Лекции по математической логике и теории алгоритмов
А. Л. Семенов, 2022
Лекция 12

1 Исчисления и алгоритмы

Определение 1. *Двоичная цепочка* — цепочка в двоичном алфавите 0, 1.

При работе с исчислениями и алгоритмами объекты рассмотрения «конструктивны», к такому виду объектов относятся, например, конечные графы. Мы не будем пытаться дать общее определение конструктивного объекта и даже рассматривать что-то более общее, чем цепочки в конечном алфавите. Более того, и цепочки мы будем как правило рассматривать двоичные.

1.1 Исчисления и породимые множества

В математике, математической логике и в нашем курсе важную роль играют индуктивные определения, мы с ними уже встречались. Индуктивные определения могут использоваться при работе с различными классами объектов, в том числе и с произвольными множествам. Для нас, однако, важен случай, когда эти объекты «конструктивны», о чем мы уже говорили выше.

Определение 2. *Исчисление* — это однозначно понятный человеку текст, включающий:

- *алфавит исчисления*, включающий исходный алфавит 0,1
- множество *имен классов* (обычно конечное); значениями имен являются (возможно, пересекающиеся) классы (типы) объектов.
- *Правило порождения* — отношение G между конечными множествами объектов, с указанием их классов, и объектами, с указанием их класса: если выполнено $G(A, b)$, то мы говорим, что объект b порождается (является порождением) множества A по правилу G.

  реализация (выяснение выполненности) Правила порождения занимает обязательно конечное (как правило, небольшое для небольших объектов) время; можно эту реализацию поручить машине.

  Заметим, что одно конечное множество может порождать бесконечно много объектов.

Множество, *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).

Определение 3. Множество объектов данного класса,*порождаемое* данным *исчислением*— множество объектов этого класса, порождаемых правилом исчисления.

Определение 4. *Породимое* множество – порождаемое каким-то исчислением.

Нетрудно сообразить, что разные данные ранее в курсе индуктивные определения описывают исчисления.

Задача 1. Укажите исчисление, порождающее все формулы логики отношений

Подсказка. Обратитесь к определению формулы из Лекции 3 Логические языки.

Введя новый класс объектов: множеств, функций и т.д. математики часто задаются вопросом о замкнутости этого класса относительно естественных операций. Вот пример такого вопроса:

Определение 5. Порождения пустого множества (всегда ли они есть?) можно называть *аксиомами*, другие элементы правила порождения — *правилами вывода*

Задача 2. Будут ли объединение, пересечение и дополнение породимых множеств – породимы?

Подсказка. Пытаясь построить исчисление для объединения и пересечения можно пытаться просто «смешать» два исчисления для двух породимых множеств и «потом разбираться». Однако при таком подходе возникает трудность: порождая новые объекты мы можем иногда использовать правило первого исчисления, иногда правило второго. Нужно как-то не смешивать, а «разделить» порождение в двух исчислениях.
В случае дополнения ситуация, кажется, еще хуже. Как можно породить то, что породить нельзя?

Пусть фиксировано некоторое исчисление.

Определение 6. *Вывод* (в данном исчислении) – 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.*

Задача 3. Доказать, что объект порождаем исчислением $\Leftrightarrow$ у него есть вывод.

Подсказка. Воспользуйтесь определением порождаемости, где есть требование замкнутости и минимальности. Попробуйте также использовать большую идею объединения возрастающей цепочки множеств.

1.2 Алгоритмы и вычислимые функции

Определение 7. *Алгоритм* — это однозначно понятный человеку текст, включающий *алфавит*, содержащий двоичный, цепочки в этом алфавите будут объектами алгоритма, и задающий (описывающий) обязательно завершающиеся (обычно за небольшое по сравнению с обозрением объекта время):

- *Проверку остановки*, дающую в применение к объекту И или Л;
- *Действие переработки*, дающее в применении к объекту другой объект.

Вычисление алгоритма — последовательность объектов:

- первый объект последовательности – произвольная двоичная цепочка – *исходное данное*,
- для всякого элемента x последовательности, если последовательность конечна, то кроме последнего:
  - если Проверка остановки для x дает значение И, то следующий элемент последовательности является последним в последовательности и называется *результатом вычисления.*
  - следующий элемент последовательности существует и получается из x Действием переработки.

Переход от объекта последовательности к следующему мы называем *шагом (работы) алгоритма.*

Таким образом, прежде чем действовать, мы выясняем, не нужно ли будет, сделав очередной шаг, завершить деятельность. На самом первом шаге мы можем добавить к аргументу дополнительные символы, заготовить несколько его копий, как частей объекта, запустить какие-то вспомогательные алгоритмы, тоже описанные в Действии переработки и т.д. Зная, что очередной шаг будет последним, мы убираем все, что не войдет в результат.

Определение 8. *Функция, вычисляемая алгоритмом* — это отображение, ставящее в соответствие исходному данному результат вычисления алгоритма, начинающегося с этого исходного данного.
Если последовательность для какого-то исходного данного бесконечна, то результата нет и значение функции на таком аргументе не определено.

Определение 9. *Вычислимая функция* — функция, вычисляемая каким-либо алгоритмом.

Задача 4. Существует алгоритм со следующим свойством:
В последовательности, начинающейся с пустой цепочки, где каждый следующий элемент получается применением алгоритма к предыдущему, встречаются все двоичные цепочки.

Подсказка. В случае натуральных чисел алгоритм мог бы просто добавлять единицу.

Определение 10. Построенный в предыдущей задаче алгоритм назовем *последовательным алгоритмом*.

Задача 5. Пары двоичных цепочек можно кодировать двоичными цепочками. При этом есть алгоритм, который из кода пары цепочек извлекает первый элемент пары и алгоритм, который из кода пары извлекает второй элемент пары.

Определение 11. Строящийся в этой задаче алгоритм назовем *алгоритмом кодирования пар*. Зафиксировав какой-то алгоритм кодирования, *код* пары цепочек a, b будем обозначать $[a, b]$.

Подсказка. Можно обходить бесконечную двухмерную таблицу.

Определение 12. *Перечислимое множество*: пустое, или являющееся множеством всех значений всюду определенной вычислимой функции.

Задача 6. Множество перечислимо $\Leftrightarrow$ оно породимо.

Подсказка. Попробуем установить соответствие между правилом порождения и действием переработки.

Задача 7. Объединение и пересечение перечислимых множеств перечислимы.

Подсказка. Строящийся алгоритм, конечно, каким-то образом должен запускать алгоритмы для каждого из множеств.

Конечно, утверждение задачи вытекает из предшествующих результатов о породимых множества и из эквивалентности породимости и перечислимости.

Определение 13. Множество называется *разрешимым*, если его характеристическая функция вычислима.

Задача 8. Множество разрешимо $\Leftrightarrow$ оно и его дополнение перечислимы.

Подсказка. Имея алгоритм для «разрешения», то есть вычисляющий характеристическую функцию, надо построить два алгоритма для перечисления.
Имея два алгоритма для перечисления, надо построить алгоритм для характеристической функции.

Задача 9. Функция вычислима $\Leftrightarrow$ ее график перечислим.
Область определения любой вычислимой функции перечислима.

Подсказка. Мы можем дождаться окончания любого процесса, лишь бы он закончился.

Алгоритмы – это тексты в некотором алфавите. К кодированию текстов мы уже прибегали, обсуждая теорему Гёделя о неполноте. Фиксируем некоторый способ кодирования всех текстов в этом алфавите цепочками в двоичном алфавите.

Определение 14. Функция называется *универсальной*, если она в применении к каждому коду $[p, x]$ пары $< p, x >$, где p — код алгоритма, а x — цепочка, находит результат применения алгоритма с кодом p к исходному данному x.

Задача 10. Универсальная функция вычислима.

Подсказка. Если вы поняли предшествующий текст, в частности, определение универсальной функции, то вы можете ее вычислять.

Определение 15. Определим "диагональную" функцию D равенством:

$$D(x) = U([x, x])0.$$

Тогда D — функция, вычисляемая некоторым алгоритмом. Обозначим код этого алгоритма (двоичную цепочку) через d.

Наша диагональная функция напоминает о Канторовой диагонали и Гёделевой диагонали. Дадим строкам и столбцам бесконечной таблицы имена – двоичные цепочки. В клетки поместим значение функции с кодом в строке на аргументе - имени столбца клетки. Тогда D получается «порчей» – дописыванием нуля к значениям, стоящим на диагонали таблицы.

Задача 11. Может ли функция D быть всюду определенной?

Подсказка. Чему равно $D(d)$?

Задача 12. Можно ли продолжить функцию D до всюду определенной?

Подсказка. А почему бы и нет?

Задача 13. Можно ли продолжить функцию D до вычислимой всюду определенной?

Подсказка. Решение для диагональной функции с пустой клеткой здесь не проходит.

Задача 14. Может ли область определения функции D быть разрешимой?

Подсказка. Если бы область определения была разрешимой, можно было бы доопределить D до всюду определенной.

Напомним, что область определения D перечислима, как у всякой вычислимой функции.

Таким образом мы построили не разрешимое перечислимое множество.

Способ построения такого множества напоминает способ доказательства несчетности множества последовательностей нулей и единиц (другими словами — характеристических функций подмножеств натурального ряда). Он также напоминает способ доказательства не определимости истины. Можно сказать, что это все — примеры большой идеи **Диагонали**.