Введение в математическую логику и теорию алгоритмов

Лекция 10

Алексей Львович Семенов

План

- Исчисления
- Породимые множества
- Грамматики. Тезис Поста
- Алгоритмы
- Вычислимые функции
- Алгоритмы Маркова. Тезис Чёрча

Общее понятие исчисления. Предварительные определения

- Цепочка = конечная последовательность, которая может быть и пустой – Λ. Длина цепочки – число элементов в ней.
- Алфавит = конечная цепочка символов.
- Слово (в данном алфавите) – цепочка символов этого алфавита.
- Ансамбль слов в данном алфавите – все слова. Часто алфавит = {0, 1}.
- Ансамбль цепочек слов в данном алфавите – все цепочки слов.
- Ансамбль списков в данном алфавите – все списки, элементами которых являются слова или списки (индуктивное определение).
- **Задача**. Дать подробные индуктивные определения для понятий с этого экрана.

Действия и проверки. Описания

- Действие – исходное понятие. Действие:
  - слово, являющееся текстом на понятном человеку языке; может выполняться и человеком, и каким-то устройством,
  - можно применить к любому исходному данному из фиксированного ансамбля, при этом ясно, что всегда получается результат применения – элемент (возможно, другого) фиксированного ансамбля.
- Действие – задает всюду определенную функцию.
- Проверка – действие с результатом 0 или 1.
- Проверка задает характеристическую функцию множества (где она дает 1), мы говорим, что она допускает его элементы (а другие – не допускает).

Исчисления. Создаваемые объекты

- Исчисление в данном ансамбле – это пара из двух проверок:
- <проверка создания, проверка окончания>.
- Проверка создания применяется к цепочке объектов, проверка окончания – к объекту.
- Создаваемый исчислением объект определяется так:

  если проверка создания допускает цепочку объектов a_0, …, a_n и все элементы этой цепочки, кроме последнего – создаваемы, то и последний элемент создаваем.
- Если проверка создания допускает цепочку из одного элемента, то его называют начальным объектом (в некоторых контекстах – аксиомой).
- **Задача**. Что, если таких у данного исчисления нет?

Исчисления. Породимые множества

- Объект, порождаем данным исчислением, если он создаваем и его допускает проверка окончания.
- Исчисление порождает множество – из всех порождаемых им объектов – множество, порождаемое этим исчислением.
- Породимое множество – множество, порождаемое каким-то исчислением.

Эмиль Пост (11.02.1897 — 21.04.1954)

Вывод

- Фиксируем исчисление.
- Если $a_0, \ldots, a_n$ допускается проверкой создания, то говорим, что a_n создается из $a_0, \ldots, a_{n-1}$ (в данном исчислении).
- Вывод объекта a – цепочка объектов S, каждый из которых создается из какой-то цепочки объектов, встретившихся в S раньше него.
- **Задача**. Объект создаваем тогда и только тогда, когда у него имеется вывод.
- **Задача**. Пусть дано исчисление. Как организовать процесс выписывания всех выводов?
- **Задача**. Пусть дано исчисление. Как организовать процесс выписывания всех порождаемых (в нем) объектов (и только их)?

Примеры

Почему исчисление *K* модальной логики – это исчисление?

Проверка создания допускает:

- Цепочки из одной формулы (аксиомы):
  - тавтологии,
  - все формулы $(\Box (A \to B)) \to (\Box A \to \Box B)$.
- Цепочки из двух формул:

  $\langle A, \Box A \rangle$,

  $\langle A$, подстановка в A формул вместо имен$\rangle$.
- Цепочки из трех формул

  $\langle A, A \to B, B \rangle$.

Проверка окончания

Все подходит.

Задача. Описать все проверки подробно.

Задача. Почему определение формулы – это исчисление?

Теоремы замкнутости для исчислений

Т. Объединение и пересечение породимых множеств породимы.

- **Д**. Объединение.
- А: <Проверка создания А, Проверка окончания А>,
- Б: <Проверка создания Б, Проверка окончания Б>.
- **Идея**:
- Создаем слова, следуя Проверке А и следуя Проверке Б,
- Берем то, что создано или по той, или по другой проверке.
- Проблема: как не перемешивать «по ходу дела» проверки А и Б?
- Выход: Метки для объектов, создаваемых по разным проверкам: Ах, Бу. Считаем, что символы А и Б в алфавит исчисления не входят.

Продолжение. Породимость объединения

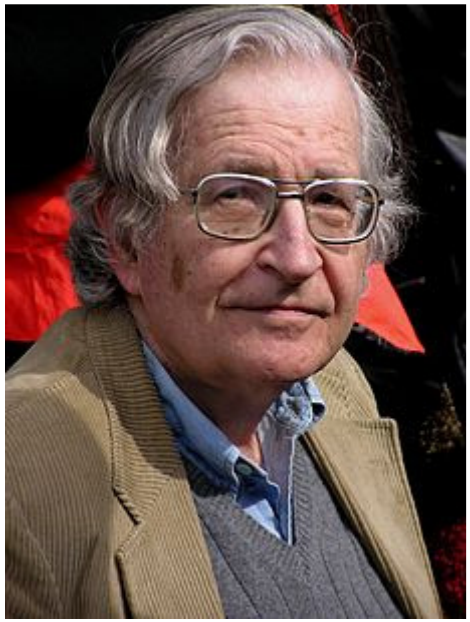
- Припишем ко всем элементам цепочки, входящей в ту или иную Проверку создания, в начале символы А или Б.
- Объединим полученные проверки.
- Проверка создания допускает еще все пары <Ах,х>, где проверка окончания А допускает х, и пары <Бх, х>, где проверка окончания Б допускает х.
- **Задача**. Проверка окончания?

Задача. Почему пересечение проверок – проверка?

Задача. Доказать теорему для пересечения.

Задача. Как обстоит дело с дополнением?

Грамматики
(Ноам Хомски, 07.12.1928 -)



Определение.

Грамматика Г – это цепочка <Σ,Ω,П,S>

- Σ – основной алфавит Г
- Ω – вспомогательный алфавит Г
- S – начальный символ Г
- Σ ∩ Ω = Ø, объединение Σ и Ω – это алфавит Г, обозначим его Δ.
- П – это конечное множество пар слов в алфавите Δ. Эти пары называются заменами.

Грамматика

определяет исчисление Г*.

Проверка создания Г* допускает:

- S
  - Всякий вывод в исчислении начинается с S.
- Для каждой замены <u,v> из П все пары вида <tup,tvp>, где t, p – произвольные слова в алфавите Δ.
  - Один шаг вывода состоит в замене в слове некоторого входящего в него u на v.
- Правило окончания для грамматики Г допускает все слова в алфавите Σ.
  - Порождаемые слова не могут содержать букв из вспомогательного алфавита.

Описание грамматики – слово в конечном алфавите.

Примеры грамматик

- В них, следуя традиции, и для наглядности, используется символ стрелочки в заменах и выводах.
- **Задача.** Как породить все цепочки из 0 и 1?
- **Решение.** Г = $\langle \Sigma, \Omega, \Pi, S \rangle$, основной алфавит $\Sigma = \{0, 1\}$, вспомогательный алфавит $\Omega = \{S\}$,

  $\Pi = \{S \to S0, \quad S \to S1, \quad S \to \Lambda\}$.

  Пример вывода: $S \to S0 \to S10 \to S010 \to S0010 \to 0010$.

- **Задача.** Как породить все десятичные числа? (Пример десятичного числа: – 3.141592.) Как породить все свободные переменные логики отношений?
- **Задача.** Что делает грамматика с основным алфавитом {*a*}, вспомогательным {*S*, *B*, *M*, *E*} и правилами
- $\Pi = \{S \to BaE,\ B \to BM,\ Ma \to aaM,\ ME \to E,\ B \to \Lambda,\ E \to \Lambda\}$?
- **Задача.** Как породить все слова, состоящие из одинакового количества букв *a* и *b* ?
- **Задача.** Построить множество, которое породить грамматикой нельзя.

Тезис Поста (вариант).

Всякое породимое множество порождается некоторой грамматикой.

Алгоритмы

- Алгоритм – описание вида
- А
- Пока Б
  - В
- Г
- где А, В, Г – действия, Б – проверка
- А - начало
- Б - продолжение
- В - переработка
- Г - извлечение результата.
- Интуитивные соображения…

Применение алгоритма

к исходному данному x – это (конечная или бесконечная) последовательность: x_0, x_1, x_2, …, x_n, …, получаемая так:

- $x_0 = x$,
- x_1 – результат применения действия А к x_0,
- Прежде чем получать следующий элемент последовательности, нужно к последнему, уже полученному, скажем, x_i , применить проверку Б.
  - Если проверка не допускает x_i , то к нему применить действие Г, результат этого применения будет последним в последовательности.
  - Если проверка допускает x_i , то к нему применить действие В, то, что получится, и будет следующим элементом последовательности x_{i+1}.
- Результат применения алгоритма к x – последний элемент последовательности.

Вычислимые функции

- Алгоритм задает (вычисляет) функцию, возможно, не всюду определенную.
- Функция, вычисляемая каким-то алгоритмом, называется вычислимой.
- Одно из основных понятий данного курса и математики.

Универсальность

- Фиксирован язык описания алгоритмов.
- Можно считать, что алфавит = {0, 1}.
- Любое слово – описание, быть может, – нигде не определённой функции.
- Универсальный алгоритм: $У (\langle P, x \rangle) = P(x)$.
- Вот этот алгоритм:

  Начало: выясняет, является ли исходное данное парой, где первый элемент – это алгоритм. Если нет – результат не определен, если да, скажем, это P, выделяем в алгоритме P его составные части и применяем **Начала** P к x.

  Продолжение: Продолжение P в применении x.

  Переработка: Переработка P в применении x.

  Извлечение результата: осуществляется соответствующим «блоком» P.

Диагонали (повторение)

В квадратной таблице выписано (по горизонтали) конечное число цепочек.

Как построить цепочку, которой нет в таблице?

1	2	8	2	3
0	3	7	3	5
7	6	0	3	9
8	2	7	5	8
3	0	6	5	0

Диагональ (конечная)

№чл.посл.	0	1	2	3	4
0	1	2	8	2	3
1	0	3	7	3	5
2	7	6	0	3	9
3	8	2	7	5	8
4	3	0	6	5	0

Взять диагональ и испортить её в каждом члене.

Например, прибавляем 1 и получаем 24161.

t(i,j) – это элемент, стоящий в i–ой строке и j–ом столбце.

Цепочка a(i) = t(i,i) +1.

Пусть она в строке c. Тогда t(c,c) = t(c,c) +1. Противоречие.

Диагональ несчетности (Кантора)

Аргумент № функц.	0	1	2	3	4	…
0	**1**	0	0	1	0	
1	1	**1**	0	1	0	
2	1	0	**0**	1	1	
3	0	0	0	**1**	0	
4	0	1	0	1	**0**	

…

• Функция, которой нет в таблице – это 1 – t(i,i), не t(i,i), нули заменили на единицы, единицы – на нули.

О словах

- Двоичные слова
- Линейный порядок:

Λ, 0, 1, 00, 01, 10, 11, 000, 001,…

- Кодирование слов в произвольном алфавите и элементов других ансамблей двоичными словами

Существуют ли невычислимые функции?

- Сколько есть вычислимых функций?
- Не больше, чем описаний.
- Не больше, чем слов.
- Счетное количество.
- Всех функций – несчетное.

Диагональ невычислимости

Имена строк и столбцов – слова.

В клетке (i,j) – значение функции с описанием i на аргументе j.
Если значения нет, оставим клетку пустой:

Аргумент Функция	0	1	01	…	1111
0				…	
1				…	
01	11	1	0111	…	01
……	…	…	…	…	…
001…	0	11	111	…	(диаг) 10

Диагональ невычислимости

Аргумент Функция	0	1	01	…	1111
0				…	
1				…	
01	11	1	0111	…	01
……	…	…	…	…	…
001…	0	11	111	…	(диаг) 10

- «Диагональная» функция t(i,i) вычислима. Вычислима и «испорченная диагональ» d(i), которая равна нулю, если t(i,i) > 0, и равна 1, если t(i,i) = 0. Кажется, что d не должно быть в таблице! Если она в строке i, то d(i,i) и ноль, и не ноль. В чем тут дело? В клетке (i) не стоит ничего!

Диагональ невычислимости

- **Задача**. Может ли функция d(i,i) быть всюду определенной?
- **Задача**. Можно ли продолжить функцию d(i,i) до всюду определенной?

Алгоритмы Маркова

- **Определение.** Описание алгоритма Маркова – это цепочка $\Phi = \langle \Sigma, \Delta, \Pi \rangle$, где
- Σ – *основной алфавит* Φ, у нас $\{0,1\}$, $\Sigma \subseteq \Delta$,
- Π – цепочка слов вида $u \to v$ или $u \to\bullet\ v$ – *замен* Φ; u, v – в алфавите Δ; $\to, \bullet \notin \Delta$,
- u называется левой частью замены, v – правой.
- Замены, содержащие $\to\bullet$, называются заключительными.

Алгоритмы Маркова

- Алгоритм?
  - Задан объект – слово
- Начало:
  - не требуется делать ничего – тождественное преобразование объекта.
- Продолжение:
  - 0, если нет ни одной замены, левая часть которой входит в объект, или в объекте есть •; иначе 1.
- Переработка:
  - Найти первую замену, левая часть которой входит в объект, найти первое ее вхождение в объект и заменить его на правую часть этой замены.
- Извлечение результата
  - Стереть в слове все символы •, которые есть.

Пример:

Алфавит из трех символов: 0, 1 и дополнительного символа | (палочки). Цепочка замен:

|0 → 0||

1 → 0|

0 →

Исходный объект: 101. Работа алгоритма:

0|01

00||1

00||0|

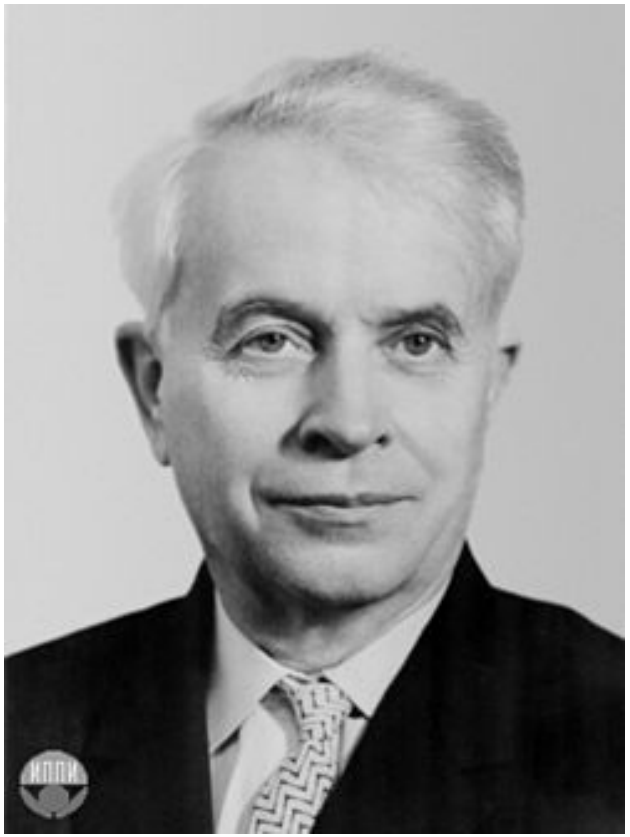
00|0|||

000|||||

00|||||

0|||||

|||||



Андрей Марков мл.

(9 [22].09.1903 – 11.10.1979)

Тезис Черча (вариант).

Всякая вычислимая функция вычислима некоторым алгоритмом Маркова.

Алонзо Чёрч (14.06.1903 — 11.08. 1995)

Алгоритмические проблемы

- Проблемы построения алгоритмов
- Проблема построения алгоритма разрешения – вычислимость характеристической функции

10-ая Проблема Гильберта

- Построить алгоритм, который по всякому алгебраическому уравнению от нескольких неизвестных с целыми коэффициентами выясняет, есть ли у него решение в целых числах.
- Отрицательное решение: Юрий Матиясевич (02.03.1947 -)