Теория алгоритмов
Вычислимость

Математическая логика и теория алгоритмов

Алексей Романов

20 декабря 2023 г.

МИЭТ

Алгоритмы и программы

- Алгоритм задаётся программой на каком-то абстрактном языке программирования.
- Язык позволяет работать с натуральными числами произвольной величины.
- Или другими объектами, задаваемыми конечными словами в конечном алфавите (элементами какого-то *дискретного множества*).
- Не взаимодействует с внешним миром, кроме получения аргумента/ов и возвращения результата.
- Не использует случайных чисел и других источников недетерминированности.
- Конкретные примеры таких языков в конце лекции.

Функции, вычисляемые алгоритмами

- Каждый алгоритм (программа) тогда вычисляет какую-то функцию.
- Эта функция может быть не везде определена (если алгоритм не останавливается или останавливается, не выдав осмысленного результата).
- В этом разделе функции по умолчанию могут быть частичными, всюду определённость оговаривается особо.
- Мы говорим в первую очередь о функциях $\mathbb{N} \to \mathbb{N}$, но областями определения и значений могут быть и любые другие дискретные множества.
- Разные алгоритмы могут вычислять одну и ту же функцию.

Вычислимые функции

- Функция *вычислима*, если какой-то алгоритм её вычисляет.
- Например, нигде не определённая функция вычислима

Вычислимые функции

- Функция *вычислима*, если какой-то алгоритм её вычисляет.
- Например, нигде не определённая функция вычислима алгоритмом, который зацикливается на любом входе.

Вычислимые функции

- Функция *вычислима*, если какой-то алгоритм её вычисляет.
- Например, нигде не определённая функция вычислима алгоритмом, который зацикливается на любом входе.
- Или функция $x \mapsto x^2$.
- Зависит ли это от языка?

Вычислимые функции

- Функция *вычислима*, если какой-то алгоритм её вычисляет.
- Например, нигде не определённая функция вычислима алгоритмом, который зацикливается на любом входе.
- Или функция $x \mapsto x^2$.
- Зависит ли это от языка? Скажем, если там нет операции умножения?

Вычислимые функции

- Функция *вычислима*, если какой-то алгоритм её вычисляет.
- Например, нигде не определённая функция вычислима алгоритмом, который зацикливается на любом входе.
- Или функция $x \mapsto x^2$.
- Зависит ли это от языка? Скажем, если там нет операции умножения?
- Есть невычислимые функции $\mathbb{N} \to \mathbb{N}$

Вычислимые функции

- Функция *вычислима*, если какой-то алгоритм её вычисляет.
- Например, нигде не определённая функция вычислима алгоритмом, который зацикливается на любом входе.
- Или функция $x \mapsto x^2$.
- Зависит ли это от языка? Скажем, если там нет операции умножения?
- Есть невычислимые функции $\mathbb{N} \to \mathbb{N}$ просто потому, что всех таких функций $\mathfrak{c}$, а вычислимых $\aleph_0$.
- Из простых вычислимых функций можно строить более сложные. Например, композиция вычислимых функций вычислима.

Разрешимые множества

- Множество $A \subseteq \mathbb{N}$ *разрешимо*, если есть алгоритм, позволяющий проверить принадлежность к нему любого натурального числа: получает на вход n и выдаёт 1, если $n \in A$ и 0, если $n \notin A$.
- То есть вычислима характеристическая функция $\chi_A(n) =$ если $n \in A$ то 1 иначе 0.

Разрешимые множества

- Множество $A \subseteq \mathbb{N}$ *разрешимо*, если есть алгоритм, позволяющий проверить принадлежность к нему любого натурального числа: получает на вход n и выдаёт 1, если $n \in A$ и 0, если $n \notin A$.
- То есть вычислима характеристическая функция $\chi_A(n) =$ если $n \in A$ то 1 иначе 0.
- Может быть известно, что множество разрешимо (или что функция вычислима), но неизвестно, как именно. Классический пример: $A = \{n \mid$ в десятичной записи π есть n нулей подряд$\}$.

Разрешимые множества

- Множество $A \subseteq \mathbb{N}$ *разрешимо*, если есть алгоритм, позволяющий проверить принадлежность к нему любого натурального числа: получает на вход n и выдаёт 1, если $n \in A$ и 0, если $n \notin A$.
- То есть вычислима характеристическая функция $\chi_A(n) =$ если $n \in A$ то 1 иначе 0.
- Может быть известно, что множество разрешимо (или что функция вычислима), но неизвестно, как именно. Классический пример: $A = \{n \mid$ в десятичной записи π есть n нулей подряд$\}$.
- Утверждение: объединение, пересечение и дополнение разрешимых множеств разрешимы.
- Утверждение: декартово произведение разрешимых множеств разрешимо (как подмножество $\mathbb{N} \times \mathbb{N}$).

Перечислимые множества

- Множество $A \subseteq \mathbb{N}$ *перечислимо*, если есть алгоритм, позволяющий подтвердить принадлежность к нему, но не опровергнуть: получает на вход n и выдаёт 1, если $n \in A$ и ничего не выдаёт, если $n \notin A$.
- То есть вычислима полухарактеристическая функция $\bar{\chi}_A(n) =$ если $n \in A$ то 1 иначе не определена.
- Все следующие утверждения эквивалентны:
    - A перечислимо.
    - $A = \{n : \mathbb{N} \mid f(n)$ определена$\}$ для какой-то вычислимой f.
    - $A = \{f(n) \mid n : \mathbb{N}\}$ для какой-то вычислимой f.
    - Есть алгоритм без входа, перечисляющий элементы A.
- Утверждение: объединение и пересечение перечислимых множеств перечислимы.
- Утверждение: декартово произведение перечислимых множеств перечислимо.
- Утверждение: если множество и его дополнение перечислимы, то оно разрешимо.

Универсальные вычислимые функции

- Пусть есть какое-то множество $X \subseteq \mathbb{N} \to \mathbb{N}$ функций одного аргумента.
- Функция двух аргументов $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ называется *универсальной для X*, если $X = \{m \mapsto f(n, m) \mid n : \mathbb{N}\}$.
- *Универсальная вычислимая функция* это вычислимая функция двух аргументов, универсальная для класса всех вычислимых функций одного аргумента.
- Теорема: существует универсальная вычислимая функция.
- Доказательство: перенумеруем все программы нашего языка, например, в порядке возрастания длины, а при одинаковой длине по алфавиту: $p_0, p_1, \ldots$
- Определим $U(n, m) = p_n(m)$ (то есть считаем программы, пока дойдём до p_n, и запускаем на аргументе m).

Универсальные и всюду определённые функции

- Теорема: не существует всюду определённой вычислимой функции, универсальной для класса всюду определённых вычислимых функций одного аргумента.
- Доказательство: предположим, что такая функция есть, и обозначим $V(n, m)$. Определим $V'(n) = V(n, n) + 1$.

Универсальные и всюду определённые функции

- Теорема: не существует всюду определённой вычислимой функции, универсальной для класса всюду определённых вычислимых функций одного аргумента.
- Доказательство: предположим, что такая функция есть, и обозначим $V(n, m)$. Определим $V'(n) = V(n, n) + 1$. Она не совпадает ни с одним сечением V (почему?).

Универсальные и всюду определённые функции

- Теорема: не существует всюду определённой вычислимой функции, универсальной для класса всюду определённых вычислимых функций одного аргумента.
- Доказательство: предположим, что такая функция есть, и обозначим $V(n,m)$. Определим $V'(n) = V(n,n) + 1$. Она не совпадает ни с одним сечением V (почему?).
- Почему это доказательство не работает для класса всех вычислимых функций и функции U?

Универсальные и всюду определённые функции

- Теорема: не существует всюду определённой вычислимой функции, универсальной для класса всюду определённых вычислимых функций одного аргумента.
- Доказательство: предположим, что такая функция есть, и обозначим $V(n, m)$. Определим $V'(n) = V(n, n) + 1$. Она не совпадает ни с одним сечением V (почему?).
- Почему это доказательство не работает для класса всех вычислимых функций и функции U? $U(n, n)$ может быть не определена, тогда нельзя доказать, что $U' \neq m \mapsto U(n, m)$.

Универсальные и всюду определённые функции

- Теорема: не существует всюду определённой вычислимой функции, универсальной для класса всюду определённых вычислимых функций одного аргумента.
- Доказательство: предположим, что такая функция есть, и обозначим $V(n, m)$. Определим $V'(n) = V(n, n) + 1$. Она не совпадает ни с одним сечением V (почему?).
- Почему это доказательство не работает для класса всех вычислимых функций и функции U? $U(n, n)$ может быть не определена, тогда нельзя доказать, что $U' \neq m \mapsto U(n, m)$.
- Теорема: существует вычислимая функция, не имеющая всюду определённого вычислимого продолжения.
- Доказательство: $U'(n)$ как раз такая.

Неразрешимость проблемы остановки

- Теорема: существует перечислимое неразрешимое множество (имеющее неперечислимое дополнение).

Неразрешимость проблемы остановки

- Теорема: существует перечислимое неразрешимое множество (имеющее неперечислимое дополнение).
- Доказательство: это область определения $U'(n)$.

Неразрешимость проблемы остановки

- Теорема: существует перечислимое неразрешимое множество (имеющее неперечислимое дополнение).
- Доказательство: это область определения $U'(n)$.
- Не существует алгоритма, который по произвольной программе p и аргументу n определял бы, закончится ли вычисление $p(n)$.
- Другими словами, область определения универсальной вычислимой функции U неразрешима.

Неразрешимость проблемы остановки

- Теорема: существует перечислимое неразрешимое множество (имеющее неперечислимое дополнение).
- Доказательство: это область определения $U'(n)$.
- Не существует алгоритма, который по произвольной программе p и аргументу n определял бы, закончится ли вычисление $p(n)$.
- Другими словами, область определения универсальной вычислимой функции U неразрешима.
- Доказательство: если она разрешима, то определим такую функцию: $U''(n) =$ если $U(n,n)$ определена, $U(n,n)+1$, иначе 0. Получается, что она:
  1. вычислима;
  2. всюду определена;
  3. продолжение U'.

  Противоречие!

Теорема (Успенского–)Райса

- Свойство программ P называется *семантическим*, если оно определяется только функцией, которую вычисляет программа.
- То есть если две программы p_1 и p_2 обе вычисляют одну и ту же функцию, то $P(p_1) \Leftrightarrow P(p_2)$.

Теорема (Успенского–)Райса

- Свойство программ P называется *семантическим*, если оно определяется только функцией, которую вычисляет программа.
- То есть если две программы p_1 и p_2 обе вычисляют одну и ту же функцию, то $P(p_1) \Leftrightarrow P(p_2)$.
- Например, «Программа останавливается на любых входных данных» это

Теорема (Успенского–)Райса

- Свойство программ P называется *семантическим*, если оно определяется только функцией, которую вычисляет программа.
- То есть если две программы p_1 и p_2 обе вычисляют одну и ту же функцию, то $P(p_1) \Leftrightarrow P(p_2)$.
- Например, «Программа останавливается на любых входных данных» это семантическое свойство.
- А «длина программы больше 100 символов»

Теорема (Успенского–)Райса

- Свойство программ P называется *семантическим*, если оно определяется только функцией, которую вычисляет программа.
- То есть если две программы p_1 и p_2 обе вычисляют одну и ту же функцию, то $P(p_1) \Leftrightarrow P(p_2)$.
- Например, «Программа останавливается на любых входных данных» это семантическое свойство.
- А «длина программы больше 100 символов» нет.

Теорема (Успенского–)Райса

- Свойство программ P называется *семантическим*, если оно определяется только функцией, которую вычисляет программа.
- То есть если две программы p_1 и p_2 обе вычисляют одну и ту же функцию, то $P(p_1) \Leftrightarrow P(p_2)$.
- Например, «Программа останавливается на любых входных данных» это семантическое свойство.
- А «длина программы больше 100 символов» нет.
- Свойство нетривиально, если есть программы, для которых оно истинно, и программы, для которых оно ложно.
- Теорема Райса: любое семантическое нетривиальное свойство программ неразрешимо.

Теорема (Успенского–)Райса

- Свойство программ P называется *семантическим*, если оно определяется только функцией, которую вычисляет программа.
- То есть если две программы p_1 и p_2 обе вычисляют одну и ту же функцию, то $P(p_1) \Leftrightarrow P(p_2)$.
- Например, «Программа останавливается на любых входных данных» это семантическое свойство.
- А «длина программы больше 100 символов» нет.
- Свойство нетривиально, если есть программы, для которых оно истинно, и программы, для которых оно ложно.
- Теорема Райса: любое семантическое нетривиальное свойство программ неразрешимо.
- Проблема остановки это частный случай: свойство «программа p даёт результат при запуске с аргументом 0» семантическое и нетривиальное.

Простые модели вычислений

- Мы пока почти ничего не говорили про конкретные языки.
- Для доказательств часто оказывается удобно работать не с привычными C, Python и т.д., а со специальными простыми языками.
- Особенно при доказательстве неразрешимости каких-то задач или невычислимости каких-то функций

Простые модели вычислений

- Мы пока почти ничего не говорили про конкретные языки.
- Для доказательств часто оказывается удобно работать не с привычными C, Python и т.д., а со специальными простыми языками.
- Особенно при доказательстве неразрешимости каких-то задач или невычислимости каких-то функций (первое это частный случай второго).
- Чаще всего это делается сведением задачи к проблеме остановки (или к другой задаче, для которой уже доказали неразрешимость).
- То есть показывается, что мы можем «эмулировать» произвольную программу на нашем языке в терминах этой задачи так, чтобы решение задачи позволило сказать, остановится программа или нет.
- А это проще, если язык небольшой и программы на нём устроены просто.

Регистровая машина

- Программа регистровой машины состоит из конечной пронумерованной последовательности команд следующих видов:
    - (переменная) $:=$ 0
    - (переменная1) $:=$ (переменная2)
    - (переменная)$++$
    - (переменная)$--$
    - goto (номер команды)
    - if (переменная)=0 goto (номер1) else goto (номер2)
    - stop
- Каждая такая программа очевидно использует конечное число переменных (или регистров).
- Значения переменных — натуральные числа, в начале все 0.
- Если переменная имеет значение 0 и уменьшается на 1, у неё остаётся значение 0.

Вычисление функций на регистровой машине

- Каждая такая программа вычисляет определённую функцию.
- Пусть среди регистров программы есть $i_1, \ldots, i_n$ и нет i_{n+1}, тогда у функции n аргументов.
- Поместим в регистры $i_1, \ldots, i_n$ значения аргументов функции.
- Начнём выполнять команды начиная с первой, если команда не goto, переходим к следующей.
- Если мы дойдём до конца программы или выполним команду stop, то значение функции это содержимое регистра o на этом шаге.
- Если ни того, ни другого не произойдёт, то функция на этих аргументах не определена.

Машины Тьюринга

- Определение мТ
- Картинка
- Более важны исторически, чем удобны для работы
- TODO

Примитивно и частично рекурсивные функции

- Базовые функции
- Операторы композиции и примитивной рекурсии
- Примитивно рекурсивные функции
- Оператор минимизации
- Частично рекурсивные функции
- TODO

Другие примеры моделей вычислимости

- Языки BlooP и FlooP, алгорифмы Маркова, машины с доступом к памяти?

Тезис Чёрча–Тьюринга

- Эквивалентность вычислимости функций на машинах Тьюринга, алгорифмах Маркова, машинах Поста и частичной рекурсивности
- Полнота по Тьюрингу
- Тезис Чёрча–Тьюринга
- (Не)возможность доказательства
- TODO