Санкт-Петербургский Политехнический Университет
Петра Великого
Институт прикладной математики и механики
Кафедра “Прикладная математика”

Алгоритм автоматической проверки правильности преобразования комбинаторных выражений

Диссертация на соискание степени магистра

Выполнил: студент группы 63601/2 В. И. Кацман
Руководитель: д. т. н., проф. Ф. А. Новиков

Автоматическая проверка решений

- Практическая деятельность - существенная часть обучения
- При решении типовых практических задач редко требуется изобретение “принципиально новой” модели
- Компетентность студентов во многом определяется степенью владения наиболее ходовым математическим аппаратом

⇓

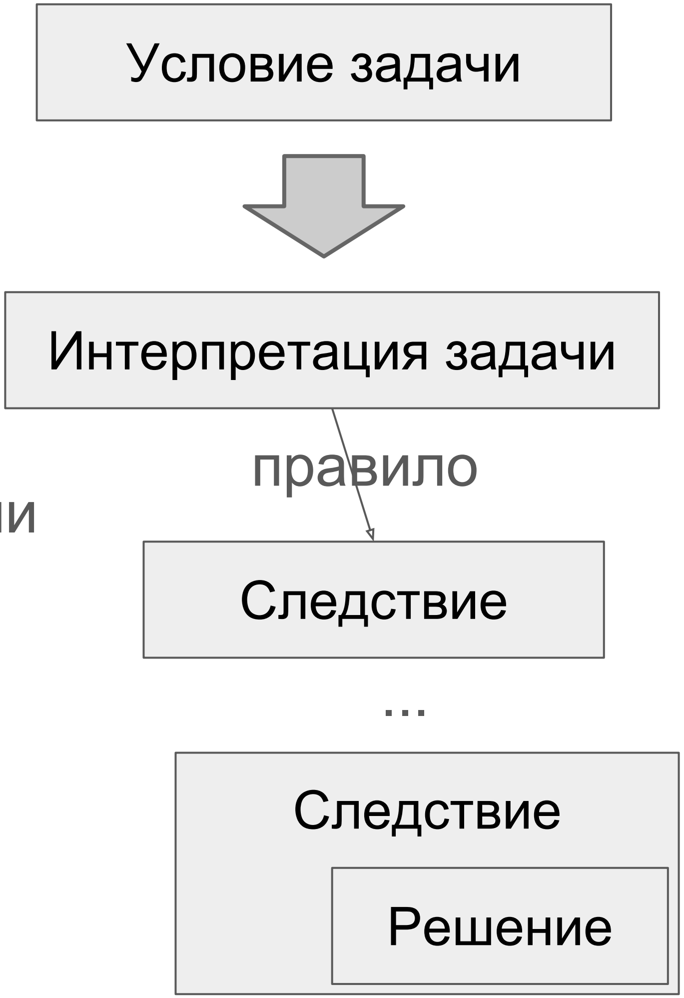
Благоприятные условия для постановки вопроса об автоматизации проверки решений типовых задач

Процесс решения задачи

1. Чтение и интерпретация условия

2. Пока решение не найдено:

   Интерпретация известной информации в виде правил, по которым можно расширять интерпретацию задачи следствиями



Проверка решения задачи

- *Корректное* расширение - расширение, соответствующее возможностям существующей теории

- *Тривиальное* расширение - разовое расширение, для которого умственных усилий у студента достаточно

Для проверки решения требуется проверить корректность и тривиальность всех расширений

Комбинаторное символьное выражение

Символьное выражение, содержащее:

- Натуральные числа, переменные
- Знаки арифметических действий (‘+’, ‘-’, ‘*’, ‘/’, ‘^’, ‘!’)
- Скобки
- Многоместные функции ($\sum_{i=m}^{n} f(i)$, $\prod_{i=m}^{n} f(i)$)
- Комбинаторные функции - функции, которые можно определить с помощью комбинаторных символьных выражений от их аргументов

Допустимые преобразования

- Множество допустимых преобразований P - множество отображений множества комбинаторных символьных выражений в себя же

- Допустимое преобразование p ∈ P:
  - Соответствует разрешенному преобразованию
  - Имеет вес w(p):
    - Для преобразований ‘+’, ‘-’, ‘*’, ‘/’: w(p) = 0
    - Для остальных преобразований: w(p) = 1

<u>Примеры</u>: p: *a + 2*(a - b) = 3*a - 2*b, w(p) = 0*
p: *C (m,n) = m! / (m - n)! / n!, w(p) = 1*

Эквивалентные выражения

- Комбинаторные символьные выражения L и R <u>N-эквивалентны</u> над P, если L и R можно привести к третьему комбинаторному символьному выражению E с помощью применения допустимых преобразований $p \in P$ с суммарным весом не более N

> <u>Пример</u>: L = *C (m,n)* и *R = m! / (m - n)! / n!* - 1-эквивалентны

- Цепочки таких преобразований L к E и R к E - <u>доказательство N-эквивалентности</u> L и R

- Комбинаторные символьные выражения L и R <u>эквивалентны</u> над P, если $\exists$ N, что L и R N-эквивалентны

Обоснование модели

- Нулевые веса преобразований ‘+’, ‘-’, ‘*’, ‘/’:
  - оперирование этими преобразованиями активно практиковалось в ходе школьного обучения
- Единичные веса остальных преобразований:
  - как раз те преобразования, которые студентами практикуются
- Выполнение N-эквивалентности:
  - обеспечивает корректность и тривиальность расширения (заключающегося в переходе от L к R)

Постановка задачи

- Дано:
  - Два комбинаторных символьных выражения L и R
  - Максимальный вес преобразований N
  - Множество допустимых преобразований P

$$L = P(n) + P(m)U(m,n-1)$$

$$R = A(n,n) + n + P(m-1)U(m,n)$$

- Требуется:
  - Проверить N-эквивалентность L и R над P
  - В случае N-эквивалентности предъявить её доказательство

$$L = ..?.. = R$$

Проблемы существующих методов

- Нормализация
  - не позволяет оценить тривиальность перехода между выражениями
  - подходит не для всех комбинаторных символьных выражений

```
L -> NF(L)
R -> NF(R)

NF(L) =?= NF(R)
```

- Сравнение значений выражений при различных значениях переменных
  - допускает ошибки только второго рода
  - не позволяет оценить тривиальность перехода между выражениями

```
for (p : значения переменных)
    if (L(p) != R(p))
        return false
return true
```

Граф преобразований G(V, E)

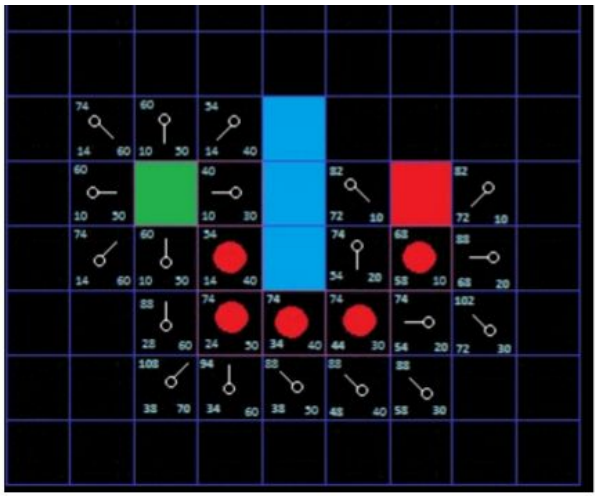
- Вершина $v \in V$:
  - соответствует некому комбинаторному символьному выражению и всем 0-эквивалентым ему выражениям

- Ребро $(u, v) \in E$:
  - соответствует наличию в P допустимого преобразования, с помощью которого выражение, соответствующее u преобразуется в выражение, соответствующее v.

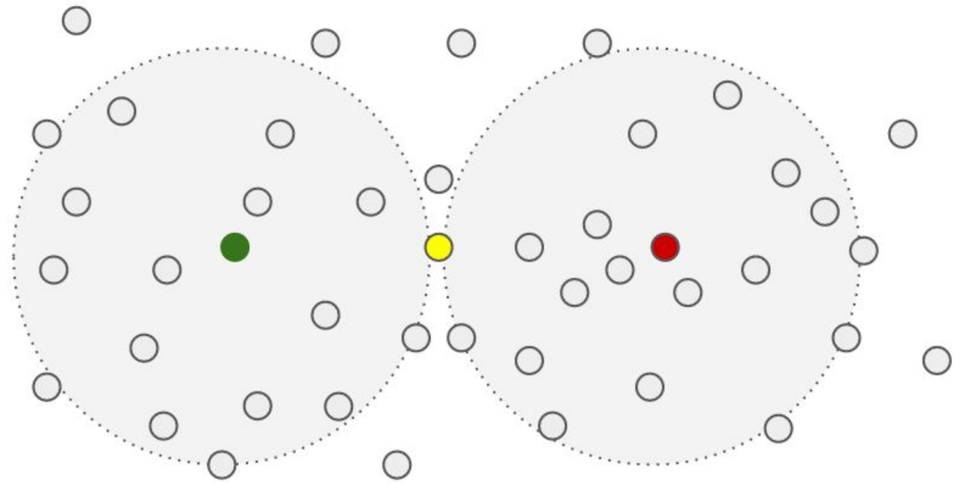
Задача проверки / доказательства N-эквивалентности L и R

Задача проверки существования / поиска пути длины не больше N в графе G между вершинами, соответствующими L и R

Идея оптимизации перебора возможных преобразований



алгоритм A*



Двусторонняя система продукций

Можно одновременно выполнять два экземпляра алгоритма A* - использовать двустороннюю систему продукций с эвристикой для предварительной оценки расстояния

Эвристика оценки расстояния между выражениями L’ и R’

1. Преобразование разности выражений (L’ - R’):
   1.1. Избавление от операций деления: домножение (L’ - R’) на все встретившиеся в разности делители
   1.2. Введение новых переменных, соответствующих операндам операций сложения, умножения, вычитания, не содержащим эти операциям (F - множество введенных переменных)
   1.3. В полученном выражении раскрываются все скобки и приводятся подобные слагаемые
   1.4. Получается многочлен K, зависящий от исходных переменных и переменных множества F
2. Расстояние между L’ и R’ (dist(L’, R’)) считается равным мощности множества F в многочлене К
   2.1. Соответствует числу различных преобразований с ненулевым весом

L’ = P(n) + u!/m!
R’ = n! + (u!+k)/m!

L’-R’ = (P(n) + u!/m!) - (n! + (u!+k)/m!)

1.1 (P(n)*m! + u!) - (n!*m! + (u!+k))

1.2 (a*b + d) - (c*b + (d+k))

1.3 a*b - c*b + k

1.4 K = a*b - c*b + k

dist(L’,R’)=|{a,b,c}|=<u>3</u>

Свойства эвристики dist (L’, R’)

- Если dist (L’, R’) = 0, то L’ и R’ эквивалентны тогда и только тогда, когда K = 0.
  - Так как, если dist (L’, R’) = 0, то разность L’ и R’ зависит только от многочлена от исходных переменных (K).

- Если dist (L’, R’) = 0 и K = 0, то L’ и R’ 0-эквивалентны.
  - Так как для преобразования L’ к R’ требуются только преобразования над операциями сложения, вычитания, умножения и деления, а веса таких преобразований равны 0.

Если исходные выражения L и R удалось свести к таким L’ и R’, что dist(L’, R’) = 0 - задачу проверки/доказательства N-эквивалентности можно считать решенной

Алгоритм проверки N-эквивалентности

АЛГОРИТМ:

1. Для всех пар вершин (u, v), соответствующих L и смежным для L вершинам, R и смежным для R вершинам соответственно:
   1.1. Если dist(u,v) = 0 - алгоритм завершает работу
2. Инициализация:
   2.1. В CL (CR) помещаем вершины, соответствующие L (R)
   2.2. В OL (OR) помещаем вершины, смежные соответствующим L (R)
3. Пока OL или OR не пустые:
   3.1. Если OL не пустая - выполняется процедура извлечения вершины из OL
   3.2. Если OR не пустая - выполняется процедура извлечения вершины из OR
4. Алгоритм завершает работу: L и R не N-эквивалентны

OL и OR - очереди открытых вершин для L и R
CL и CR - списки закрытых вершин для L и R

Процедура извлечения вершины u из очереди OL (OR):

1. Для каждой смежной u вершине v, такой что:
   a). Расстояние от v до вершины, соответствующей L(R) не больше N
   b). $v \notin OL \cup OR \cup CL \cup CR$
   1.1. prior = min(v’ ∈ CR (CL), dist(v,v’))
      1.1.1. Если при вычислении prior оказалась, что dist(v,v’) = 0 - алгоритм завершает работу
   1.2. Добавление v в OL(OR) с приоритетом, обратным prior
2. Удаление u из OL(OR); Помещение u в CL(CR)

Эффективность алгоритма

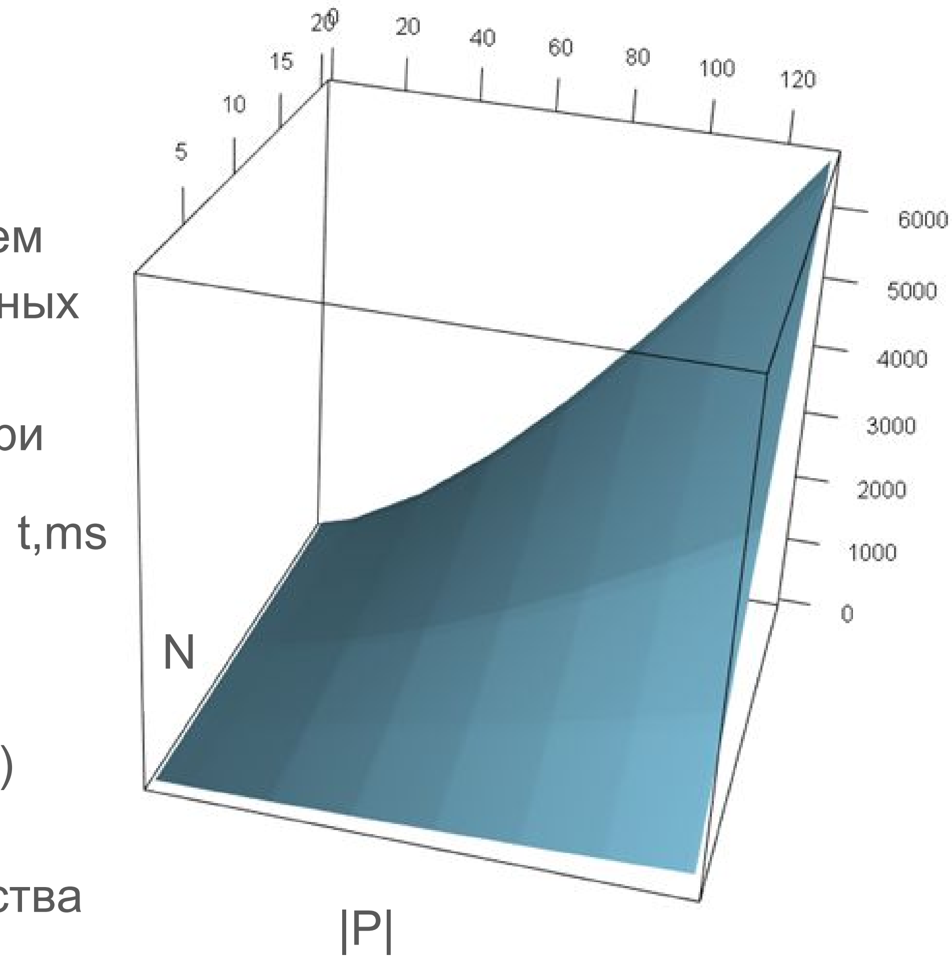
- В худшем случае: $O((2 * |P|)^{N*2})$
  - оценка следует из необходимости рассмотреть каждую вершину, расстояние от которой до L (R) не больше N, а таких вершин $O((2 * |P|)^{N})$, и для каждой из таких вершин требуется посчитать оценку расстояния еще до $O((|P|)^{N})$ вершин

- В проверяемых решениях задач большинство соседних выражений в цепочке преобразований имеют схожую структуру

На практике ожидается, что алгоритм в большинстве случаев будет завершать работу по результатам вычисления функции dist и перебирать небольшой объем вершин

Экспериментальная оценка эффективности (от N и |P|)

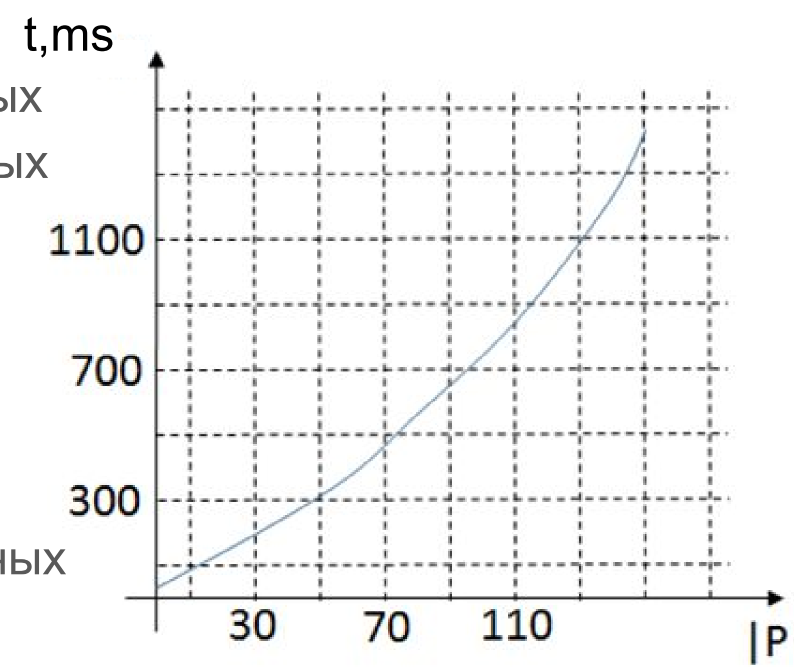
- Эксперимент
  - Данные: 100 000 пар комбинаторных символьных выражений, полученных путем случайных замен терминальных символов из пар равных выражений, используемых при описании решений задач
  - Результат: Усредненные значения t - времени работы алгоритма (в миллисекундах) для проверки N-эквивалентности для множества допустимых преобразований мощности |P|



Экспериментальная оценка эффективности (от |P| при N = 1)

- Эксперимент
  - Данные: 100 000 пар комбинаторных символьных выражений, полученных путем случайных замен терминальных символов из пар равных выражений, используемых при описании решений задач

  - Результат: Зависимость усредненных значений t - времени работы алгоритма (в миллисекундах) от мощности множества допустимых преобразований |P|



Для используемого при проверке решений задач по комбинаторике |P| = 32 в среднем, эквивалентность одной пары выражений проверяется за 2мс

Оценка возможности использования системы

- Два случая опытной эксплуатации
  - Студенты 2-го курса пишут решения предложенных им задач на языке системы в текстовых файлах, могут запускать программу, автоматически проверяющую решения, и модифицировать свои решения в соответствии с результатами проверки.

    В конце студенты получают баллы в соответствии с количеством правильно решенных задач и комментариев к проверяющей системе

  - 1-й эксперимент: 15 мин на решение задач => 7 из 12 студентов решили по 1 задаче
  - 2-й эксперимент: 90 мин на решение задач => в среднем, каждый студент решил по 4-ре задачи

Студенты смогли понять язык выражений и писать на нем решения задач

Задачи на втором эксперименте

По комбинаторике:

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 F(n+3) - F(n) = 2 * F(n+1)
4. Выразить z из выражения: S1(z,3) = U(3,z) - 3*U(2,z) + z
5. Выразить y из выражения: U(k,y)*C(2y,y)/C(y) = U(k,y+1) - U(k,y)
6. Доказать, что S1(m,n)*C(n) = S2(m-1,n-1)*A(2*n,n-1) + n*S2(m-1,n)*A(2*n,n-1)
7. Доказать, что B(m) = S(n,0,m,S1(m,n)/P(n))

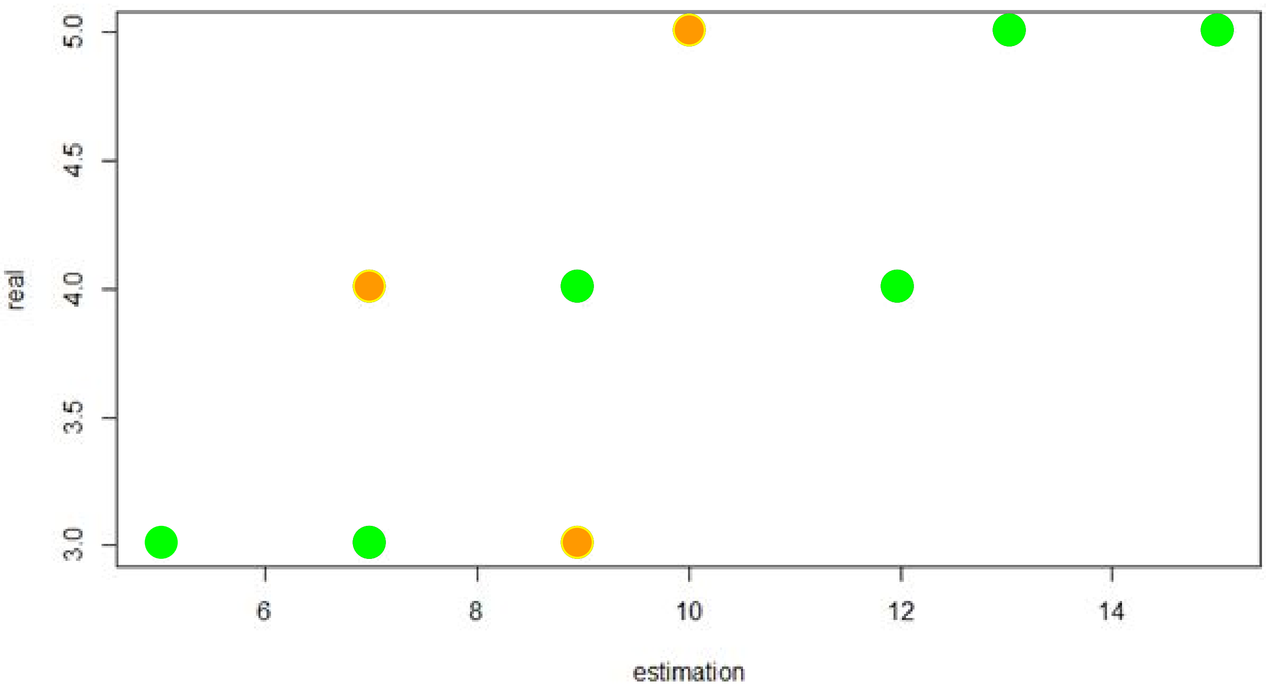
По теории множеств, булевым функциям и методу резолюций:

1. Доказать, что C(V(C(A,B),C(A,N(B))),N(A)) = O()
2. Доказать, что C(C(A,I(B, B)),N(A)) = O()
3. Доказать, что C(I(A,N(A)), I(N(A),A)) = O()
4. Доказать, что V(A,B) = V(C(A,B),V(C(N(A),B),C(A,N(B))))
5. Доказать, что C(V(A,A),C(V(N(B),A),N(A))) = O()
6. Доказать, что N(I(I(I(A,B),A),A)) = O()
7. Доказать, что S(A, B) = C(V (A, B), V (N(A), N(B)))
8. Доказать, что C(C(I(B,A), I(A,B)), C(I(B,N(A)), I(N(A),B))) = O()

Оценка возможности использования системы

- Второй эксперимент проходил 01.06; 05.06 был экзамен.

  Зависимость оценок, полученных студентами по итогам семестра (real) от баллов, полученных за решение задач на эксперименте (estimation):



Корреляция ~ 0.7 => алгоритм проверки + задачи могут являться значащим фактором при выставлении оценки за семестр

Заключение

1. Разработан и реализован алгоритм автоматической проверки эквивалентности комбинаторных символьных выражений относительно переменного набора правил, проведены оценки временной трудоемкости разработанного алгоритма

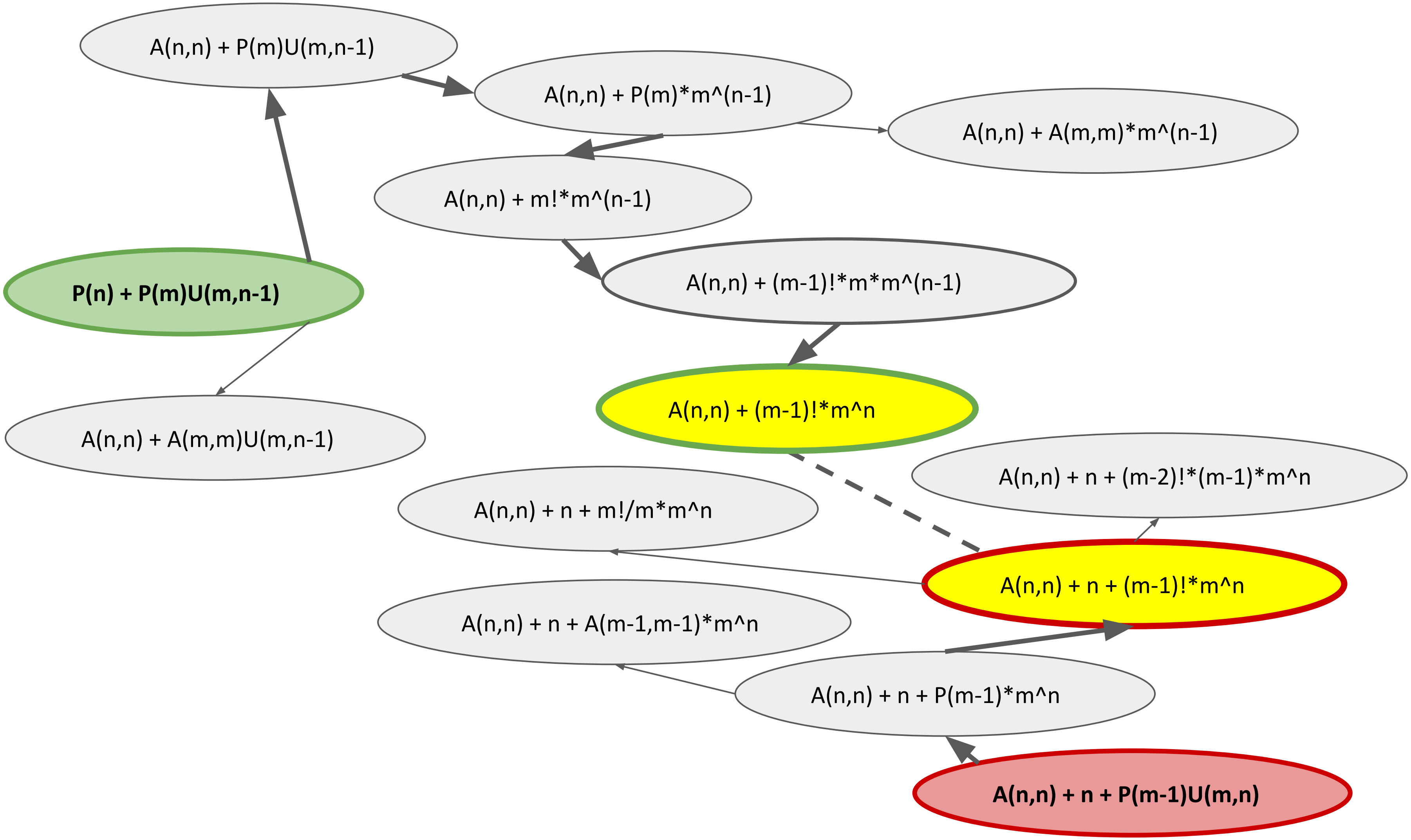
2. Предложен язык для записи решений, подобраны задачи

3. Проведены эксперименты с целью оценки возможности применения разработанной системы в образовательном процессе и степени удобства разработанного языка для описания решений задач.

ВЫВОД: алгоритм и язык описания решений подходят для использования в системе автоматической проверки студенческих решений задач

Дополнительные слайды

Пример работы алгоритма



Дальнейшие исследования

- Подбор не единичных весов преобразований
- Подбор оптимального N
- Автоматическая генерация задач
- Добавление паттернов математической логики
- Расширение поддерживаемых символьных выражений
- Продумывание и реализация удобного интерфейса

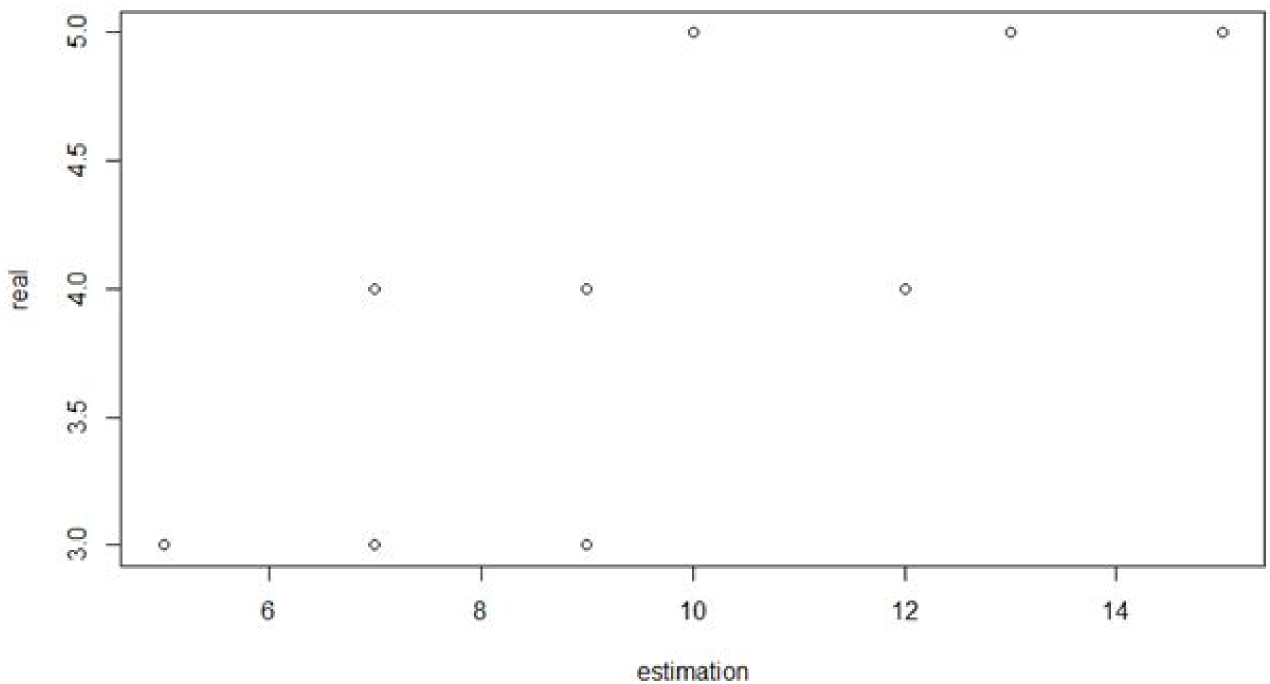
Символьные выражения в системах online-обучения

	Canvas	Stepic Coursera WeBWorK Udacity	edX	Moodle	This
Сравнение выражений с преобразованиями +, -, *, / и скобками	+	+	+		+
Визуализация выражений со стандартными математическими функциями			+		
Сравнение выражений с произвольными функциями		+	+		+
Сравнение выражений с преобразованиями над комбинаторными функциями					+
Возможность автоматической проверки решения задачи, а не только ответа					+

Оценка возможности использования системы

- Второй эксперимент проходил 01.06; 05.06 был экзамен.

  Зависимость оценок, полученных студентами по итогам семестра (real) от баллов, полученных за решение задач на эксперименте (estimation):



Корреляция ~ 0.7 => система проверки + задачи могут являться значащим фактором при выставлении оценки за семестр

Структуры данных алгоритма проверки N-эквивалентности

- Очереди открытых вершин для L и R: OL и OR.
  Списки закрытых вершин для L и R: CL и CR.

- Процедура извлечения вершины u из очереди OL (OR):

1. Для каждой смежной u вершине v, такой что:
   a). Расстояние от v до вершины, соответствующей L(R) не больше N
   b). $v \notin OL \cup OR \cup CL \cup CR$
   1.1. prior = min(v’ $\in$ CR (CL), dist(v,v’))
      1.1.1. Если при вычислении prior оказалась, что dist(v,v’) = 0 - алгоритм завершает работу
   1.2. Добавление v в OL(OR) с приоритетом, обратным prior

2. Удаление u из OL(OR); Помещение u в CL(CR)

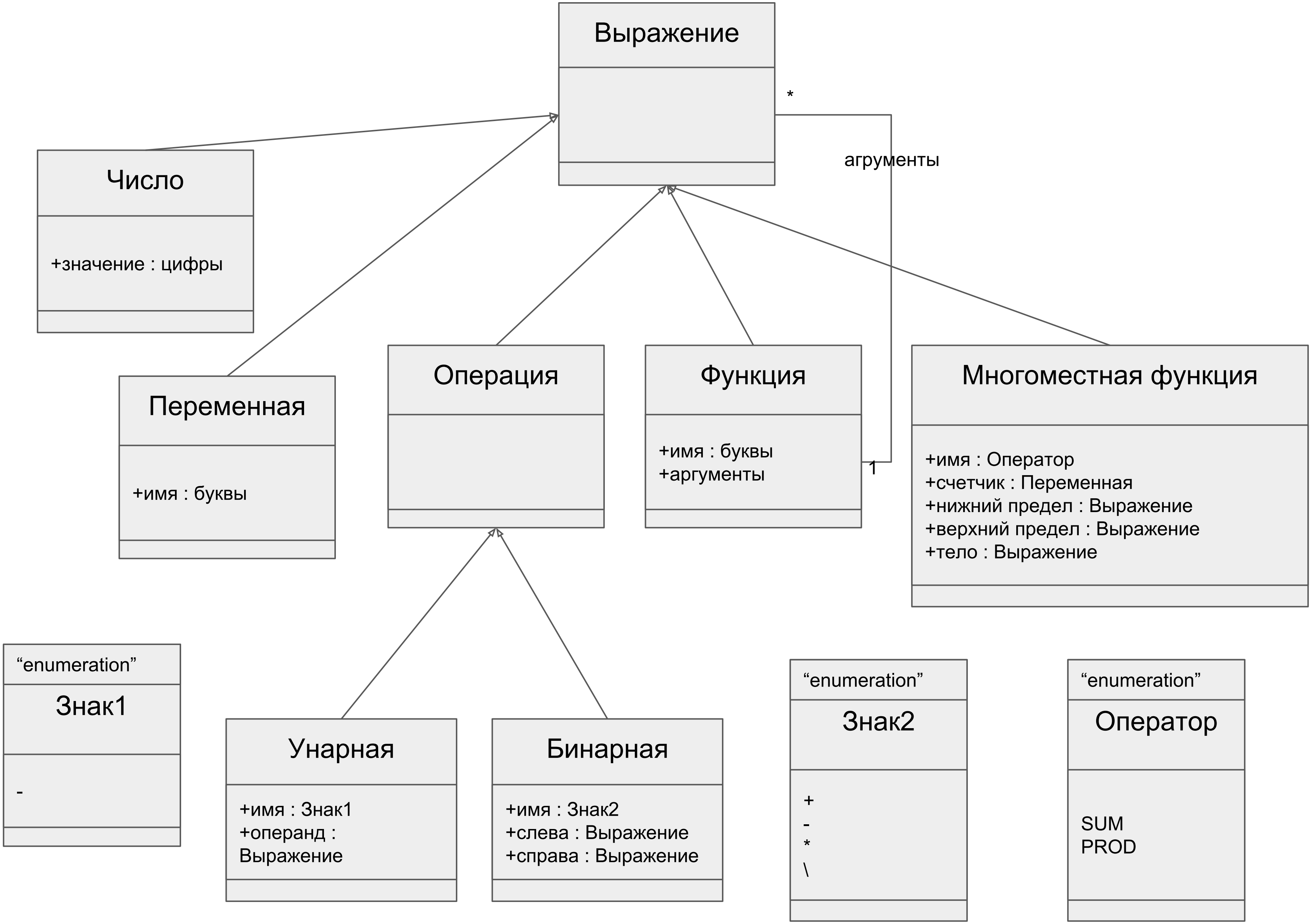
Алгоритм проверки N-эквивалентности

1. Для всех пар вершин (u, v), соответствующих L и смежным для L вершинам, R и смежным для R вершинам соответственно:
   1.1. Если dist(u,v) = 0 - алгоритм завершает работу

2. Инициализация:
   2.1. В CL (CR) помещаем вершины, соответствующие L (R)
   2.2. В OL (OR) помещаем вершины, смежные соответствующим L (R)

3. Пока OL или OR не пустые:
   3.1. Если OL не пустая - выполняется процедура извлечения вершины из OL
   3.2. Если OR не пустая - выполняется процедура извлечения вершины из OR

4. Алгоритм завершает работу: L и R не N-эквивалентны


Выражение
*
агрументы
1
Число
+значение : цифры
Переменная
+имя : буквы
Операция
Функция
+имя : буквы
+аргументы
Многоместная функция
+имя : Оператор
+счетчик : Переменная
+нижний предел : Выражение
+верхний предел : Выражение
+тело : Выражение
“enumeration”
Знак1
-
Унарная
+имя : Знак1
+операнд : Выражение
Бинарная
+имя : Знак2
+слева : Выражение
+справа : Выражение
“enumeration”
Знак2
+
-
*
\
“enumeration”
Оператор
SUM
PROD

Символьные выражения в системах online-обучения

	Canvas	Stepic	Coursera	edX	WeBWorK	Moodle	Udacity	This
Сравнение выражений с преобразованиями +, -, *, /	+	+	+	+	+		+	+
Визуализация выражений со стандартными математическими функциями				+				
Сравнение выражений с произвольными функциями		+	+	+	+		+	+
Сравнение выражений с преобразованиями над комбинаторными функциями								+
Возможность автоматической проверки решения задачи, а не только ответа								+

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 $E \in R$, то предикат «$E = 0$» неразрешим.