САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ ПЕТРА ВЕЛИКОГО

На правах рукописи

Кацман Виктор Игоревич

МОДЕЛИ И МЕТОДЫ АВТОМАТИЧЕСКОЙ ПРОВЕРКИ РЕШЕНИЙ ЗАДАЧ

Специальность: 05.13.11 «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей»

Выпускная квалификационная работа аспиранта

Научный руководитель:
Доктор технических наук, с.н.с.
Новиков Федор Александрович

Санкт-Петербург – 2021

Содержание

ВВЕДЕНИЕ

Решение типовых учебных задач является одной из самых существенных составляющих образовательного процесса в наше время. Для твердого усвоения полученного материала, обучаемым требуется решить большое число учебных задач, что требует проверки большого числа решений. Одним из основных элементов изучения технических наук является решение учащимися больших объемов задач на практику навыков работы с формулами.

При решении существенной части таких задач учащимся требуется построить или вывести неизвестную им последовательность преобразований на основе уже известных им правил преобразования. Значительная часть подобных решений описывается:

1. Цепочками преобразований над выражениями – описывают последовательности преобразований над символьными выражениями, которые удовлетворяют отношениям порядка, которые налагают, в том число, необходимость равенства или последовательного убывания всех элементов цепочки. Такая цепочка преобразований может быть решением задачи на доказательство равенства или неравенства символьных выражений.
2. Цепочками преобразований над логическими утверждениями над символьными выражениями. Логическое утверждение над выражениями соответствует их удовлетворению логическому условию, а точнее, неравенству либо равенству символьных выражений, а также совокупности или системе других логических утверждений.

Проверка правильности совершения учащимся цепочек преобразования требует проверки корректности и тривиальности (не слишком большой сложности для студента) всех преобразований в цепочке. Нынешний уровень развития технологий дает возможность подойти к автоматической проверке правильности таких цепочек преобразования на принципиально новом уровне.

Совмещение приведенных видов цепочек преобразования дает возможность записать решения значительного числа учебных задач. Автоматическая проверка правильности всех переходов в решениях этих задач дает возможность значительно увеличить производительность преподавательского труда без потери качества. Таким образом, она дает возможность проверять существенно большие числа задач, а это дает возможность увеличить объемы решаемых учащимися задач и, тем самым, повысить уровень образовательного процесса в целом.

Кроме повышения числа проверяемых задач, автоматизация проверки дает возможность существенно улучшить коммуникацию, то есть позволит масштабировать процесс в том числе в местах, где качественное образование получить существенно сложнее. Кроме того, результаты проверки могут стать доступны сразу после вывода учащимся решения: ожидание преподавателя не потребуется. Также проверка корректности и тривиальности всех переходов даст возможность обучаемому сразу узнать о недостаточно поясненном или неправильном переходе и дорабатавать решение сразу, без потери контекста. Таким образом, и обучаемый может тратить свое время более продуктивно.

Вместе описанные возможности автоматизации проверки решений типовых учебных задач открывают простор для значительного улучшения образовательного процесса, а также увеличивают доступность образования на высоком уровне благодаря возможности масштабирования. Рост качества образования позитивно влияет на рост прогресса во многих областях. Это актуально не только для России, но и для многих других стран.

1. Процесс решения типовых учебных задач

1.1 Задачи на оперирование формулами

Существенную долю учебных задач в области технических наук составляют ***задачи на закрепление навыков оперирования формулами***. В таких задачах от учащихся требуется совершить определенные действия над ***символьными выражениями*** или ***логическими утверждениями*** – комбинациями условий над символьными выражениями.

В данной работе рассматриваются исключительно те ***символьные выражения***, которые задаются следующей порождающей грамматикой:

1. Терминальные символы:
   - 1.1. Имена функций: {‘exp’, ‘ln’, ‘sin’, ‘cos’, ‘tg’, ‘ctg’, ‘sh’, ‘ch’, ‘th’, ‘cth’, ‘asin’, ‘acos’, ‘atg’, ‘actg’, ‘abs’, ‘log’, ‘mod’, ‘P’, ‘F’, ‘C’, ‘U’, ‘A’, ‘V’, ‘S1’, ‘S2’,’ B’},
   - 1.2. Цифры: {0-9},
   - 1.3. Буквы: {‘a’-’z’, ‘A’-’Z’},
   - 1.4. Левые унарные операции: {‘-’, ‘¬’},
   - 1.5. Правые бинарные операции: {‘!’, ‘`’},
   - 1.6. Бинарные операции: {‘+’, ‘-‘, ‘*’, ‘/‘, ‘^‘, ‘∧’, ‘∨’, ‘⊕’ , ‘≡’ , ‘→’ , ‘\’ },
   - 1.7. Итераторные функции:
     - 1.7.1. Сумма символьных выражений: {‘∑’},
     - 1.7.2. Произведение символьных выражений: {‘∏’},
   - 1.8. Зарезервированные имена: {‘π’, ‘e’},
   - 1.9. Скобки: {‘(’, ‘)’},
   - 1.10. Разделители: {‘,’, ‘.’}.

2. Нетерминальные символы:
   - 2.1. СВ - символьное выражение,
   - 2.2. ИФ - имя функции,
   - 2.3. БК - буква,
   - 2.4. Ц - цифра,
   - 2.5. ЛУО - левая унарная операция,
   - 2.6. ПУО - правая унарная операция,
   - 2.7. БО - бинарная операция,
   - 2.8. ИТФ - имя итераторной функции,
   - 2.9. ЗИ - зарезервированные имена,
   - 2.10. П - переменная,
   - 2.11. ИмП - имя переменной,
   - 2.12. Чис - число,

2.13. СЧ– счетчик суммы или произведения комплексных символьных выражений, шаг счетчика: единица,
2.14. МинЗС – минимальное значение счетчика,
2.15. МаксЗС – максимальное значение счетчика,
2.16. СА - список аргументов,
2.17. ЧЧ - часть числа.

3. Правила:
3.1. ИФ → exp | ln | sin | cos | tg | ctg | sh | ch | th | cth | asin | acos | atg | actg | abs | log | compconj | P | F | C | U | A | V | S1 | S2 | B
3.2. БК → a-zA-Z
3.3. Ц → 0-9
3.4. БО → + | - | * | / | mod | ∧ | ∨ | ⊕
3.5. ЛУО → - | ¬
3.6. ПУО → ! | `
3.7. ИТФ → ∑ | ∏
3.8. ЗИ → π | e
3.9. ИмП → БК | ИмП БК
3.10. П → ИмП | ЗИ
3.11. ЧЧ → Ц | ЧЧ Ц
3.12. Чис → ЧЧ | ЧЧ.ЧЧ
3.13. СА → СВ | СА, СВ
3.14. СВ → СВ БО СВ | ЛУО СВ | СВ ПУО | (СВ) | ИФ (СА) | П | Чис | ИТФ (СЧ , МинЗС , МаксЗС , СВ)
3.15. МинЗС → СВ
3.16. МаксЗС → СВ
3.17. СЧ → СВ

4. Начальные символы:
4.1. СВ

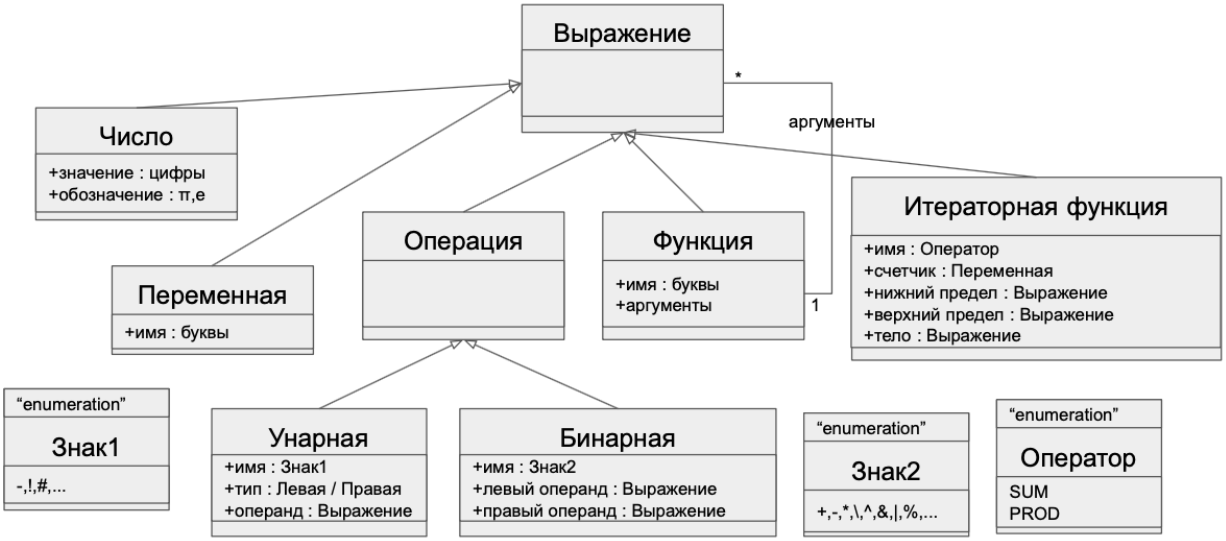

Рис. 1. Модель символьных выражений

Ниже примеры символьных выражений, удовлетворяющих и неудовлетворяющих приведенной выше грамматике:

1. $\frac{1}{n!}\sum_{i=1}^{k}(-1)^{k-i} * C(k,i) * i^n -$ удовлетворяет; здесь C – число сочетаний: $C(m,n) = \frac{m!}{(m-n)!n!}$
2. 0.(3) – бесконечная периодическая дробь – неудовлетворяет. Удовлетворяет в записи $1/3$

Логическое утверждение (logical proposition) [71] – утверждение над символьными выражениями, может быть истинно или ложно. В данной работе рассматриваются логические утверждения, которые задают либо равенство символьных выражений, либо отношение порядка над символьными выражениями, либо логическую конъюнкцию других рассматриваемых утверждений, либо конъюнкцию.

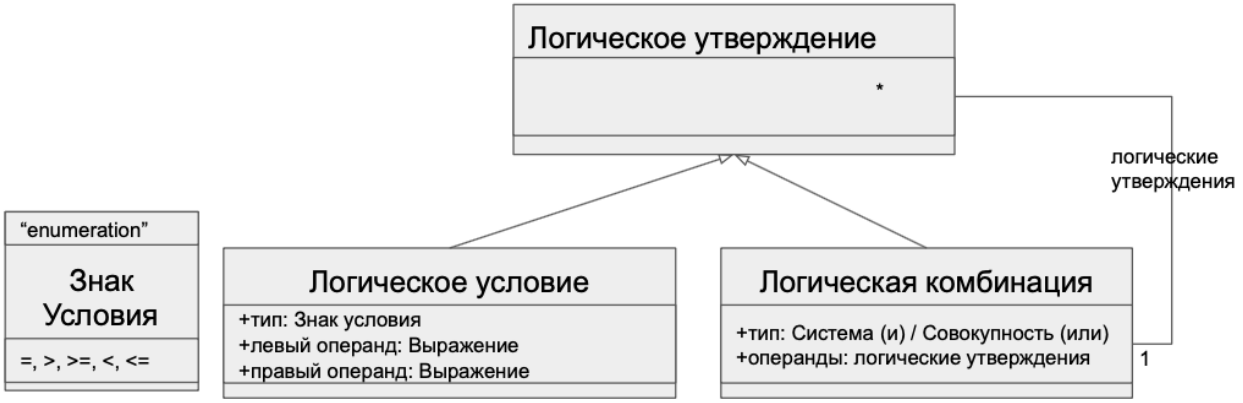

Рис. 2. Модель логических утверждений над символьными выражениями

Примеры логических утверждений, рассматриваемых в данной работе:

1. $\sin(x) < \cos(x) + 1$

2. $\begin{cases} x + y = 4 \\ \left[\begin{array}{l} x * y = 6 \\ x * y = 9 \end{array}\right. \end{cases}$

Точно верные в контексте рассматриваемой задачи логические утверждения являются ***фактами***. А именно:

1. Данные исходно логические утверждения, обозначенные в условии этой задачи как истинные.
2. Логические утверждения, доказанные по ходу решения этой задачи.

Задача на оперирование формулами имеет ввиду наличие ***множества символьных выражений и/или логических утверждений E***.

Множество правил преобразования – множество отображений множества ***E*** в себя же, элементы которого соответствуют правилам преобразованиям, которые учащиеся могут применять при решении данной им задачи.

Применение правила преобразования к элементу множества ***E*** – замена фрагментов этого элемента на другие элементы множества ***E***.

Примеры возможных правил преобразования:

1. $a^2 - b^2 \iff (a - b) * (a + b)$
2. $a + c = \mathrm{b} + \mathrm{c} \iff \mathrm{a} = \mathrm{b}$

Примеры применения правила преобразования:

1. Применение правила $a^2 - b^2 \iff (a - b) * (a + b)$ к выражению $x + ((a^3)^2 - (5 - y)^2)$ дает выражение с примененным правилом $x + ((a^3) - (5 - y))((a^3) + (5 - y))$.
3. Применение правила $a + c = \mathrm{b} + \mathrm{c} \iff \mathrm{a} = \mathrm{b}$ к логическому утверждению $\sin(x) + a = \cos(x) + a$ дает логическое утверждение с примененным правилом $\sin(x) = \cos(x)$.

Итоговое решение учащегося представляется в виде ***цепочки преобразований над символьными выражениями и логическими утверждениями***. Задаются порождающей грамматикой:

1. Терминальные символы:
   1.1. Знаки: {‘<’, ‘>’, ‘≤’, ‘≥’, ‘=’, ‘=>’}
   1.2. Перенос строки: {‘\n’}
   1.3. Символьное выражение
   1.4. Скобки: {‘{’, ‘[’, }’, ‘]’} - фигурные и квадратные скобки, трактуемые, соответственно, как система или совокупность. Скобка_большая_фигурная_системная, Скобка_большая_квадратная_системная

2. Нетерминальные символы:
   2.1. Знак - знаки, связывающие выражения
   2.2. Знак_следования - знак логического перехода
   2.3. Перенос_строки
   2.4. Логическое_утверждение
   2.5. Расширенное_утверждение
   2.6. Перечень_утверждений
   2.7. Система_логических_утверждений
   2.8. Совокупность_логических_утверждений
   2.9. Правило
   2.10. Правило_преобразования_выражений
   2.11. Цепочка_преобразований_выражений
   2.12. Цепочка_преобразования_логических_утверждений

3. Правила:
   3.1. Знак → < | > | ≤ | ≥ | =
   3.2. Знак_следования → =>
   3.3. Перенос_строки → \n
   3.4. Логическое_утверждение → Выражение Знак Выражение | Система_логических_утверждений | Совокупность_логических_утверждений
   3.5. Расширенное_утверждение → Логическое_утверждение | Цепочка_преобразований_выражений | Цепочка_преобразования_логических_утверждений
   3.6. Перечень_утверждений → Расширенное_утверждение | Расширенное_утверждение Перенос_строки Перечень_утверждений
   3.7. Система_логических_утверждений → Скобка_большая_фигурная_системная Перечень_утверждений
   3.8. Совокупность_логических_утверждений → Скобка_большая_квадратная_системная Перечень_утверждений
   3.9. Правило_преобразования_выражений → [Цепочка_преобразований_выражений]
   3.10. Правило → Правило_преобразования_выражений | [Цепочка_преобразования_логических_утверждений]
   3.11. Цепочка_преобразований_выражений → Выражение | Выражение Знак Цепочка_преобразований_выражений | Цепочка_преобразований_выражений Знак Правило_преобразования_выражений Знак Цепочка_преобразований_выражений
   3.12. Цепочка_преобразования_логических_утверждений → Логическое_утверждение | Логическое_утверждение Знак_следования Цепочка_преобразования_логических_утверждений |

Логическое_утверждение Перенос_строки Цепочка_преобразования_логических_утверждений | Цепочка_преобразования_логических_утверждений Знак_следования Правило Знак_следования Цепочка_преобразования_логических_утверждений

4. Начальные символы:
   4.1. Цепочка_преобразования_логических_утверждений
   4.2. Цепочка_преобразования_выражений

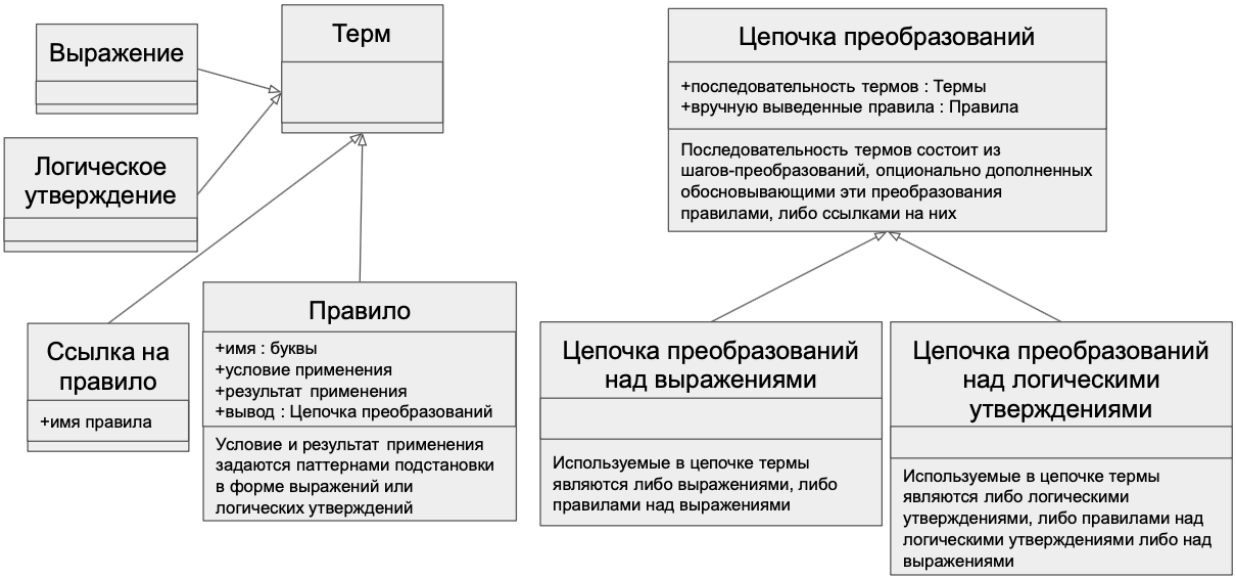

Рис. 3. Модель цепочек преобразования

Рассматриваемые в работе примеры цепочек преобразований:

$$4*\cos(\frac{\pi}{6}-x)*\sin(\frac{\pi}{3}-x)=2*(\sin(\frac{\pi}{2}-2*x)+\sin(\frac{\pi}{6}))=2*(\sin(\frac{\pi}{2}-2*x)+0.5)=$$
$$=\frac{2*\sin(x)(\cos(2*x)+0.5)}{\sin(x)}=\frac{2*\sin(x)\cos(2*x)+\sin(x)}{\sin(x)}=\frac{(\sin(3*x)-\sin(x))+\sin(x)}{\sin(x)}=\frac{\sin(3*x)}{\sin(x)}$$

Рис. 4. Цепочка преобразований «доказать тригонометрическое тождество»

$$C(x+1,x)=k=>$$
$$[C(x+1,x)=(x+1)!/x!/((x+1)-x)!=(x+1)!/x!/1!=(x+1)!/x!/1=x!*(x+1)/x!/1=x+1]$$
$$x+1=k=>$$
$$(x+1)-1=k-1=>$$
$$x=k-1$$

Рис. 4. Пример цепочки преобразований «решить уравнение»

Решение задачи начинается с интерпретирования учащимся ее условия. Далее из доступных ему ресурсов, в том числе собственной памяти, учащийся начинает извлекать информацию, применимую к этой задаче, выявляет по ней правила преобразования, позволяющие преобразовывать символьные выражения и логические утверждения, имеющиеся в задаче.

Далее учащимся строится план решения задачи: по каким правилам и в каком порядке данные ему выражения требуется преобразовывать. После трансформирует данные выражения и факты по последовательно выбираемым правилам преобразования из те, которые известны ему заранее или тех, которые выводятся им в ходе решения задачи, исходя из уже известных. Такие итерации производятся обучающимся до момента, в котором он либо бросит решать решаемую задачу, либо получит правильное на его взгляд, решение этой задачи. не получит удовлетворяющее его решение задачи, либо не бросит ее решать.

Учащийся записывает полученное решение в виде цепочки преобразований, затем она передается учителю для проверки правильности. В некоторых случаях такое решение дополняется пояснениями, объясняющими возможность использования преобразования в текущей ситуации.

Затем выведенная учащимся цепочка преобразований и пояснения передаются учителю для оценки решения задачи. И при выявлении степени правильности преподаватель использует исключительно условие задачи и переданный учащимся материал.

В ходе оценки правильности учитель проверяет:

1. Соответствие начального и конечного элементов цепочки преобразований исходно данному условию задачи.
2. Правильность переходов во всех парах соседних элементов в предоставленной обучаемым цепочке.
   Если присутствуют вложенные внутренние цепочки преобразований, которые выводят правила, то учителю требуется проверить все переходы в этой вложенной внутренней цепочке, а также переходы, которые обосновываются выведенными в этой цепочке внутренними правилами.

Переходы (их правильность) проверяются по следующим критериям:

1. ***Корректность*** – насколько выполненные учащимся преобразования соответствуют теории. Корректный переход возможен по тем аксиомам и теоремам, которые присуществуют в рассматриваемой области.
   1.3.Примеры:
   1.3.1. Преобразование $x \vee (\mathrm{x} \wedge y) = x$ корректное.
   1.3.2. Преобразование $x \vee (\mathrm{x} \wedge y) = \mathrm{x} \wedge y$ некорректное.

2. ***Тривиальность*** – достаточно ли прост переход, чтобы для его совершения учащемуся хватило умственных усилий. Так, переход будет считаться тривиальным, когда он не чересчур сложен для учащегося при его познаниях на рассматриваемый момент. Задача оценки тривиальности перехода – уменьшить вероятность случайного пропуска учащимся важных шагов в ходе решения рассматриваемой задачи и вероятность непонимания этих шагов. Чтобы превратить нетривиальный переход в проходящий проверку на тривиальность, учащемуся нужно вывести объясняющее его правило, либо разбить его на более простые.

   2.3.Примеры:

   2.3.1. Преобразование $x \vee (x \wedge y) = x$ нетривиальное и не может быть совершено за один переход.

   2.3.2. Преобразование $x \vee (x \wedge y) = (x \wedge 1) \vee (x \wedge y)$ тривиальное и может быть совершено за один переход.

Когда все переходы в рассматриваемой цепочке преобразований одновременно и корректны, и тривиальны, результатом является выведенное либо требуемое в задаче тождество, либо требуемое в задаче символьное выражение, либо логическое утверждение, которое удовлетворяет критериям из условия задачи – решение, которое представленно рассматриваемой цепочкой преобразований, размечается как правильное и для учащегося засчитывается. В ином случае решение полагается неправильным, учитель указывает учащемуся точку, с которой рассуждения превратились либо в неправильные, либо также в недостаточно обоснованные.

Примеры:

1. Решение $x \vee (x \wedge y) = (x \wedge 1) \vee (x \wedge y) = x \wedge (1 \vee y) = x \wedge 1 = x$ правильное для задачи «Докажите, что $x \vee (x \wedge y) = x$», так как все переходы одновременно и корректны и тривиальны.

1.2 Альтернативные возможные подходы к записи и нотации решений

Создание языка, позволяющего записывать решения типовых учебных задач – значимый элемент автоматической проверки решений таких задач. Степень применимости языка предлагается оценивать по таким критериям:

1. Возможность однозначно интерпретировать записанные решения.
2. Удобство процесса записи.
3. Сложность в усвоении.

Примечание: удовлетворение языка 2-м и 3-му критериям может проверяться исключительно экспериментально.

Обозначенная в 1.1 система записи решений учебных задач не является безальтернативной. Есть много отличных подходов, принятых в других образовательных платформах и системах других стран, многие из которых широко используются сейчас.

Существующие нотации, позволяющие записывать математические преобразования в цифровом виде разделяются на 2 типа по признаку совпадения форматов ввода и получения введенных преобразований. Так, TeX [86] и MathML [92] потребуют ввода математических преобразований на специальном определенном языке, существенно отличающемся по сравнению с привычным для обучающихся языка решений типовых учебных задач «на бумаге». Далее они компилируют преобразования, записанные на этом языке в формат, более привычный для восприятия. Овладение подобным языком потребует отдельной подготовки, не связанной с изучением предмета и сложной для большой доли учащихся – плох подходит для систем автоматической проверки.

Наоборот, другие нотации позволяют записывать математические преобразования сразу в выходном итоговом формате, привычному для среднестатистических учащихся. В эти нотации относятся:

1. Запись в одну строку, с применение определенных обозначений при обозначении частей математических преобразований, которые принято обозначать путем разделения на множество строк. Минусом подобных подходов к записи является неудобство их восприятия. Так, языки программирования (Python, Java, C++, Kotlin, …); языки математических сред (R, Mathematica, …) и другие.

```
double d = 4 * y * y * (x * x + y * y - X * X);
auto q = 2 * (x * x + y * y);
auto alpha1 = d < 0 ? 0 : acos((2 * X * x + sqrt(d)) / q);
auto alpha2 = d < 0 ? 0 : acos((2 * X * x - sqrt(d)) / q);
```

Рис. 5. Пример записи математических выкладок на языке C++

2. Ввод математических преобразований в формате, похожем на записи «на бумаге», благодаря предлагаемому набору паттернов связей и расположений символов, которые соответствуют общераспространенному расположению частей выражений для описываемых ими примитивов. Такие нотации включают наличие средства ввода, позволяющего выбирать правила и паттерны. Так, средства ввода выражений, основанные на этом принципе, предоставляет Wiris [132] и MathQuill [93].

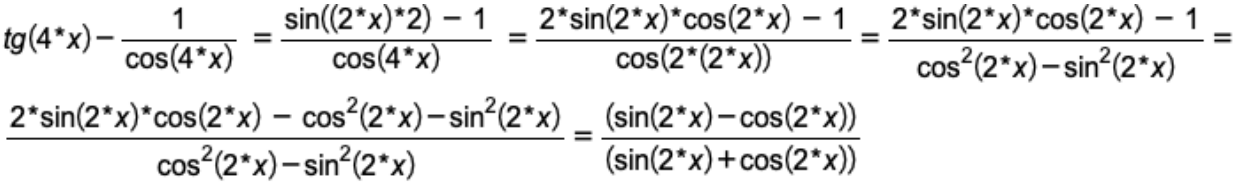

Рис. 6. Пример записи математических выкладок в Wiris

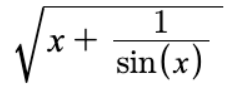

Рис. 7. Примеры записи математических выкладок в MathQuill

Приведенные нотации определяют способы введения математических преобразований разных видов, при этом могут не подразумевать автоматическую интерпретацию этих выкладок. Она требует специального интерпретируемого языка для ввода цепочек преобразований над символьными выражениями и логическими утверждениями.

Разработка новой хорошей нотации – трудоемкий процесс, требующий разработки специальных комплексов, реализующих эту нотацию, в дополнение к распространению этой нотации в нынешних условиях существования уже широко зарекомендовавших себя других нотаций. Они бывают разных видов, кроме того, занимаются многие научные организации.

Поэтому в данной работе было принято решение разрабатывать язык для ввода цепочек преобразований на основе других нотаций, уже существующих и предполагающих возможность записи математических преобразований сразу в выходном формате. Одновременно была обозначена задача сделать нотацию, способную понимать выкладки одновременно поверх более одной распространенных существующих нотаций, с целью упрощения.

В настоящее время существует много таких языков записи символьных выражений. Они дают возможность ввести символьные выражения, затем скомпилировать их в исполняемый код, либо в структуры, которые предназначены для последующей трансформации. Одновременно, такие языки дают возможность вычислять значение символьных выражений для заданных значений переменных.

Основные различия между форматами записи наблюдаются в:

1. Записи степени. В части языков она пишется как «^» (C++), в отличных языках как «**» (Python). В части языков в тригонометрических фукнциях позволяется запись степени сразу за именем тригонометрической функции (Wiris).
2. Записи итераторных функций. В большом числе языков такие функции могут быть описаны только процедурой вычисления или через определение операции, вслед за которой следуют аргументы в скобках: имя счетчика, затем нижний предел, затем верхний предел, затем выражение, с которым

осуществляется преобразование (Python). В других специализированных системах есть опция записывать такие функции, записывая счетчик и нижний предел под именем операции, а затем верхний предел – над именем операции (Wiris).

```
sum(range(n ** 2+1))
def prod(l, u, e):
    result = 1
    for i in range(l, u+1):
        result *= e(i)
    return result

print(prod(1, n, (lambda i: i)))
```

Рис. 8. Примеры записи итераторной функции в Python

$$\sum_{i=1}^{n} i^2 \qquad \prod_{i=1}^{n} i = n!$$

Рис. 9. Примеры записи итераторных функций в Wiris

3. Именах функций. Например, «tg» или «tan».
4. Специфических операциях, например, комбинаторных чисел. В части языков такие операции задаются только функцией вычисления (C++), а в иных языках такие операции поддерживаются (Python), существуют различия между формами записи, которые допустимы. Так, число сочетаний это: $C(m,n) = \binom{m}{n} = C_m^n = \frac{m!}{(m-n)!n!}$.

Логические утверждения аналогично поддерживаются разными языками. В большом числе случаев они определяются как расширение нотации выражений: они могут быть связаны операциями сравнения или равенства, а еще логическими операциями: дизъюнкцией и конъюнкцией. Подобно символьным выражениям, логические утверждения компилируются в исполняемый код или в определенные структуры, которые предназначены для их преобразования; затем возможна проверка их правильности при заданных значениях переменных.

Специализированные математические пакеты содержат также возможности для применения правил преобразования к существенной части логических утверждений, которые задают несложные уравнения, или системы

таких уравнений. Так, WolframAlpha [133] предоставляет возможности решения уравнений из областей линейной алгебры, тригонометрии, и многих других.

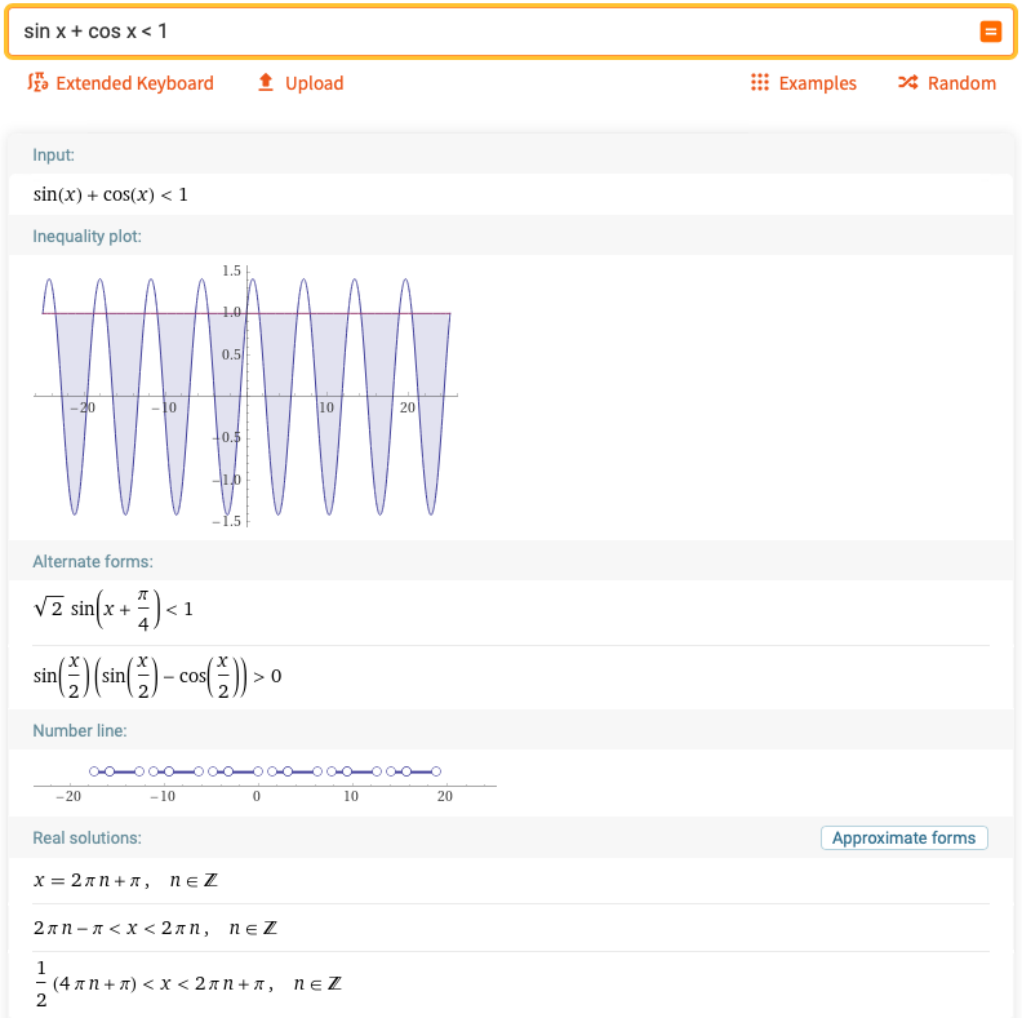

Рис. 10. Пример решения неравенства WolframAlpha

Значительно хуже ситуация с системами для записи и интерпретации решений. Обнаружены исключительно решения, дающие возможности записи цепочек преобразования без интерпретации и проверки. Такие системы делятся на 2 типа: блокноты, позволяющие записывать произвольные математические преобразования (Wiris); а также блокноты, выводящие структурированные наборы полей для записи решений определенных видов. Вполне приемлемый по удобству интерфейс для записи решений простых задач школьного уровня предоставляет eMathStudio [139].

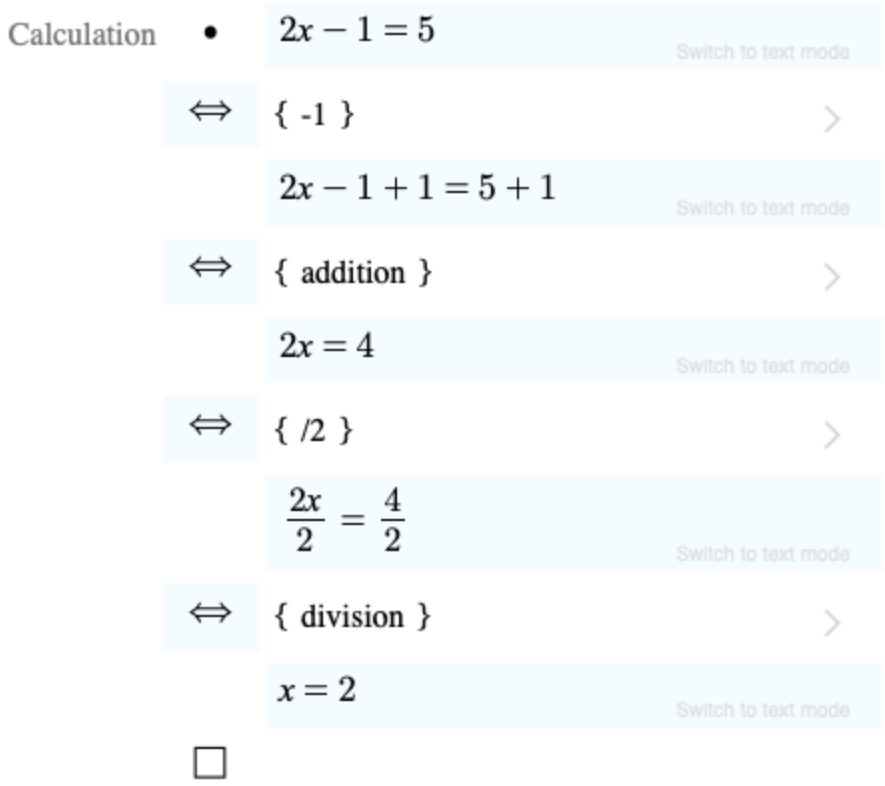

Рис. 11. Пример решения в eMathStudio

Системы первой группы позволяют только записывать решения. Во второй группе есть возможности для проверки решений, но они не годятся для очень многих из рассматриваемых в работе задач. Для:

1. Учета области определения.
2. Использования итераторных функций
3. Использования функций, которые определяются через итераторные например, комбинаторных чисел.
4. Проверки правильности решений с выражениями с числом символьных переменных больше двух.

Одновременно такие системы предполагают чересчур суровые требования к тривиальности решений, они не подходят для студентов младших курсов и обучающихся старших классов, так как те облают достаточно высокой степенью владения арифметическими операциями.

1.3 Существующие способы автоматической проверки

Самый распространенный способ автоматической проверки решений - проверка по ответу, а именно сравнение ответа, который нашел обучающийся, с тем ответом, который обозначил преподаватель. Значительно менее используемыми являются методы пошаговой верификации решений, которые

подразумевают проверку не только ответа, но также хода рассуждений, по которому обучаемый дошел до ответа.

1.3.1 Автоматическая проверка по ответу

Ключевые составляющие:

1. Определение языка той предметной области, используемого для записи ответов.
2. Разработка языка для критериев проверки по ответу. Например
    a. Вхождение или его отсутствие в символьное выражение или логическое утверждение для конкретной переменной.
    b. Вхождение или его отсутствие в символьное выражение или логическое утверждение операции с возможными ограничениями на аргументы.
3. Разработка алгоритма проверки ответа, который задан учащимся, с правильным. На нынешний момент есть 3 способа проверки: посимвольное сравнение, нормализация, а затем посимвольное сравнение, а также метод вычислительных экспериментов.

Посимвольное сравнение это частным случай проверки на основе нормализации.

1.3.1.1 Методы, основанные на нормализации

Пусть дан класс выражений E. Совокупность нормальной формы и последовательности преобразований элементов E должны соответствовать перечисленным далее условиям:

1. Предопределенная цепочка преобразований задана алгоритмом.
2. Нормальные формы выражений E равны если и только если эти выражения равны.
3. Нормальная форма существует для выражений E и есть возможность получить ее из любого выражения E через применение к нему предопределенного набора преобразований.

Проверить равенство заданных выражений E возможно приведя их к нормальной форме, а затем сравнивая эти формы. Кроме проверки, этот метод дает возможность предъявить доказательство равенства: цепочки преобразований.

Главный недостаток этого подхода - маленькая мощность класса символьных выражений, для которых эквивалентность проверяется сведением к нормальной форме. Это следствие из теоремы Ричардсона [115], где доказана алгоритмическая неразрешимость задачи сравнения двух символьных выражений.

Теорема Ричардсона: Пусть R – класс выражений созданных из переменной x, рациональных чисел и двух вещественных чисел π и $\ln 2$ с помощью операций сложения, умножения, композиции и функций exp и абсолютная величина. Тогда, если выражение E из R, то предикат «$E = 0$» неразрешим.

Число π невозможно выразить выражениями из R, в которых нет $\ln 2$. Аналогично, $\ln 2$ невозможно выразить выражениями R, в которых нет π. Подтверждений или опровержений возможности выражения π и $\ln 2$ друг через друга выражениями R обнаружить не удалось, в обоих случаях есть возможность заменить $\ln 2$ и π на переменные так, чтобы теорема Ричардсона осталась актуальной:

1. Когда возможности выражения π и $\ln 2$ друг через друга выражениями R не существует – заменим их на независимые друг от друга переменные. Множество правил, применимых к выражениям, не изменится, теорема останется актуальной.
2. Когда возможность выражения π и $\ln 2$ друг через друга выражениями R существует – заменим $\ln 2$ переменной, число π – выражением от этой переменной (оно существует, так как через переменную, которая соответствует $\ln 2$ можно выразить функции, обратные exp). Аналогично, множество правил, применимых к выражениям, не изменится, теорема останется актуальной.

В итоге такой замены R превращается в класс символьных выражений. Задача проверки равенства символьного выражения нулю алгоритмически неразрешима. Поэтому и задача оценки совпадения ответов, которые заданны в форме символьных выражений, неразрешима произвольным детерминированным алгоритмом, включая алгоритмы, использующие нормализацию.

Однако методы, которые основанные на нормальных формах, активно применяются при сравнения выражений из многих программных комплексов, WolframAlpha [133], Mathematica [97], Maple [89] – на практике и дают вполне применимые результаты, дают возможность проверять равенство символьных выражений большом числе случаев.

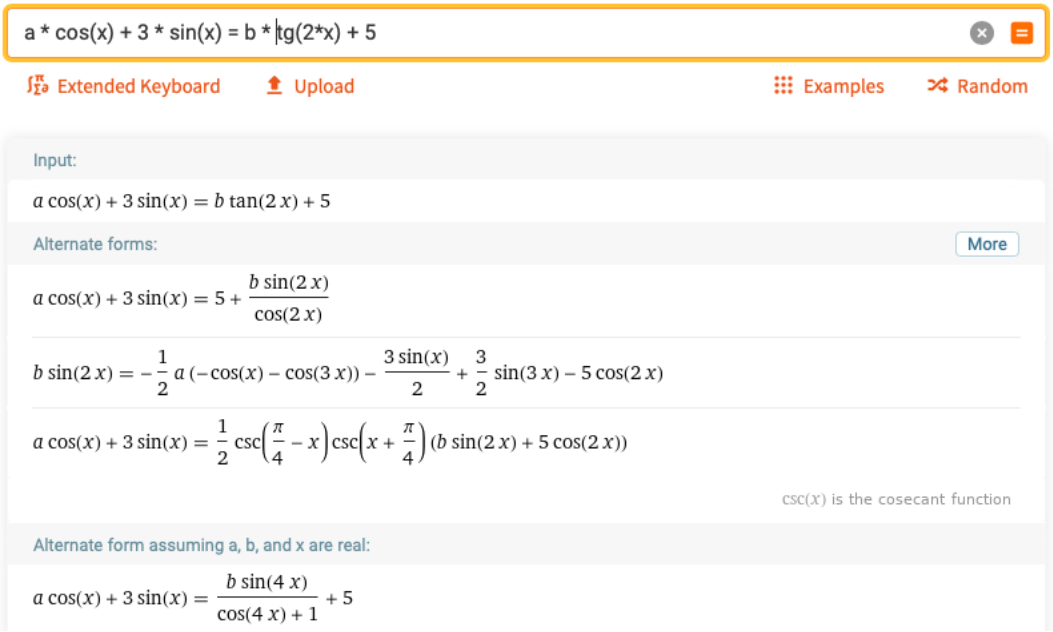

Рис. 13. Пример сведений при разборе проверки равенства выражений в программном комплексе WolframAlpha

Но для проверки тривиальности применение такого подхода слишком трудоемко: потребуется задание, а затем перебор очень большого числа возможных комбинаций. В данной работе предлагается метод оптимизации этого перебора.

1.3.1.2 Методы на базе вычислительных экспериментов

Идея в сравнении ответа с заданным преподавателем через проверку их совпадения для различных значений параметров. Метод подразумевает случайный или предусловленный выбор точек, затем для всех точек проводится серия вычислительных экспериментов: координаты точек подставляются как значения символьных переменных в выражениях, а потом полученные значения выражений сравниваются. Когда значения хотя бы в одной из точек не совпали – полагаем их неравными; иначе полагаем их равными.

Этот метод вероятностный, так как значения вычисленных выражений сравнивается лишь в выбранных точках, а не во всех возможных значениях переменных. И алгоритм может не найти ту точку, в которой сравниваемые выражения не равны или принять выражения, которые не равны за равные, или неправильный ответ, данный учащимся, принять за правильный.

Главными проблемами, которые возникают при применении этого метода вычислительных экспериментов для целей работы, являются:

1. Выбор критериев оценки точности вычислений.
2. Подбирать точки, в которых символьные выражения потом будут сравниваться.

3. Нет возможности проверить тривиальность при переходе между соседними шагами.

1.3.2 Автоматическая проверка по шагам

В ней проверяется не только ответ, но также ход решения рассматриваемой задачи, она решает многие проблемы, возникающие при автоматической проверке решений по ответу. Предлагает полагать решение правильным, если приведено его доказательство.

Такой подход полагает, что есть формальное описание задачи, а также её решения, и по ним можно предъявить логическую цепочку, подтверждающую, что предъявленное решение является решением задачи.

Допустим используется исчисление предикатов первого порядка. Тогда определим:

1. Данное множество допустимых X
2. Что неизвестно: множество возможных ответов Y
3. Предусловие $P(x)$, ему должны соответствовать входные данные
4. Постусловие $Q(x, y)$, связывающее входные и выходные данные.

Тогда решение задачи сводится к конструктивному доказательству для теоремы существования такого вида:

$$\forall x \in X \Big(\big(P(x) \to \exists y \in Y (Q(x, y)) \big) \Big).$$

После сколемизации используемого квантора существования можно найти «неизвестную» функцию f, она получает ответ по имеющимся исходным данным, а также удовлетворяет условию

$$P(x) \to Q(x, f(x))$$

В том числе в нотации логики Хоара

$$\{P(x)\}\, y \coloneqq f(x)\, \{Q(x, y)\}.$$

Функцию f учащийся должен предъявить в качестве решения задачи. Тогда оценка правильности решения задачи сводится к доказательству этого утверждения в любой из выше приведённых форм.

Методы пошаговой проверки решений задач требуют более индивидуального подхода в задачах разных типов, и их сложнее реализовать. Поэтому на практике встречающиеся случаи его применения редки, не подходят для использования в широком круге рассматриваемых задач на оперирование формулами.

Так, метод такой пошаговой проверки решений используется в eMathStudio, ориентированом за задачи на правила оперирования простыми арифметическими операциями. Создатели системы тщательно продумали

возможные виды переходов и их нотации [65-71]. Сама система проверки в каждом переходе посмотрит все преобразования и, когда находит правило, обосновывающее переход, положит переход правильным. А в противном случае нетривиальным или некорректным.

Создатели eMathStudio провели исследования о роли влияния применения этой системы на исследуемый образовательный процесс; в том числе собирали отзывы обучавшихся, от которых ожидались записанные решения задач в предложенном создателями формате. В результате исследований запись решений учебных задач в выбранном формате отнимает у обучаюхся больше времени, зато ведет к лучшему осознанию материала и выработке навыка построения цепочек рассуждений [69].

Рис. 14. Пример пошаговой проверки решения в eMathStudio

Однако существующие на нанешний день подходы предполагают чересчур жесткие критерии проверки тривиальности, так за один переход возможно совершение только простого преобразования. Так, нетривиальным и недопустимым, будет приведение существенного числа однородных слагаемых в многочлене за один переход. На практике эти преобразования элементарны для студентов и учащихся в старших классах, они часто выполняются за один переход в независимости от числа слагаемых и раскрываемых скобок.

Дополнительно существует ряд способов, нацеленных на автоматическую верификацию решений в определенных предметных областях, например геометрические построения, эйлеров путь в графах, и других. Используемые там методы не получается использовать для проверки решений задач на преобразование формул.

2. Язык записи цепочек преобразований и их проверка эффективным перебором правил

Положим решение задачи дано в виде цепочки преобразований над выражениями и утверждениями. Проверка решения задачи подразумевает проверку корректности, а также тривиальности всех переходов между всеми соседними элементами в цепочке преобразований, включая пошаговую проверку выводимых дополнительных правил – вложенных цепочек преобразования.

Критерии корректности и тривиальности получаются по заданному преподавателем ***взвешенному множеству допустимых правил преобразования***.

Взвешенное множество допустимых правил преобразования – совокупность:

1. ***Множества допустимых правил преобразования*** P – множества правил преобразования, доступны в ходе решения задачи
2. ***Весов правил преобразования*** – отображение $W : P \to \{x \mid x \in Q \wedge x \in (0; 1]\}$.

Рациональное число, соответствующее допустимому преобразованию, является ***весом правила преобразования***, определяет его сложность.

Пример:

Множество $P = \{C(m,n) = \frac{m!}{(m-n)!n!}; V(m,n) = \frac{(m+n-1)!}{(m-1)!n!}\}$ и $W = \{C(m,n) = \frac{m!}{(m-n)!n!} \to 0.6; V(m,n) = \frac{(m+n-1)!}{(m-1)!n!} \to 0.7\}$ - взвешенное множество допустимых правил преобразования. 0.6 и 0.7 - веса правил $C(m,n) = \frac{m!}{(m-n)!n!}$ и $V(m,n) = \frac{(m+n-1)!}{(m-1)!n!}$.

Взвешенное множество допустимых правил преобразования определяет ***взвешенное множество суперпозиций допустимых правил преобразования*** – объединение:

1. ***Множества суперпозиций допустимых правил преобразования*** P* – множества допустимых правил преобразования, расширенное суперпозициями содержащихся внутри него отображений. Суперпозиция правил также правило, его применение к выражению соответствует применению правил, составляющих суперпозицию.
2. ***Весов суперпозиций правил преобразования*** – тотального отображения $W^* : P^* \to Q$, где сопоставляемое суперпозиции число являются суммами весов правил, входящих в суперпозицию - ***вес суперпозиции правил преобразования***.

Пример:

$C(m,n) + V(m,n) \to \frac{m!}{(m-n)!n!} + \frac{(m+n-1)!}{(m-1)!n!}$ - суперпозиция $C(m,n) = \frac{m!}{(m-n)!n!}; V(m,n) = \frac{(m+n-1)!}{(m-1)!n!}$. Для $W = \{C(m,n) = \frac{m!}{(m-n)!n!} \to 0.6; V(m,n) = \frac{(m+n-1)!}{(m-1)!n!} \to 0.7\}$, суммарный вес 1.3.

Переход между соседними L и R в цепочке преобразований корректный, тогда и только тогда, когда существует обосновывающая суперпозиция правил. Такая, что, при применении ее к L получится R, или при применении ее к R получится L.

Переход между соседними L и R в цепочке преобразований считается корректным и тривиальным, тогда и только тогда, когда существует обосновывающая суперпозиция правил с весом не больше 1.

Ожидается, что преподаватель подбирает веса правил так, чтобы возможные переходы были не чересчур сложными для обучающих, но также учащимся не требовалось слишком детально расписывать простые преобразования.

Также заданные правила могут иметь ***условия применимости***, которым обязательно удовлетворять для их применения к элементу цепочки. Разновидности условий:

1. По типу элемента цепочки
2. По типу переходов в цепочке
    a. Между выражениями: равенство или сравнение.
    b. Между логическими утверждениями: эквивалентность или следствие.
3. По области определения: условию на допустимые значения в правиле, с которым его можно применить.

Примеры:

a. Правило $\sqrt[2]{x} \Longrightarrow x$ верно исключительно при $x > 0$.
b. Правило a $*$ c $< b * c \Longrightarrow a < b$ верно исключительно при $c > 0$.

Для удовлетворения условию требуется чтобы выполнялось одно из:

- Корректен и тривиален переход между одним из утверждений в системе и утверждением, которое описывает область определения правила преобразования
- В ходе применении правила преобразования вместо всех символьных переменных подставляются константы, такие что при них истинно утверждение, которое описывает область определения.

Примеры:

a. Правило $\sqrt[2]{x} \Longrightarrow x$, верное исключительно при $x > 0$:
   i. Не применимо к выражению $\sqrt[2]{x}$ в произвольной цепочке преобразований.
   ii. Применимо к символьному выражению $\sqrt[2]{x}$ в логическом утверждении: $\begin{cases} x > 0 \\ y + 4 = x \\ \sqrt[2]{x} = y \end{cases}$
   iii. Применимо к символьному выражению $\sqrt[2]{4}$ в произвольной цепочке преобразований.

b. Правило $\mathrm{a} * \mathrm{c} < b * c \Longrightarrow a < b$, верное исключительно при $c > 0$:
   i. Не применимо к утверждению $\mathrm{a} * \mathrm{c} < b * c \Longrightarrow a < b$ в произвольной цепочке преобразований.
   ii. Применимо к утверждению $\mathrm{a} * (a + d) < b * (a + d)$ в: $\begin{cases} a + d > 0 \\ \mathrm{a} * (a + d) < b * (a + d) \\ x > 0 \end{cases}$
   iii. Применимо к выражению утверждению $\mathrm{a} * 3 < b * 3$ в произвольной цепочке преобразований.

4. По контексту обозначений: требуется ли строгое соответствие между переменными в правиле и задаче, или это правило может быть применимо и для любых других обозначений.

Примеры:

a. Правило $\sin(x + y) = \sin(x) * \cos(y) + \sin(y) * \cos(x)$ верно для любых x и y и применимо при любых обозначениях.
b. Правило $I = U/R$ (закон Ома) верно только для I, обозначающей силу тока, U – напряжение и R – сопротивление.
c. Правило $\cos(2 * a) = 0.5$ в задаче «вычислите $\cos^2 a$ при условии $\cos(2 * a) = 0.5$» верно только для конкретного a в задаче.

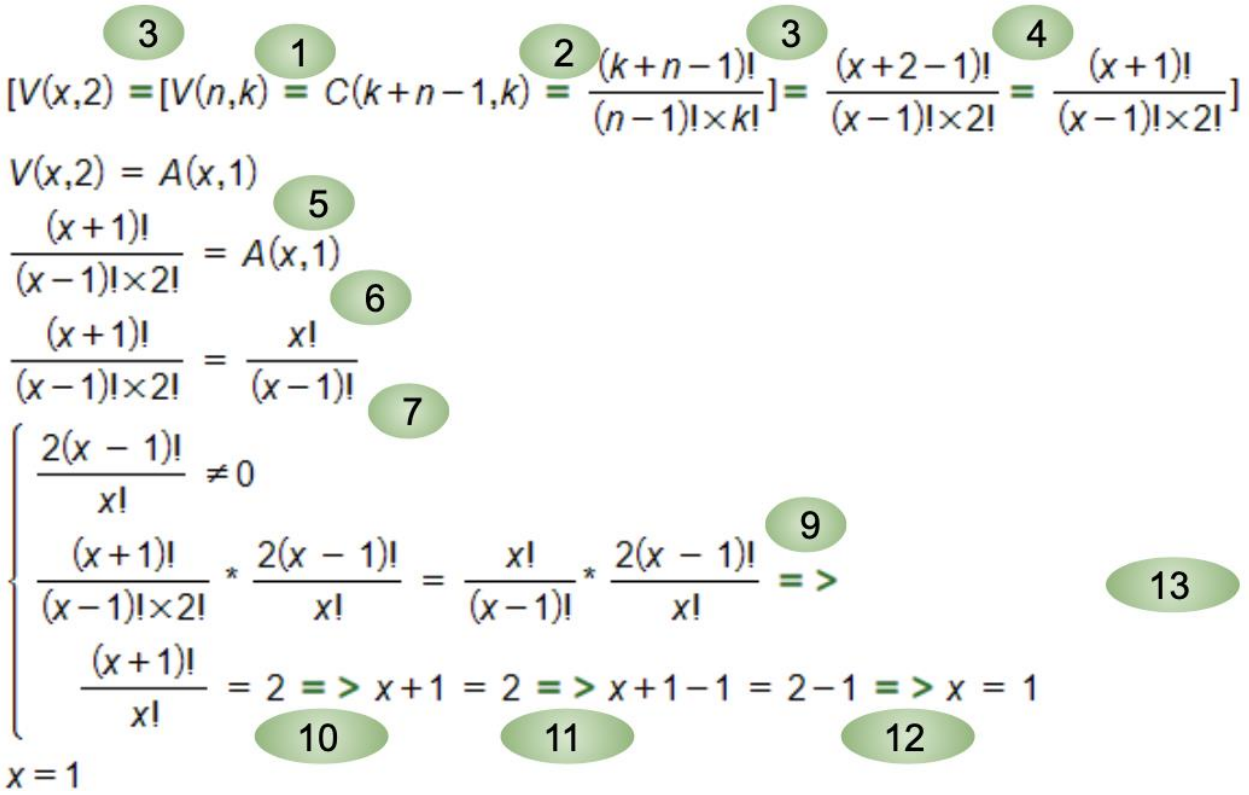

Рис. 15. Пример последовательности проверки переходов в цепочке преобразований, решающей уравнение $V(x,2) = A(x,1)$

Доступные переходы между элементами из цепочки преобразований предлагается описать ***взвешенным ориентированным графом преобразований*** D (V, E). ***Вершины*** E такого ориентированного графа соответствуют элементам рассматриваемых цепочек преобразования. ***Дуги*** - связывают их, тогда и только тогда, когда в множестве допустимых правил существует правило преобразования, при применении которого к выражению или утверждению, которое соответствует стартовой вершине дуги, получится элемент, который соответствуют финальной вершине рассматриваемой дуги. Вес этой дуги будет равен наименьшему из весов подобных правил преобразования.

Теперь задача проверки корректности совершенного перехода между элементами рассматриваемой цепочки преобразований сведется к задаче проверки наличия пути между вершинами графа, которые им соответствуют. А задача проверки тривиальности совершенного перехода накладывает ограничение на длину такого пути и сводится к задаче проверки наличия пути с суммой весов дуг в пути не превышающей 1.

Число вершин и дуг в таком орграфе преобразований неограниченно, так как неограниченное число результатов применений суперпозиций допустимых

правил к выражениям и утверждениям. Поэтому при проверке существования пути с весом не превышающем 1 неприменимы алгоритмы поиска пути, которые основаны на рассмотрении всех ребер и вершин в графе.

Но ограничение максимальной длины пути, которое задается требованием тривиальности, позволяет ограничить время работы алгоритма. Также сократить время работы позволяет использование эвристики для оценки расстояния для исследования преимущественно путей, более вероятно подтверждающих корректность и тривиальность перехода.

На базе комбинации этих идей в работе разработана эвристика оценки расстояния и модификация алгоритма A* [38] для проверки правильности переходов в цепочке преобразований **BASC** (Bidirectional A* Search Combination).

Пусть даны 2 элемента u, v из цепочки преобразований. Нужно построить оценки расстояния $d(u, v)$ между соответствующими u, v вершинами в ориентированном графе преобразований, чтобы:

1. Временная трудоемкость вычисления этой оценки $\tilde{d}(u, v)$ занимала константное время от числа вершин;
2. Минимизировалось число ошибок вида: $(\tilde{d}(u_1, v) < \tilde{d}(u_2, v)) \wedge (d(u_1, v) > d(u_2, v))$, где $u_1 u_2, v$ – вершины ориентированного графа.

Оценкой расстояния между элементом u и множеством элементов V $\tilde{d}(u, V)$ назовем минимальную из оценок расстояний $\tilde{d}(u, v)$, где $v \in V$.

Положим оценку расстояния между двумя символьными выражениями u, v равна сумме величин:

1. Минимальная сумма значений чисел в ячейках ***матрицы несоответствия подвыражений***, которые выбраны так, что в каждой строке и каждом столбце матрицы встретится хотя бы одна ячейка.

   По строкам такой матрицы несоответствия подвыражений располагаются выражения, которые соответствуют всем ветвям дерева выражения u с аргументами, которые не являются листовыми; по столбцам – точно также для выражения v. В ячейках матрицы расположены нули, если вхождения по строке и столбцу совпадают и единицы – если не совпадают. Под совпадением имеется в виду полное совпадение функции и списка аргументов, ожидается что они идут в том же поорядке.

2. Число переменных, которые входят в выражение u и не входят в v.
3. Число переменных, которые входят в выражение v и не входят в u.

Оценка расстояния для логических утверждений u, v равна наименьшей сумме значений в ячейках ***матрицы несоответствия фрагментов логических утверждений***, которые выбраны так, что в каждой строке и каждом столбце этой матрицы встретится не меньше чем одна ячейка.

Такая матрица несоответствия фрагментов логических утверждений определяется аналогично матрице несоответствия выражений, но:

1. По столбцам и строкам расположены фрагменты деревьев утверждений, которые включают в себя логические условия над выражениями.
2. Для оценки несоответствия узлов Совокупность_логических_утверждений и Система_логических_утверждений за коэффициент несоответствия предлагается брать сумму числа тех утверждений, находящихся в первом сравниваемом узле и не находящиеся во втором и наоборот.

Алгоритм BASC для проверки переходов

Вход: 2 элемента проверяемой цепочки преобразований l и r; взвешенный ориентированный граф преобразований D (V, E).

Выход: 1, когда переход между l и r и корректен, и тривиален; иначе 0.

Используемые структуры данных:

1. Очереди открытых вершин, соответствующие l и r: Q_l и Q_r. Вершина попадает в очередь открытых вершин после того, как алгоритм нашел элемент, который ей соответствует, как результат применения допустимого преобразования к элементу, который соответствуют рассматриваемой вершине, взятой из очереди открытых вершин. В процессе работы алгоритма рассматриваются вершины из очередей открытых вершин для l и r, которые имеют наибольший приоритет. После того, как вершина рассмотрена, она удаляется из очереди и попадает в список закрытых вершин C_l и C_r.
    a. В момент добавления вершины в Q_l её приоритет будет определяться по оценке расстояния от неё до множества в C_r: чем меньше эвристика оценки расстояния – тем выше ее приоритет и тем раньше эта вершина будет извлечена. Точно также, в момент добавлении вершины в Q_r её приоритет определяется по оценке расстояния от этой вершины до множества в C_l.
    b. После рассмотрения вершины из очереди открытых вершин для l или r в эту же очередь попадут все вершины, которые смежны с рассматриваемой и еще не были в очередях открытых вершин.
2. Списки закрытых вершин, соответствующие l и r: C_l и C_r. Вершина попадет в C_l после того, как будет рассмотрена и удалена из Q_l. Аналогично,

вершина попадет в C_r после того, как будет рассмотрена и удалена из Q_r. Составленные списки закрытых вершин C_l и C_r используются при расстановке приоритетов вершин в очередях открытых вершин.

Процедура рассмотрения и извлечения вершины из очереди открытых вершин:

1. Выбирать и рассмотреть вершину с максимальным приоритетом из Q_l (либо Q_r). Для всех из соответствующих смежным ее вершинам элементов вершины u, не попадавшей в очереди открытых вершин ($u \notin Q_l \cup Q_r \cup C_l \cup C_r$) и такой, что $d(l,u) \leq 1$ (либо $d(u,r) \leq 1$), выполняем шаги:

1.1 MIN_DIST = Оценка расстояния между u и r (либо l).

1.2 Для всех v из $Q_r \cup C_r$ (либо $Q_l \cup C_l$) и такого, что $d(l,u) + d(v,r) \leq 1$ (либо $d(l,v) + d(u,r) \leq 1$):

1.2.1 Если u и v одинаковы – завершить процедуру и вернуть 1; имеющаяся суперпозиция преобразований из l в v и из v в r подтверждает корректность и тривиальность перехода.

1.2.2 Оценить расстояние между u и v. Если это расстояние меньше MIN_DIST, то обновляем MIN_DIST = Оценка расстояния между u и v.

1.3 Добавить вершину u в список открытых вершин Q_l (либо Q_r), с приоритетом по MIN_DIST и расстояний до l (либо r).

2. Удалить рассмотренную вершину из Q_l (либо Q_r) и поместить в список закрытых вершин C_l (либо C_r).
3. Вернуть 0.

Алгоритм:

1. Если l и r совпадают – вернуть 1.
2. Поместить вершины, соответствующие l и r в Q_l и Q_r соответственно.
3. Пока Q_l либо Q_r не пустая:
   3.1. Если Q_l не пуста – выполнить процедуру рассмотрения и извлечения вершины из Q_l. Если процедура возвращает 1 – возвращаем 1.
   3.2. Если Q_r не пуста – выполнить процедуру рассмотрения и извлечения вершины из Q_r. Если процедура возвращает 1 – возвращаем 1.
4. Вернуть 0.

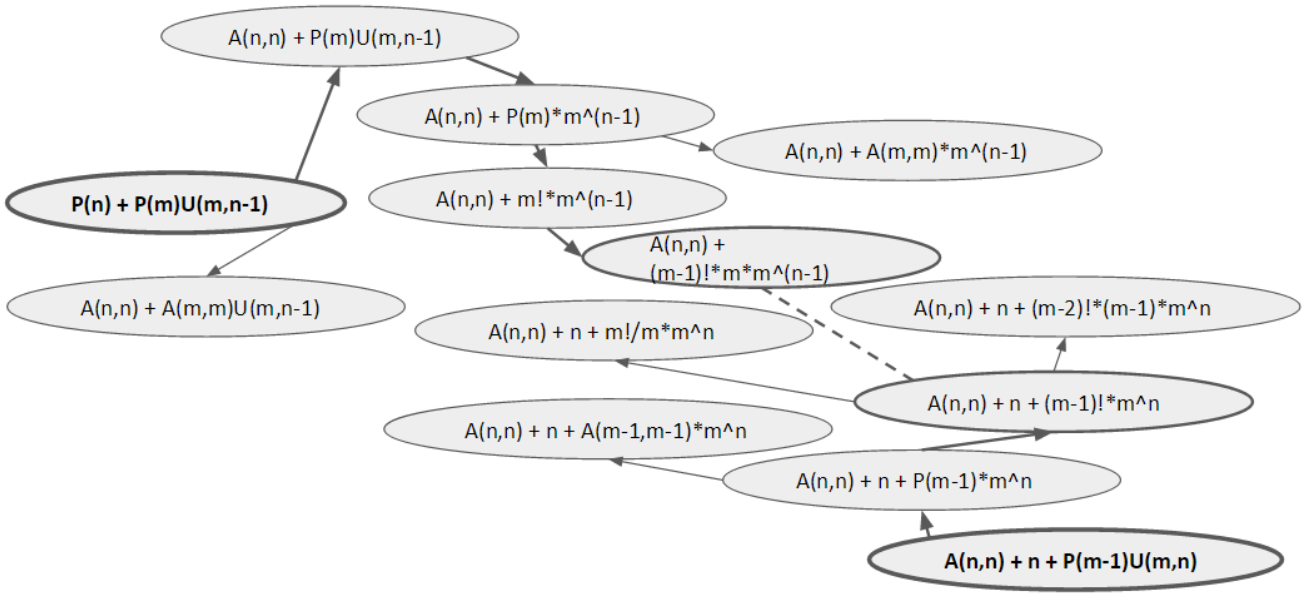

Рис. 16. Пример: схема работы BASC при проверке равенства “P(n) + P(m)U(m, n-1)” и “A(n,n) + n + P(m-1)U(m, n)”

Временная эффективность алгоритма:

В самом худшем случае алгоритм в ходе работы переберет все вершины u и v, для которых верно условие $d(l,u) + d(v,r) \leq 1$ (либо $d(l,v) + d(u,r) \leq 1$), а это соответствует экспоненциальному времени работы от вероятного числа правил, которые могут быть применены за один переход. При случае естественных правильных переходов алгоритм работать значительно быстрее за счет рассмотрения в-основном разумных правил, применение которых будет делать выражения более похожими друг на друга.

Описанные эвристики значительно понижают приоритеты, по которым рассматриваются суперпозиции правил преобразования, добавляющих в рассматриваемые выражения однозначно лишние переменные или функции. Недостаточно качественно работает ранжирование простых правил, которые соответствуют свойствам коммутативности, ассоциативности, и многим другим частоприменяемым, но при этом не очень существенно меняющим рассматриваемое выражение либо утверждение.

Самым существенным это будет при проверке тех переходов, в которых естественно применение значительного числа простых правил преобразования. Так, общепринятым является совершение приведения однородных слагаемых в ходе одного перехода, в котором будут применяться суперпозиции десятков правил, от сложения до раскрытия скобок. Описанных

эвристик не хватает, чтобы перебор таких правил стал достаточно эффективным.

То есть, если задать часто применимым правилам такие веса, чтобы они могли применяться в ожидаемом при стандартном решении учебных задач объеме, алгоритм BASC сможет их проверять достаточно быстро.

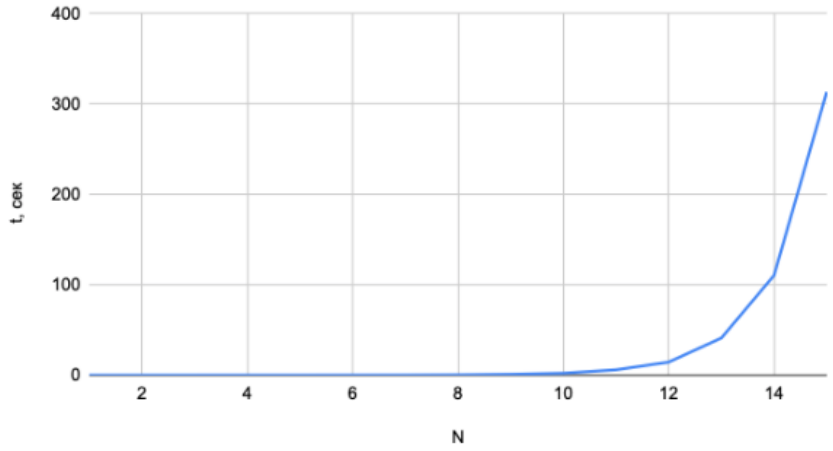

Рис. 17. График усредненной зависимости времени работы алгоритма t (сек) от числа однородных слагаемых N.

3. Метод вычислительных экспериментов для проверки корректности преобразований

Базовые идеи методов, которые основаны на вычислительных экспериментах, описаны в первом разделе. Основное преимущество такого подхода – возможность проверки корректности совершаемых переходов в независимости от числа применяемых в них правил. А именно, метод вычислительных экспериментов дает возможность проверить корректность вычислимых утверждений совершением серии вычислительных экспериментов-тестов с различными значениями параметров утверждения, которые проводятся с целью отыскать набор значений параметров, на котором утверждение не будет выполняться. Если соответствующий набор значений находится – утверждение полагается ложным, иначе – истинным. То есть, качество проверки определяется именно вероятностью классифицировать некорректный переход как корректный. Основной вклад в избежание подобных ошибок вносит алгоритм подбора совокупностей значений параметров, при которых утверждение будет проверяться; а именно, от репрезентативности таких подобранных наборов значений параметров зависит вероятность ошибки при проверке.

В ходе проверке переходов над выражениями или утверждениями проверяются равенство или отношение порядка над символьными

выражениями, эквивалентность фактов или следствие над утверждениями. Параметрами логических утверждений являются переменные выражений, которые в него входят. Так как вычисление утверждения – это результат вычисления логической комбинации результатов проверки входящих в него равенств и отношений порядка, задача проверки переходов над ними сводится к задаче проверки переходов над выражениями.

Существующие алгоритмы проверки равенств дают довольно точные результаты при случайном подборе совокупностей значений переменных в большинстве случаев, кроме выражений с небольшой областью определения, при ней возрастает риск непопадания выбранного значения переменной в эту область определения.

При проверке неравенств вероятность ошибки существенно выше, так как выполнение отношения порядка в каком-то числе случайных точках не гарантирует удовлетворения этому отношению на всей области определения.

Для решения проблемы с маленькой областью определения в работе предлагается перенести вычисления в комплексную область. Так, если при взятом наборе значений символьных переменных существует неопределенное выражение, вычисляются значения обоих выражений в комплексных числах и сравниваются. Когда значения не совпадают – полагаем выражения неравными.

Однако комплексный логарифм принимает бесконечное число значений $\mathrm{Ln}\, z = \mathrm{Ln}\left(r * e^{i(\varphi+2\pi k)}\right) = \ln r + i(\varphi + 2\pi k), \quad k \in Z.$ Также, бесконечное множество значений могут принимать комплексные функции, которые определяются на основе логарифма, к примеру обратные тригонометрические.

Итак, оба выражения потенциально принимают бесконечное счетное множество значений. Будем фиксировать одну из голоморфных ветвей в каждом из вычислений многозначной функции; ту, где $k = 0$, то есть будем считать результатом комплексного логарифма число, для которого мнимая часть попадает в $[0; 2\pi)$.

Но тогда перестает соблюдаться равенство

$$\sum_{j=1}^{n} \mathrm{Ln}(z_j) = \mathrm{Ln}\left(\prod_{j=1}^{n} z_j\right)$$

, так как

$$\mathrm{Ln}\left(\prod_{j=1}^{n} z_j\right) = \mathrm{Ln}\left(\prod_{i=1}^{n} r_j * e^{i\varphi_j}\right) = \mathrm{Ln}\left(\left(\prod_{i=1}^{n} r_j\right) * \left(e^{i\sum_{j=1}^{n}\varphi_j}\right)\right) =$$

$$= ln\left(\prod_{i=1}^{n} r_j\right) + i\left(\sum_{j=1}^{n}\varphi_j - 2\pi l\right) = \sum_{j=1}^{n}\mathrm{Ln}(r_j) + i\left(\sum_{j=1}^{n}\varphi_j - 2\pi l\right)$$

, где l будет таким, что $0 \le \sum_{j=1}^{n}\varphi_j - 2\pi l < 2\pi$ поскольку мы возьмем голоморфную ветвь логарифма, в которой мнимая часть находится в $[0; 2\pi)$; а

$$\sum_{j=1}^{n}\mathrm{Ln}(z_j) = \sum_{j=1}^{n}\mathrm{Ln}(r_j) + i\left(\sum_{j=1}^{n}\varphi_j\right)$$

Получаем, что при l отличных от нуля, сумма логарифмов неравна логарифму произведения.

Обобщая, аргументы многозначной комплексной функции могут быть произвольными выражениями, произвольным образом зависящими от взятого набора значений переменных. Будем считать, что аргумент комплексного числа равномерно распределен на $[0; 2\pi)$. Равенство будет выполняться, когда $0 \le \sum_{j=1}^{n}\varphi_j < 2\pi$; вероятность его выполнения равна вероятности, с которой сумма аргументов попадет в $[0; 2\pi)$.

Пусть ξ_j – случайная величина, которая соответствует аргументу комплексного числа. Пусть $p_{\xi_j}(x)$ – плотность распределения, поскольку распределение равномерное $p_{\xi_j} = \frac{1}{2\pi}$. Положим эти случайные величины независимыми для разных комплексных чисел. Получим распределение n-мерного вектора $(\xi_1, \xi_2, \dots, \xi_n)$ – равномерное распределение в n-мерном кубе, ребро которого равно 2π.

Покажем, что для двух логарифмов (n=2) такая вероятность равна $\frac{1}{2}$:

$$P(\xi_1 + \xi_2 \le 2\pi) = \int_0^{2\pi}\int_0^{2\pi-x} p_{\xi_1}(x)\, p_{\xi_2}(y)\, dydx =$$

$$= \int_0^{2\pi}\int_0^{2\pi-x} \frac{1}{4\pi^2}\, dydx = \int_0^{2\pi} \frac{2\pi - x}{4\pi^2}\, dx = \frac{1}{2}$$

Покажем, что для n логарифмов вероятность равна $\frac{1}{n!}$:

$P(\xi_1 + \xi_2 + \cdots + \xi_n \le 2\pi)$ равно отношению объема множества X = $\{x = (x_1, x_2, \dots, x_n) \mid x_i \ge 0, \sum_{i=n}^{n} x_i < 2\pi\}$ к объему n-мерного куба. Объем n-мерного куба $(2\pi)^n$.

Соответственно, множество X соответствует пирамиде, которая отсекается гиперплоскостью $\sum_{i=n}^{n} x_i = 2\pi$ от прямого угла. Объем такой пирамиды равен $\frac{(2\pi)^n}{n!}$ [95].

Целевая вероятность равна $\frac{(\frac{(2\pi)^n}{n!})}{(2\pi)^n} = \frac{1}{n!}$.

Получили, что при проверке перехода между символьными выражениями с n логарифмами вероятность равенства символьных выражений в вычислительном эксперименте со случайной совокупностью значений переменных $\frac{1}{n!}$ если эти выражения равны. В случае проведения m вычислительных экспериментов вероятность k равенств в вычислительных экспериментах получится равной $\frac{m!}{(m-k)!*k!} * \left(\frac{1}{n!}\right)^k * \left(1 - \frac{1}{n!}\right)^{m-k}$.

Вероятность получения как минимум двух подтверждений равенства из m совершаемых вычислительных экспериментов равна разности 1 и вероятностей 0 и 1 успехов соответственно: $1 - \left(1 - \frac{1}{n!}\right)^m - \frac{m}{n!} * \left(1 - \frac{1}{n!}\right)^{m-1}$; вероятность получения как минимум одного подтверждения равна $1 - \left(1 - \frac{1}{n!}\right)^m$.

Алгоритм:

Вход: 2 выражения l и r.

Выход: 1, если $l = r$; иначе 0.

1. Выбрать желательное число вычислительных экспериментов $m = 8 * n!$ по числу логарифмов n.
2. Обрезать m по порогу, который соответствует максимально допустимому числу экспериментов, выполнимому на конкретной ЭВМ за приемлемое время.
3. Вычислить вероятность как минимум 2-х подтверждений равенства. Если больше 0.995, то $expectedeq = 2$, иначе $expectedeq = 1$
4. В каждом из экспериментов:
    a. Случайно сгенерировать значения переменных в $[-\max num; maxnum]$, где $\max num$ – максимальная из констант в сравниваемых нами выражениях.
    b. Сравнить значения выражений при этих значениях переменных. Если они равны, то:
        i. $expectedeq := expectedeq - 1$
        ii. Если $expectedeq = 0$, вернуть 1.
5. Вернуть 0.

Для неравенств описанный выше подход, который предполагает сравнение символьных выражений в комплексной области, не может использоваться, так как в ней отсутствует отношение порядка. Ключевыми задачами становятся подбор точек в области определения и выбор наиболее репрезентативных точек для сравнения в них.

Разработанный алгоритм основывается на решении задачи минимизации метрик различия входных выражений итерационными методами, а именно комбинацией методов имитации отжига и градиентного спуска. При получении точки, выходящей за область определения решается задача минимизации модуля мнимой части метрик различия. При нахождении точки, в которой не выполняется проверяемое отношение порядка, оно полагается некорректным; если такая точка не находится, то корректным.

Вход: 2 выражения l и r, суммарно содержание n символьных переменных; проверяемое отношение порядка.

Выход: 1, когда l и r удовлетворяют отношению порадка; иначе 0.

Характеристики различия:

- Разность $l - r$ для отношений порядка $l > r$ (или $l \geq r$); иначе $r - l$. Если получается найти набор значений символьных переменных, при которых разность не больше 0 (или меньше 0 если отношение порядка нестрогое), такой набор значений является контрпримером, который опровергает проверяемое отношение порядка: алгоритм вернет 0.
- Частное $|l|/|r|$ когда отношение порядка $l > r$ (или $l \geq r$); иначе $|r|/|l|$. Если получается найти набор значений символьных переменных, при которых рассматриваемое частное не больше 1 (или меньше 1 если отношение порядка нестрогое) и проверяемое отношение порядка нарушается, такой набор значений является контрпримером, который опровергает проверяемое отношение порядка: алгоритм вернет 0.

Поиск подпространства, которое содержит корни разности $\boldsymbol{l}$ и $\boldsymbol{r}$:

1. В разности l и r ищем самые быстрорастущие (быстроубывающие) функции.
2. Положим $\max num$ равным максимальной из констант в сравниваемых символьных выражениях (если константы отсутствуют – то 1). Далее ищем точки в n-мерном параллелепипеде с центром в (0; 0) и с параллельными осям.
3. Если положительная разность наиболее быстрорастущих (модулей быстроубывающих) функций и суммы модулей оставшихся функций в вершинах параллелепипеда и на случайно выбранных точках вне его, но в пределах параллелепипеда с в 2 раза более длинными ребрами, выбираем

параллелепипед с ребрами $2 * \max num$ за искомое подпространство (оно с высокой вероятностью содержит искомый контрпример). Иначе увеличиваем ребра параллелепипеда в 2 раза и повторяем шаг 3.

Начальное приближение: случайно генерируемый кортеж значений символьных переменных в n-мерном параллелепипеде.

Минимальное число вычислительных экспериментов: квадрат суммы числа имеющихся узлов в деревьях сравниваемых выражений.

Минимизация мнимой части:

1. Пока модуль мнимой части метрики различия при рассматриваемом кортеже переменных больше ε, параллельно:
    a. Выбираем другое случайное начальное приближение. Переходим к проверке условия для него.
    b. Вычисляем разностные приближения производных в точке, задаваемой рассматриваемым кортежем значений, в n направлениях. Для каждого из этих направлений подбираем бинарным поиском коэффициент длины шага λ с которым модуль мнимой части примет наименьшее значение. За следующий новый набор значений выбираем тот, который соответствует направлению и λ, с которыми достигается наименьшее значение у модуля мнимой части. С этим набором значений символьных параметров переходим к проверке условия для него.
2. При нахождении набора значений символьных параметров, который попадает в область определения, прерываем оставшиеся параллельно выполняемые итерации.

Минимизация характеристики различия:

1. Пока контрпример не нашелся и не выполнено минимальное число итераций, параллельно:
    a. Выбираем другое случайное начальное приближение. Переходим к проверке условия для него.
    b. Если функция характеристики различия автоматически дифференцируема, то дифференцируем, выбираем направление наибольшего убывания градиента. Подбираем бинарным поиском коэффициент длины шага λ с которым метрика различия примет наименьшее значение. За следующий набор значений параметров выбираем соответствующий λ, при котором достигается наименьшее значение метрики различия. Переходим с полученным набором значений символьных параметров к проверке условия.

c. Вычисляем разностные приближения производных в точке, задаваемой рассматриваемым набором значений, в n направлениях. Для каждого из направлений бинарным поиском подбираем коэффициент длины шага λ при котором характеристика различия принимает минимальное значение. За следующий набор значений параметров выбираем соответствующий таким направлению и λ, при которых достигается наименьшее значение характеристики различия. Переходим с полученным набором значений параметров к проверке условия.

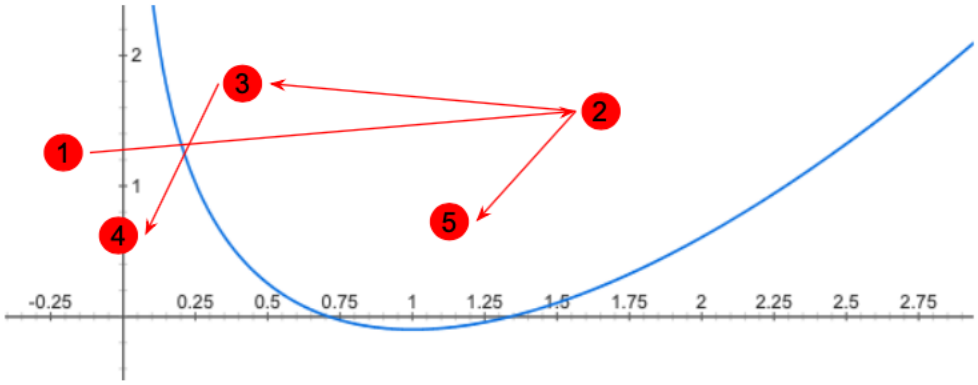

Рис. 18. Пример работы при проверке $(x+\frac{1}{x})\sqrt{x} > 2\sqrt{x} + 0.1$.

Время проверки переходов, где применяется существенное число правил, в алгоритме не зависит от числа правил, только от числа символьных переменных, также сложности вычисления выражений, их области определения и числа многозначных функций.

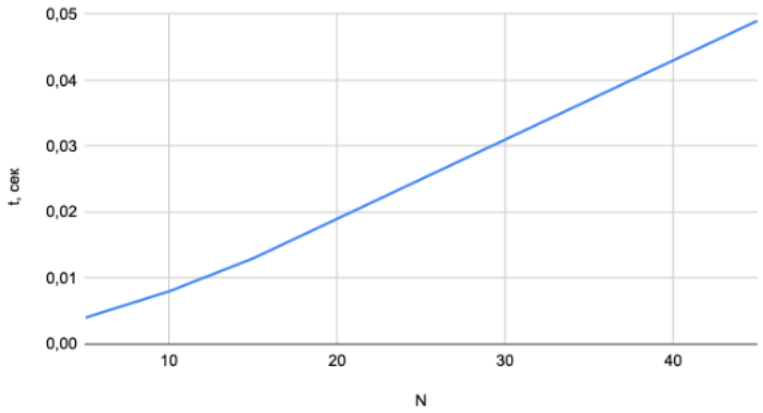

Рис. 19. График усредненной зависимости времени работы алгоритма t (сек) от числа однородных слагаемых N, которые можно привести при 5-10 различных переменных.

4. Метод проверки корректности и тривиальности цепочек преобразований

Назовем ***операциями с нулевым весом*** операции (и функции), такие что все правила, содержащие исключительно эти операции имеют вес 0 и дают возможность совершать все преобразования над выражениями, содержащими только эти операции за число шагов, которое не превышает некое натуральное N, которое задается для всех переходов.

К фрагментам символьных выражений, которые содержат только операции с нулевым весом, за один переход могут примениться любые суперпозиции правил преобразования если результаты их применения тоже содержат исключительно операции с нулевым весом. Так задаются преобразования, которыми обучающиеся владеют в совершенстве; и проверка тривиальности преобразований в такой ситуации не нужна.

Такими операциями с нулевым весом являются сложение, вычитание и умножение. При решении задач учащиеся имеют возможность неограниченно преобразовывать многочлены, в том числе, приводить однородные слагаемые в них.

Введем понятие сокращенного взвешенного ориентированного графа преобразований. Вершины этого ориентированного графа соответствуют выражениям или утверждениям, которые могут быть получены друг из друга путем применения преобразований над операциями с нулевым весом. Также как в обычном ориентированном графе, дуги в этом орграфе связывают вершины если и только если есть правило, в результате применения которого к элементу, который соответствует стартовой вершине дуги, будет получаться элемент, который соответствует конечной вершине этой дуги. Вес дуги равен наименьшему из весов этих правил.

Пример:

Положим операциями с нулевым весом сложение, вычитание и умножение. Рассмотрим задачу сократить дробь $\frac{P(x)*Q(x)}{Q(x)}$, P(x) и Q(x) – полиномы.

Все дроби $\frac{P'(x)}{Q'(x)}$, где $P'(x) = P(x) * Q(x)$ и $Q'(x) = Q(x)$ будут находиться в одном классе элементов с исходной дробью $\frac{P(x)*Q(x)}{Q(x)}$ и относиться к одной вершине. А полиномы, которые равны $P(x)$ будут соответствовать

другому классу, так как для преобразования к нему будет необходимо применение правила над делением, а оно имеет не нулевой вес.

Задача проверки существования пути между двумя вершинами с ограничением на максимальную длину в сокращенном орграфе решается аналогичным алгоритмом, но требует существенно меньшего объема перебора.

Главной задачей является определение относятся ли 2 выражения либо 2 утверждения к одному классу. Для этого разработан ***алгоритм*** сравнения двух выражений либо утверждений l и r без применения правил ***CWS (Compare Without Substitutions)***:

Вход: 2 выражения либо утверждения l и r; список операций с нулевым весом.

Выход: 1 если l и r относятся к одному классу; 0 иначе.

Алгоритм:

1. Заменить в l и r все вхождения операций с ненулевыми весами другими новыми переменными. При этом вхождения одних и тех же операций с равными аргументами заменяются одними и же переменными; в противном случае разными. Аргументы будут считаться равными тогда и только тогда, когда деревья их разбора будут полностью совпадать, либо они содержат только операции с нулевым весом и являются равными при проверке методом вычислительных экспериментов.
   Полученные после замены символьные выражения либо логические утверждения обозначить l' и r' соответственно.

2. Вернуть результат проверки равенства l' и r' методом на основе вычислительных экспериментов. Проверка тривиальности не требуется, так как сущности, которые получены после описанной замены, содержат исключительно операции с нулевым весом.

Разработанный алгоритм автоматической проверки

Вход: цепочка преобразования выражений либо утверждений s; взвешенное множество допустимых правил преобразования, список операций с нулевым весом.

Выход: 1 если s корректна и тривиальна; иначе 0, а также левый, и правый элементы первого некорректного либо нетривиального перехода.

Алгоритм:

1. Разбить s на переходы в сочетании с актуальными для них правилами.

2. Для всех переходов между l и r:
   2.1. Запустить алгоритм двунаправленного перебора правил BASC для l и r на сокращенном взвешенном орграфе преобразований. Для всех отбираемых суперпозиций правил:
      2.1.1. Применить суперпозицию к l и r и получить l' и r' соответственно.
      2.1.2. Заменить в l' и r' все вхождения операций с ненулевым весом новыми переменными.
      2.1.3. Проверить корректность перехода между l'' и r'' методом вычислительных экспериментов. Если переход корректен, перейти к выбору и проверке следующего перехода на шаге 2.
   2.2. Подтверждающая переход суперпозиция правил не найдена, вернуть 0, l и r.
3. Некорректные либо нетривиальные переходы не найдены, вернуть 1.

Экспериментально установлено: разработанный алгоритм позволяет проверять корректность приведенных ниже цепочек преобразования с вероятностью ошибки менее 0.1% и за время, не выходящее за одну секунду.

$$4*\cos(\frac{\pi}{6}-x)*\sin(\frac{\pi}{3}-x)=2*(\sin(\frac{\pi}{2}-2*x)+\sin(\frac{\pi}{6}))=2*(\sin(\frac{\pi}{2}-2*x)+0.5)=$$
$$=\frac{2*\sin(x)(\cos(2*x)+0.5)}{\sin(x)}=\frac{2*\sin(x)\cos(2*x)+\sin(x)}{\sin(x)}=\frac{(\sin(3*x)-\sin(x))+\sin(x)}{\sin(x)}=\frac{\sin(3*x)}{\sin(x)}$$

$$[tg(2*x)=4]$$
$$[\sin(2*x)=2*\sin(x)*\cos(x)=\frac{2*tg(x)*\cos(x)*\cos(x)}{1}=2*\mathrm{tg}(x)*\cos^2(x)=2*\mathrm{tg}(x)*\frac{1}{tg^2(x)+1}=\frac{2*\mathrm{tg}(x)}{1+\mathrm{tg}^2(x)}]$$
$$[\cos(2*x)=2*\cos^2(x)-1=2*\frac{1}{\mathrm{tg}^2(x)+1}-1=\frac{1-tg^2(x)}{1+tg^2(x)}]$$
$$\sin(4*x)+\cos(4*x)*ctg(2*x)=\sin(4*x)+\cos(4*x)/tg(2*x)=\sin(2*(2*x))+\cos(4*x)/4=$$
$$\frac{2*tg(2*x)}{1+tg^2(2*x)}+\cos(2*(2*x))/4=\frac{2*tg(2*x)}{1+tg^2(2*x)}+\frac{1-tg^2(2*x)}{1+tg^2(2*x)}/4=$$
$$=\frac{2*4}{1+4^2}+\frac{1-4^2}{1+4^2}/4=1/4$$

$$8*x+4*a=5*b+4*x-4*a^2$$
$$4*x=5*b-4*a-4*a^2$$
$$x=1.25*b-a-a^2$$

$$C(x+1,x)=k$$
$$[C(x+1,x)=(x+1)!/x!/((x+1)-x)!=(x+1)!/x!/1!=(x+1)!/x!/1=x!*(x+1)/x!/1=x+1]$$
$$x+1=k$$
$$(x+1)-1=k-1=>$$
$$x=k-1$$

В цепочках преобразований, которые решают учебные задачи на преобразование формул приемлемо ограничение в 1-2 правила на переход в зависимости от их сложности. Эта конфигурация дает возможность проверять почти все цепочки преобразования в учебных задачах за время в пределах одной секунды.

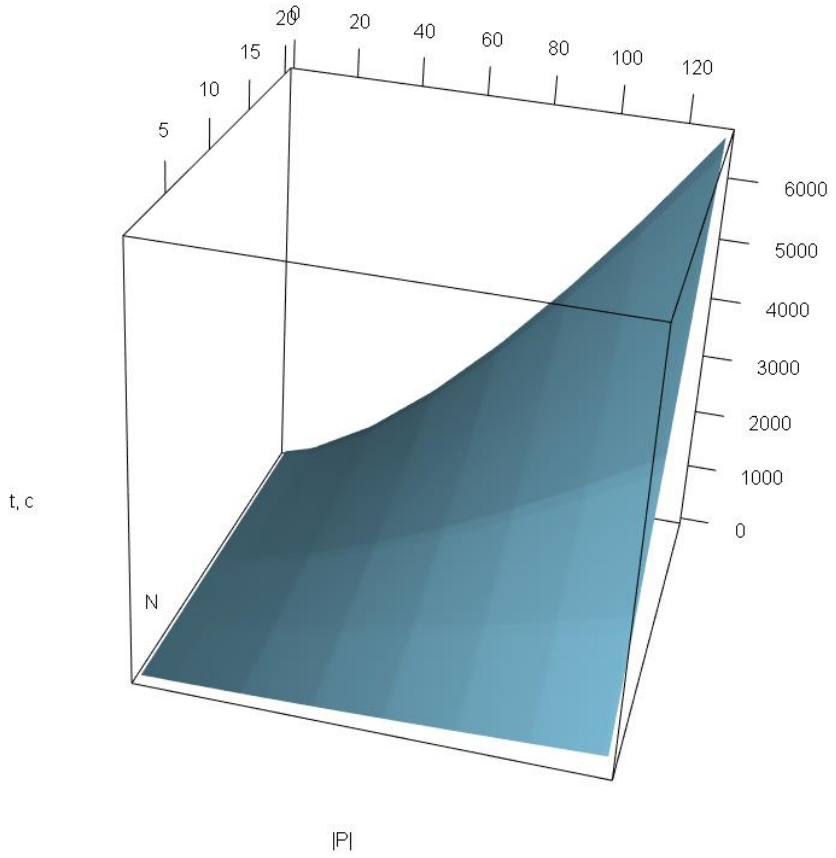

Рис. 21. Усредненная зависимость времени работы алгоритма проверки перехода t, сек от числа правил преобразования |P| и числа правил, которые можно применять за переход N

5. Опытная эксплуатация разработанных подходов

С целью проверки удобства разработанного языка и применимости алгоритма студентам СПБПУ кафедры прикладной математики в ходе 3-х лет предлагалось решать задачи по дискретной математике на созданном языке.

Пример списка предлагавшихся студентам задач:

1. Доказать, что A(m,n) = A(m-1,n) + n*A(m-1,n-1)
2. Выразить v из выражения: C(v+1,v) = k
3. Доказать, что m*C(m-1,n-1) = n*C(m,n)
4. Доказать, что V(m,n)/A(k,k-m+1) = A(m+n-1,m+n-1-k)/P(n)
5. Доказать, что S1(m,n)*C(n) = S2(m,n)*A(2*n,n-1)
6. Выразить x из выражения: V(x,2) = A(x,1)
7. Выразить z из выражения: S1(z,3) = U(3,z) - 3*U(2,z) + z
8. Выразить y из выражения: U(k,y)*C(2*y,y)/C(y) = U(k,y+1) + U(k,y)

, здесь:

- U(m,n) – число размещений с повторениями
- A(m,n) – число размещений
- P(n) – число перестановок
- C(m,n) – число сочетаний
- V(m,n) – число сочетаний с повторениями
- S1(m,n) – число Стирлинга первого рода
- S2(m,n) – число Стирлинга второго рода
- B(n) – число Белла
- C(n) – число Каталана

По итогам из решавших задачи более 95% студентов решили хотя бы одну из задач в ходе первых 20 минут. Более 80% решили хотя бы одну задачу в ходе первых 10 минут.

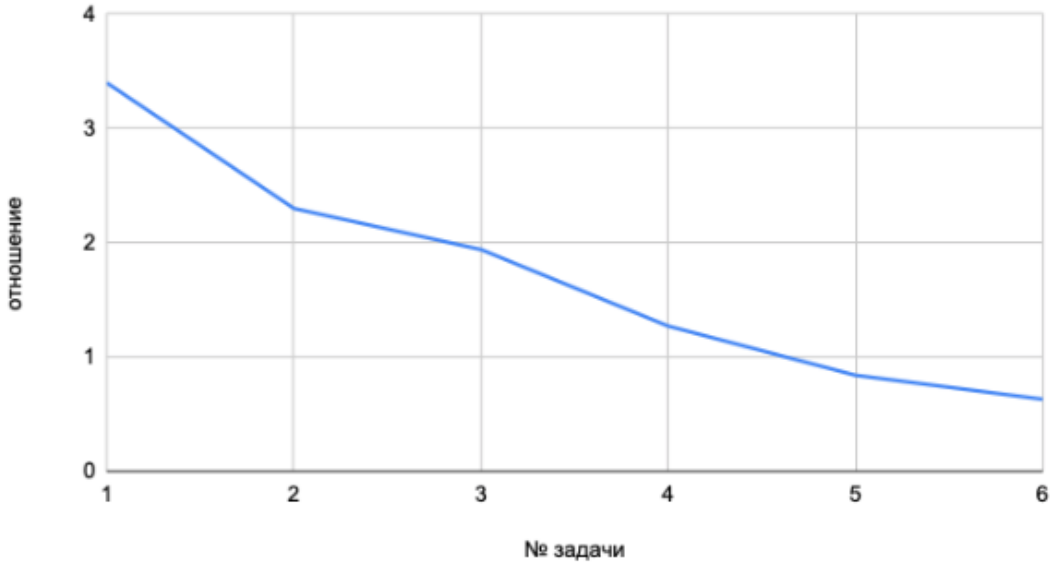

Рис. 22. Усредненная зависимость отношения времени решения задачи в разработанной нотации ко времени решения задач на бумаге от номера решаемой задачи

Отдельно проводилась оценка экономии преподавательского времени - сравнением времени, затраченного на проверку решения студентов, которые выполненны на бумаге, с временем, которое затрачено на настройку системы и обработку результатов. В итоге проверка решений «на бумаге» занимала в среднем в 4 раза больше времени, чем подготовка системы к их автоматической проверке и обработка результатов.

При этом настройка системы– разовое действие, оно может быть масштабировано на большое число студентов без дополнительных временных затрат со стороны преподавателя. Следовательно, реальная экономия преподавательского времени значительно выше.

ЗАКЛЮЧЕНИЕ

В работе разработан метод автоматической проверки решений типовых учебных задач на развитие навыков оперирования формулами. Он позволяет проверять и корректность, и тривиальность выполняемых учащимися переходов, а также удовлетворяет необходимым критериям для применения в образовательном процессе.

ЛИТЕРАТУРА

1. Айвазян С.А. и др. Прикладная статистика: основы моделирования и первичная обработка данных - 1983.

2. Андреева, Е.А. Вариационное исчисление и методы оптимизации: учеб. пособие / Е.А. Андреева, В.М. Цирулева. Тверь: ТвГУ, 2001. 576 с.
3. Антамошкин А.Н. Оптимизация функционалов с булевыми переменными / Томск: ТГУ. 1986. -129с.
4. Антипов, М. А., Бондарко, М. В., Вавилов, Н. А., Генералов, А. И., Демченко, О. В., Дыбкова, Е. В., Жуков, И. Б., Зильберборд, И. М., Карпов, Д. В., Лузгарёв, А. Ю., Пименов, К. И., Семёнов, А. А., & Шмидт, Р. А. Задачи по алгебре. Комплексные числа и многочлены / 2011
5. Ахо, А. Теория синтаксического анализа, перевода и компиляции (Тома 1-2. Синтаксический анализ) / А. Ахо, Дж. Ульман. — М. : Мир, 1978.
6. А. В. Ахо, Д. Э. Хопкрофт, Д. Д. Ульман, Структуры данных и алгоритмы / М., СПб., Киев: «Вильямс», 2001.
7. Башмаков М.И., Поздняков С.Н., Резник Н.А. Информационная среда обучения / Монография. СПб: СВЕТ, 1997.
8. Бахвалов Н.С Численные методы / Т.l. -М: Наука, 1973 г.
9. Береснев В.А., Гимади Э.К., Дементьев В.Т / Экстремальные задачи стандартизации. Новосибирск, Наука, 1978.
10. Брудно А. Л. Метод Лобачевского / Квант. № 4 (1989), С. 51—53.
11. Васильев В.Ф. Численные методы решения экстремальных задач / М., Наука, 1981.
12. Выготский Л.С. Мышление и речь / Изд. 5, испр. - М.: Лабиринт, 1999. - 352 с.
13. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация / М.: Мир, 1985.
14. Гиндикин С. Г. Алгебра логики в задачах / М. Наука 1972
15. Гинзбург С. Математическая теория контекстно-свободных языков / М. Мир. 1970.
16. Гэри М., Джонсон Д., Вычислительные Машины и Труднорешаемые Задачи / М., Мир, 1982.
17. Дамбраускас А.П. Симплекс поиск / М.: Энергия. 1979
18. Дасгупта С. и др. Алгоритмы / С. Дасгупта, Х. Пападимитриу, У. Вазирани; Пер. с англ. под ред. А. Шеня. — М.: МЦНМО, 2014 ISBN 978-5-4439-0236-4
19. Дейкстра Э. Дисциплина программирования = A discipline of programming / 1-е изд. - М.: Мир, 1978. - С. 275.

20. Демидович Б.П. Сборник задач и упражнений по математическому анализу / Учебное пособие. 13-е изд, 1997, Изд-вво Моск. ун-та, ЧеРо, ISBN 5-211-03645-X

21. Демьянов В.Ф., Васильев Л.В. Недифференцируемая оптимизация. М.: Наука, 1981.-384с.

22. Демьянков В.З., Панкрац Ю.Г., Лузина Л. Г. Универсальная грамматика. Краткий словарь когнитивных терминов / Под общ. ред. Е.С. Кубряковой. — М.: МГУ. 1997.

23. Денис Дж. Шнабель Р. Численные методы безусловной оптимизации и решения нелинейных уравнений. М.: Мир,- 1988.

24. Егорушкин О. П., Колбасина И. В., Сафонов К. В. О применении многомерного комплексного анализа в теории формальных языков и грамматик / 2017. Т. 36. Л" 2. С. 76-89.

25. Карманов В.Г. Математическое программирование / М., Наука, 1975.

26. Карпов Ю. Г. Теория и технология программирования. Основы построения трансляторов. — СПб.: БХВ-Петербург. 2005.

27. Кацман В. И., Новиков Ф. И., 2020. Автоматическая проверка цепочек преобразований над символьными фактами на основе метода перебора логических правил. XXI век: итоги прошлого и проблемы настоящего, технические науки, 2020 №3(51) Т.9, 23–28, 10.46548/21vek-2020-0952-0003.

28. Кацман В. И., Новиков Ф. И., 2020. Игрофикация процесса решения типовых учебных задач на основе выбора правил преобразования / Современная наука, актуальные проблемы теории и практики, естественные и технические науки №9 2020, 63-69, 10.37882/2223-2966.2020.09.18.

29. Кацман В. И., 2020. Автоматическая проверка цепочек преобразований символьных неравенств на основе перебора логических правил / Приборы и системы. Управление, контроль, диагностика №11 2020, 19-24, 10.25791/pribor.11.2020.1221.

30. Кацман В. И., Новиков Ф. И. 2018. Автоматизация проверки решения задач на основе перебора логических правил / Информатизация непрерывного образования (Informatization of continuing education), Москва, Материалы. международной научной конференции, 33–36.

31. Кибрик А. Е. Проблема синтаксических отношений в универсальной грамматике. Вып. XI. М.: Прогресс. 1982. С. 5-36.

32. Корбут А.А., Финкельштейн Ю.Ю. Дискретное программирование. и., Наука, 1969.

33. Кормен, Т., Лейзерсон, Ч., Ривест, Р. Алгоритмы: построение и анализ = Introduction to Algorithms / Пер. с англ. под ред. А. Шеня. — М.: МЦНМО, 2000. — 960 с. — ISBN 5-900916-37-5.

34. Манцеров Д.И., Перченок О.В., Поздняков С.Н. и др. Генерация математических задач и верификация решений в автоматизированных системах поддержки обучения / СПб - Изд-во СПбГЭТУ ЛЭТИ.-2012.-154 С.

35. Моисеев Н.Н. Элементы теории оптимальных систем. М., Наука, 1975.

36. Нестерова С.И., Скоков В.А. Численный анализ программ негладкой безусловной оптимизации / Экономика и математические методы. -1994.

37. Нестеров Ю.Е. Методы выпуклой минимизации / М.: Изд. МЦНМО, 2010.

38. Нильсон Н., Принципы искусственного интеллекта / М., Радио и связь, 1985.

39. Новиков Ф. А., Кацман В. И. 2020. Автоматическая проверка решений учебных задач на основе комбинации методов перебора логических правил и тестирования / Цифровые технологии в инженерном образовании: новые тренды и опыт внедрения (IT-Technologies for engineering education: new trends and implementation experience), Москва: Издательство МГТУ им. Н. Э. Баумана, 266-273.

40. Новиков, Ф. А. Визуальное конструирование программ / Информационно-управляющие системы. 2005. №6. 9-22.

41. Новиков, Ф. А. Дискретная математика / СПб: Питер, 2017. — 432.

42. Пападимитру Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. М.: Мир. - 1985.

43. Пентус А.Е., Пентус М.Р. Математическая теория формальных языков / М.: БИНОМ. Лаборатория знаний. - 2006.

44. Перельман И.И. Текущий регрессионный анализ и его применение в некоторых задачах автоматического управления / Изв. АН СССР, Энергетика и автоматика. -1960. -Т. XXIII, №2. -С.

45. Поздняков С.Н., Перченок О.В., Посов И.А. Автоматизация проверки решения геометрических задач по описанию их условий на предметно-ориентированном языке / Компьютерные инструменты в образовании. - СПб. - 2012. - № 1. - С. 37-44.

46. Сафонов К. В., Егорушкнн О. И. Системы полиномиальных уравнений, порождающих контекстно-свободные языки / Материалы Международной конференции «Решетнёвские чтения». Т. 2. — 2009. Красноярск. С. 535.

47. Сеа Ж. Оптимизация. Теория и алгоритмы М., Мир, 1973.

48. Серебряков В. А. [и др.]. Теория и реализация языков программирования / М. : МЗ Пресс, 2006.

49. Станкевич А.С. Общий подход к подведению итогов соревнований по программированию при использовании различных систем оценки / Компьютерные инструменты в образовании. - СПб. - 2011. - №2.

50. Стоцкий Э. Д. Управление выводом в формальных грамматиках / Проблемы передачи информации. — 1971. Т. 7. Вып. 3. С. 87—102.

51. Фаулер М. Предметно-ориентированные языки программирования / М. - СПб - Киев: Вильяме

52. Херц М. М. Энтропия языков, порождаемая автоматной или контекстно-свободной грамматиками с однозначным выводом / Научно-техническая информация. Серия 2. Вып. 4. — 1969.

53. Хомский Н. Три модели описания языка: Пер. с англ. / Кибернетический сборник. Вып. 2. — 1961. С. 237—266.

54. Хомский Н. О некоторых формальных свойствах грамматик / Кибернетический сборник. Нов. серия. Вып. 5. 1962. С. 279—311.

55. Хомский Н. Заметка о грамматиках непосредственно составляющих / Кибернетический сборник. Нов. серия. Вып. 5. — М.: Мир. 1962. С. 312-316.

56. Хомский Н. Синтаксические структуры / Новое в лингвистике. Вып. 2. С. 412-527.

57. Хомский Н., Шютценберже М.Н. Алгебраическая теория контекстно-свободных языков / Кибернетический сборник. Нов. серия. Вып. 3. — 1966. С. 195 242.

58. Ababneh, O. Y. New Newton’s Method with Third-order Convergence for Solving Nonlinear Equations / O. Y. Ababneh / International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering. — 2012. — Vol. 6, No. 1. — pp. 118—120.

59. Aitken, A. On Bernoulli’s numerical solution of algebraic equations / Proceedings of the Royal Society of Edinburgh. — 1926. — Vol. 46.

60. Alfred V. Aho. Currents in the Theory of Computing, 1973

61. Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman. The Design and Analysis of Computer Algorithms, 1974
62. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. Compilers: Principles, Techniques, & Tools / Second Edition, 2007
63. Amat S., Busquier S. Advances in Iterative Methods for Nonlinear Equations / Springer International Publishing, 2016.
64. Auvinen T. Harmful Study Habits in Online Learning Environments with Automatic Assessment / Proceedings of the 2015 International Conference on Learning and Teaching in Computing and Engineering. – 2015 – C. 50-57.
65. Back R.J., A calculus of refinements for program derivations / Acta Informatica, 25:593–624, 1988.
66. Back R.J., Sjöberg M., and Wright J., Field tests of the structured derivations method / Tech. Rpt. 491, Turku Centre for Computer Science, 2002.
67. Back R.J., Mannila L., Peltomäki M., and Sibelius P., Structured derivations: A logic based approach to teaching mathematics / In FORMED 2008: Formal Methods in Computer Science Education, Budapest, 2008.
68. Back R.J., Mannila L., and Wallin S., Student justifications in high school mathematics / In CERME 6, January 2009.
69. Back R.J., Mannila L., and Wallin S., " "It Takes Me Longer, but I Understand Better" - Student Feedback on Structured Derivations. In International Journal of Mathematical Edu- cation in Science and Technology, volume 41, pages 575–593, 2010.
70. Back R.J., Structured Derivations: a Unified Proof Style for Teaching Mathematics. Formal Aspects of Computing, 22(5):629–661, 2010.
71. Back R.J., Teaching Mathematics in the Digital Ave with Structured Derivations / Four Ferries Publishing, Abo Akademi University, 2016.
72. Bar-Hillel Y., Gaifman C., Shamir E. On the categorial and phrasestructure grammars / Bull. Res. Council Israel. Section F. — 1960. Vol. 9F. P. 1-16.
73. Brent R.P. Algorithms for minimization without derivatives / New Jersey. - Prentice -Hall Inc., Englewood Cliffs. 1973.
74. Coursera. [Электронный ресурс] – Режим доступа: https://ru.coursera.org/. Последний доступ 05.04.2020.
75. Игровые пазлы. [Электронный ресурс] – Режим доступа: www.dm.compsciclub.ru/app/list. Последний доступ 05.04.2020.

76. Dijkstra E. and Scholten C., Predicate Calculus and Program Semantics. Springer-Verlag, 1990.
77. Dijkstra E. The notational conventions I adopted, and why. Formal Aspects of Computing, 14:99 – 107, 2002.
78. Dorfler, W. Computer use and views of the mind / 1993 in Keitel.C and Rutheven.K, (eds), Learning from Computers: Mathematics Education and Technology, NATO ASI Series. Series F and Systems Sciences, Vol 121. London: Springer-Verlag.
79. Dreyfus, T. (Ed.). Imagery and reasoning in mathematics and mathematics education / 1994 Sante-Foy, Quebéc, Canada: Les presses de l'université Laval.
80. Dubinsky, E. (1991). Reflective Abstraction in Advanced Mathematical Thinking. In D. Tall (Ed.) Advanced Mathematical Thinking, Kluwer: Dordrecht.
81. EDX. [Электронный ресурс] – Режим доступа: https://www.edx.org/. Последний доступ 05.04.2020.
82. Euclidea. [Электронный ресурс] – Режим доступа: https://www.euclidea.xyz. Последний доступ 05.04.2020.
83. FineReader. [Электронный ресурс] – Режим доступа: https://finereaderonline.com. Последний доступ 05.04.2020.
84. Fedorchenko, L. Equivalent Transformations and Regularization in Context-Free Grammars / L. Fedorchenko, S. Baranov // Cybern. Inf. Technol. — 2015. — Vol. 14, no. 4. — Pp. 29–44.
85. GeoGebra. [Электронный ресурс] – Режим доступа: https://www.geogebra.org. Последний доступ 05.04.2020.
86. Hargreaves K., Abrahams P. Tex for the Impatient / Addison-Wesley, 1990, ISBN 978-0201513752.
87. Heller A. Stohastic transformations and automata / Mathemetical System Theory. - 1967. № 3. P. 68-73.
88. Hibbard N., Gray J. and Harrison M. Scan limited automata and context limited grammars / Information and Control. — 1969. V. 11. № 1. P. 5-14.
89. MapleSoft. [Электронный ресурс] – Режим доступа: https://www.maplesoft.com. Последний доступ 05.04.2020.
90. Math Helper Demo. [Электронный ресурс] – Режим доступа: https://www.mathhelper.online/. Последний доступ 05.04.2020.

91. Math Helper TWF. [Электронный ресурс] – Режим доступа: https://www.mathhelper.space/twf/prototype/demo.html. Последний доступ 05.04.2020.

92. MathML. [Электронный ресурс] – Режим доступа: https://www.w3.org/TR/MathML3/. Последний доступ 05.04.2020.

93. MathQuil. [Электронный ресурс] – Режим доступа: https://www. http://www.mathquill.com. Последний доступ 05.04.2020.

94. Matlab. [Электронный ресурс] – Режим доступа: https://www.mathworks.com/products/matlab.html. Последний доступ 05.04.2020.

95. Mathai A. An Introduction to Geometrical Probability. CRC Press, 1999, ISBN 9789056996819.

96. Mannila L. and Wallin S., Promoting students justification skills using structured derivations. In ICMI 19 studies, 2009.

97. Mathematica. [Электронный ресурс] – Режим доступа: https://www.wolfram.com/mathematica/. Последний доступ 05.04.2020.

98. Mitchell T. Machine Learning / McGraw Hill, 1997.

99. Moodle. [Электронный ресурс] – Режим доступа: https://moodle.org/. Последний доступ 05.04.2020.

100. Morrill G. Categorial grammar. Logical Syntax, Semantics, and Processing / Oxford: Oxford Univ. Press. 2011. 236 p.

101. Nesterov, Yu. Introductory Lectures on Convex Optimization / A Basic Course. Springer. 2004, ISBN 1-4020-7553-7.

102. Nesterov Yu. Barrier subgradient method / Mathematical Programming - 2011 - V.127, N1. - P.31-56.

103. Nesterov Yu. Dual extrapolation and its application for solving variational inequalities and related problems / Mathematical Programming - 2007 - V.109, N2-3. - P.319-344.

104. Nesterov Yu. Gradient methods for minimizing composite функцияs / Mathematical Programming - 2013 - V.140, N1. - P.125-161.

105. Nesterov Yu. Primal-dual subgradient methods for convex problems / Mathematical Programming - 2009 - V.120, N1. - P.261-283.

106. Nesterov Yu., Scrimali L. Solving strongly monotone variational and quasi- variational inequalities / Discrete and Continuous Dynamical Systems - 2011 - V.31, N4. - P.1383-1296.

107. Novikov, F., Katsman, V., 2018. Gamification of Problem Solving Process Based on Logical Rules / In Informatics in Schools. Fundamentals of Computer Science and Software Engineering, Pozdniakov, S. N. & Dagienė, V. (eds). Scopus, Springer International Publishing, 369–380.

108. Novikov, F., Katsman, V., Mosin, V., 2020. Automated Verification of Expression Transformation Chains Based on Computational Experiments / IOP Conference Series: Materials Science and Engineering. Scopus, 012132.

109. Öchsner A. and Makvandi R., Finite Elements for Truss and Frame Structures / An Introduction Based on the Computer Algebra System Maxima. Berlin: Springer-Verlag. 2019, DOI: 10.1007/978-3-319-94941-3

110. Pea, R. (1987). Cognitive technologies for mathematics education / In Schoenfeld, A. H. (Ed.) Cognitive science and mathematics education (pp. 89-122). Hillsdale, NJ: Erlbaum.

111. Peltomäki M. and Back R.J., An empirical evaluation of structured derivations in high school mathematics. In ICMI-19 studies, Taipei, Taiwan, 2009.

112. Pentus M. Lambek calculus is NP-complete / Theor. Comput. Sci. — 2006. V. 357. P. 186-201.

113. P.J.M. van Laarthoven, E.H.L. Aarts: Simulated Annealing: Theory and Applications / in Mathematics and its application, Springer Science Business Media Dordrecht, 1987.

114. Rall, L. B., Automatic Differentiation: Techniques and Applications. Lecture Notes in Computer Science. 120. Springer. 1981. ISBN 978-3-540-10861-0

115. Richardson D 1968, Some Unsolvable Problems Involving Elementary Functions of a Real Variable / Journal of Symbolic Logic 33 (4) 514-520 10.2307/2271358.

116. Rice H 1953 Classes of Recursively Enumerable Sets and Their Decision Problems / Transactions of the American Mathematical Society 74 (2) 358 10.2307/1990888.

117. Rogers H. The Theory of Recursive Functions and Effective Computability / MIT Press, 1987, ISBN: 0-262-68052-1; ISBN: 0-07-053522-1.

118. Rosenberg A. L. A machine realization of the linear context-free languages / Information and Control. — 1977. V. 10. P. 175—188.

119. Rosenkrants D. Matrix equations and normal forms for context-free grammars // J. Assoc. Computing Machinery. V. 14. — 1967. P. 501-507.

120. Saito K.R. and Nakano R. Second-order learning algorithm with squared penalty term. / In Advances in Neural Information Processing Systems 9. -Denver, CO. -1997

121. Stepic Math Lesson. [Электронный ресурс] – Режим доступа: https://stepik.org/lesson/9176/step/1?unit=1721.

122. SymPy. [Электронный ресурс] – Режим доступа: https://www.sympygamma.com. Последний доступ 05.04.2020.

123. R. Tibshirani, J. Friedman Introduction to Statistical Learning / ISBN-13: 978-1461471370.

124. T. Hastie, R. Tibshirani, J. Friedman The elements of Statistical Learning Data Mining, Inference, and Prediction / Springer, New York, NY 10.1007/978-0-387-84858-7.

125. Thurston, W. On proof and progress in mathematics / 1995 For the Learning of Mathematics 15(1), 20-36.

126. Tirronen M. and Tirronen V. A framework for evaluating student interaction with automatically assessed exercises / Koli Calling Proceedings of the 16th Koli Calling International Conference on Computing Education Research - 2016. – С. 180-181.

127. Valiant L. G. General context-free recognition in less than cubic time / J. Comp. Syst. Sci. - 1975. V. 10. P. 308-315.

128. Velten K., Mathematical Modeling and Simulation: Introduction for Scientists and Engineers, 2019, ISBN 3527407588

129. Verzhbitskii, V. M. One specification of Mann—Ishikawa iterations / V. M. Verzhbitskii, I. F. Yumanova // NNA’12: Fifth Conference on Numerical Analysis and Applications. Abstracts (June 15—20, 2012, Lozenets). — 2012.

130. WeBWorK. [Электронный ресурс] – Режим доступа: https://webwork.elearning.ubc.ca/webwork2/. Последний доступ 05.04.2020.

131. Weisstein, E., Power Tower [Электронный ресурс] – Режим доступа: https://mathworld.wolfram.com/PowerTower.html. Последний доступ 05.04.2020.

132. Wiris. [Электронный ресурс] – Режим доступа: https://www.wiris.com. Последний доступ 05.04.2020.

133. WolframAlpha. [Электронный ресурс] – Режим доступа: https://wolframalpha.com. Последний доступ 05.04.2020.

134. Wolfram S., An Elementary Introduction to the Wolfram Language / Wolfram Media, 2015.

135. Wolfram S., The Background and Vision of Mathematica / WolframResearch, 2011.

136. Wolfram S., The Foundation of Mathematics and Mathematica / WolframResearch, 1999.

137. Willman S., Linden R., Kaila E., Rajala T. and Laakso M. On study habits on an introductory course on programming / Computer Science Education-2015. – С. 276-291.

138. Younger D. H. Recognition and parsing of context-free languages in time / Information and Control. — 1967. V. 10. P. 598—605.

139. eMathStudio. [Электронный ресурс] – Режим доступа: https://emathstudio.com. Последний доступ 05.04.2020.