home `http://www.w3.org/Math/`

[15] MathJax `http://www.mathjax.org`

[16] OpenMath `http://www.openmath.org`

[17] Dave-ML Overview `http://daveml.org/`

[18] LaTeXML A LaTeX To XML Converter `http://dlmf.nist.gov/LaTeXML/`

[19] MathEdit `http://wme.lzu.edu.cn/mathedit/index.html`

[20] Connexions MathML Editor (in Beta) `http://cnx.org/matheditor`

[21] WIRIS editor `http://wiris.com/editor`

[22] The Wolfram Functions Site `http://functions.wolfram.com/`

[23] Competition on Recognition of Online Handwritten Mathematical Expressions `http://www.isical.ac.in/~crohme/`