Министерство экономического
развития и торговли РФ

Государственный университет
Высшая школа экономики

Факультет прикладной математики и кибернетики
Кафедра прикладной математики

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

На тему модернизация конструктора математических задач и серверного модуля компьютерной обучающей системы.

Студент группы № М-94
Крылов Дмитрий Вячеславович
(Ф.И.О.)

Научный руководитель
Данилов Владимир Григорьевич
(должность, звание, Ф.И.О.)

Консультант
Петров Пётр Петрович
(должность, звание, Ф.И.О.)

Москва 2013

Содержание

1 Введение

Данная дипломная работа посвящена модернизации компьютерной обучающей системы. Эта система способна генерировать различные математические задачи основываясь на формальных описаниях классов задач.

Система даёт пользователям возможность обучаться решению задач на конкретных практических примерах и дистанционно. Для этого система генерирует различные задачи для пользователей и в том случае если они решают их неправильно показывает им правильное решение этих задач.

Пользователи данной системы обучаются различным математическим приёмам путём решения задач и исследуя их правильные решения.

2 Описание обучающей системы

На верхнем уровне обучающая система состоит из двух основных подсистем: ядро системы и внешний математический модуль. Для взаимодействия с системой применяется веб-интерфейс. Также в рамках данной дипломной работы был разработан интерфейс системы на базе операционной системы Android, который поддерживает все функции системы для обучения пользователей, реализованные в данный момент времени. Android интерфейс использует javascript библиотеку MathJax для отображения формул на экране.

Для формальных описаний задач в системе используется специальный язык, который называется SimpleTask. Для описания математических выражений на этом языке применяется специальный формат, который понятен используемому математическому модулю. Каждое описание класса задач представляет собой файл написанный на языке SimpleTask и называется грамматикой.

Существует два способа добавления нового класса задач в систему. Первый способ заключается в ручном написании грамматики и её установки на сервер. Для этого пользователь должен знать язык

SimpleTask и язык используемого математического модуля. Установка грамматики проводиться с помощью отдельного сервлета, который отделён от ядра, но установлен на том же сервере. Второй способ заключается в использовании конструктора грамматик. Конструктор грамматик автоматизирует большую часть процесса написания и установки грамматик и позволяет пользователю сосредоточиться на их составлении.

В рамках данной дипломной работы в конструкторе были реализованы дополнительные функции, упрощающие процесс создания грамматик и делающие его более удобным. В работе над конструктором использовались такие библиотеки как jlatexmath для визуализации формул, ini4j для работы с файлами свойств, jRegistryKey для работы с реестром операционной системы windows, SnuggleTeX для преобразования описаний математических формул в формате MathML в формат LaTeX и обратно, и другие.

В обязанности ядра системы входит принятие и обработка html запросов на генерацию математических задач. Получив подобный запрос, ядро ищет на сервере соответствующий требуемому типу задач файл грамматики. Затем проводит синтаксический анализ найденного файла, преобразуя грамматику в древовидную структуру данных, основываясь на этой структуре генерирует задачу и отправляет её в формате XML как html отклик.

В процессе синтаксического разбора грамматики ядро системы активно использует математический модуль для различных вычислений. Большинство файлов грамматик требуют таких вычислений для возможности успешной генерации задач.

В настоящее время математическим модулем является система компьютерной алгебры Mathematica версии 5.2.

Ядро обучающей системы полностью написано на языке Java и реализовано как набор сервлетов работающих под руководством TomCat. Каждой функции ядра соответствует свой сервлет, реализующий данную функцию, используя общие для системы библиотеки.

Для связи ядра системы и математического модуля Mathematica применяется библиотека JLink. Эта библиотека была специально со-

здана разработчиками Mathematica для связи этой системы с внешними программами написанными на языке Java. С помощью этой библиотеки ядро системы образует вместе с системой Mathematica клиент-серверную архитектуру. Ядро отправляет математическому модулю запросы на вычисления математических выражений в форме специальных пакетов и получает от Mathematica результат в виде отклика. Для коммуникации между ядром системы и системой Mathematica используется специальный протокол, установленный библиотекой JLink.

Для приёма запросов от пользователя и передачи ему информации используется веб-интерфейс. Пользователи используют веб-интерфейс для того, чтобы запросить у системы задачу для решения и для того, чтобы проверить ответ. Для решения этих задач веб-интерфейс обращается к соответствующим функциям ядра.

Для пользователя типичный сеанс работы с обучающей системой выглядит следующим образом.

Рис. 1: Схема сеанса работы с системой

Пользователь выбирает интересующий его тип задачи из списка,

система находит соответствующую этому типу задач грамматику и генерирует некоторую конкретную задачу. Затем система отображает условие и предлагает пользователю ввести ответ. После того, как пользователь введёт ответ система сравнивает его с правильным. В том случае если была допущена ошибка система отображает правильное решение. Далее пользователь может попробовать решить аналогичную задачу.

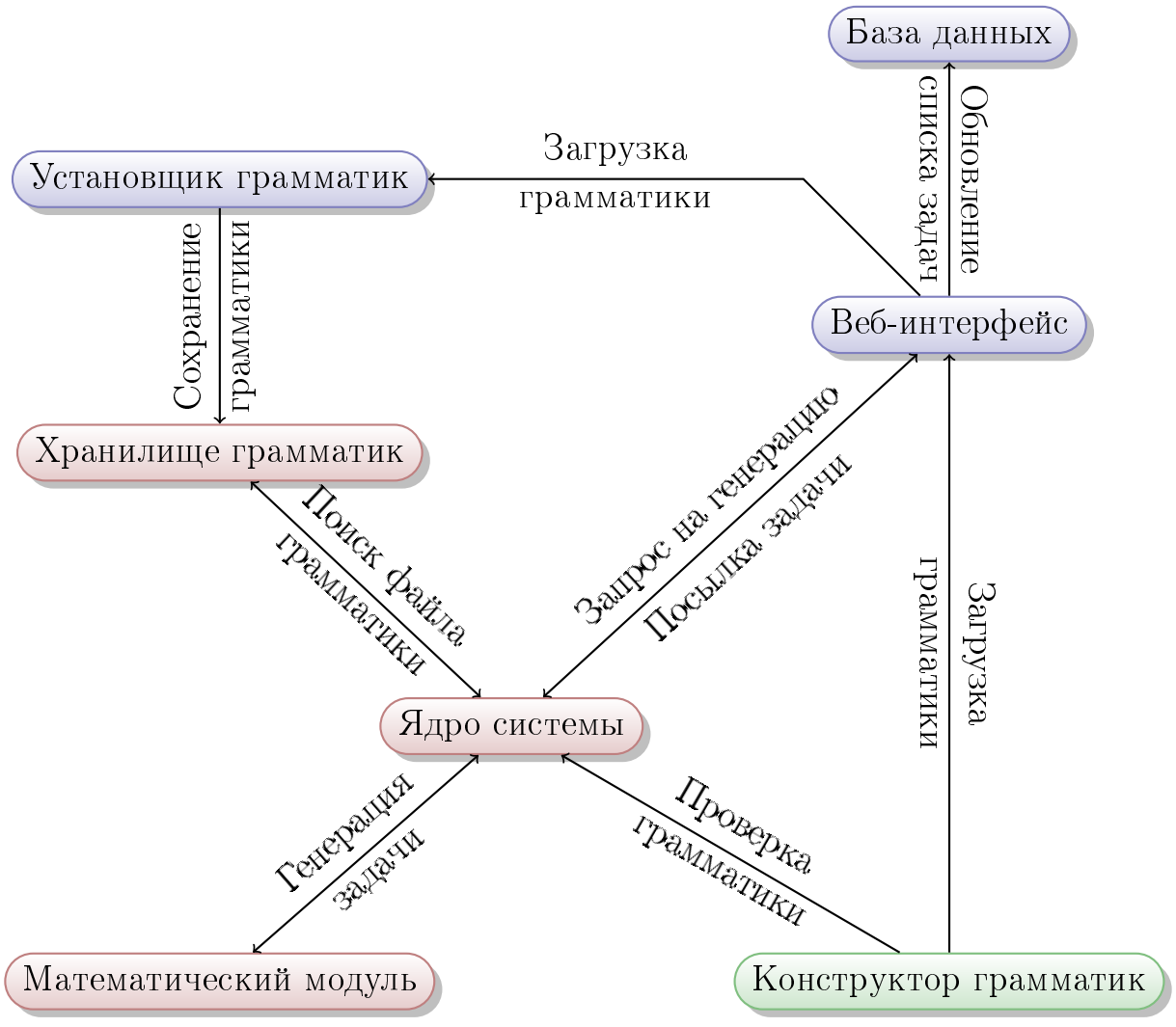

Рис. 2: Связи между модулями системы

3 Пример работы с системой через веб-интерфейс

Рассмотрим работу системы через веб-интерфейс на примере. Пусть пользователь после входа в систему послал запрос на генерацию задачи вычисления производной от произведения двух функций и числового множителя.

В результате система сгенерировала следующую задачу:

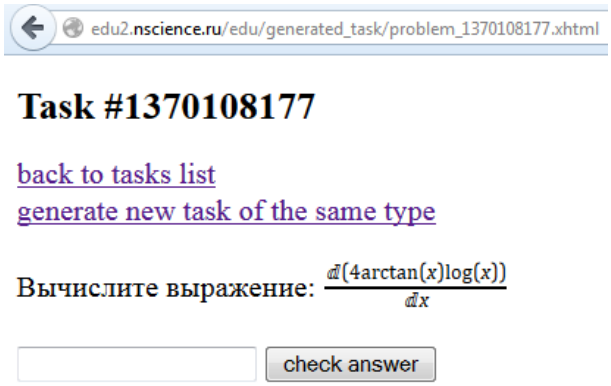

Рис. 3: Пример сгенерированной задачи

В этом примере пользователю предлагается вычислить производную от $4 \cdot \arctan(x) \cdot \log(x)$.

Страница с формулировкой задачи содержит форму, включающую в себя поле для ввода ответа и кнопку для отправки. Так как система использует программу Mathematica для работы с математическими выражениями ответ должен быть записан в формате Mathematica.

В том случае, если пользователь вводит неправильный ответ система отправляет пользователя на веб страницу с решением.

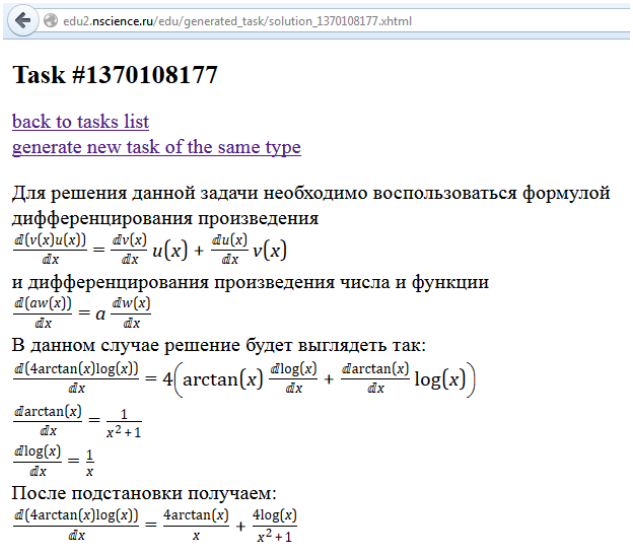

Рис. 4: Решение задачи

При повторной генерации задачи этого типа пользователю нужно будет решать уже другую задачу:

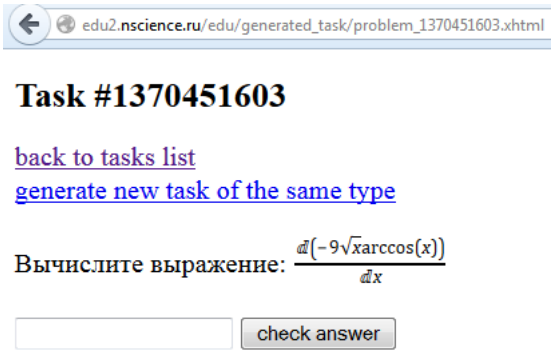

Рис. 5: Повторная генерация задачи

4 Грамматики

4.1 Структура грамматики

Описание задачи на языке SimpleTask представляет собой последовательность правил вида ⟨имя правила⟩ ⟨{определение}⟩. Определение может содержать в себе текст и ссылки на другие правила. Для того, чтобы поместить в определение одного правила ссылку на другое требуется поместить в определение первого правила имя второго правила предваряя его символом #. Например:

```
rule1 { x+123 }
rule2 { y + #rule1 }
```

В этом примере при вычислении правила rule2 будет вычислено правило rule1 и его результат подставлен вместо ссылки на первое правило. То есть результатом вычисления правила rule2 будет `y+x+123`.

Правила могут быть вычислимыми (формулами) и невычислимыми (текстом). Формулы отличаются от обычных правил тем, что их определение, прежде чем оно будет использовано в других правилах обрабатывается математическим модулем. В настоящей версии этим модулем является Mathematica 5.2. Как следствие определение правила типа «формула» должно быть корректным выражением языка Mathematica.

Для того, чтобы указать, что некоторое правило является вычислимым следует предварить открывающую фигурную скобку определения правила символом тильда. Так, в следующем примере при вычислении правила rule2 определение первого правила сначала будет передано математическому модулю для вычисления, а затем уже результат вычисления будет подставлен во второе правило.

```
rule1 ~{ 2+2 }
rule2 { y + #rule1 }
```

Таким образом, значение rule2 после вычисления будет `y+4`.

Каждая грамматика должна содержать в себе как минимум три правила отвечающие за формулирофку задачи, решение и ответ. Эти три правила являются корневыми и все прочие правила будут использованны и вычисленны только в том случае если какое-нибудь главное правило прямо или косвенно ссылается на него.

Эти правила имеют зарезервированные имена. Правило для формулировки задачи называется «problem». Правило для решения: «solution». Правило для ответа: «answer». После имён правил «problem» и «solution» должно идти ключевое слово «output».

В качестве иллюстрации приведём пример простейшей грамматики:

```
number { 42 }
problem output { Введите число #number }
answer { #number }
solution output {
    Вы правильно решите эту задачу,
    если введёте число #number
}
```

Из этой грамматики может быть сгенерирована только одна задача. Когда студент выбирает эту задачу, обучающая система выводит предложение «Введите число 42» и в том случае если студент введёт в поле ответа что-то другое выводит «Вы правильно решите эту задачу, если введёте число 42».

4.2 Принцип генерация задач

Возможность генерации большого количества задач на основе одного файла грамматики обеспечивается за счёт так называемых правил случайных значений. Эти правила являются специальными формулами, которые характеризуются тем, что их значения не определены заранее, а случайным образом выбираются из заданного списка значений при каждом обращении к грамматике.

Это достигается за счёт применения возможностей математического модуля генерации случайных чисел. Таким образом, если заменить в примере простой грамматики из придедущего подраздела первую строку на `number ~{ Random[ Integer, { 1, 42000 } ] }`, то на основе такой грамматики может быть сгенерированно 42000 различных задач, для решения которых нужно будет ввести предложенное целое число в диапозоне от 1 до 42000.

В качестве более сложного примера рассмотрим генерацию правила со значением, представляющем собой произведение функции и числового множителя.

```
t ~{ Random[ Integer, {   1, 12 } ] }

a ~{ Random[ Integer, { -10, 10 } ] }

t1  ~{ Switch[#t11, 1, Sin[x], 2, Cos[x],
                    3, Tan[x], 4, Cot[x],
                    5, ArcSin[ x ], 6, ArcCos[ x ],
                    7, ArcTan[ x ], 8, ArcCot[ x ],
                    9, Exp[ x ], 10, x^( #z ),
                    11, Log[ x ], 12, Sqrt[ x ],
                    13, x]}

f {#a*#t1}
```

В этом примере правило случайных значений `t` используется для выбора случайной функции из списка. Правило `a` используется для вычисления множителя.

Результат правила `f` оказывается зависимым от обоих правил случайных значений.

5 Конструктор грамматик

5.1 Описание

Конструктор грамматик это приложение упрощающее создание файлов грамматик и их загрузку в систему. Конструктор предоставляет удобный интерфейс для ввода элементов грамматики, таких как условие задачи, ответ и решение.

Конструктор даёт пользователям возможность создавать грамматики не изучая язык SimpleTask, использующийся в системе.

Начиная работу с конструктором пользователь может выбрать с каким элементом он желает работать. Он может как начать вводить один из трёх главных элементов так и ввести некий вспомогательный элемент.

5.2 Реализованные возможности конструктора

5.2.1 Функция проверки грамматики

Конструктор предоставляет возможность отправить созданную грамматику на проверку ядру обучающей системы. Получив запрос от конструктора на проверку грамматики ядро проверяет синтаксическую корректность и отсылает конструктору отчёт о проверке, содержащий в себе либо сообщение о том, что всё правильно, либо сообщение ошибки. Пользователь конструктора может запросить проверку грамматики в любой момент времени.

Проверка грамматики внутри ядра происходит следующим образом. Ядро загружает грамматику в систему и пытается сгенерировать задачу на её основе. Если генерация задачи была произведена успешно, то конструктору возвращается сообщение, говорящее о корректности грамматики. В случае возникновения ошибки в коде ядра генерируется исключение с текстом, описывающим имеющуюся проблему. Конструктору возвращается сообщение, содержащее этот текст. Грамматика, переданная ядру на проверку, хранится внутри системы во временном файле.

Для реализации этой задачи внутри ядра был создан специальный сервлет, обрабатывающий запросы конструктора на проверку грамматики. В конструкторе была организована функция, осуществляющая обращение к сервлету посредством html запроса типа POST. Текст грамматики передаётся сервлету в качестве параметра.

5.2.2 Функция визуализации математических выражений

При работе с конструктором и заполнением различных составляющих описания задачи пользователям часто бывает необходимо получить наглядное представление о том, как различные математические выражения, участвующие в описании будут выглядеть в окне браузера. Пользователь конструктора работает с формулами, описанными в формате, который понятен математическому модулю обучающей системы, то есть в формате Mathematica. Однако, для эффективной работы пользователю требуется иметь изображение формул и окружающего их текста в традиционной, привычной форме.

Для решения этой задачи в конструкторе была организована функция визуализации математических выражений. Визуализация построена по следующему принципу. Описание какого-либо элемента грамматики состоит из текста, который должен отображаться как есть и из математических формул, которые сначала должны быть преобразованы в соответствующие изображения. Когда пользователь отправляет команду конструктору визуализировать тот элемент грамматики с которым он в данный момент работает, конструктор анализирует введённый пользователем текст и путём синтаксического разбора находит участки текста, в которых описываются математические выражения. Далее конструктор преобразует найденные формулы в изображения, вставляет их на соответствующие места в тексте и выводит результат на экран.

Для преобразования описания формулы в изображение применяется сторонняя библиотека. При решении вопроса об используемой библиотеке было рассмотрено два варианта.

Первый вариант: библиотека JEuclid, поддерживающую функ-

цию визуализации формул в формате MathML. В этой библиотеке имелись реализации расширений компонентов SWING, способные отображать формулы. Эта библиотека удовлетворяла требованиям по функционалу, но полученные с её помощью результаты были недостаточно хорошего качества, поэтому было принято решение найти альтернативные варианты.

Второй рассмотренный вариант: библиотека jlatexmath. Эта библиотека позволяет отрисовывать формулы таким образом, как эта делает система компьютерной вёрстки TEX и следовательно качество работы этой библиотеки очень высоко. В то же время она не требует доступа к системе TEX и способна работать автономно.

Библиотека jlatexmath работает с математическими выражениями в формате LATEX. Таким образом, прежде, чем применять её требовалось решить проблему преобразования формул, написанных в формате Mathematica в этот формат.

Эта задача была решена средствами самой системы Mathematica. Когда пользователю требуется визуализировать формулу, конструктор обращается к системе Mathematica с запросом конвертировать требуемое математическое выражение из своего формата в формат LATEX. После этого конструктор передаёт полученное описание формулы библиотеке jlatexmath и получает изображение.

Конструктор имеет две возможности получить доступ к математическому модулю. В начале конструктор пытается найти систему Mathematica на локальном компьютере. Для этого он сначала анализирует файл настроек программы. В этом файле может быть указан путь к исполняемому ядру математики. Если путь действительно указан и он корректен конструктор соединяется с Mathematica по указанному пути и использует её для преобразования форматов. В противном случае конструктор пытается найти систему Mathematica, используя реестр операционной системы Windows. Это возможно, так как Mathematica при установке, записывает адрес своей рабочей папки в известный раздел реестра и конструктор может проанализировать его. Для работы с реестром в конструкторе используется нативная библиотека jRegistryKey. В том случае если систему Mathematica удалось найти через реестр операционной

системы, конструктор записывает путь к ядру Mathematica в файл настроек, для того, чтобы в следующий раз иметь возможность воспользоваться первым способом нахождения математического модуля.

Конструктор использует систему Mathematica с помощью той же библиотеки и тем же методом, что и ядро системы.

В том случае если конструктор не может обнаружить математический модуль на локальном компьютере, он делает попытку связаться с ядром обучающей системы и передать требуемую задачу ему. Для реализации этой возможности в ядре был добавлен ещё один сервлет обрабатывающий запросы на преобразования математических выражений из одного формата в другой. Для осуществления своих функций этот сервлет использует возможности системы Mathematica. Он способен преобразовывать формулы записанные в формате Mathematica в форматы MathML и LaTeX.

Таким образом для работы функции визуализации формул требуется либо установленная на локальном компьютере система Mathematica, либо соединение с интернетом. В первом случае конструктор сможет провести преобразование используя локальный математический модуль, а во втором требуемые преобразования реализуются посредством ядра обучающей системы.

5.2.3 Функция конвертирования формул в формате LaTeX в формат Mathematica

В конструкторе реализована возможность преобразования математических выражений описанных в формате LaTeX в формат Mathematica. Посредством этой функции пользователь при составлении шаблона задачи может записывать формулы в формате LaTeX, после чего конструктор будет преобразовывать их.

В конструкторе имеется встроенный конвертор формул, который осуществляет преобразование в несколько этапов. Данный конвертор представляет собой композицию нескольких сторонних конверторов, преобразующих данные, в несколько промежуточных форматов.

Сначала исходная формула, написанная в формате LaTeX пре-

образуется в презентационный MathML, затем в содержательный MathML.

Основное различие между презентационным и содержательным MathML заключается в следующем. Презентационный MathML описывает, как математическое выражение должно отображаться на экране устройства или на бумаге. По сути, презентационный MathML представляет собой набор инструкций, руководствуясь которыми програма визуализирует формулу. Этим презентационный MathML похож на LaTeX и поэтому преобразовать выражение из одного формата в другой достаточно простая задача. Содержательный MathML описывает математическую формулу с точки зрения её семантики. Другими словами, информация присутствующая в описании математического выражения на языке содержательного MathML содержит в себе данные о логических связях между математическими объектами участвующими в формуле, таких как: «x умножается на y», «применение функции f к аргументу x» или «x принадлежит множеству X».

Это означает, что описание формулы на языке презентационного MathML или LaTeX содержит в себе информацию, касающуюся правил отображения, которая не содержится в содержательном MathML. Приведём, в качестве иллюстрации следующий простой пример:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow>
    <mi>x</mi>
    <mo>*</mo>
    <mi>[</mi>
    <mi>y</mi>
    <mo>+</mo>
    <mn>1</mn>
    <mi>]</mi>
  </mrow>
</math>
```

Это описание формулы $x*[y+1]$ на языке презентационного MathML.

Оно является просто последовательностью символов, которые должны быть изображены друг за другом. В этом описании смысловая составляющая формулы не указана, однако, определены внешний вид скобок и знака умножения. Следующий пример содержит описание той же формулы на языке содержательного MathML.

```
<math xmlns='http://www.w3.org/1998/Math/MathML'>
 <apply>
  <times/>
  <ci>x</ci>
  <apply>
   <plus/>
   <ci>y</ci>
   <cn type='integer'>1</cn>
  </apply>
 </apply>
```

Здесь наоборот определена семантика формулы, а информация об отображении опущена. Эта формула может быть отображена как $x \times (y+1)$, $x \cdot [y+1]$, $x\ \{y+1\}$ и другими способами не влияющими на семантику.

Вообще говоря, существует множество языков описывающих математические формулы и всех их можно разделить на две группы. Первая группа состоит из языков, описывающих способы отображения формулы. К этой группе относятся такие языки как LaTeX и презентационный MathML. Вторая группа состоит из языков, описывающих семантику формулы. К этой группе относятся такие языки как содержательный MathML, OpenMath и почти все языки, использующиеся в системах компьютерной алгебры, в том числе и в системе Mathematica.

Преобразование, проводимые между языками одной группы не является сложной задачей, так как описания формул на этих языках содержат одну и ту же информацию.

Отсутствие информации об отображении в языках второй группы, также, не является серьёзной проблемой в том случае если эта информация не является критически важной, так как всегда можно

использовать некоторый стандартный вариант визуализации формулы.

Однако, при написании конвертора, требовалось осуществить преобразование описания формулы на языке первой группы (LaTeX или презентационный MathML) в описание на языке второй группы (формат Mathematica).

Это преобразование гораздо сложнее. Проблема заключается в том, что одному и тому же изображению формулы может соответствовать несколько конкретных формул с разной семантикой и никакой стандартной семантики определить нельзя. Поясним сказанное примером: выражение $L(x + y)$ может интерпретироваться как «L умноженное на сумму x и y» или «функция L применённая к сумме x и y». Что в действительности означает данная формула, можно определить только из контекста, в котором это выражение применяется. В отрыве от контекста, когда данный пример рассматривает человек, он обычно принимает решение на основании первого символа. Например, в выражении $f(x+y)$ символ f скорее всего означает функцию, так как этот символ часто используется для этой цели, с другой стороны в формуле $a(x + y)$ символ a скорее всего является множителем. Однако, на самом деле это лишь предположение.

Данный пример показывает, что сконструировать абсолютно корректный конвертор презентационного MathML в содержательный невозможно принципиально. Возможно, используя различные приёмы из области искуственного интелекта, построить конвертор, который будет работать правильно почти всегда.

В интернете существует довольно мало доступных программных продуктов поддерживающую данную функциональность. В конструкторе для этой используется библиотека SnuggleTeX. Данная библиотека позволяет проводить преобразование формул в формате LaTeX в презентационный MathML, а затем осуществлять операцию семантического пополнения Up-Conversion, преобразуя презентационный MathML в содержательный.

После получения содержательного MathML требовалось преобразовать его в формат Mathematica. Так как прямого конвертора не удалось найти, было решено использовать формат OpenMath в

качестве промежуточного.

OpenMath это язык описания формул основанный на XML, который был специально создан для обмена математическими выражениями между различными приложениям, использующими математику. В том числе между различными системами компьютерной алгебры и компьютерных вычислений, такими как Mathematica, Maple, Sage и другими. Как следствие, в настоящее время существует множество конверторов, преобразующих описания формул на языке OpenMath в формат различных математических систем. Такие конверторы имеют названия PhraseBooks. Для преобразования OpenMath в конструкторе используется такой конвертор.

Содержательный MathML преобразуется в OpenMath путём использования документа XSLT (eXtensible Stylesheet Language Transformations). Это язык преобразований XML документов имеющий прямую поддержку в стандартных библиотеках языка Java.

Таким образом, конвертор проводит преобразования, осуществляя следующие шаги:

1. Конвертор получает исходный текст описания математического выражения в формате LaTeX и с помощью библиотеки SnuggleTeX преобразует его в формат презентационного MathML.

2. С помощью постпроцессора библиотеки SnuggleTeX преобразует полученный результат из презентационного MathML в содержательный.

3. Используя документ XSLT и встроенный в Java преобразователь XML документов осуществляет конвертирование из содержательного MathML в формат OpenMath.

4. С помощью кодека Mathematica PhraseBook преобразует полученное выражение в формат Mathematica.

При работе с XML документами конвертор использует DOM модель документа и работающие с ней DOM парсеры из стандартной библиотеки Java.

Не смотря на то, что полученный конвертор корректно работает с большим подмножеством языка Mathematica существуют несколько важных элементов нуждающихся в дополнительной обработке. Такими элементами являются интегралы, различные формы представления производных, нижние индексы переменных и прочее. Все подобные элементы имеют поддержку в языке Mathematica, однако используемые в конверторе библиотеки не корректно работают с ними. Для обработки этих особых случаев был сконструирован ещё один конвертер, который использует первый как основной, но находит и самостоятельно конвертирует отдельные части исходного выражения, которые не поддаются его обработке. Этот конвертер расширяем и может быть использован для модификации действий основного конвертора. При проведении преобразования конструктор обращается к внешнему конвертеру, а тот по мере необходимости использует функционал основного.

5.3 Интерфейс конструктора

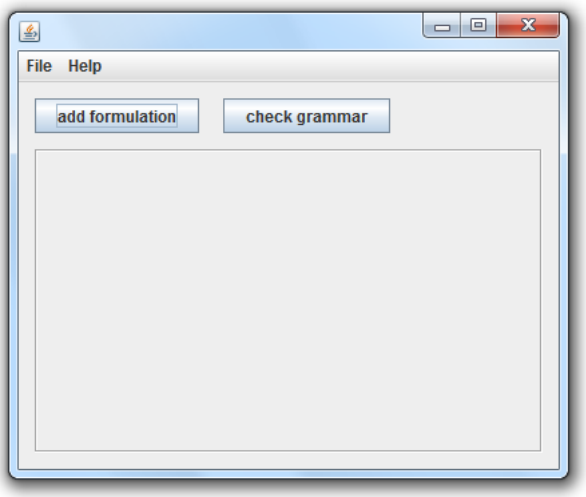

Рис. 6: Главное окно конструктора.

В этом разделе будет описан интерфейс конструктора и рассмотрен пример создания файла грамматики с его помощью.

Рассмотрим главное окно конструктора, изображённое на рисунке в начале раздела.

В верхней части окна расположенны две кнопки. Кнопка «add formulation» используется для добавления в грамматику новых элементов. Эти элементы могут быть как главними, то есть формулировкой задачи, решением задачи или ответом, так и вспомогательными. Кнопка «check grammar» служит для отправки запроса грамматики на сервер для проверки. При её нажатии конструктор основываясь на введённых элементах создаёт текст грамматики, генерирует запрос ядру обучающий системы с требованием провести проверку и отправляет созданный текст в качестве аргумента.

Ниже расположен список введённых пользователем правил из которых текст грамматики будет генерироваться. При первом запуске конструктора этот список пуст.

Дополнительные функции, касающиеся файлов грамматики в целом, доступны в меню file.

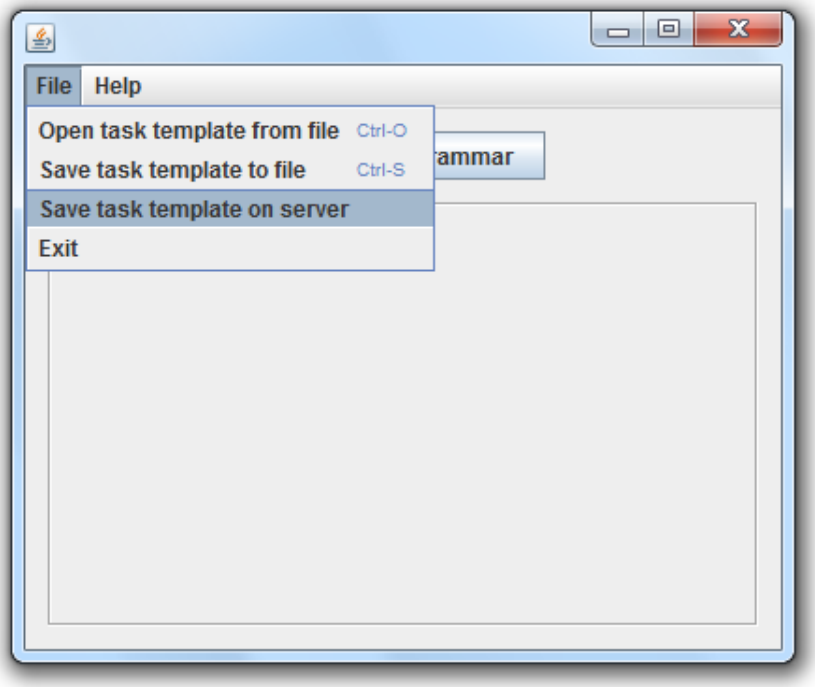

Рис. 7: Меню file.

Конструктор поддерживает возможности сохранения файла грамматики на локальном компьютере, а так же загрузку ранее сохранённых или написанных вручную грамматик.

Функция отправки готовой грамматики на сервер реализованна следующим образом. Конструктор извлекает из вайла свойств иденititификатор преподавателя и конструирует запрос к специальному модулю работающему с ядром системы на сервере. Модуль обновляет базу данных задач и добавляет новую задачу в список задач данного преподователя. С этого момента новая задача становиться доступной.

Меню help содержит один пункт: Malual. При его выборе конструктор обращается к установленной веб странице содержащей описание программы и открывает её в броузере.

Теперь обратимся к функциям добавления новых правил в грамматику. Рассмотрим окно добавления нового элемента, которое появляется при нажатии на кнопку «add formulation».

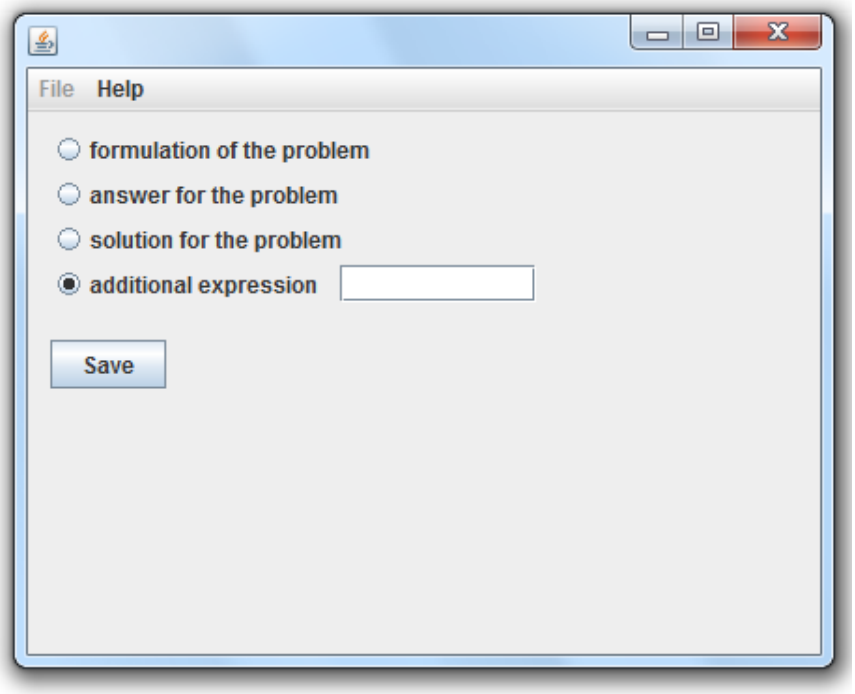

Рис. 8: Добавление нового правила.

В этом окне пользователь может выбрать какой именно элемент он хочет добавить. Пользователь может выбрать один из четырёх вариантов для того, чтобы ввести соответственно формулировку задачи, правильный ответ, решение задачи или любое другое вспомогательное правило. В последнем случае пользователю нужно ввести имя дополнитеьного правила в текстовом поле справа от элемента списка. При нажатии на кнопку «Save» правило будет добавлено в грамматику и можно будет приступить к его редактированию.

После добавления в грамматику правила, отвечающего за формулировку задачи главное окно приложения будет выглядеть следующим образом:

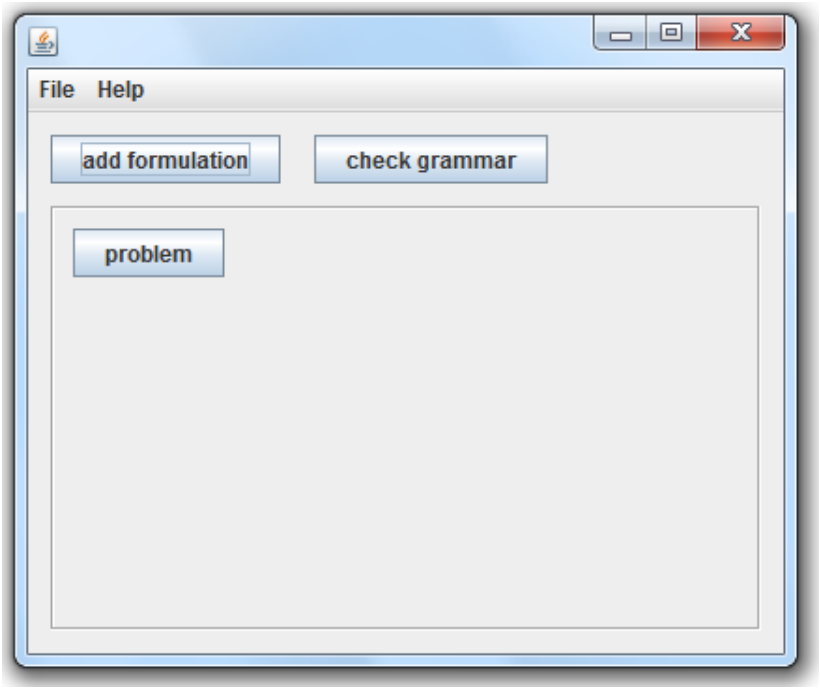

Рис. 9: Добавление формулировки задачи.

Теперь список добавленных элементов содержит одно правило. Все правила добавляются в список элементов в виде кнопок. При нажатии на кнопку соответствующую некоторому правилу можно приступить к его редактированию. Все правила, которые были только что добавлены изначально пусты.

Рассмотрим окно для редактирования правил.

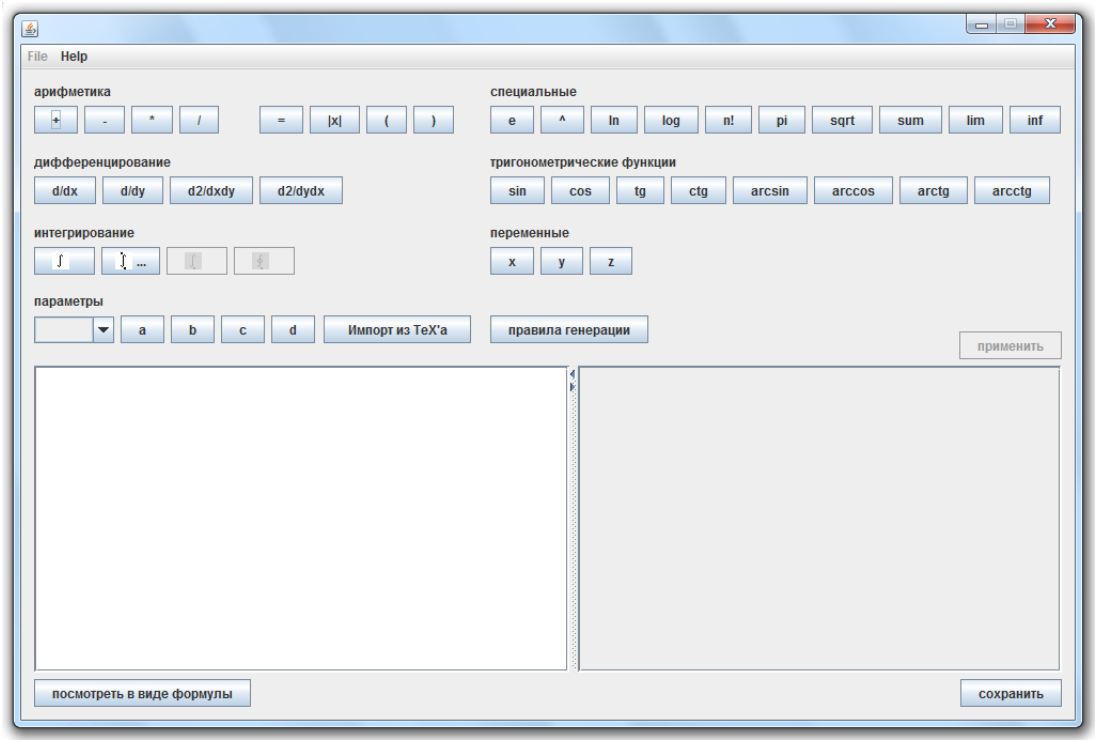

Рис. 10: Окно редактирование правил.

В верхней части окна расположены большое количество кнопок облегчающих ввод различных математических выражений. Арифметические операторы, разные математические символы, элементарные функции, интегралы и производные. Все подобные кнопки работают по одному принципу При нажатии на какую-нибудь из них в текстовое поле внизу слева вставляется соответствующий кнопке шаблон языка Mathematica. В том случае если ввод шаблона требует дополнительных параметров, например, в случае интегралов требуется ввести переменную после дифференциала, справа от кнопки «правила гнерации» появляются текстовые поля в которых можно будет ввести требуемые значения и вставка шаблона в этом случае производится после нажатия кнопки применить.

Для того чтобы сделать описание интерфейса более наглядным рассмотрим пример создания грамматики для простой задачи. В ка-

честве примера рассмотрим задачу нахождение предела выражения вида $\frac{\sin(\alpha x)}{\beta x}$ при $x \to 0$. Составим грамматику, генерирующую задачи с изменяемыми параметрами α и β. Пусть параметр α будет принадлежать диапазону от 3 до 17 а параметр β принадлежать диапазону от 5 до 42.

Начнём с создания правил случайных значений для параметров a и b. Для этого создадим и добавим в грамматику два правила случайных значений с именами a и b.

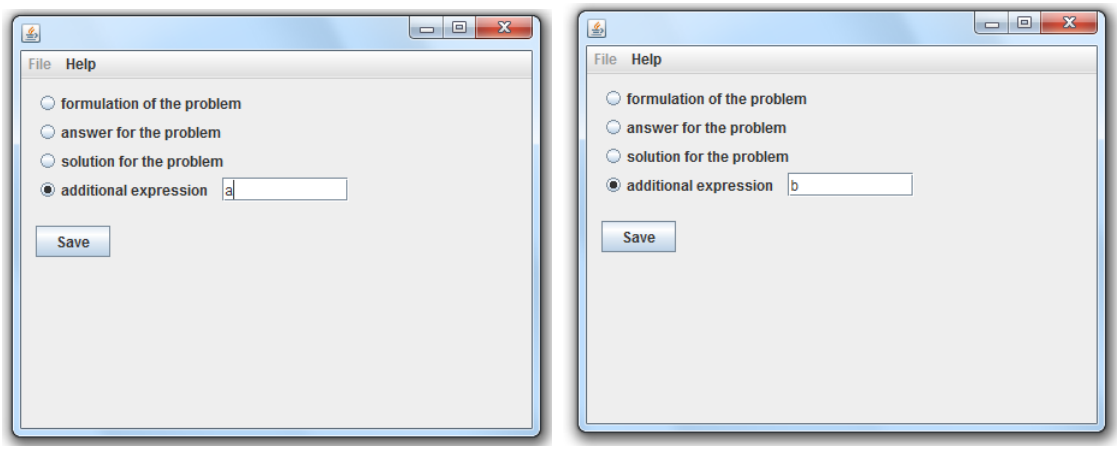

Рис. 11: Добавление параметров «a» и «b».

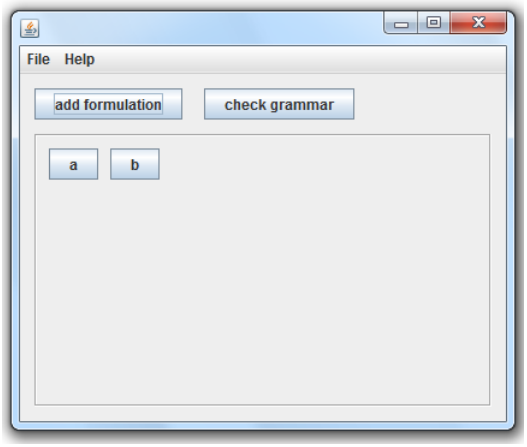

Рис. 12: Список элементов после добавления.

Теперь приступим к их редактированию. В окне редактирования правил есть кнопка «правила генерации». Она специально предназначена для ввода правил случайных значений. При нажатии на неё открывается следующее окно.

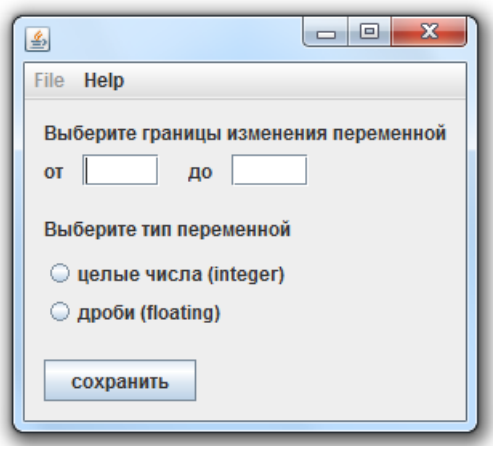

Рис. 13: Окно для генерации случайных значений.

В этом окне можно выбрать желаемый тип случайного значения, целочисленный или действительный, а также диапазон значений в пределах которого должен находиться настраеваемый параметр.

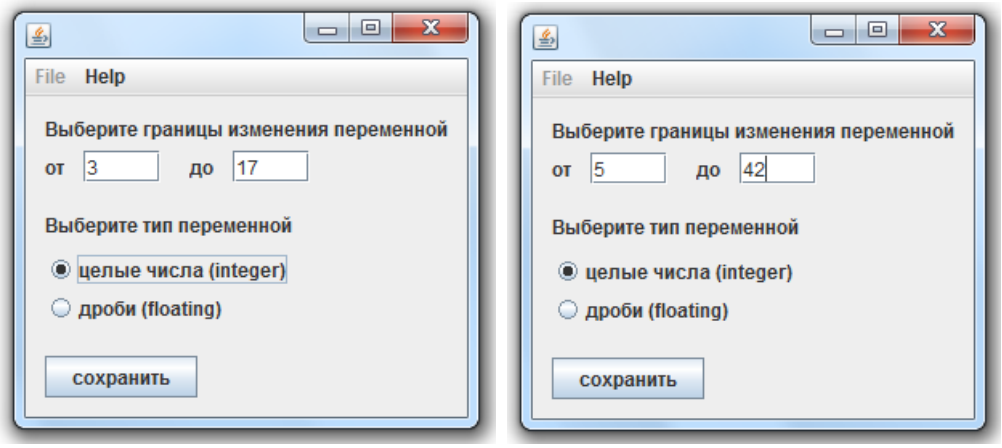

Рис. 14: Настройка параметров a и b.

Нижняя часть окна редактирования разделена на две области. Левая область является текстовым полем для ввода правила. Сюда

нужно вводить весь текст и ссылки на другие элементы, которые будут использлваться. В правой области расположена панель отображения формул. При нажатии на кнопку «посмотреть в виде формулы» конструктор считывает текст из поля ввода и преобразует его в графическое изображение.

Для параметра «a» окно редактирования правил будет выглядеть следующим образом:

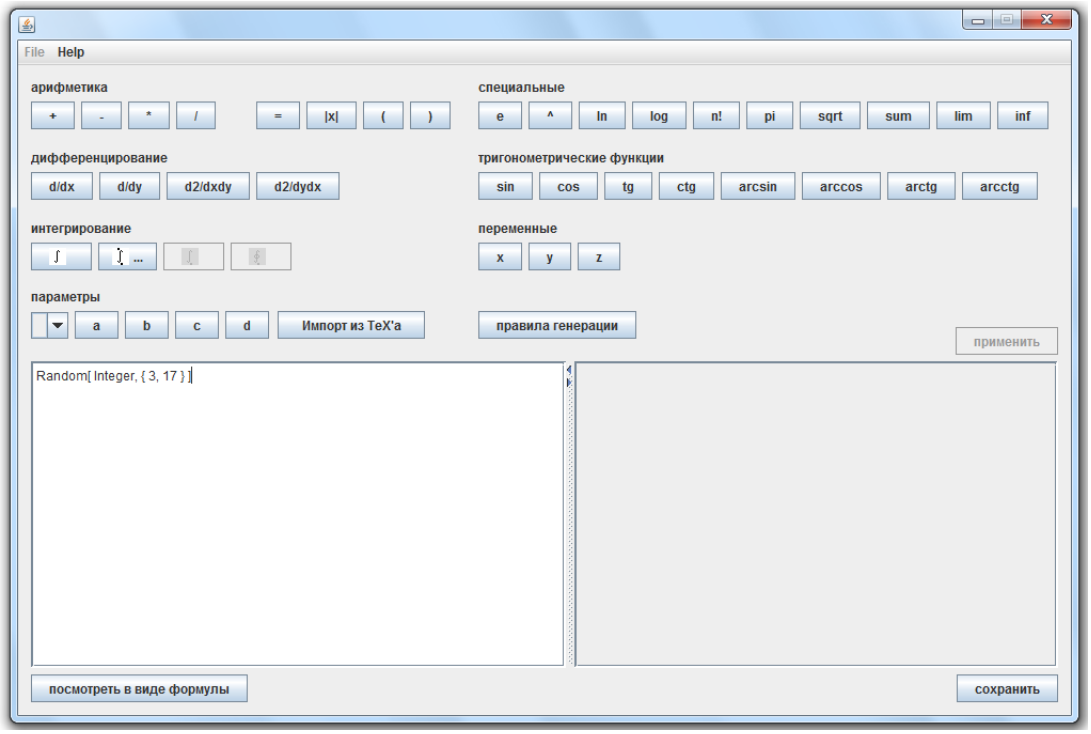

Рис. 15: Окно редактирования для правила случайных значений.

В поле ввода текста правила, появилось выражение язака Mathematica `Random[ Integer, { 3, 17 } ]`. Это выражение, будучи выполненым ядром Mathematica, даст в результате случайное целое число от 3 до 17.

Далее добавляем в грамматику формулировку задачи и заполняем её. Панель параметров, расположенные прямо над полем ввода содержит кнопки для вставки самых ссылок на другие правила. В выпадающем списке слева содержатся именна всех созданных в

грамматике правил.

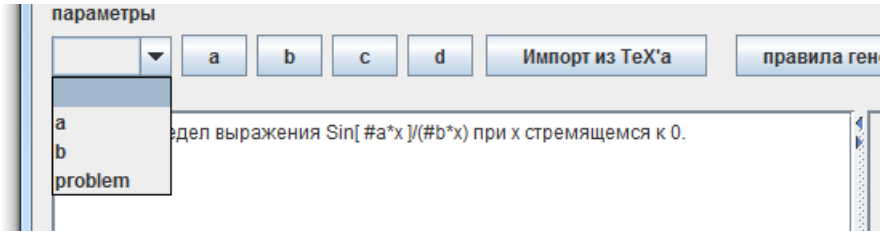

Рис. 16: Список правил.

Поле ввода текста правила будет выглядеть следующим образом.

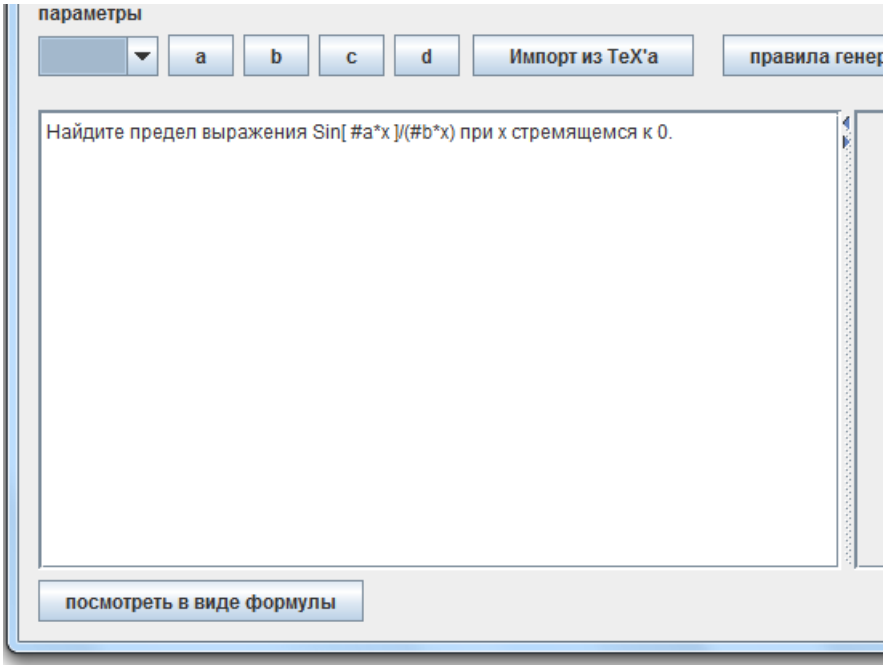

Рис. 17: Формулировка задачи.

Приступим к вводу решения. Добавим в грамматику соответствующее правило и отредактируем его. Для того, что бы продемонстрировать функцию визуализации формул, дополнительно щёлкнем по кнопке «посмотреть в виде формулы». В результате получим следующее:

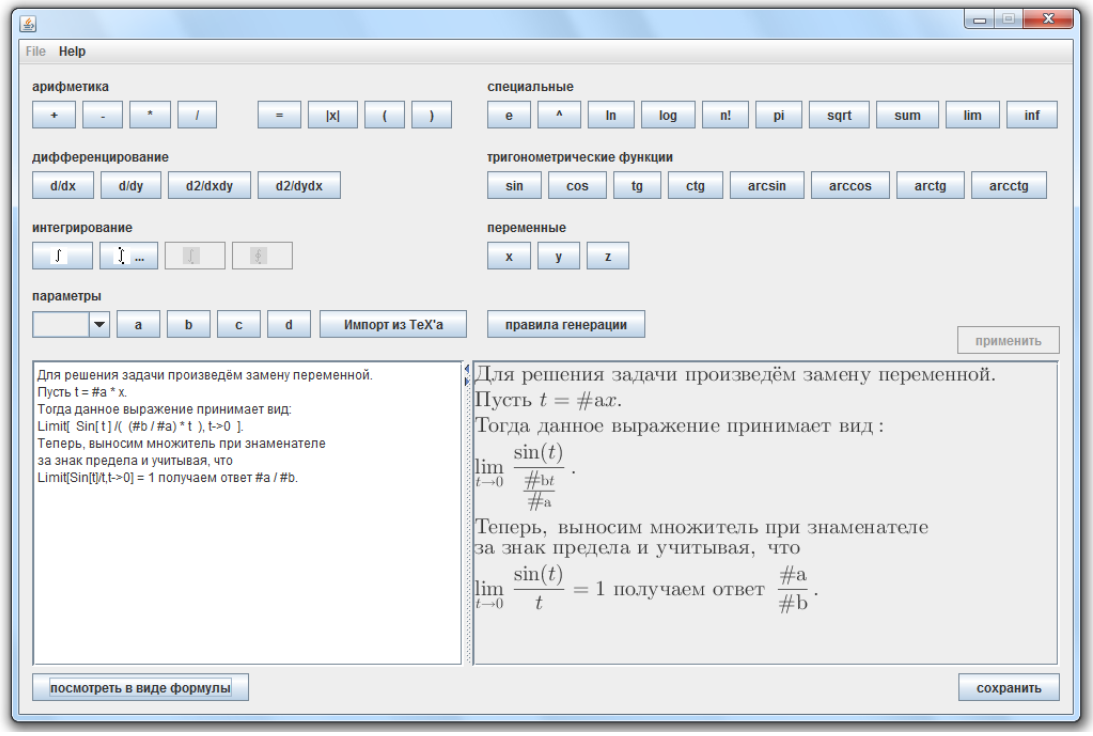

Рис. 18: Решение задачи.

Теперь остаётся только ввести правльный ответ на задачу. Обучающая система будет сравнивать его с ответом, введённым пользователем. Для этого добавляем элемент «answer» и пишем в нём `#a/#b`.

После добавления всех правил главное окно приложения будет выглядеть следующим образом:

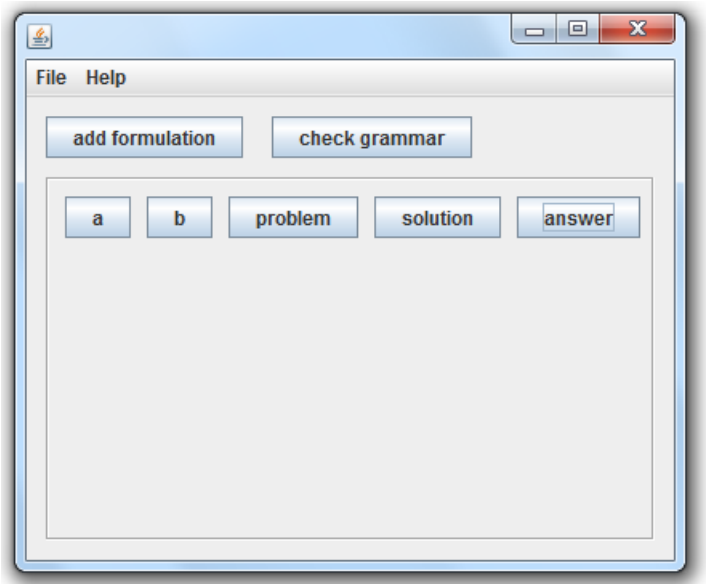

Рис. 19: Главное окно с готовой грамматикой.

Теперь можно сохранить результат в файле или отправить грамматику на сервер. В результате своей работы конструктор сгенерирует следующий текст:

```
a ~{ Random[ Integer, { 3, 17 } ] }
b ~{ Random[ Integer, { 5, 42 } ] }
problem output
{
Найдите предел выражения Sin[ #a*x ]/(#b*x)
    при x стремящемся к 0.
}

solution output
{
Для решения задачи произведём замену переменной.
Пусть t = #a * x.
Тогда исходное выражение принимает вид
Limit[  Sin[ t ] /(  (#b / #a) * t  ), t->0  ].
Теперь, выносим множитель при знаменателе
```

```
за знак предела и учитывая, что
Limit[Sin[t]/t,t->0] = 1 получаем ответ #a / #b.
}

answer ~{ #a / #b }
```

6 Интерфейс пользователя для операционной системы Android

Интерфейс пользователя для операционной системы Android это специальное приложение, позволяющие пользователям использовать обучающую систему с помощью мобильных устройств на базе Android. Это приложение позволяет пользователям подключаться к обучающей системе, посылать ей запросы на генерацию задач и получать сгенерированные задачи. Далее пользователь может послать системе ответ на задачу для того чтобы ядро его проверила. Если ответ на задачу не правильный Andriod приложение отображает решение сгенерированной задачи.

Таким образом описываемый интерфейс пользователя поддерживает весь обучающий процесс для пользователя системы.

Изначально окно приложения содержит элементы управления, позволяющие пользователю войти в систему. При вводе логина Android интерфейс проверяет, зарегистрирован ли в системе пользователь с таким логином. Эта проверка осуществляется с помощью html запроса соответствующей php странице веб-интерфейса. Запрос имеет тип Get. Проверяемый логин пользователя посылается веб-интерфейсу в качестве параметра.

В том случае если пользователь с таким логином зарегистрирован Android интерфейс посылает веб-интерфейсу запрос с требованием вернуть список типов задач, доступных этому пользователю. Веб-интерфейс хранит информацию о пользователе в файлах cookie. При обращении к веб-интерфейсу интерфейс Android устанавливает в своём html запросе требуемые cookie и благодаря этому веб-интерфейс возвращает отклик содержащий нужный список классlв

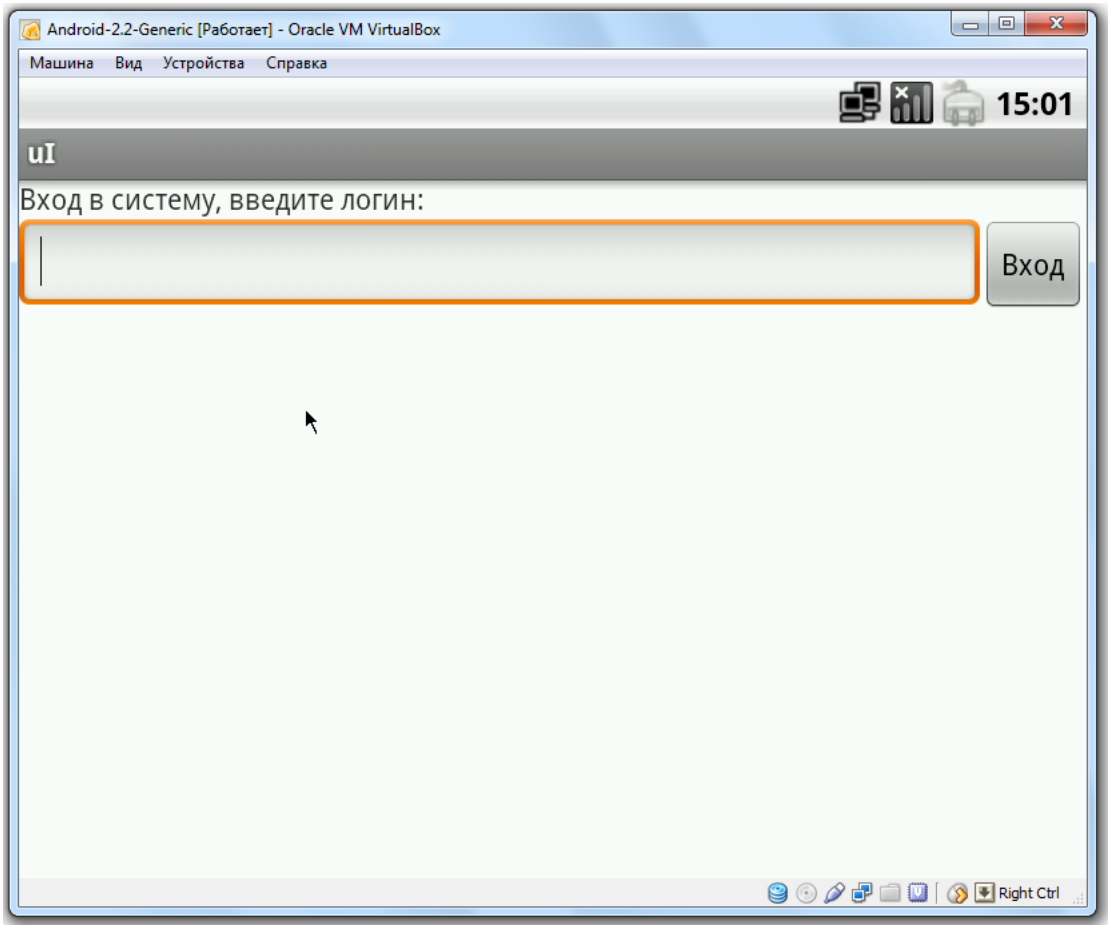

Рис. 20: Вход в систему

задач. Отклик веб-интерфейса возвращается в виде html, поэтому для получения нужного списка Android интерфейс проводит синтаксический анализ отклика и извлекает нужные элементы.

Приложение отображает полученный список задач в виде списка элементов управления каждый из которых относиться к одному из доступных классов задач. При клике на один из таких элементов управления приложение конструирует соответствующий html запрос к ядру системы. В этом запросе указан тип задачи, который нужно будет использовать для генерации. После принятия запроса ядро генерирует задачу и возвращает отклик в формате XML содержащий в себе формулировку задачи, правильный ответ и решение задачи.

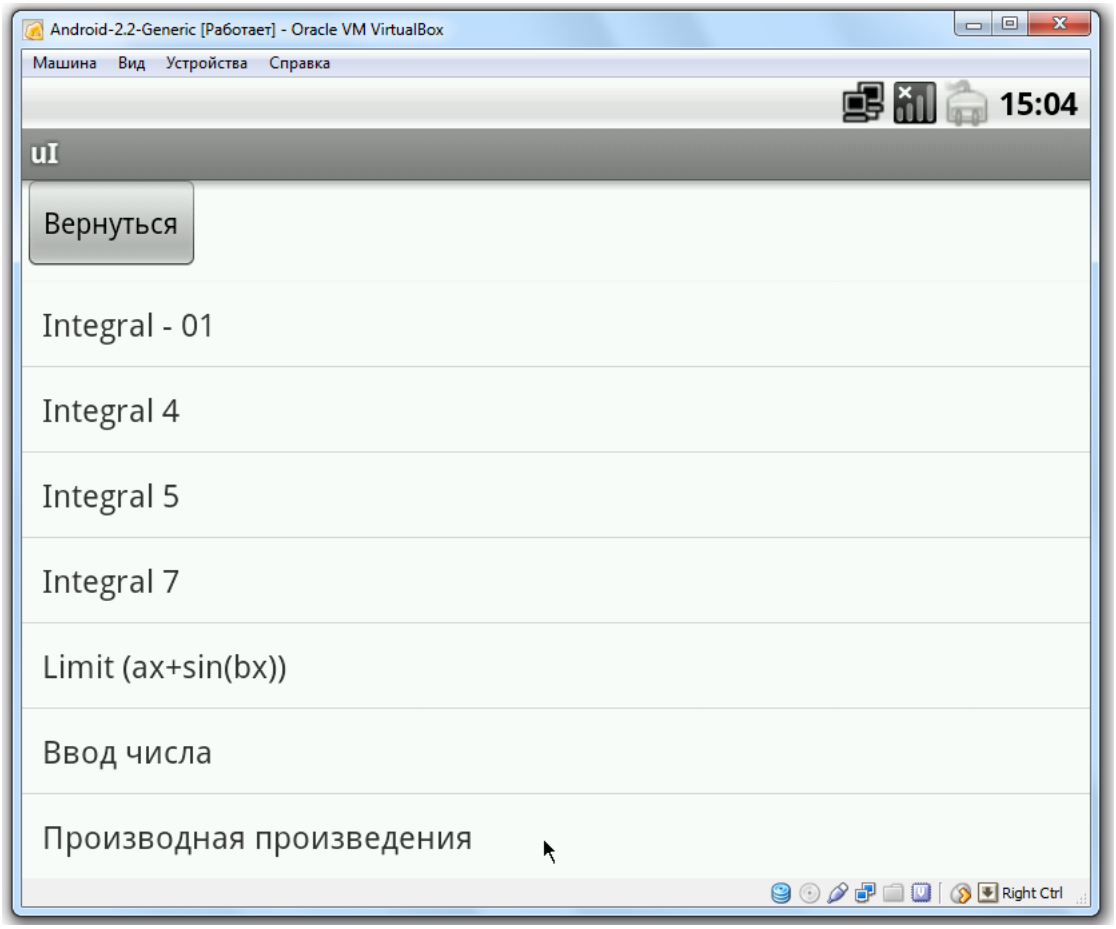

Рис. 21: Список доступных классов задач

После получение сгенерированной задачи Android интерфейсу требуется отобразить на экране условие задачи. Формулировка задачи в XML отклике представляет собой тело html страницы содержащее простой текст и описание математических формул в формате MathML. Отображение формулировки задачи реализованно с помощью встроенного элемента Android WebView, служащего для отображения на экране содержания html страниц. Для того чтобы это стало возможным Android приложение генерирует html страницу, содержащую формулировку задачи и передаёт её элементу WebView.

Самая большая проблема при разработке Android интерфейса заключалась в том, что элемент WebView не поддерживает описа-

ние математических выражений в формате MathML. Следовательно формулы не отображаются в нём как требуется. Для решения этой проблемы была применена javascript библиотека MathJax. Для того, чтобы подключить требуемую библиотеку достаточно было подключить к генерируемой html странице ссылку на скрипт MathJax. В настоящей версии Android интерфейса соответствующая ссылка вставляется в заголовок html страницы формулировки задачи, а также страницы отображающей решение.

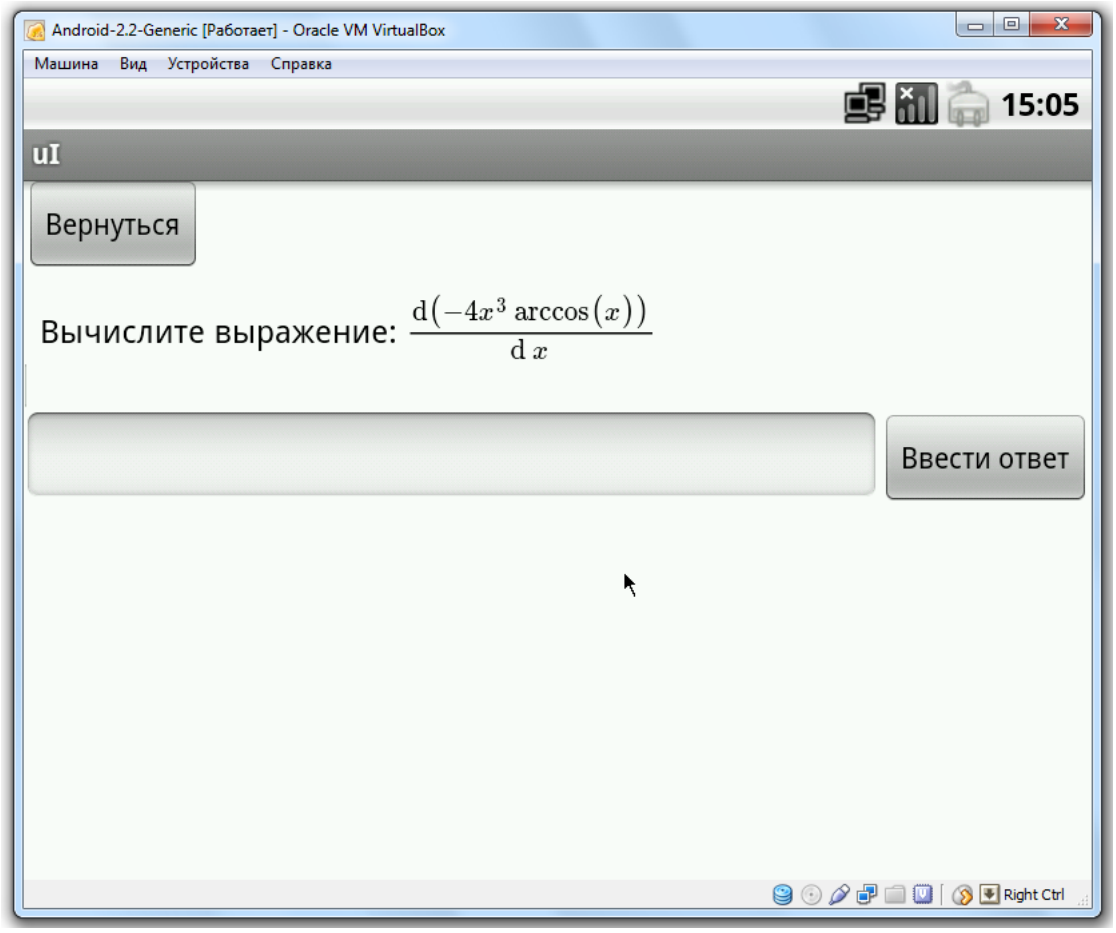

Рис. 22: Пример отображения условия задачи

Находясь в состоянии отображения условия задачи Android интерфейс предоставляет пользователю поле для ввода ответа и кнопку для отправки этого ответа ядру. При нажатии на кнопку приложение

конструирует html запрос к ядру системы, содержащий ответ введённый пользователем и првильный ответ. Android приложение не может самостоятельно сравнить ответы по той причине, что правильный ответ может быть введён пользователем в различных формах. Для того, чтобы корректно сравнивать правильный и пользовательский ответ приложению требовалось бы устанавливать математическую эквивалентность между ответами, что в общем слуучае является слишком сложной задачей для Android приложения. К томуже эту задачу нельзя упростить в силу того, что в системе в перспективе возможно существование типов задач произвольной сложности. По этим причинам математическую эквивалентность проверяет ядро системы используя для этого свой математический модуль. Результат сравнения ответов передаётся обратно Android интерфейсу в html отклике. Если приложение получает негативный результат, то есть результат, говорящий о том, что пользователь ошибся. Система отображает правильное решение задачи.

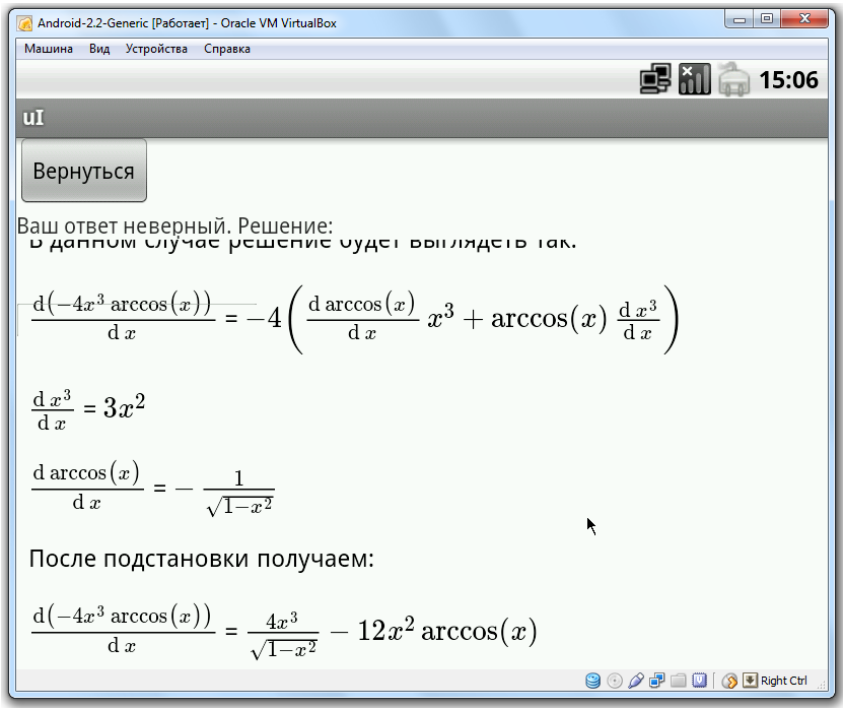

Рис. 23: Пример отображения решения задачи

При отображении решения применяется тотже механизм, что и при отображении формулировки задачи. В данном примере решение задачи не умещается на экране эмулятора целиком. Поэтому отображаемая html страница прокручена в самый низ, к отображаемому ответу.

Библиотека MathJax обрабатывает html страницу, находя в ней описание математических формул в формате MathML и заменяя их изображениями. Эта обработка проводится автоматически при каждом отображении страницы.

После правильного или неправильного решения задачи пользователь может вернуться обратно к списку доступных классов задач кликнув по кнопке «Вернуться».

7 Приложение

Данное приложение содержит исходные тексты программ, отвечающие за функционал, реализованный в рамках настоящей дипломной работы.

Исходные тексты реализованных фунций конструктора

Converter.java

```
package v3;

/**
 *
 * @author Dmitriy Krylov
 */
public abstract class Converter
{
    public  abstract String convert( String input )
            throws IncorrectSyntaxException;
}
```

WolfMathToLaTeXConverter.java

```
package v3;

import java.io.IOException;
import java.util.LinkedList;
import v3.MathConnection.MathConnectionException;

/**
 *
 * @author Dmitriy Krylov
 */
public
class    WolfMathToLaTeXConverter
extends  Converter
{
    @Override
    public String convert( String input )
    {
        if( input == null )
        {
            throw new NullPointerException( "input is null" );
        }

        if( input.equals("") )
        {
```

```
    return "";
}

for( int remainingAttempts = 2
       ; remainingAttempts > 0
       ; remainingAttempts--
){
    try
    {
        Log.msg( "convert "+input+" with local math module" );

        MathConnection mc = MathConnection.getInstance();

        String request  =  "TeXForm[HoldForm["+input+"]]";
        String result   =  mc.sendRequest( request );

        if( result != null )
        {
            Log.msg( "success" );
            return result;
        }
    }
    catch( MathConnectionException ex )
    {
        Log.msg( "fail: "+ex.getLocalizedMessage() );
        Log.msg( "try again" );
    }
}

try
{
    Log.msg( "convert "+input+" with remote math module" );

    java.util.List<String[]> params = new LinkedList<>();

    params.add(new String[]{ "inputFormat" , "WolfMath" });
    params.add(new String[]{ "outputFormat", "LaTeX"    });
    params.add(new String[]{ "text"        , input      });

    CSendTemplateToPost postRequest = new CSendTemplateToPost();

    //посылка запроса
    String url = "http://int.pm.miem.edu.ru/"
              + "taskpower3_war_exploded/converterServlet.do";

    String result = postRequest.sendPostRequest( url, params );

    String prefix = "response page source:";
    int start = result.indexOf(prefix);
    if( start != -1 )
    {
        start += prefix.length() + 2;
        result = result.substring( start );
        if( result.startsWith("error:") == false )
        {
            return result;
        }
    }
}
catch( IOException ex )
{
```

```
            Log.msg( "fail: "+ex.getLocalizedMessage() );
        }

        return null;
    }

    //------------------------------------------------------------------------

    String getErrorMsg()
    {
        return "";
    }
}
```

LaTeXToWolfMathConverter.java

```
package v3;

import java.io.CharArrayWriter;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;
import java.util.Locale;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;
import nl.tue.win.riaca.mathematica.codec.MathematicaCodec;
import nl.tue.win.riaca.openmath.codec.CodecEncodeException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import uk.ac.ed.ph.snuggletex.InputError;
import uk.ac.ed.ph.snuggletex.SnuggleEngine;
import uk.ac.ed.ph.snuggletex.SnuggleInput;
import uk.ac.ed.ph.snuggletex.SnuggleSession;
import uk.ac.ed.ph.snuggletex.internal.util.XMLUtilities;
import uk.ac.ed.ph.snuggletex.upconversion.MathMLUpConverter;
import uk.ac.ed.ph.snuggletex.upconversion.UpConversionFailure;
import uk.ac.ed.ph.snuggletex.upconversion.UpConversionOptionDefinitions;
import uk.ac.ed.ph.snuggletex.upconversion.UpConversionOptions;
import uk.ac.ed.ph.snuggletex.upconversion.UpConversionUtilities;
import uk.ac.ed.ph.snuggletex.upconversion.internal.UpConversionPackageDefinitions;
import uk.ac.ed.ph.snuggletex.utilities.MathMLUtilities;
import uk.ac.ed.ph.snuggletex.utilities.MessageFormatter;

/**
 *
 * @author Dmitriy Krylov
 */
public
class     LaTeXToWolfMathConverter
extends   Converter
{
    private final Locale locale;
```

```
private final SnuggleEngine engine;
private final UpConversionOptions upOpts;
private final MathMLUpConverter upConverter;
private final Transformer cMathMLToOpenMath;

private StringBuilder errorsBuffer = null;

//-------------------------------------------------------------------------

private static LaTeXToWolfMathConverter instance = null;
public  static LaTeXToWolfMathConverter getInstance()
{
    if( instance != null )
    {
        Log.msg( "exist instance" );
    }
    else
    {
        Log.msg( "new instance" );
        instance = new LaTeXToWolfMathConverter();
    }

    return instance;
}

//-------------------------------------------------------------------------

private LaTeXToWolfMathConverter()
{
    Locale.setDefault( Locale.ENGLISH );
    String doMaxima = UpConversionOptionDefinitions.DO_MAXIMA_NAME;
    locale = Locale.getDefault();

    try
    {
        engine=new SnuggleEngine();
        engine.addPackage( UpConversionPackageDefinitions.getPackage() );

        upConverter = new MathMLUpConverter();

        upOpts=new UpConversionOptions();
        upOpts.setSpecifiedOption( doMaxima, "false" );

        TransformerFactory factory = TransformerFactory.newInstance();

        InputStream  xslStream;
        StreamSource xslSource;

        xslStream = getClass().getResourceAsStream( "cmmltoom.xsl" );
        xslSource = new StreamSource( xslStream );

        cMathMLToOpenMath = factory.newTransformer( xslSource );

    }
    catch( TransformerConfigurationException ex )
    {
        throw new RuntimeException( ex );
    }
    finally
    {
        Locale.setDefault( locale );
```

```
    }
}

//-------------------------------------------------------------------------

@Override
public String convert( String latex ) throws IncorrectSyntaxException
{
    assert latex != null;

    String result = null;

    conversion:try
    {
        Locale.setDefault( Locale.ENGLISH );
        errorsBuffer = new StringBuilder();

        // ----

        Document mathDoc = XMLUtilities
                          . createNSAwareDocumentBuilder()
                          . newDocument();

        Element root = mathDoc.createElement( "root" );
        mathDoc.appendChild( root );

        // ----

        SnuggleSession session = engine.createSession();
        session.parseInput( new SnuggleInput(latex) );
        session.buildDOMSubtree( root );

        // ----

        List<InputError> errors = session.getErrors();

        if( errors.isEmpty() == false )
        {
            for( InputError error : errors )
            {
                String errorMsg = MessageFormatter
                                 . formatErrorAsString( error );

                errorsBuffer.append('\n').append( errorMsg );
            }

            throw new IncorrectSyntaxException( errorsBuffer.toString() );
        }

        // ----

        Element pMathML = null;
        NodeList  nodes = root.getChildNodes();

        for( int i = 0; i < nodes.getLength(); ++i )
        {
            Node node = nodes.item( i );
            if( MathMLUtilities.isMathMLElement(node) )
            {
                if( pMathML == null )
                {
```

```
                pMathML = (Element) node;
            }
            else
            {
                String msg = "More than one math element.";
                throw new RuntimeException( msg );
            }
        }
    }

    if( pMathML == null )
    {
        throw new RuntimeException( "No math input." );
    }

    // ----

    mathDoc.removeChild( root );
    mathDoc.appendChild( pMathML );

    mathDoc = upConverter
            . upConvertSnuggleTeXMathML( mathDoc, upOpts );

    // ----

    List<UpConversionFailure> fails =
    UpConversionUtilities.extractUpConversionFailures( mathDoc );

    if( fails.isEmpty() == false )
    {
        for( UpConversionFailure fail : fails )
        {
            String errorMsg = UpConversionUtilities
                            . getErrorMessage( fail );

            errorsBuffer.append('\n').append( errorMsg );
        }

        throw new IncorrectSyntaxException( errorsBuffer.toString() );
    }

    // ----

    mathDoc = MathMLUtilities.isolateAnnotationXML(
                  mathDoc.getDocumentElement(),
                  MathMLUpConverter.CONTENT_MATHML_ANNOTATION_NAME);

    Element cMathML = mathDoc.getDocumentElement();

    // ----

    CharArrayWriter caw = new CharArrayWriter();

    cMathMLToOpenMath.transform( new DOMSource(cMathML)
                               , new StreamResult(caw));

    String openMath = caw.toString();

    // ----

    MathematicaCodec codec = new MathematicaCodec();
```

```
            String wolfMath = codec.encode( openMath );

            result = wolfMath;
        }
        catch( CodecEncodeException ex )
        {
            throw new IncorrectSyntaxException( ex.getMessage() );
        }
        catch( TransformerException | IOException ex )
        {
            Log.msg( ex.getLocalizedMessage() );
            throw new RuntimeException( ex );
        }
        finally
        {
            Locale.setDefault( locale );
        }

        return result;
    }

    //-------------------------------------------------------------------------

    public static void main( String[] args ) throws TransformerException, IncorrectSyntaxException
    {
        new LaTeXToWolfMathConverter().convert( "$x*(y+1)$" );
    }

    public void print( Object obj )
    {
        System.out.println( obj );
    }
}
```

SpecialCaseConverter.java

```
package v3;

import java.util.Arrays;
import java.util.List;

/**
 *
 * @author Dmitriy Krylov
 */
public
class   SpecialCaseConverter
extends Converter
{
    private final Converter converter;

    private final static
            List <Class <? extends SpecialCase> > cases = Arrays.asList(
                DerivativeSpecialCase.class,
                IntegralSpecialCase.class,
                SumSpecialCase.class,
                SubScriptSpecialCase.class);

    //-------------------------------------------------------------------------
```

```
public SpecialCaseConverter( Converter converter )
{
    this.converter = converter;
}

//-------------------------------------------------------------------------

@Override
public String convert( String input ) throws IncorrectSyntaxException
{
    SpecialCaseReport report = applySpecialCases( input );
    if( report == null )
    {
        return converter.convert( input );
    }
    assert report.isValid( input ) : "invalid special case report";

    // ----

    char    sCh       =  SpecialCase.specialChar;

    int     left      =  report.getLeft();
    int     right     =  report.getRight();
    String  specConv  =  report.getConversionResult();

    // ----

    int numSpecialChars = 0;
    for( int pos = 0; pos < left; ++pos )
    {
        char ch = input.charAt( pos );
        if( ch == sCh )
        {
            numSpecialChars++;
        }
    }

    // ----

    input = new StringBuilder( input )
          . replace( left, right, String.valueOf(sCh) )
          . toString();

    // ----

    String result = this.convert( input );
    if( result == null )
    {
        return null;
    }

    // ----

    for( int pos = 0; pos < result.length(); ++pos )
    {
        char ch = result.charAt( pos );
        if( ch == sCh )
        {
            if( numSpecialChars > 0 )
            {
```

```
                --numSpecialChars;
            }
            else
            {
                result = new StringBuilder( result )
                       . replace( pos, pos+1, specConv )
                       . toString();
                break;
            }
        }
    }

    // ----

    return result;
}

//-------------------------------------------------------------------------

private SpecialCaseReport applySpecialCases( String input )
        throws IncorrectSyntaxException
{
    SpecialCaseReport result = null;

    for( Class c : cases )
    {
        try
        {
            SpecialCase sc = (SpecialCase) c.newInstance();

            result = sc.apply( input );

            if( result != null )
            {
                break;
            }
        }
        catch( InstantiationException | IllegalAccessException ex )
        {
            throw new RuntimeException( ex );
        }
    }

    return result;
}

//-------------------------------------------------------------------------

public static void main( String[] args )
{
    try{
    LaTeXToWolfMathConverter baseConverter = LaTeXToWolfMathConverter
                                           . getInstance();
    SpecialCaseConverter conv = new SpecialCaseConverter( baseConverter );

    String input = "$\\int \\frac{x}{2} dx$";
    String convert = conv.convert( input );
    System.out.println( convert );
    }catch( Throwable e ){ }
}
```

```
}
```

SpecialCase.java

```
package v3;

/**
 *
 * @author Dmitriy Krylov
 */
public abstract class SpecialCase
{
    public static final char specialChar = '@';

    public abstract SpecialCaseReport apply( String input )
            throws IncorrectSyntaxException;
}
```

SpecialCaseReport.java

```
package v3;

/**
 *
 * @author Dmitriy Krylov
 */
public class SpecialCaseReport
{
    private final int left;
    private final int right;
    private final String conversionResult;

    // ------------------------------------------------------------------------

    public SpecialCaseReport( int left, int right, String conversionResult )
    {
        this.left = left;
        this.right = right;
        this.conversionResult = conversionResult;

        Log.msg( "("+left+","+right+")"+conversionResult );
    }

    public int getLeft()
    {
        return left;
    }

    public int getRight()
    {
        return right;
    }

    public String getConversionResult()
    {
        return conversionResult;
    }
```

```
    public boolean isValid( String input )
    {
        return ( left  > 0
              && right > 0
              && left  < right
              && right <= input.length() );
    }
}
```

DerivativeSpecialCase.java

```
package v3;

import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 *
 * @author Dmitriy Krylov
 */
public
class     DerivativeSpecialCase
extends  SpecialCase
{
    private static final String fracCmd = "\\frac";

    private LaTeXParser parser = null;

    private String diffType  = null;
    private String diffFunc  = null;
    private String diffOrder = null;

    private ArrayList<String> diffVars = null;
    private ArrayList<String> diffOrders = null;

    private boolean derivativeReject = false;
    private boolean firstOrder = true;

    private Converter converter = new SpecialCaseConverter(
            LaTeXToWolfMathConverter.getInstance() );

    //-------------------------------------------------------------------------

    @Override
    public  SpecialCaseReport apply( String input )
            throws IncorrectSyntaxException
    {
        assert input != null;

        SpecialCaseReport report = null;

        apply:
        {
            int entryIndex = input.indexOf( fracCmd );
            if( entryIndex == -1 )
            {
                break apply;
```

```
        }

        parser = new LaTeXParser( input, entryIndex );
        parser.readLaTeXCommand();

        parseNumerator();
        if( derivativeReject )
        {
            break apply;
        }

        parseDenominator();
        if( derivativeReject )
        {
            break apply;
        }

        String converted = convertDerivative();

        int start  =  entryIndex;
        int end    =  parser.getCurrentPos();

        report = new SpecialCaseReport( start, end, converted );
    }

    return report;
}

private void parseNumerator() throws IncorrectSyntaxException
{
    boolean success = false;
    String numerator = extractBlock();
    if( numerator == null || numerator.isEmpty() )
    {
        return;
    }

    try
    {
        LaTeXParser lp = new LaTeXParser( numerator );
        String prefix = lp.readBlock();

        boolean correctPrefix = prefix.equals("d")
                             || prefix.equals("\\partial");
        if( correctPrefix )
        {
            String func;
            String order;

            String next = lp.readBlock();
            if( ! next.equals("^") )
            {
                firstOrder = true;
                order = "1";
                func = next;
            }
            else
            {
                firstOrder = false;
                order = lp.readBlock();
                func = lp.readBlock();
```

```
            }

            if( lp.endOfInput() )
            {
                diffFunc  = func;
                diffType  = prefix;
                diffOrder = order;
                success = true;
            }
        }
    }
    catch( IncorrectSyntaxException ex )
    {
        //This is not derivative. Just return.
    }

    derivativeReject = !success;
}

private void parseDenominator() throws IncorrectSyntaxException
{
    parse:{

    derivativeReject = true;

    String denominator = extractBlock();
    if( denominator == null || denominator.isEmpty() )
    {
        break parse;
    }

    LaTeXParser lp = new LaTeXParser( denominator );

    String firstBlock = lp.readBlock();
    if( ! firstBlock.equals(diffType) )
    {
        break parse;
    }

    if( firstOrder )
    {
        String var = lp.readBlock();
        if( ! lp.endOfInput() )
        {
            break parse;
        }

        diffVars = new ArrayList<>(1);
        diffOrders = new ArrayList<>(1);
        diffVars.add( var );
        diffOrders.add( "1" );
    }
    else
    {
        final int readVar = 1;
        final int readOrder = 2;
        final int startRead = 3;
        final int determOrder = 4;
        final int compliteRead = 5;

        String var = null;
```

```
String order = null;

ArrayList<String> vars = new ArrayList<>();
ArrayList<String> orders = new ArrayList<>();

boolean correctReading = true;

lp.setCurrentPos( 0 );
int nextAction = startRead;

reading:
while( true )
{
    if( lp.endOfInput()
     && nextAction != compliteRead
     && nextAction != determOrder
    ){
        correctReading = false;
        break reading;
    }

    switch( nextAction )
    {
        case compliteRead:
            vars.add( var );
            orders.add( order );

            if( lp.endOfInput() )
            {
                correctReading = true;
                break reading;
            }

            nextAction = startRead;
            break;

        case startRead:
            String diff = lp.readBlock();
            if( !diff.equals(diffType) )
            {
                break parse;
            }
            nextAction = readVar;
            break;

        case readVar:
            var = lp.readBlock();
            nextAction = determOrder;
            break;

        case determOrder:
            if( lp.endOfInput() )
            {
                order = "1";
                nextAction = compliteRead;
                break;
            }

            int currPos = lp.getCurrentPos();
            if( lp.readBlock().equals("^") )
            {
```

```
                        nextAction = readOrder;
                    }
                    else
                    {
                        lp.setCurrentPos( currPos );
                        order = "1";
                        nextAction = compliteRead;
                    }

                    break;

                case readOrder:
                    order = lp.readBlock();
                    nextAction = compliteRead;
                    break;
            }
        } // reading:

        if( !correctReading || vars.isEmpty() )
        {
            break parse;
        }

        this.diffVars = vars;
        this.diffOrders = orders;
    }

    derivativeReject = false;
    }//parse:
}

private String convertDerivative() throws IncorrectSyntaxException
{
    StringBuilder result = new StringBuilder( "D[" );
    result.append( convert(diffFunc) );

    int len = diffVars.size();
    for( int i=0; i<len; ++i )
    {
        result.append( "," + "{" );
        result.append( convert(diffVars.get(i)) );
        result.append( "," );
        result.append( convert(diffOrders.get(i)) );
        result.append( "}" );
    }

    result.append( "]" );
    return result.toString();
}

private boolean inBracket(String text)
{
    assert text != null;
    String txt = text.trim();

    int len = txt.length();
    if( len < 2 )
    {
        return false;
    }
```

```
        char first = txt.charAt( 0 );
        char last  = txt.charAt( len - 1 );

        return parser.isLeftBracket(first)
            && parser.oppositeBracket(first) == last;
    }

    public static void main( String[] args )
    {
        try
        {
            new DerivativeSpecialCase().apply( "$\\frac{ d x  }{dy}$");
        }
        catch( IncorrectSyntaxException ex )
        {
        }
    }

    private String extractBlock()
            throws IncorrectSyntaxException
    {
        String result = null;
        String denominator = parser.readBlock().trim();

        if( inBracket(denominator) )
        {
            denominator = denominator
                        . trim()
                        . substring( 1, denominator.length() - 1 )
                        . trim();

            if( ! denominator.isEmpty() )
            {
                result = denominator;
            }
        }

        return result;
    }

    private String convert( String latex ) throws IncorrectSyntaxException
    {
        return converter.convert( '$'+latex+'$' );
    }
    private void firstOrderCase()
    {

    }

    private void higherOrderCase()
    {

    }
}
```

SubScriptSpecialCase.java

```
package v3;
```

```
/**
 *
 * @author Dmitriy Krylov
 */
public
class     SubScriptSpecialCase
extends  SpecialCase
{
    private int startIndex = -1;
    private int endIndex   = -1;

    private LaTeXParser parser = null;

    private final Converter converter;

    //-------------------------------------------------------------------------

    public SubScriptSpecialCase()
    {
        converter = new SpecialCaseConverter(
                LaTeXToWolfMathConverter.getInstance());
    }

    //-------------------------------------------------------------------------

    /* Этот специальный случай должен применяться ПОСЛЕ
     * специальных случаев, обрабатывающие выражения,
     * которые могут содержать нижний предел, таких как
     * интегралы и суммы.
     */

    /* Метод ищет во входной строке случай применения
     * нижнего индекса, извлекает элемент, обладающий
     * нижним индексом и сам нижний индекс, конвертирует
     * их и составляет соответствующее выражение на языке
     * Mathematica.
     */
    @Override
    public  SpecialCaseReport apply( String input )
            throws IncorrectSyntaxException
    {
        assert input != null;

        int entryIndex = input.lastIndexOf( '_' );
        if( entryIndex == -1 ){ return null; }

        parser = new LaTeXParser( input, entryIndex );

        String element   = extractSubscriptElement();
        String subscript = extractSubscript();

        String converted = "Subscript"
                         + "["
                         +     convert( element )
                         +     ","
                         +     convert( subscript )
                         + "]"
                         ;

        return new SpecialCaseReport( startIndex, endIndex, converted );
    }
```

```
    //-------------------------------------------------------------------------

    private String extractSubscriptElement()
            throws IncorrectSyntaxException
    {
        int oldPos = parser.getCurrentPos();
        String result = parser.readPreviousBlock();
        startIndex = parser.getCurrentPos();
        parser.setCurrentPos( oldPos + 1 );
        return result;
    }

    //-------------------------------------------------------------------------

    private String extractSubscript()
            throws IncorrectSyntaxException
    {
        String result = parser.readBlock();
        endIndex = parser.getCurrentPos();
        return result;
    }

    //-------------------------------------------------------------------------

    private String convert( String latex ) throws IncorrectSyntaxException
    {
        return converter.convert( '$'+latex+'$' );
    }
}
```

IntegralSpecialCase.java

```
package v3;

import v3.LaTeXParser.Limits;

/**
 *
 * @author Dmitriy Krylov
 */
public
class    IntegralSpecialCase
extends  SpecialCase
{
    private final String intCmd = "\\int";
    private final String header = "Integrate";
    private final Converter converter;
    private LaTeXParser parser;

    //-------------------------------------------------------------------------

    public IntegralSpecialCase()
    {
        converter = new SpecialCaseConverter(
                LaTeXToWolfMathConverter.getInstance() );

    }
```

```
//-----------------------------------------------------------------------

/* apply
 * Метод принимает в качестве входного параметра
 * строку содержащую описание формулы на языке LaTeX,
 * ищет в отоэ строке описание интеграла и конвертирует
 * его в формат Wolfram Mathematica.
 *
 * Если интеграл удалось найти
 * Возвращает объект содержащий индексы начала и конца
 * найденного интегралла и результат конвертации.
 *
 * Иначе возвращает null
 *
 * Генерирует исключение если интеграл удалось найти но
 * не удалось конвертировать.
 */
@Override
public  SpecialCaseReport apply( String input )
        throws IncorrectSyntaxException
{
    assert input != null;

    //Найдём первое вхождение команды иинтеграла языка LaTeX.

    //Будем считать найденное вхождение началом интеграла.
    //Если строка не содержит этой команды будем считать что
    //она не содержит описание интеграла.
    //В этом случае возвращаем null

    //Требуется определить есть ли у рассматримваемого интеграла
    //нижний и верхний предел.
    //Найдём следующий за командой интеграла непробельный символ
    //Если он является одним из символов '_' или '^'
    //обрабатываем интеграл как определённый.
    //Иначе обрабатываем интеграл как неопределённый.

    //Если обнаружено что у интеграла есть только верхний
    //или только нижний предел, или если у интеграла
    //два или более нижних или верхних предела синтаксис
    //интеграла считается некорректным и метод кидает
    //соответствующее исключение.

    int entryIndex = input.lastIndexOf( intCmd );

    if( entryIndex == -1 )
    {
        return null;
    }

    parser = new LaTeXParser( input, entryIndex );

    parser.readLaTeXCommand();

    // ---

    int startPos = entryIndex;

    String conversionResult;
    parser.skipSpaces();
    switch( parser.currentChar() )
```

```
    {
        case '_':
        case '^':
            conversionResult = treatAsDefiniteIntegral();
            break;

        default :
            conversionResult = treatAsIndefiniteIntegral();
            break;
    }

    int endPos = parser.getCurrentPos();

    return new SpecialCaseReport( startPos, endPos, conversionResult );

}

//-------------------------------------------------------------------------

private String treatAsDefiniteIntegral() throws IncorrectSyntaxException
{
    Limits limits        =  parser.readLowerAndUpperLimits();
    String lowerLimit    =  limits.lower;
    String upperLimit    =  limits.upper;

    String integrand     =  readIntegrand();
    String differential  =  readDifferential();

    return header
         + "["
         +    convert( integrand )
         + ","
         +    "{"
         +       convert( differential )
         +       ","
         +       convert( lowerLimit )
         +       ","
         +       convert( upperLimit )
         +    "}"
         + "]";
}

//-------------------------------------------------------------------------

private String treatAsIndefiniteIntegral() throws IncorrectSyntaxException
{
    String integrand     =  readIntegrand();
    String differential  =  readDifferential();

    return header
         + "["
         + convert( integrand )
         + ","
         + convert( differential )
         + "]";
}

//-------------------------------------------------------------------------

private String readIntegrand() throws IncorrectSyntaxException
{
```

```
        parser.skipSpaces();

        int startIndex = parser.getCurrentPos();

        boolean lastBlockIsMathOperator = false;

        for(;;)
        {
            String readBlock = parser.readBlock();

            if( readBlock.equals("d")
             && lastBlockIsMathOperator == false
            ){
                break;
            }

            lastBlockIsMathOperator = parser.isMathOperator( readBlock );
        }

        int endIndex = parser.getCurrentPos() - 1;

        return parser.getPart( startIndex, endIndex );
    }

    //-------------------------------------------------------------------------

    private String readDifferential() throws IncorrectSyntaxException
    {
        parser.skipSpaces();
        return parser.readBlock();
    }

    //-------------------------------------------------------------------------

    private String convert( String text ) throws IncorrectSyntaxException
    {
        return converter.convert( '$'+text+'$' );
    }

    //-------------------------------------------------------------------------

    public static void main(String[] args) throws IncorrectSyntaxException
    {
        new IntegralSpecialCase().apply( "$a+b+s+\\int_{y}^{x} a+d+s+x dx$");
    }
}
```

SumSpecialCase.java

```
package v3;

import v3.LaTeXParser.Limits;

/**
 *
 * @author Dmitriy Krylov
 */
public
class    SumSpecialCase
```

```
extends  SpecialCase
{
    private final String sumCmd = "\\sum";

    private String convert = null;
    private String sumBody = null;
    private String lowerLimit = null;
    private String upperLimit = null;

    private LaTeXParser parser;

    //-------------------------------------------------------------------------

    @Override
    public  SpecialCaseReport apply(String input)
            throws IncorrectSyntaxException
    {
        if( input == null )
        {
            throw new RuntimeException( "argument is null" );
        }

        int intEntryIndex = input.lastIndexOf( sumCmd );
        if( intEntryIndex == -1 )
        {
            return null;
        }

        parser = new LaTeXParser( input, intEntryIndex );

        int startPos = intEntryIndex;

        readSumLimits();
        readSumBody();
        convertSum();

        int endPos = parser.getCurrentPos();

        return new SpecialCaseReport( startPos, endPos, convert );
    }

    //-------------------------------------------------------------------------

    private void readSumLimits() throws IncorrectSyntaxException
    {
        Limits limits = parser.readLowerAndUpperLimits();

        lowerLimit = limits.lower;
        upperLimit = limits.upper;
    }

    //-------------------------------------------------------------------------

    private void readSumBody() throws IncorrectSyntaxException
    {
        sumBody = parser.readBlock();
    }

    //-------------------------------------------------------------------------

    private void convertSum() throws IncorrectSyntaxException
```

```
    {
        if( lowerLimit == null
          || upperLimit == null
          || sumBody    == null
        ){
            throw new IncorrectSyntaxException();
        }

        String cmdHeader = "Sum";

        SpecialCaseConverter conv =
                new SpecialCaseConverter(
                    LaTeXToWolfMathConverter.getInstance() );

        String cSumBody     =  conv.convert( "$$"+sumBody+"$$" );
        String cLowerLimit  =  conv.convert( "$$"+lowerLimit+"$$" );
        String cUpperLimit  =  conv.convert( "$$"+upperLimit+"$$" );

        convert = cmdHeader
                + "["
                +     cSumBody
                + ","
                +     "{"
                +         cLowerLimit
                +         ","
                +         cUpperLimit
                +     "}"
                + "]";
    }

    //-------------------------------------------------------------------------

    public static void main( String[] args )
    {}
}
```

LaTeXParser.java

```
package v3;

/**
 *
 * @author Dmitriy Krylov
 */
public class LaTeXParser
{
    private final int length;
    private final String input;

    private int currentPos = -1;

    //-------------------------------------------------------------------------

    public LaTeXParser()
    {
        this( null, -1 );
    }

    public LaTeXParser( String input )
```

```
{
    this( input, 0 );
}

public LaTeXParser( String input, int currentPos )
{
    this.input = input;
    this.currentPos = currentPos;
    length = input.length();
}

//-------------------------------------------------------------------------

public String getPart( int startIndex, int endIndex )
{
    return input.substring( startIndex, endIndex );
}

public char getChar( int pos )
{
    return input.charAt( pos );
}

//-------------------------------------------------------------------------

public Limits readLowerAndUpperLimits() throws IncorrectSyntaxException
{
    String lowerLimit;
    String upperLimit;

    switch( currentChar() )
    {
        case '_':
            lowerLimit = readLowerLimit();
            upperLimit = readUpperLimit();
            break;

        case '^':
            upperLimit = readUpperLimit();
            lowerLimit = readLowerLimit();
            break;

        default :
            String error = "expect upper or lower limit";
            throw new IncorrectSyntaxException( error );
    }

    return new Limits( lowerLimit, upperLimit );
}

//-------------------------------------------------------------------------

public String readLowerLimit() throws IncorrectSyntaxException
{
    skipSpaces();
    if( currentChar() != '_' )
    {
        String errorMessage = "expect lower limit";
        throw new IncorrectSyntaxException( errorMessage );
    }
```

```
    nextNonSpace();
    return readBlock();
}

//-------------------------------------------------------------------------

public String readUpperLimit() throws IncorrectSyntaxException
{
    skipSpaces();
    if( currentChar() != '^' )
    {
        String errorMessage = "expect upper limit";
        throw new IncorrectSyntaxException( errorMessage );
    }

    nextNonSpace();
    return readBlock();
}

//-------------------------------------------------------------------------

public String readBlock() throws IncorrectSyntaxException
{
    skipSpaces();
    char c = currentChar();

    if( c == '\\' )
    {
        return readLaTeXCommand();
    }

    if( isLeftBracket(c) )
    {
        return readBreacketsBlock( c );
    }

    return readChar();
}

//-------------------------------------------------------------------------

public String readPreviousBlock() throws IncorrectSyntaxException
{
    int endPos = lookupPreviousNonSpacePos() + 1;
    int startPos = -1;

    char c = getChar( endPos - 1 );
    if( isRightBracket(c) )
    {
        startPos = lookupLeftBracketPos( endPos - 1 );
    }
    else
    if( isLatinLetter(c) )
    {
        int cmdPos = lookupBeginLaTeXCommand( endPos - 1 );

        if( cmdPos != -1 )
        {
            startPos = cmdPos;
        }
        else
```

```
        {
            startPos = endPos - 1;
        }
    }
    else
    {
        startPos = endPos - 1;
    }

    setCurrentPos( startPos );
    return getPart( startPos, endPos );
}


//-----------------------------------------------------------------------

public String readLaTeXCommand() throws IncorrectSyntaxException
{
    skipSpaces();

    if( currentChar() != '\\' )
    {
        String errorMessage = "expect LaTeX command";
        throw new IncorrectSyntaxException( errorMessage );
    }

    nextChar();

    if( currentChar() == ' ' )
    {
        return "\\ ";
    }

    if( isLatinLetter(currentChar()) )
    {
        int startCommand = getCurrentPos() - 1;

        do
        {
            nextChar();
        }
        while( Character.isLetter(currentChar()) );

        return input.substring( startCommand, getCurrentPos() );
    }

    throw new IncorrectSyntaxException( "expect laTeX command" );
}

//-----------------------------------------------------------------------

public void skipSpaces() throws IncorrectSyntaxException
{
    while( isSpace(currentChar()) )
    {
        nextChar();
    }
}

//-----------------------------------------------------------------------
```

```
public char currentChar() throws IncorrectSyntaxException
{
    if( currentPos >= input.length() )
    {
        String errorMessage = "end of input";
        throw new IncorrectSyntaxException( errorMessage );
    }

    return input.charAt( currentPos );
}

//-----------------------------------------------------------------------

public void setCurrentPos( int newPos )
{
    currentPos = newPos;
}

//-----------------------------------------------------------------------

public int getCurrentPos()
{
    return currentPos;
}

//-----------------------------------------------------------------------

public char nextNonSpace() throws IncorrectSyntaxException
{
    if( !isSpace(currentChar()) )
    {
        nextChar();
    }

    skipSpaces();
    return currentChar();
}

//-----------------------------------------------------------------------

public  int lookupPreviousNonSpacePos( )
        throws IncorrectSyntaxException
{
    int result = -1;
    boolean nonSpaceFound = false;

    for( int pos = getCurrentPos() - 1
           ; pos >= 0 && !nonSpaceFound
           ; pos--
    ){
        char c = input.charAt( pos );
        if( Character.isWhitespace(c) == false )
        {
            result = pos;
            nonSpaceFound = true;
        }
    }

    return result;
}
```

```
//-----------------------------------------------------------------------

public boolean isLatinLetter( char ch )
{
    char lch = Character.toLowerCase( ch );
    return "qwertyuioplkjhgfdsazxcvbnm".indexOf( lch ) != -1;
}

//-----------------------------------------------------------------------

public boolean isLeftBracket( char c )
{
    switch( c )
    {
        case '(':
        case '[':
        case '{':
            return true;

        default :
            return false;
    }
}

//-----------------------------------------------------------------------

public boolean isRightBracket( char c )
{
    switch( c )
    {
        case ')':
        case ']':
        case '}':
            return true;

        default :
            return false;
    }
}

//-----------------------------------------------------------------------

public char oppositeBracket( char c )
{
    switch( c )
    {
        case '(': return ')';
        case ')': return '(';
        case '[': return ']';
        case ']': return '[';
        case '{': return '}';
        case '}': return '{';
        default : throw new IllegalArgumentException();
    }
}

//-----------------------------------------------------------------------

public boolean isSpace( char c )
{
    return Character.isWhitespace( c );
```

```
}

//-----------------------------------------------------------------------

public int lookupLeftBracketPos( int rightBracketPos )
{
    int result = -1;

    char rightBracket = input.charAt( rightBracketPos );
    char leftBracket  = oppositeBracket( rightBracket );

    int pos = rightBracketPos - 1;
    boolean found = false;

    while( pos >= 0 && !found )
    {
        char currentChar = input.charAt( pos );

        if( currentChar == leftBracket )
        {
            result = pos;
            found = true;
        }
        else
        if( isRightBracket(currentChar) )
        {
            pos = lookupLeftBracketPos( pos );
        }

        pos--;
    }

    return result;
}

//-----------------------------------------------------------------------

public boolean isMathOperator(String block)
{
    if( block.length() != 1 )
    {
        return false;
    }

    switch( block.charAt(0) )
    {
        case '+':
        case '-':
        case '*':
        case '/':
        case '^':
            return true;
        default :
            return false;
    }
}

//-----------------------------------------------------------------------

private String readBreacketsBlock( char leftBracket )
        throws IncorrectSyntaxException
```

```
{
    assert isLeftBracket( leftBracket );
    char rightBracket = oppositeBracket( leftBracket );

    int startIndex = getCurrentPos();

    do
    {
        nextChar();
        if( isLeftBracket(currentChar()) )
        {
            readBreacketsBlock( currentChar() );
        }
    }
    while( currentChar() != rightBracket );

    nextChar();
    int endIndex = getCurrentPos();

    return input.substring( startIndex, endIndex );
}

//-------------------------------------------------------------------------

private void nextChar()
{
    currentPos++;
}

private void previousChar()
{
    currentPos--;
}

//-------------------------------------------------------------------------

private String readChar() throws IncorrectSyntaxException
{
    char returnedChar = currentChar();
    nextChar();
    return String.valueOf( returnedChar );
}

private int lookupBeginLaTeXCommand( int endIndex )
{
    int pos = endIndex + 1;
    int result = -1;

    while( --pos >= 0 && result == -1 )
    {
        char ch = input.charAt( pos );
        if( isLatinLetter(ch) == false )
        {
            if( ch == '\\' )
            {
                result = pos;
            }
        }
    }

    return result;
```

```
    }

    //-------------------------------------------------------------------------

    public boolean endOfInput()
    {
        return !(currentPos < length);
    }

    //-------------------------------------------------------------------------

    public static class Limits
    {
        public final String upper;
        public final String lower;

        public Limits( String lower, String upper )
        {
            this.lower = lower;
            this.upper = upper;
        }
    }
}
```

LaTeXInputDialog.java

```
package v3;

import java.awt.BorderLayout;
import java.awt.Container;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JEditorPane;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.text.JTextComponent;

/**
 *
 * @author Dmitriy Krylov
 */
public
class     LaTeXInputDialog
extends   JFrame
{
    private final String txtInsert = "Вставить";
    private final String txtCancel = "Отменить";
    private final SpecialCaseConverter converter;

    private final JTextComponent target;
```

```
private final JEditorPane editorPane;

private StringBuilder text = null;
private int currPos   = -1;
private int startMath = -1;
private int finalMath = -1;

//---------------------------------------------------------------------------

public LaTeXInputDialog( JTextComponent target )
{
    this.setSize( 600, 400 );
    this.setDefaultCloseOperation( JDialog.DO_NOTHING_ON_CLOSE );

    this.target = target;

    editorPane = new JEditorPane();
    editorPane.setFont( new Font(null, Font.PLAIN, 18) );

    Container contentPane = this.getContentPane();
    contentPane.setLayout( new BorderLayout() );

    JScrollPane scrollPane = new JScrollPane();
    scrollPane.setViewportView( editorPane );

    JPanel buttonPane = new JPanel();
    buttonPane.setLayout( new BoxLayout( buttonPane, BoxLayout.X_AXIS) );

    JButton bInsert = new JButton( txtInsert );
    JButton bCancel = new JButton( txtCancel );

    buttonPane.add( Box.createHorizontalGlue() );
    buttonPane.add( bInsert );
    buttonPane.add( Box.createHorizontalStrut(5) );
    buttonPane.add( bCancel );
    buttonPane.add( Box.createHorizontalGlue() );

    contentPane.add( scrollPane, BorderLayout.CENTER );
    contentPane.add( buttonPane, BorderLayout.SOUTH  );

    bInsert.addActionListener( new ActionListener() {
        @Override public void actionPerformed(ActionEvent e) {
            insert();
        }
    });

    bCancel.addActionListener( new ActionListener() {
        @Override public void actionPerformed(ActionEvent e) {
            cansel();
        }
    });

    converter = new SpecialCaseConverter(
            LaTeXToWolfMathConverter.getInstance() );
}

//---------------------------------------------------------------------------

private void insert()
{
    String sText = editorPane.getText();
```

```
    if( sText.isEmpty() )
    {
        close();
        return;
    }

    try
    {
        text = new StringBuilder( sText );
        currPos = 0;

        while( readMath() )
        {
            String math = text.substring( startMath, finalMath );
            String convertedMath = converter.convert( math );

            text.replace( startMath, finalMath, convertedMath );
        }

        // ----

        StringBuilder targetText = new StringBuilder( target.getText() );
        targetText.insert( target.getCaretPosition(), text );

        target.setText( targetText.toString() );
        close();
    }
    catch( IncorrectSyntaxException ex )
    {
        JOptionPane.showMessageDialog( this, ex.getMessage() );
    }
}

//-----------------------------------------------------------------------

private void cansel()
{
    close();
}

//-----------------------------------------------------------------------

private int findStartMath()
{
    int result = -1;

    Pattern p = Pattern.compile( "(\\$\\$)|(\\$)|(\\\\\\()|(\\\\\\[)" );
    Matcher m = p.matcher( text );

    if( currPos < text.length() && m.find(currPos) )
    {
        result = m.start();
        currPos = m.end();
    }

    return result;
}

//-----------------------------------------------------------------------
```

```
    private int findFinalMath()
    {
        int result = -1;

        Pattern p = Pattern.compile( "(\\$\\$)|(\\$)|(\\\\\\\\)|(\\\\\\])" );
        Matcher m = p.matcher( text );

        if( currPos < text.length() && m.find(currPos) )
        {
            result = m.end();
            currPos = m.end();
        }

        return result;
    }

    //-----------------------------------------------------------------------

    private boolean readMath()
    {
        startMath = findStartMath();
        finalMath = findFinalMath();

        return startMath != -1
            && finalMath != -1;
    }

    //-----------------------------------------------------------------------

    private void close()
    {
        this.setVisible( false );
    }
}
```

DirFinder.java

```
package v3;

import java.io.File;
import java.io.FileFilter;
import java.util.LinkedList;

/**
 *
 * @author Dmitriy Krylov
 */
class DirFinder
{
    public static File findSubDir( File rootDir, String target )
    {
        assert rootDir != null;
        assert target  != null;
        assert rootDir.isDirectory();

        LinkedList <File> queue = new LinkedList<>();
        DirFilter df = new DirFilter();

        for(; rootDir != null; rootDir = queue.poll() )
```

```
        {
            File[] listFiles = rootDir.listFiles( df );

            for( File testingDir : listFiles )
            {
                if( testingDir.getName().equals(target) )
                {
                    return testingDir;
                }

                queue.add( testingDir );
            }
        }

        return null;
    }

    static class DirFilter implements FileFilter {
        @Override public boolean accept(File file) {
            return file.isDirectory();
        }
    };

}
```

MathConnection.java

```
package v3;

import com.wolfram.jlink.Expr;
import com.wolfram.jlink.KernelLink;
import com.wolfram.jlink.MathLinkException;
import com.wolfram.jlink.PacketArrivedEvent;
import com.wolfram.jlink.PacketListener;
import com.wolfram.jlink.PacketPrinter;

/**
 *
 * @author Dmitriy Krylov
 */
public
class       MathConnection
implements  AutoCloseable, PacketListener
{
    private static MathConnection activeConnection = null;

    //-------------------------------------------------------------------------

    public static MathConnection getInstance() throws MathConnectionException
    {
        if( activeConnection == null )
        {
            Log.msg( "create new connection" );
            activeConnection = new MathConnection();
        }
        else
        {
            Log.msg( "return exist connection" );
        }
```

```
    return activeConnection;
}

//-------------------------------------------------------------------------

private String      location    =  null;
private String      errorMsg    =  null;
private String      result      =  null;
private KernelLink  kernelLink  =  null;
private boolean     waitError   =  false;

public String getLocation() { return location; }
public String getError()    { return errorMsg; }

//-------------------------------------------------------------------------

public MathConnection() throws MathConnectionException
{
    Log.msg();

    MathKernelFinder finder = new MathKernelFinder();
    finder.find();

    location    =  finder.getLocation();
    kernelLink  =  finder.getKernel();

    addListeners();
}

//-------------------------------------------------------------------------

public String sendRequest( String request ) throws MathConnectionException
{
    if( request == null )
    {
        throw new IllegalArgumentException( "request must be non null" );
    }

    Log.msg( request );

    result = null;
    errorMsg = null;

    result = kernelLink.evaluateToInputForm( request, 0 );

    if( result == null )
    {
        Throwable lastError = kernelLink.getLastError();

        close();
        throw new MathConnectionException( lastError );
    }

    Log.msg( request+" -> "+result );

    if( result.equals("$Failed") )
    {
        result = null;
    }
```

```
    return result;
}

//-------------------------------------------------------------------------

@Override
public void close()
{
    Log.msg();
    if( kernelLink != null )
    {
        kernelLink.evaluateToInputForm( "CloseFrontEnd[]", 0 );
        kernelLink.close();
        kernelLink = null;
    }
    if( this == activeConnection )
    {
        Log.msg( "this is active connection" );
        activeConnection = null;
    }
}

//-------------------------------------------------------------------------

@Override
public
boolean packetArrived(PacketArrivedEvent e) throws MathLinkException
{
    Expr expr = null;

    try
    {
        expr = ((KernelLink) e.getSource()).getExpr();
        String incomingMessage = expr.toString();

        switch( e.getPktType() )
        {
            case KernelLink.MESSAGEPKT:
            {
                waitError = true;
            }
            break;

            case KernelLink.TEXTPKT:
            {
                if( waitError )
                {
                    waitError = false;
                    if( errorMsg == null )
                    {
                        errorMsg = incomingMessage;
                    }
                    else
                    {
                        errorMsg += "\n" + incomingMessage;
                    }
                }
            }
            break;

            default:
```

```
                    break;
                }
            }
            catch( MathLinkException ex )
            {
                Log.msg( ex.getMessage() );
                throw ex;
            }
            finally
            {
                expr.dispose();
            }

            return true;
        }

        //-------------------------------------------------------------------------

        private void addListeners()
        {
            kernelLink.addPacketListener( this );
            kernelLink.addPacketListener( new PacketPrinter() );
        }

        //-------------------------------------------------------------------------

        public static class MathConnectionException extends Exception
        {
            public MathConnectionException( String  msg ) { super( msg ); }
            public MathConnectionException( Throwable e ) { super( e );   }
        }

        //-------------------------------------------------------------------------

        public static void main(String[] args)
        {
            try{
            try( MathConnection mc = new MathConnection() )
            {
                mc.sendRequest( "ToExpression[\"\\\\sin(2)+2\",TeXForm]" );
                Thread.sleep( 7000);
                mc.sendRequest( "ToExpression[\"\\\\sin(2)+2\",TeXForm]" );
            }
            }catch( MathConnectionException | InterruptedException e )
            {
                Log.msg( e.getMessage() );
            }
        }
}
```

MathKernelFinder.java

```
package v3;


import ca.beq.util.win32.registry.RegistryKey;
import ca.beq.util.win32.registry.RegistryValue;
import ca.beq.util.win32.registry.RootKey;
import com.wolfram.jlink.KernelLink;
```

```
import com.wolfram.jlink.MathLinkException;
import com.wolfram.jlink.MathLinkFactory;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import v3.MathConnection.MathConnectionException;


/**
 *
 * @author Dmitriy Krylov
 */
public class MathKernelFinder
{
    private static final long    timeout      =  10000;
    private static final String  kernelName   = "mathkernel.exe";
    private static final String  libProperty  = "com.wolfram.jlink.libdir";

    private String      kernelLocation  =  null;
    private KernelLink  kernelLink      =  null;
    private boolean     isFound         =  false;

    private SettingManager settings = SettingManager.getSettingManager();

    public String       getLocation() {return kernelLocation;}
    public KernelLink   getKernel()   {return kernelLink;    }
    public boolean      isFound()     {return isFound;       }

    //-----------------------------------------------------------------------

    public void find() throws MathConnectionException
    {
        Log.msg( "start search" );

        findUsingIniFile();
        if( isFound ) {return;}

        findUsingRegister2();
        if( isFound ) {return;}

        throw new MathConnectionException( "can't find math module" );
    }

    //-----------------------------------------------------------------------

    private void tryConnect( String location )
    {
        Log.msg( location );

        File kernelFile = new File( location );

        if( kernelFile.canExecute() == false )
        {
            Log.msg( "file does not exist or is not executable." );
            return;
        }
```

```
    if( System.getProperty(libProperty) == null )
    {
        File kernelDir = kernelFile.getParentFile();
        File jlinkdir = DirFinder.findSubDir( kernelDir, "JLink" );

        if( jlinkdir == null )
        {
            Log.msg("can't find libdir");
            return;
        }

        String jlinkdirPath = jlinkdir.getPath();

        Log.msg( "libdir: "+jlinkdirPath );
        System.setProperty( libProperty, jlinkdirPath );
    }

    try
    {
        String[] creatingArgs = {
            "-linkmode", "launch", "-linkname", location
        };

        KernelLink link = MathLinkFactory
                        .createKernelLink( creatingArgs );

        link.connect( timeout );
        link.discardAnswer();

        // connection success.

        isFound = true;
        kernelLink = link;
        kernelLocation = location;

        Log.msg( "connected" );
    }
    catch( MathLinkException e )
    {
        // connection fail.
        Log.msg( e.getMessage() );
    }

}

//-------------------------------------------------------------------------

private void findUsingIniFile()
{
    Log.msg();

    String path = settings
                .getSetting( "Mathematica path", "PathToMath" );

    if( path == null )
    {
        Log.msg( "can't get setting PathToMath" );
    }
    else
    {
        tryConnect( path );
```

```
    }
}

//-------------------------------------------------------------------------

private void findUsingRegister1()
{
    Log.msg();

    try
    {
    String[] cmdLine = { "reg", "query"
                       , "HKLM\\SOFTWARE\\Wolfram Research"
                       , "/s"
                       , "/t", "REG_SZ"
                       };

    Process reg = Runtime.getRuntime().exec( cmdLine );
    int retCode = reg.waitFor();

    if( retCode != 0 )
    {
        Log.msg( "fail in reg process: " + retCode );
        return;
    }

    try
    (
        BufferedReader regOut =
            new BufferedReader(
                new InputStreamReader(
                    reg.getInputStream()))
    ){

        Pattern pathPattern = Pattern.compile( "\\S+?:\\\\(.+\\\\)" );

        for( String line  = regOut.readLine()
                  ; line != null
                  ; line  = regOut.readLine()
        ){

            Matcher m = pathPattern.matcher( line );

            if( m.find() )
            {
                String path = m.group() + kernelName;

                tryConnect( path );

                if( isFound )
                {
                    settings.setSetting( "Mathematica path"
                                       , "PathToMath"
                                       ,  path
                                       );

                    Log.msg( "success: "+path );
                    break; // success.
                }
            }
```

```
            } // for

        } // try-with-resources

        }
        catch( InterruptedException | IOException ex )
        {
            Log.msg( ex.getMessage() );
        }
    }

    //-----------------------------------------------------------------------

    private void findUsingRegister2()
    {
        Pattern p = Pattern.compile( "^([c-zC-Z]+):(\\\\.*)*\\.(.*)$" );

        RegistryKey wolfkye = new RegistryKey( RootKey.HKEY_LOCAL_MACHINE
                                             , "SOFTWARE\\Wolfram Research"
                                             );

        RegistrySelector rs = new RegistrySelector( p );
        ArrayList <RegistryValue> findValueData = rs.findValueData( wolfkye );

        for( RegistryValue v : findValueData )
        {
            String path = v.getData().toString();

            path = path.substring( 0, path.lastIndexOf("\\") + 1 )
                       .concat( kernelName );

            tryConnect( path );

            if( isFound )
            {
                settings.setSetting( "Mathematica path"
                                   , "PathToMath"
                                   ,  path
                                   );

                Log.msg( "success: "+path );
                break; // success.
            }
        }
    }
}
```

MathPresentationBuilder.java

```
package v3;

import javax.swing.BoxLayout;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import org.scilab.forge.jlatexmath.TeXConstants;
import org.scilab.forge.jlatexmath.TeXFormula;
import v3.MathConnection.MathConnectionException;
```

```
/**
 *
 * @author User
 */
public
class  MathPresentationBuilder
{
    private int fontSize = 20;

    private String input = null;

    private final int readMath = 1;
    private final int readText = 2;
    private int readNow = readText;

    private JPanel target = new JPanel();
    private StringBuilder currLine = new StringBuilder();
    private StringBuilder errorMsg = new StringBuilder();

    enum TermType { func, number, brackets, list };

    //-------------------------------------------------------------------------

    public MathPresentationBuilder()
    {
        target.setLayout( new BoxLayout(target, BoxLayout.Y_AXIS) );
    }

    //-------------------------------------------------------------------------

    public JPanel build( String inputText )
        throws MathConnectionException
    {
        this.input = inputText.replaceAll( "\r\n", "\n" )
                              .replaceAll( "\r", "\n" );

        int currentPos = 0;
        while( true )
        {
            Range math = findMath( currentPos );
            if( math == null )
            {
                break;
            }

            if( math.begin > currentPos )
            {
                String text = input.substring( currentPos, math.begin );
                treatText( text );
            }

            String tMath = input.substring( math.begin, math.end );
            treatMath( tMath );

            if( math.end < input.length() )
            {
                if( input.charAt(math.end) == '\n' )
                {
                    newLine();
                }
            }
```

```
            currentPos = math.end;
        }
        if( currentPos < input.length() )
        {
            treatText( input.substring(currentPos) );
        }
        addLine();
        return target;
    }

    //-------------------------------------------------------------------------

    private void treatMath( String math ) throws MathConnectionException
    {
        if( math.contains("#") )
        {
            StringBuilder workMath = new StringBuilder( math );
            int len = math.length();
            int start = math.indexOf( "#" );
            while( start != -1 )
            {
                int curr = start;
                while( curr < len && isLatinLetter(workMath.charAt(curr)) )
                {
                    ++curr;
                }
                workMath.insert( start, '\"' );
                workMath.insert( curr+1, '\"' );

                start = workMath.indexOf( "#", curr+2 );
                len += 2;
            }
            math = workMath.toString();
        }

        WolfMathToLaTeXConverter converter = new WolfMathToLaTeXConverter();
        String latex = converter.convert( math );
        if( latex != null )
        {
            currLine.append( latex );
        }
        else
        {
            if( errorMsg == null )
            {
                errorMsg = new StringBuilder();
            }

            errorMsg.append( "Error in " )
                    .append( math )
                    .append( ":\n" )
                    .append( converter.getErrorMsg() )
                    .append( "\n" );
        }
    }

    //-------------------------------------------------------------------------

    private void treatText( String text )
    {
```

```
    if( text.contains("\n") )
    {
        int count = 0;
        String[] split = text.split("\n");
        for( String s: split )
        {
            treatText(s);
            if( count++ < split.length - 1 )
            {
                newLine();
            }
        }
        if( text.endsWith( "\n") )
        {
            newLine();
        }
        return;
    }

    String latex = "\\text{"+text+"}";
    currLine.append( latex );
}

//-----------------------------------------------------------------------

private boolean isUnary(char ch)
{
    return "-+".indexOf( ch ) != -1;
}

//-----------------------------------------------------------------------

private boolean isSpace(char ch)
{
    return " \t".indexOf( ch ) != -1;
}

//-----------------------------------------------------------------------

private boolean isMathOperator(char ch)
{
    return "*+-/<=>^".indexOf( ch ) != -1;
}

//-----------------------------------------------------------------------

private boolean isDiggit(char ch)
{
    return "0123456789".indexOf( ch ) != -1;
}

//-----------------------------------------------------------------------

private boolean isLatinLetter(char ch)
{
    return "qwertyuioplkjhgfdsazxcvbnmQWERTYUIOPLKJHGFDSAZXCVBNM#"
         . indexOf( ch ) != -1;
}

//-----------------------------------------------------------------------
```

```
private void newLine()
{
    addLine();
    currLine = new StringBuilder();
}

//-----------------------------------------------------------------------

private void addLine()
{
    Log.msg();

    String latex = currLine.toString();

    target.add( new JLabel( new TeXFormula( latex )
                          . new TeXIconBuilder()
                          . setStyle( TeXConstants.STYLE_DISPLAY )
                          . setSize( fontSize )
                          . build()
                          ) );
}

//-----------------------------------------------------------------------

Range findMath( int startPos )
{
    int startFind = startPos;
    while( true )
    {
        int beginMath = findBeginMath( startFind );
        if( beginMath == -1 )
        {
            return null;
        }

        int endMath = readMath( beginMath );
        if( endMath == -1 )
        {
            startFind = beginMath + 1;
            continue;
        }

        return new Range( beginMath, endMath );
    }
}

//-----------------------------------------------------------------------

int findBeginMath( int startPos )
{
    int length = input.length();
    int currentPos = startPos;
    for( ; currentPos < length; ++currentPos )
    {
        char ch = input.charAt( currentPos );
        if( isFirstCharInTerm(ch) )
        {
            return currentPos;
        }
    }
    return -1;
```

```
}

//-----------------------------------------------------------------------

int readMath( int startPos )
{
    int endMath = -1;
    int beginTerm = startPos;
    while( true )
    {
        int endTerm = readTerm( beginTerm );
        if( endTerm == -1 )
        {
            break;
        }

        endMath = endTerm;

        int nextNonSpacePos = skipSpaces( endTerm );
        if( nextNonSpacePos == -1 )
        {
            break;
        }

        char nonSpace = input.charAt( nextNonSpacePos );
        if( ! isMathOperator(nonSpace) )
        {
            break;
        }

        if( nonSpace == '-' )
        {
            if( nextNonSpacePos + 1 < input.length() )
            {
                if( input.charAt(nextNonSpacePos + 1) == '>' )
                {
                    ++nextNonSpacePos;
                }
            }
        }

        beginTerm = nextNonSpacePos + 1;
    }
    return endMath;
}

//-----------------------------------------------------------------------

boolean isFirstCharInTerm( char test )
{
    return isLatinLetter(test)
        || isDiggit(test)
        || isUnary(test)
        || test == '('
        || test == '{';

}

//-----------------------------------------------------------------------

int readTerm( int startPos )
```

```
{
    int nonSpacePos = skipSpaces( startPos );

    TermType type = determineTermType( nonSpacePos );
    if( type == null )
    {
        return -1;
    }

    switch( type )
    {
        case list:     return readListTerm( nonSpacePos );
        case func:     return readFuncTerm( nonSpacePos );
        case number:   return readNumberTerm( nonSpacePos );
        case brackets: return readBracketsTerm( nonSpacePos );

        default:
            String error = "unexpected type of term";
            throw new IllegalStateException( error );
    }
}

//-----------------------------------------------------------------------

TermType determineTermType( int startPos )
{
    char first = input.charAt( startPos );

    if( isLatinLetter(first) )
    {
        return TermType.func;
    }

    if( isDiggit(first) )
    {
        return TermType.number;
    }

    if( first == '(' )
    {
        return TermType.brackets;
    }

    if( first == '{' )
    {
        return TermType.list;
    }

    if( isUnary(first) )
    {
        int newStart = skipSpaces( startPos + 1 );
        if( newStart != -1 )
        {
            return determineTermType( newStart );
        }
    }

    return null;
}

//-----------------------------------------------------------------------
```

```
int readFuncTerm( int startPos )
{
    int length = input.length();
    int endHeader = skipHeader( skipUnary(startPos) );
    if( endHeader == length )
    {
        return length;
    }

    int nonSpacePos = skipSpaces( endHeader );
    if( nonSpacePos == -1 )
    {
        return length;
    }

    char nonSpace = input.charAt( nonSpacePos );
    if( nonSpace == '[' )
    {
        int beginSequence = nonSpacePos + 1;
        int endSequence = readSequenceMath( beginSequence, ']' );
        if( endSequence != -1 )
        {
            return skipSpaces( endSequence ) + 1;
        }
        else
        {
            return -1;
        }
    }

    return endHeader;
}

//-----------------------------------------------------------------------

private int readSequenceMath( int startPos, char terminator )
{
    int beginMath = startPos;
    while( true )
    {
        if( beginMath >= input.length() )
        {
            return -1;
        }

        int endMath = readMath( beginMath );
        if( endMath == -1 )
        {
            return -1;
        }

        int nextNonSpacePos = skipSpaces( endMath );
        if( nextNonSpacePos == -1 )
        {
            return -1;
        }

        char nonSpace = input.charAt( nextNonSpacePos );
        if( nonSpace == terminator )
        {
```

```
            return endMath;
        }
        else
        if( nonSpace == ',' )
        {
            beginMath = nextNonSpacePos + 1;
            continue; //Read next math.
        }
        else
        {
            return -1;
        }
    }
}

//-----------------------------------------------------------------------------

private int readNumberTerm( int startPos )
{
    int length = input.length();
    int currentPos = startPos;
    for( ; currentPos < length; ++currentPos )
    {
        char ch = input.charAt( currentPos );
        if( ! isDiggit(ch) )
        {
            return currentPos;
        }
    }
    return length;
}

//-----------------------------------------------------------------------------

private int readListTerm( int startPos )
{
    int beginMathSequence = skipSpaces( startPos + 1 );
    if( beginMathSequence == -1 )
    {
        return -1;
    }

    int endSequenceMath = readSequenceMath( beginMathSequence, '}' );
    if( endSequenceMath == -1 )
    {
        return -1;
    }

    int endList = skipSpaces( endSequenceMath );
    if( endList == -1 )
    {
        return -1;
    }

    return endList + 1;
}

//-----------------------------------------------------------------------------

private int readBracketsTerm( int startPos )
{
```

```
    int beginMath = skipSpaces( startPos + 1 );
    if( beginMath == -1 )
    {
        return -1;
    }

    int endMath = readMath( beginMath );
    if( endMath == -1 )
    {
        return -1;
    }

    int endList = skipSpaces( endMath );
    if( endList == -1 )
    {
        return -1;
    }

    return endList + 1;
}

//-----------------------------------------------------------------------

private int skipHeader( int startPos )
{
    int length = input.length();
    int currentPos = startPos;
    for( ; currentPos < length; ++currentPos )
    {
        char ch = input.charAt( currentPos );
        if( ! isLatinLetter(ch) )
        {
            return currentPos;
        }
    }
    return length;
}

//-----------------------------------------------------------------------

int skipUnary( int startPos )
{
    return isUnary( input.charAt(startPos) )
        ? skipSpaces( startPos + 1 )
        : startPos;
}

//-----------------------------------------------------------------------

private int skipSpaces( int startPos )
{
    int length = input.length();
    int currentPos = startPos;
    for( ; currentPos < length; ++currentPos )
    {
        char ch = input.charAt( currentPos );
        if( ! isSpace(ch) )
        {
            return currentPos;
        }
    }
```

```
        return -1;
    }

    //-----------------------------------------------------------------------

    class Range
    {
        public final int begin;
        public final int end;
        public Range( int begin, int end )
        {
            this.begin = begin;
            this.end = end;
        }
    }
}
```

MathPresentationPanel.java

```
package v3;

import javax.swing.JScrollPane;
import v3.MathConnection.MathConnectionException;

/**
 *
 * @author Dmitriy Krylov
 */
public
class     MathPresentationPanel
extends   JScrollPane
{
    public void build( String text ) throws MathConnectionException
    {
        setViewportView( new MathPresentationBuilder().build(text) );
        revalidate();
    }
}
```

SettingManager.java

```
package v3;

import java.io.File;
import java.io.IOException;
import org.ini4j.Ini;

/**
 *
 * @author Dmitriy Krylov
 */
public class SettingManager
{
    private String settingFileLocation = "settings.ini";

    //-----------------------------------------------------------------------
```

```
    private static SettingManager settingManager = new SettingManager();

    public static SettingManager getSettingManager()
    {
        return settingManager;
    }

    private SettingManager() {}

    //-----------------------------------------------------------------------

    public String getSetting( String section, String setting )
    {
        Log.msg( "get setting: " + setting + " ("+section+")" );

        try
        {
            Ini ini = new Ini( new File(settingFileLocation) );
            return ini.get( section, setting );
        }
        catch( IOException ex )
        {
            Log.msg( "error: " + ex.getMessage() );
        }

        return null;

    }

    public boolean setSetting( String section, String setting, String value )
    {
        boolean retValue = false;

        Log.msg( "set setting: " + section + " " + setting + " = " + value );
        try
        {
            Ini ini = new Ini( new File(settingFileLocation) );
            ini.put( section, setting, value );
            ini.store();

            retValue = true;
        }
        catch( IOException ex )
        {
            Log.msg( "error: " + ex.getMessage() );
        }

        return retValue;
    }
}
```

RegistrySelector.java

```
package v3;

import ca.beq.util.win32.registry.RegistryException;
import ca.beq.util.win32.registry.RegistryKey;
import ca.beq.util.win32.registry.RegistryValue;
import ca.beq.util.win32.registry.RootKey;
```

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.URL;
import java.util.ArrayDeque;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Queue;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


public class RegistrySelector
{
    private final static String DLL_NAME = "jRegistryKey.dll";
    private final Pattern pattern;

    private Queue <RegistryKey> keys = new ArrayDeque<>();
    private Queue <RegistryValue> queue = new ArrayDeque<>();

    //-------------------------------------------------------------------------

    public RegistrySelector(Pattern p)
    {
        this.pattern = p;
    }

    //-------------------------------------------------------------------------

    public ArrayList <RegistryValue> findValueData( RegistryKey root )
    {
        for( RegistryKey currKey = root
         ;               currKey != null
         ;               currKey = keys.poll()
        ){
            try
            {
                if( currKey.hasSubkeys() )
                {
                    Iterator subkeys = currKey.subkeys();
                    while( subkeys.hasNext() )
                    {
                        keys.add( (RegistryKey) subkeys.next() );
                    }
                }

                if( currKey.hasValues() )
                {
                    Iterator values = currKey.values();
                    while( values.hasNext() )
                    {
                        RegistryValue value = (RegistryValue) values.next();
                        String stringValue = value.getData().toString();
                        Matcher m = pattern.matcher( stringValue );
                        if( m.matches() )
                        {
                            Log.msg( value.getStringValue() );
                            queue.add( value );
                        }
```

```
                }
            }
        }
        catch( RegistryException ex )
        {
            Log.msg( ex.getMessage() );
        }
    } // for

    return new ArrayList <> ( queue );
}

//-----------------------------------------------------------------------

public static void init()
{
    try
    {
        Log.msg();
        URL dll = RegistrySelector.class.getResource( DLL_NAME );

        if( dll == null )
        {
            throw new RuntimeException( "can't find "+DLL_NAME );
        }

        File temp = File.createTempFile( "library", ".dll" );
        temp.deleteOnExit();

        try (
            InputStream  is = dll.openStream();
            OutputStream os = new FileOutputStream( temp );
        ){
            int readBytes = -1;
            byte[] buffer = new byte[1024];

            while( (readBytes = is.read(buffer)) != -1 )
            {
                os.write(buffer, 0, readBytes);
            }
        }
        Log.msg("load");
        RegistryKey.initialize( temp.getAbsolutePath() );

    }
    catch( IOException ex )
    {
        Log.msg( ex.getMessage() );
    }
}

//-----------------------------------------------------------------------

public static void main(String[] args)
{

    String wolf = "SOFTWARE\\Wolfram Research";
    RegistryKey wolfkye = new RegistryKey( RootKey.HKLM, wolf );
    Pattern p = Pattern.compile("^([c-zC-Z]+):(\\\\.*)*\\.(.*\\.exe)$");

    //Pattern p = Pattern.compile(".*");
```

```
        RegistrySelector rs = new RegistrySelector( p );

        rs.findValueData( wolfkye );

    }
}
```

IncorrectSyntaxException.java

```
package v3;

/**
 *
 * @author Dmitriy Krylov
 */
public
class     IncorrectSyntaxException
extends   Exception
{
    public IncorrectSyntaxException(String msg)
    {
        super( msg );
    }

    public IncorrectSyntaxException()
    {
        super();
    }
}
```

Исходные тексты интерфейса под Andriod

MainActivity.java

```
package com.example.test1app;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.StringWriter;
import java.net.HttpCookie;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpUriRequest;
import org.apache.http.client.protocol.ClientContext;
```

```
import org.apache.http.cookie.Cookie;
import org.apache.http.impl.client.BasicCookieStore;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.impl.cookie.BasicClientCookie;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.protocol.BasicHttpContext;
import org.apache.http.protocol.HttpContext;

import android.os.Bundle;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.res.AssetManager;
import android.view.Menu;
import android.view.View;
import android.webkit.WebView;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;

public class MainActivity extends Activity {

    private final String BASE_ADDRESS = "http://edu2.nscience.ru";
    private final String KERNEL_ADDRESS = "http://int.pm.miem.edu.ru/taskpower3_war_exploded";

    private final String PROF_ADDRESS = BASE_ADDRESS + "/prof/";
    private final String TEST_REGISTER = BASE_ADDRESS + "/ajax/professor_auth.php";
    private final String CHECK_ANSWER = KERNEL_ADDRESS + "/compareAnswer.do";
    private final String GET_TASK = KERNEL_ADDRESS + "/getTask.do";

    private final HttpClient httpClient = new DefaultHttpClient();
    private String cookiesLine_ = null;
    private String userLogin_ = null;

    private ListView taskList = null;

    private String problem;
    private String solution;
    private String answer;
    private String problemPage;
    private String SolutionPage;

    //-------------------------------------------------------------------------
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }


    public void clickHandler(View target) {
        switch( target.getId() ) {
```

```
        case R.id.userEnterButton:
            userEnter();
            break;
        case R.id.returnHomeButton:
            setContentView(R.layout.activity_main);
            break;
        case R.id.returnTaskListButton:
            readAndShowListOfTask( userLogin_ );
            break;
        case R.id.answerEnteredButton:

            boolean rigthAnswer = checkAnswer();
            setContentView(R.layout.tast_result);

            TextView text = (TextView) this.findViewById(
                    R.id.solutionResultText );

            if( rigthAnswer ){
                text.setText("Ваш ответ верный");
            } else {
                text.setText("Ваш ответ неверный. Решение:");
                WebView web = (WebView) this.findViewById( R.id.webview );
                web.getSettings().setJavaScriptEnabled( true );
                web.loadDataWithBaseURL("fake", SolutionPage,
                        "text/html", "utf-8", "");
            }

            break;
        default:
            //do nothing.
            break;
        }
}

//-----------------------------------------------------------------------

private boolean checkAnswer() {

    EditText et = (EditText) this.findViewById( R.id.taskAnswerEditText );

    String real_answer = answer;
    String user_answer = "("+et.getText().toString()+")";

    try {
        String result =
        httpRequestPost(
                CHECK_ANSWER,
                "user_answer", user_answer,
                "real_answer", real_answer );
        return result.trim().equals("True");

    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return false;
}

//-----------------------------------------------------------------------

public void userEnter() {
```

```
    EditText et = (EditText) this.findViewById( R.id.userLoginEditText );
    String userLogin = et.getText().toString();

    try {
        boolean userRegistered = testUserRegistered( userLogin );
        if( userRegistered ){
            readAndShowListOfTask( userLogin );
            userLogin_ = userLogin;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

//-----------------------------------------------------------------------

private void readAndShowListOfTask(String userLogin) {

    try {
        cookiesLine_ = "login=" + userLogin;
        String page;
        page = httpRequestPost( PROF_ADDRESS );

        final String[] tasks = makeTaskList( page );
        final String[] taskNames = new String[tasks.length/2];
        final String[] taskRefs = new String[tasks.length/2];

        for( int i=0; i<taskNames.length; ++i ){
            taskNames[i] = tasks[2*i];
            taskRefs[i] = tasks[2*i+1];
        }

        ArrayAdapter<String> listAdapter = new ArrayAdapter<String>(
                this,
                R.layout.simplerow,
                taskNames
        );
        setContentView(R.layout.task_list);
        taskList = (ListView)findViewById(R.id.taskList);
        taskList.setAdapter( listAdapter );
        taskList.setOnItemClickListener(
                new MyOnItemClickListener( taskRefs ));

    } catch (Exception e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
}

//-----------------------------------------------------------------------

private String[] makeTaskList(String page) {

    List<String> taskList = new ArrayList<String>();

    String findPattern = "<a href='/edu/index.php?task=";
    String ref = null;
    String name = null;

    int patLen = findPattern.length();
```

```
    int start = 0;
    int end = 0;

    do{
        start = page.indexOf( findPattern, end );
        if( start != -1 ){
            start += patLen;
            end = page.indexOf( "&", start );
            if( end != -1 ){
                ref = page.substring( start, end );
                start = page.indexOf( '>', end );
                if( start != -1 ){
                    start += 1;
                    end = page.indexOf( "</a>", start );
                    if( end != -1 ){
                        name = page.substring( start, end );
                        taskList.add( name );
                        taskList.add( ref );
                    }
                }
            }
        }
    } while( start != -1 && end != -1 );

    String[] taskListArray = new String[taskList.size()];
    return taskList.toArray( taskListArray );
}

//-----------------------------------------------------------------------

private void requestTask( String taskRef ) throws Exception {

    setContentView(R.layout.task_view);
    WebView web = (WebView) this.findViewById( R.id.webview );
    web.getSettings().setJavaScriptEnabled( true );

    String taskDescription;
    taskDescription = httpRequestGet( GET_TASK, "task", taskRef );

    problem = extractXMLByTag( taskDescription, "problem" );
    solution = extractXMLByTag( taskDescription, "solution" );
    answer = extractXMLByTag( taskDescription, "answer" );

    InputStream is = getAssets().open( "htmlpattern" );
    String htmlPattern = convertStreamToString( is );

    problem = problem.replace("&#8518;", "d")
                     .replace("&#8519;", "e");
    solution = solution.replace("&#8518;", "d")
                       .replace("&#8519;", "e");

    problemPage = htmlPattern.replace( "!!!", problem );
    SolutionPage = htmlPattern.replace( "!!!", solution );
    web.loadDataWithBaseURL("fake", problemPage, "text/html", "utf-8", "");

    //web.loadUrl("http://www.mathjax.org/demos/tex-samples/");
}

//-----------------------------------------------------------------------
```

```
private String extractXMLByTag( String page, String tag ) {
    String patternString = "<" + tag + ".*?>(.*?)</" + tag + ">";

    int flags = Pattern.MULTILINE | Pattern.DOTALL;
    Pattern p = Pattern.compile( patternString, flags );

    Matcher m = p.matcher( page );

    if( m.find() ){
        return m.group( 1 );
    }
    return null;
}


//-------------------------------------------------------------------------

public boolean testUserRegistered( String userLogin )
        throws Exception {

    String result = httpRequestGet( TEST_REGISTER, "login", userLogin );
    return result.trim().equals( "1" );
}

//-------------------------------------------------------------------------

public String httpRequestGet( String request, String ... params )
        throws Exception {

    if( params.length % 2 != 0 ){
        String error = "number of params is not even";
        throw new IllegalArgumentException( error );
    }

    String resultRequest = request;

    if( params.length != 0 ){
        resultRequest += "?";
        for( int i = 0; i < params.length; i+=2 ){
            resultRequest += params[i]+"="+params[i+1];
        }
    }

    HttpGet getRequest = new HttpGet( resultRequest );
    return httpRequest( getRequest );
}

//-------------------------------------------------------------------------

public String httpRequestPost( String request, String ... params )
        throws Exception {

    if( params.length % 2 != 0 ){
        String error = "number of params is not even";
        throw new IllegalArgumentException( error );
    }

    HttpPost postRequest = new HttpPost( request );

    if( params.length != 0 ){
```

```
        final int pairNum = params.length/2;
        List<NameValuePair> postParameters =
                new ArrayList<NameValuePair>( pairNum );

        for( int i = 0; i<params.length; i+=2 ){
            postParameters.add(
                new BasicNameValuePair( params[i], params[i+1] ));
        }

        UrlEncodedFormEntity formEntity =
                new UrlEncodedFormEntity( postParameters );

        postRequest.setEntity(formEntity);
    }

    return httpRequest( postRequest );
}

//-----------------------------------------------------------------------

private String httpRequest( HttpUriRequest request )
        throws Exception {

    if( cookiesLine_ != null ){
        request.setHeader("Cookie", cookiesLine_);
    }
    HttpResponse response = httpClient.execute( request );

    InputStream is = null;
    try {
        is = response.getEntity().getContent();
        return convertStreamToString( is );
    } finally {
        if( is != null ){
            is.close();
        }
    }
}

//-----------------------------------------------------------------------

public static String convertStreamToString( InputStream is )
        throws IOException {

    if( is == null ){
        throw new IllegalArgumentException("arg is null");
    }

    BufferedReader in;
    in = new BufferedReader(new InputStreamReader( is ));

    StringBuffer sb = new StringBuffer();
    String separator = System.getProperty("line.separator");

    String line;
    while( (line = in.readLine()) != null ) {
        sb.append( line + separator );
    }

    String result = sb.toString();
    return result;
```

```
    }

    //=============================================================

    class MyOnItemClickListener
    implements AdapterView.OnItemClickListener {

        final String[] taskRefs;

        public MyOnItemClickListener( String[] taskRefs ) {
            this.taskRefs = taskRefs;
        }

        @Override
        public void onItemClick(
                AdapterView<?> arg0,
                View arg1, int position, long arg3) {

            try {
                requestTask( taskRefs[position] );
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

activityMain.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/userEnterTextFieldTitle"
        android:layout_width="wrap_content"
           android:layout_height="wrap_content"
        android:text="@string/userEnterTextFieldTitle" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <EditText
            android:id="@+id/userLoginEditText"
            android:inputType="text"
            android:layout_weight="1.0"
            android:layout_width="0dp"
            android:layout_height="wrap_content" />

        <Button
            android:id="@+id/userEnterButton"
            android:onClick="clickHandler"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/userEnterButtonText" />

    </LinearLayout>

</LinearLayout>
```

taskList.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/returnHomeButton"
        android:onClick="clickHandler"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/returnHomeButtonText" />

    <ListView
        android:id="@+id/taskList"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

taskView.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/returnTaskListButton"
        android:onClick="clickHandler"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/returnTaskListButtonText" />

    <WebView
        android:id="@+id/webview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
    />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <EditText android:id="@+id/taskAnswerEditText"
```

```
                android:inputType="text"
                android:layout_weight="1.0"
                android:layout_width="0dp"
                android:layout_height="wrap_content" />

        <Button     android:id="@+id/answerEnteredButton"
                android:onClick="clickHandler"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/answerEnteredText" />
    </LinearLayout>

</LinearLayout>
```

tastResult.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

     <Button
         android:id="@+id/returnTaskListButton"
         android:onClick="clickHandler"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="@string/returnTaskListButtonText" />

     <TextView
          android:id="@+id/solutionResultText"
          android:textIsSelectable="true"
          android:layout_width="wrap_content"
         android:layout_height="wrap_content" />

    <WebView
         android:id="@+id/webview"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
    />

</LinearLayout>
```

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">uI</string>
    <string name="action_settings">Settings</string>
    <string name="userEnterTextFieldTitle">Вход в систему, введите логин:</string>
    <string name="userEnterButtonText">Вход</string>
    <string name="returnHomeButtonText">Вернуться</string>
    <string name="returnTaskListButtonText">Вернуться</string>
    <string name="answerEnteredText">Ввести ответ</string>

</resources>
```

Исходные тексты реализованных сервлетов ядра

ConverterServlet.java

```
package servlet;

import configuration.ConfigurationContextHelper;
import math.MathAware;
import math.MathException;
import math.MathModule;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

public class ConverterServlet extends HttpServlet implements MathAware {

    public void doPost( HttpServletRequest req, HttpServletResponse res )
        throws ServletException, IOException
    {
        String result = null;
        res.setCharacterEncoding("UTF8");
        PrintWriter out = res.getWriter();

        math = ConfigurationContextHelper
              .getConfigurationContext()
              .getBean(MathModule.class);

        String inputFormat = req.getParameter( "inputFormat" );
        String outputFormat = req.getParameter( "outputFormat" );

        String treated_text = req.getParameter("text");

        if( inputFormat == null
         || outputFormat == null
         || treated_text == null )
        {
            result = "error:";
            if( inputFormat == null )
                result += "   inputFormat is not specified";
            if( outputFormat == null )
                result += ", outputFormat is not specified";
            if( treated_text == null )
                result += ", treated_text is not specified";
        }
        else
        {
            if( inputFormat.equals("LaTeX")
             && outputFormat.equals("WolfMath") )
            {
                try
                {
                    String request = "ToExpression[\""
                                   + treated_text
                                   + "\",TeXForm,Hold]";

                    result = math.Evaluate( request );
```

```
                result = result.substring( 5, result.length() - 1 );
            }
            catch( MathException e )
            {
                result = "error: " + e.getMessage();
                e.printStackTrace();
            }
        }
        else
        if( inputFormat.equals("WolfMath")
         && outputFormat.equals("LaTeX") )
        {
            try
            {
                String request = "TeXForm[HoldForm["+treated_text+"]]";
                result = math.Evaluate( request );
            }
            catch (Exception e)
            {
                result = "error: " + e.getMessage();
                e.printStackTrace();
            }
        }
        else
        {
            result = "error: unsupported conversion";
        }
    }

    out.println( result );

    out.close();

}

public void setMath(MathModule math) {
    this.math = math;
}

protected MathModule math;
}
```

CheckGrammarServlet.java

Данный сервлет был модифицирован. В данной дипломной работе в него была добавлена возможность возвращать сообщение с описанием возникшей ошибки.

```
package servlet;

import java.lang.Exception;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.*;
import grammar.CheckGrammar;
```

```java
/**
 * Author: Zhukov Dmitry
 * Date: 17.05.11
 */
public class CheckGrammarServlet extends HttpServlet {

    public void doPost( HttpServletRequest req, HttpServletResponse res )
        throws ServletException, IOException
    {

        res.setCharacterEncoding( "UTF8" );
        PrintWriter out = res.getWriter();

        String fileContent = req.getParameter( "filecontent" );

        try {

            CheckGrammar checkGrammar = new CheckGrammar();
            Boolean status = checkGrammar.checkValid( fileContent );

            out.println( status.toString() );

            if( status == false )
            {
                out.println( checkGrammar.getErrorMsg() );
            }
        }
        catch (Exception e) {

            out.println("error");
            System.out.println( e.getMessage() );
        }

        out.close();

    }

}
```