Математическая логика и теория алгоритмов

Алексей Львович Семенов

Заведующий кафедрой математической логики и теории алгоритмов
Академик Российской академии наук
и Российской академии образования

Лекция 12. Исчисления и алгоритмы

План:

- Исчисления
- Породимые множества, операции над ними
- Алгоритмы
- Вычислимость
- Перечислимые, разрешимые множества
- Исчисления и алгоритмы
- Не-вычислимость

Исчисления

Были примеры индуктивных определений: Если *A, B* - формулы, то (*A* → *B*) формула.

Фиксирован **алфавит** символов, например, {0,1}. Основные понятия обобщаются на более общие структуры, чем цепочки

Исчисление

- ***Алфавит*** расширение исходного (нам могут понадобиться дополнительные символы
- ***Правила*** - тексты, понятные человеку, реализуемые за конечное, обычно небольшое время, можно их заложить в машину, задающие:
  - Конечное число (возможно, пересекающихся) ***классов (типов)*** объектов (правило классификации)
  - ***правило порождения*** – отношение *C* между конечными множествами объектов и объектами: если выполнено *C(A,b)*, то мы говорим, что объект *b* **порождается** (**является порождением**) *A* по правилу *C*.

Заметим, что у одного конечного подмножества может быть бесконечно много порождений, обычно не так.

Множество *S* **порождаемое** данным **правилом** – это наименьшее множество *S* со следующим свойством:

- Для всякого конечного подмножества в *S* любое его порождение правилом (их может не быть) лежит в *S* (замкнутость)

Множество, **порождаемое** данным **исчислением** – каждая из частей порождаемого правилом множества, выделяемых каким-либо классом.

Породимое множество – порождаемое каким-то исчислением

Пример из логики отношений

Исчисление для порождения всех формул логики отношений

Классы:

- переменные
- имена объектов
- *n*-местные имена отношений
- атомные формулы
- формулы.

Правило порождения:

- Если … - имена объектов или переменные, … - имя *n*-местного отношения, то … атомная формула
- …

Примеры и комментарии

- Порождения пустого множества (всегда ли они есть) можно называть аксиомами, другие элементы правила порождения – ***правилами вывода***
- Примеры:
  - Формулы логики отношений, модальной логики. Высказывания логики отношений – класс формул без свободных переменных
  - Доказуемые формулы логики отношений, теории множеств
  - **Определимость**. Фиксируем конечное (для простоты) множество имен исходных отношений. Тогда все формулы, строящиеся из этих имен (как атомных), задают пространство определимости, порожденное исходным множеством.

Операции над породимыми множествами

Множество называется ***породимым***, если оно порождается некоторым исчислением.

3. Доказать, что объединение, пересечение и дополнение породимых множеств – породимы.

- Введем два новых алфавита (для симметрии – два, а не один)
- 0’, 1’…; 0”, 1”…
- Объединим правила и добавим…

Дальше мы увидим, как нужно подправить утверждение этой задачи

Вывод в исчислении

Вывод – цепочка объектов, такая, что для каждого объекта x в ней существует (конечное) множество объектов, предшествующих x, такое, что x получается из него по правилу порождения. Последний элемент цепочки называется ***результатом вывода***.

3. Объект порождаем исчислением ⇔ у него есть вывод.

Д. Лемма. Породимое правилом множество = объединение бесконечной возрастающей последовательности множеств. Каждый ее член получается добавлением к предыдущему всех порождений его конечных подмножеств.

Большая идея: каждый элемент бесконечного объединения появился на конечном шаге.

Замкнутость – ждем пока появятся все элементы конечного подмножества.

Минимальность: Первый момент, когда появилось что-то лишнее. Почему оно появилось? –… противоречие.

Алгоритмы

Алгоритм -- это

Дополнительный алфавит

Текст, понятный человеку, задающий:

- Действие ввода
- Действие переработки
- Проверка остановки
- Действие выхода

Функция, ***вычисляемая алгоритмом*** задается следующими действиями:

a) применяем к аргументу действие ввода,

b) применяем действие переработки, проверку остановки, если И, то переходим к c), если Л, то -- снова b)

c) Применяем действие вывода – получаем значение функции

Если последовательность для аргумента бесконечна, то значение функции не определено.

Вычислимая функция – вычисляемая каким-то алгоритмом

Перечислимость

3. Породимость ⇔ Перечислимость

Д. Перечислимое – породимо. Опишем Правило порождения, в которое, по существу, включена работа алгоритма

- Из пустого множества можно породить любую цепочку в алфавите 01 снабдив ее пометкой a) (напр. добавить символ a) в конце)
- Из цепочки с пометкой a) можно породить, не обращая внимания на пометку, цепочку, получающуюся применением действием ввода, после этого заменить пометку на b)
- Из цепочки с пометкой b) можно породить, не обращая внимания на пометку, цепочку, получающуюся применением действием переработки, после чего сохранить пометку b), если проверка остановки дает Л, и заменить пометку на c), если проверка остановки дает И
- Из цепочки с пометкой c) можно породить, не обращая внимания на пометку, цепочку, получающуюся применением действием выхода, и убрав пометку
- Проверяем, что если есть шаги алгоритма, то есть вывод и обратно.

Перечислимость

О. ***Перечислимое*** множество – пустое, или множество значений всюду определенной вычислимой функции.

З. Породимость ⇔ Перечислимость

Д. Всякое пустое множество породимо и перечислимо. В непустом фиксируем элемент.

Породимое – перечислимо

- Выводы можно кодировать, например, цепочками в алфавите 01
- Получив на вход объект, алгоритм пытается его раскодировать в вывод. Если это не получается, выдаем фиксированный объект. Если получается, выдаем результат вывода.

Перечислимость. Операции

3. Функция вычислима ⇔ ее график (коды пар) перечислим

→ Будем пытаться рассматривать всякое исходное данное, как код: аргумент и время работы. Если за это время результат не получится, будем выдавать в качестве значения фиксированный элемент графика. Если результат получится, то выдадим ставший известным элемент графика

← Получив на вход аргумент функции, начнем перечислять все элементы графика. Если функция для данного аргумента определена, мы получим когда-то пару, где есть нужный аргумент и значение

3. Множество перечислимо ⇔ оно является областью определения какой-либо вычислимой функции.

→ Получив исходное данное, начнем перечислять все элементы множества, если получим в какой-то момент исходное данное, заканчиваем работу.

← Получив исходное данное, пытаемся вычислить функцию. Если получается результат, исходное данное выдаем , как результатю

перебирать входы и время работы для данного алгоритма

Замкнутость перечислимости. Разрешимость

З. Замкнутость класса перечислимых множеств относительно объединения, пересечения, дополнения…

Можно доказать непосредственно, или получить из доказанного о породимых множествах

О. Множество называется разрешимым, если его характеристическая функция вычислима

У. Множество разрешимо ⇔ оно и его дополнение перечислимы.

Д. Для пустого множества очевидно.

→ Будем выдавать результат, равный исходному данному, для элементов множества, и фиксированный элемент, для элементов не из множества

← Получив исходное данное, запускаем «параллельно» (например, чередуя шаги) перечисление множества и его дополнения. Ответ о принадлежности множеству обязательно получим.

Универсальная вычислимая функция

У. Пары двоичных цепочек можно кодировать цепочками. При этом есть алгоритм, который из пары цепочек извлекает первый элемент и алгоритм, который из пары извлекает второй элемент.

У. Алгоритмы (тексты) можно кодировать цепочками в двоичном алфавите.

У. Существует алгоритм, который в применении к каждой паре $<p,x>$, где p – алгоритм, а x – цепочка, находит результат применения алгоритма p к исходному данному x.

Этот алгоритм вычисляет ***универсальную функцию*** $U(<p,x>)$.

Реальный компьютер получает программу и исходное данное и вычисляет результат...

Существуют ли невычислимые функции?

3. Сколько есть вычислимых функций?

Не больше, чем алгоритмов.

Не больше, чем текстов (цепочек символов)

- Счетное количество
- Всех функций – несчетное.

Попытаемся указать невычислимую функцию «более явно»

Как построить то, чего нет?

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

Как построить цепочку, которой нет в таблице?

Диагональ

№ \ №	1	2	3	4	5
1	1	2	8	2	3
2	0	3	7	3	5
3	7	6	0	3	9
4	8	2	7	5	8
5	3	0	6	5	0
	2	4	1	6	1

+1

Диагональ несчетности (Кантор)

Арг. / Фун.	1	2	3	4	5	…
1	1	0	0	1	0	
2	1	1	0	1	0	
3	1	0	0	1	1	
4	0	0	0	1	0	
5	0	1	0	1	0	
…	0	0	1	0	1	

отрицание

Таблица вычислимых функций

Арг. / Фун.	Λ	0	1	…	1111
Λ				…	
0				…	
1	11	1	0111	…	01
……	…	…	…	…	…
1111	0	11	111	…	10

D(x): Дописываем 0 в конце

01110 …10***0***

3. Будет ли функция D(x) вычислимой?

Диагональ невычислимости

У. Может ли функция D быть всюду определенной?

У. Можно ли продолжить функцию D до всюду определенной?

У. Можно ли продолжить функцию D до вычислимой всюду определенной?

Область определения вычислимой функции перечислима?

У. Может ли область определения функции D быть разрешимой?

Лекция 12. Исчисления и алгоритмы

Что было:

- Исчисления
- Породимые множества, операции над ними
- Алгоритмы
- Вычислимость
- Перечислимые, разрешимые множества
- Исчисления и алгоритмы
- Не-вычислимость