

Конспект по теоретической информатике
2020-2021 гг
Современное программирование, факультет математики и
компьютерных наук, СПбГУ
(лекции Гирша Эдуарда, Пузыниной Светланы и Дмитрия
Соколова)

Тамарин Вячеслав и СП-2019

June 18, 2021

Contents



I	Теория сложности вычислений	2
1	Введение в теорию сложности вычислений	3
1.1	Машины Тьюринга	3
1.1.1	Напоминания	3
1.1.2	Детерминированная машина Тьюринга	4
1.2	Классы сложности	7
1.2.1	Классы DTime и P	7
1.3	Недетерминированная машина Тьюринга	8
1.4	Сведения и сводимости	9
1.4.1	Трудные и полные задачи	10
1.4.2	Булевы схемы	11
1.5	Теорема Кука-Левина	12
1.5.1	Оптимальный алгоритм для \overline{NP} -задачи	12
1.6	Полиномиальная иерархия	14
1.6.1	Полиномиальная иерархия	14
1.7	$\Sigma^k P$ -полная задача	16
1.7.1	Коллапс полиномиальной иерархии	17
1.8	Классы, ограниченные по времени и памяти	17
1.8.1	PSPACE -полная задача	18
1.8.2	Иерархия по памяти	19
1.8.3	Если памяти совсем мало...	19
1.8.4	Иерархия по времени	20
1.9	Полиномиальные схемы	20
1.9.1	Схемы фиксированного полиномиального размера	21
1.9.2	Класс NSpace	22
1.10	Логарифмическая память	22
1.11	Равномерные полиномиальные схемы	24
1.11.1	Замкнутость NSpace относительно дополнения	27
1.12	Вероятностные вычисления	28
1.12.1	Классы с односторонней ошибкой	28
1.12.2	Связь с другими классами	29
1.13	Общая картина	31
1.14	Квантовые вычисления	31

II	Теория вычислимости	33
1	Вычислимость. Система вычислимости по Клини	34
1.1	Рекурсивные функции	34
1.1.1	Простейшие функции	34
1.1.2	Операторы	34
1.1.3	Функции	35
1.1.4	Оператор ограниченной минимизации	37
1.1.5	Предикаты	38
1.1.6	Канторовская нумерация	39
1.1.7	Теоремы про рекурсии	41
1.2	Равносильность МТ и ЧРФ	42
1.3	Функция Аккермана	45
2	Разрешимые и перечислимые множества	48
2.1	Определения	48
2.2	Перечислимые множества	49
2.3	Универсальные функции	52
2.3.1	Перечислимое неразрешимое множество	53
2.3.2	Главные универсальные функции	55
2.3.3	Теорема Райса	57
2.4	Иммунные и простые множества	58
2.5	Теорема о неподвижной точке	59
2.6	m -сводимость	60
2.7	Проблема соответствия Поста (PCP)	62
2.8	T -сводимость (по Тьюрингу)	64
2.9	Арифметическая иерархия	65
2.10	Еще про T -сводимость	67
2.11	Теорема об арифметической иерархии	69
2.11.1	Утверждения для доказательства в обратную сторону	69
2.11.2	Относительная вычислимость: эквивалентные определения	70
2.12	Классификация множеств в иерархии	72
III	Теории информации	74
1	Информация по Хартли	75
1.1	Базовые свойства	75
1.2	Угадывание числа	78
1.3	Задача выполнимости	80
1.4	Рассада голубей	80
2	Информация по Шеннону	81
2.1	Определения и свойства	81
2.1.1	Энтропия	81

2.1.2	Условная энтропия	82
2.2	Взаимная информация	85
2.3	Применение энтропии	87
2.3.1	Взвешивания монеток	87
2.3.2	Оценка на биномиальные коэффициенты	87
2.3.3	Подсчет углов в графе	88
3	Теория кодирования	89
3.1	Префиксные коды	89
3.2	Примеры эффективных кодов	91
3.2.1	Код Шеннона-Фано	91
3.2.2	Код Хаффмана	92
3.2.3	Арифметическое кодирование	93
3.3	Кодирование с ошибками	93
4	Коммуникационная сложность	96
4.1	Детерминированная модель	96
4.1.1	Нижние оценки для детерминированного случая	96
4.2	Методы оценки коммуникационной сложности	98
4.2.1	Метод ранга	98
4.2.2	Fooling Set	98
4.3	Балансировка протоколов	99
4.4	Сложная функция. Теорема Шеннона	100
4.5	Теорема Карчмера-Вигдерсона	101
4.6	Вероятностная модель	102
4.6.1	Эффективные протоколы для EQ и GT	102
4.7	Теория информации в коммуникационной сложности	103
4.7.1	Подсчет функции индексов	103
5	Колмогоровская сложность	107
5.1	Определения	107
5.2	Применение	107
5.3	Условная сложность	108

Исходный код на https://github.com/tamarinvs19/theory_university

Огромное спасибо студентам СП за помощь в написании и редактировании конспектов!

 Некоторые доказательства были опущены на лекции или разбирались только на практике, но написаны нами. Они выделены оранжевыми символами. 

Index

k -местная частичная функция, 34

$\Delta^i P$, 15

NC^i , 24

$\Pi^i P$, 15

Π_n , 65

$\Sigma^i P$, 15

Σ_n , 65

QBF_k , 16

\widetilde{NP} , 7

\tilde{P} , 7

$\overline{3-SAT}$, 12

ВН, задача об ограниченной подстановке, 11

BPP, 28

BQP, 32

CIRCUIT_EVAL, 25

CIRCUIT_SAT, 11

DSPACE, 17

DTime, 7

L, 23

NC, 24

NL, 23

NP, 7, 9

NPSpace, 22

NSpace, 22

P, 7, 8

PH, 15

PSPACE, 17

P/poly, 20

QBF, 18

RP, 28

STCON, 22

Size, 20

ZPP, 28

logspace-равномерное семейство схем, 24

Свойство функций, 57

булевы схемы, 11

время работы МТ, 4

индивидуальная задача, 3

информация по Хартли, 75

используемая память, 4

класс дополнений, 14

конструируемая по времени функция, 7

конструируемая по памяти функция, 17

кусочное задание функции, 42

массовая задача, 3

машина Тьюринга детерминированная, 4

машина Тьюринга недетерминированная, 8

машина Тьюринга оракульная, 10

общерекурсивная функция, 36

оператор минимизации, 35

оператор ограниченной минимизации, 37

оператор примитивной рекурсии, 34

оператор суперпозиции, 34

перечислимое множество, 49

полиномиальная иерархия, 14

полиномиально ограниченная задача, 7

полиномиально проверяемая задача, 7

полиномиальные схемы, 20

полная задача, 10

предикаты, 38

примитивно рекурсивная функция, 35

проекция, 51

простейшие функции, 34

равномерное семейство схем, 24

равносильность МТ и **ЧРФ**, 42

разрешимое множество, 48

рекурсия возвратная, 41

рекурсия совместная, 41

сведение по Карпу, 9

сведение по Левину, 9

сведение по Тьюрингу, 10

сводимость по Тьюрингу, 64

теорема Иммермана-Клапана, 27

теорема Карпа-Липтона, 21

теорема Кука-Левина, 12

теорема об арифметической иерархии, 69

трудная задача, 10

универсальная функция, 52

функция Аккермана, 45

частично рекурсивная функция, 35

энтропия, 81

язык, 3

Part I

Теория сложности вычислений

Chapter 1

Введение в теорию сложности вычислений

Лекция 1
5 nov

1.1 Машины Тьюринга

1.1.1 Напоминания

Обсудим, что мы решаем.

Обозначение.

- Алфавит будет бинарный $\{0, 1\}$;
- Множество всех слов длины n : $\{0, 1\}^n$;
- Множество всех слов конечной длины $\{0, 1\}^*$;
- Длина слова x : $|x|$.

Определение 1

Язык (задача распознавания, decision problem) — $L \subseteq \{0, 1\}^*$.

Индивидуальная задача — пара, первым элементом которой является условие, а второй – решение; принадлежит $\{0, 1\}^* \times \{0, 1\}^*$.

Массовая задача — некоторое множество индивидуальных задач, то есть бинарное отношение на $\{0, 1\}^*$.

Определение 2

Будем говорить, что алгоритм **решает задачу поиска** для массовой задачи R , если для условия x он находит решение w , удовлетворяющее $(x, w) \in R$.

Можем сопоставить массовой задаче, заданной отношением R , язык

$$L(R) = \{x \mid \exists w: (x, w) \in R\}.$$

Пример 1.1.1: Массовая задача и соответствующий язык

$$\text{FACTOR} = \{(n, d) \mid d \mid n, 1 < d < n\}.$$

Здесь условием задачи является натуральное число n , а решением некоторый (не 1, и не n) делитель числа n .

Данной задаче соответствует язык

$$L(\text{FACTOR}) = \text{множество всех составных чисел}.$$

1.1.2 Детерминированная машина Тьюринга

Определение 3: Детерминированная машина Тьюринга

Детерминированная машина Тьюринга —

- конечный алфавит (с началом ленты и пробелом): $\Sigma = \{0, 1, \triangleright, _ \}$;
- несколько лент, бесконечных в одну сторону;
- читающие/пишущие головки, по одной на каждую ленту;
- конечное множество состояний, в том числе начальное (q_S/q_0), принимающее (q_Y/q_{acc}) и отвергающее (q_N/q_{rej});
- управляющее устройство (программа), содержащее для каждого q, c_1, \dots, c_k одну инструкцию вида

$$(q, c_1, \dots, c_k) \mapsto (q', c'_1, \dots, c'_k, d_1, \dots, d_k),$$

где $q, q' \in Q$ — состояния, $c_i, c'_i \in \Sigma$ — символы, обозреваемые головками, $d_i \in \{\rightarrow, \leftarrow, \cdot\}$ — направления движения головок.

ДМТ **принимает** входное слово, если заканчивает работу в q_{acc} , и **отвергает**, если заканчивает в q_{rej} .

ДМТ M **распознает язык** A , если принимает все $x \in A$ и отвергает все $x \notin A$.

$$A = L(M).$$

Замечание. Обычно есть отдельная лента только для чтения, куда записаны входные данные, и лента только для вывода, куда нужно поместить ответ. Остальные ленты будут рабочими.

Замечание. Изначально на входной ленте вход и пробелы, на остальных пробелы, головки в крайней левой позиции, а состояние есть q_S .

Определение 4

Время работы машины M на входе x — количество шагов (применений инструкций) до достижения q_{acc} или q_{rej} .

Используемая память — суммарное крайнее правое положение всех головок на *рабочих* лентах.

Теорема 1.1.1. Для любого $k \in \mathbb{N}$, работу ДМТ M с k рабочими лентами, работающую t шагов, можно промоделировать на ДМТ с двумя рабочими лентами за время $\mathcal{O}(t \log t)$, где константа $\mathcal{O}(\dots)$ зависит только от размеров записи машины M .

□

- Перестроим исходную МТ:
 - Запишем все ленты в одну строку по символу из всех лент по очереди.
 - Будем бегать «лентой по головке»: выровняем все ленты, чтобы головки стояли друг над другом и далее будем сдвигать нужную ленту.
 - Заметим, что двустороннюю ленту можно смоделировать на односторонней, причем с увеличением количества операций в константу раз: разрезаем двустороннюю пополам и записываем элементы через один.

- Теперь поймем, как экономично сдвигать ленты в однострочной записи.

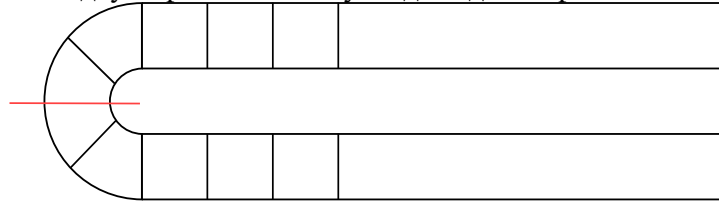
Разобьем строку на блоки начиная от позиции головки в две стороны: справа блоки R_i , слева L_i . При этом $|L_i| = |R_i| = 2^i$. Раздвинем символы, заполняя пустоту специальными символами пустоты, так, чтобы в каждом блоке ровно половина элементов были пустыми.

Далее будем поддерживать такое условие:

По очереди берем по одному элементу из каждой ленты

a_1	b_1	c_1	a_2	b_2	\dots
-------	-------	-------	-------	-------	---------

Разрезаем двустороннюю ленту на две односторонние



Синхронизация головок

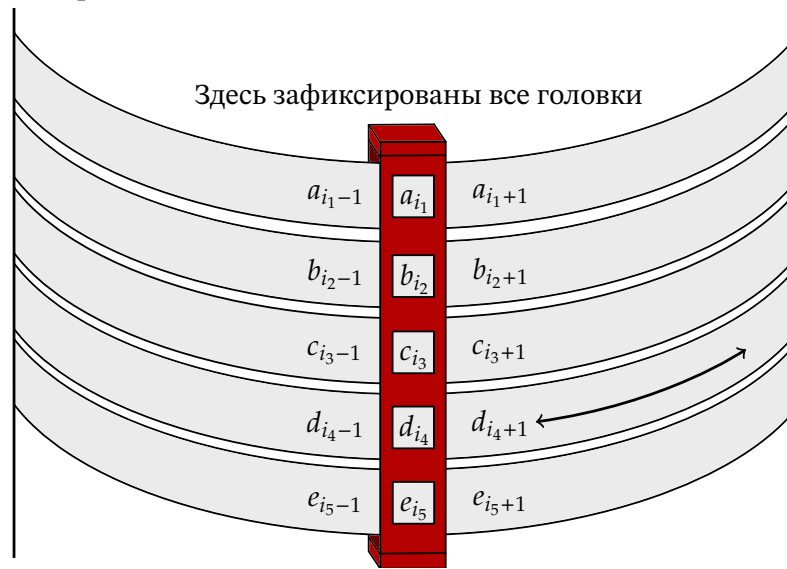


Figure 1.1: Построение новой МТ

1. В блоке либо информация, либо пусто, либо наполовину (ровно) пусто
2. L_i пустой, когда R_i полный
3. L_i наполовину пустой, когда R_i наполовину полный
4. L_i полный, когда R_i пустой

Пусть нужно подвинуть головку влево. Найдём слева первый не пустой блок L_i . Возьмем из него правую половину и разложим по пустым $L_{<i}$ так, чтобы порядок сохранился и каждый из $L_{<i}$ стал полупустым, а первый символ попал под головку.

Так получится сделать, так как всего перемещаемых символов 2^{i-1} , а в j -й блок будет помещено 2^{j-1} символов, поэтому всего в $L_{<i}$ поместится

$$1 + 2 + 4 + \dots + 2^{i-2} = 2^{i-1} - 1.$$

И один символ под головку.

Чтобы инвариант сохранился нужно теперь исправить правую часть.

Так как первые $i - 1$ левых блоков были пусты, первые $i - 1$ правых блоков полны, а R_i пуст (либо наполовину полон, в зависимости от первоначального состояния L_i). Заполним половину в R_i символами из R_{i-1} . Теперь R_{i-1} пустой, а меньшие полные. Прделаем ту же операцию еще раз для $i - 1$, потом для $i - 2$ и так далее. Заметим, что по отношению к R_i все сохранилось: если L_i был наполовину полон, то стал пуст, а R_i - полон (так как до этого был наполовину полон). Полностью аналогичный случай, если бы L_i был полон.

Когда мы дойдем до R_1 , положим туда элемент из-под головки.

Итого, инвариант сохранился.

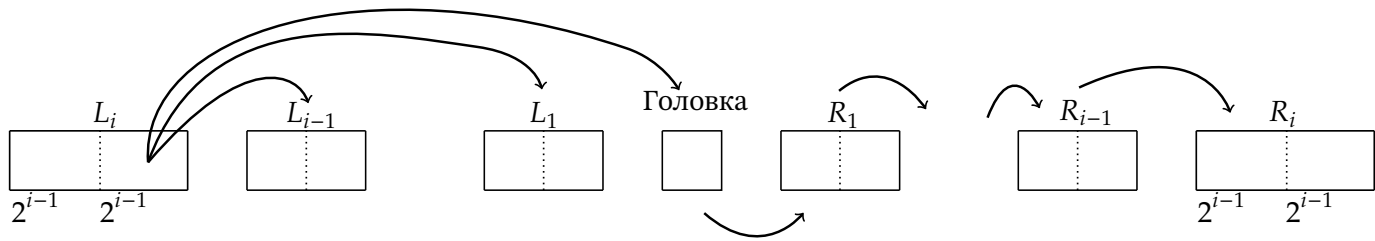


Figure 1.2: Структура блоков

- Посчитаем количество операций. В алгоритме переносим различные отрезки из одного места в другое. Чтобы делать это за линию, сначала скопируем нужный участок на вторую ленту, а затем запишем с нее.

Тогда при перераспределении происходит $c \cdot 2^i$ операций:

каждый символ переносили константное число раз (на вторую ленту, со второй ленты) плюс линейное перемещение от L_i к R_i несколько раз (символов всего 2^{i-1} в каждой половине, итого суммарно перенесли 2^i символов).

Докажем, что с i -м блоком происходят изменения не чаще 2^{i-1} шагов. Пусть L_i хотя бы наполовину заполнен. Когда мы забрали половину из него, мы заполнили все $L_{<i}$ и $R_{<i}$ наполовину.

Поэтому, чтобы изменить L_i еще раз, нужно сначала опустошить все $L_{<i}$. При перераспределении в левой части становится на один элемент меньше, а всего там 2^{i-1} заполненное место. Для того, чтобы все они ушли из левой половины, придется совершить 2^{i-1} сдвигов.

Итого, для t шагов исходной машины будет

$$\sum_i^{\log t} c \cdot 2^i \cdot \frac{t}{2^{i-1}} = O(t \log t).$$

Почему сумма до логарифма? Заметим, что так как мы сделали не более чем t шагов изначально, то мы не можем на каждой ленте занять более чем t ячеек. В таком случае, самый большой блок не будет больше t . А его индекс, соответственно не будет больше, чем $\log t$.

Замечание. Так как до этого у нас было k лент и все сдвиги могли происходить одновременно, то здесь мы делаем сдвиг каждой ленты по очереди. То есть на самом деле мы получаем $O(k \cdot t \log t)$, но k в данном случае просто константа (почти точно константа может стать намного больше, чем k , но всё равно останется константой).

Теорема 1.1.2 (Об универсальной МТ). Существует ДМТ U , выдающая на входе (M, x) тот же результат, что дала бы машина M на входе x , за время $O(t \log t)$, где t — время работы M на входе x .

□ Используем прием из прошлой теоремы 1.1.1.

Замечание. Единственный способ узнать результат машины M на строке x , это промоделировать вычисление машины M на входе. Мы никогда не можем заранее сказать результат этого вычисления, и заканчивается ли оно вообще. Поэтому, моделировать вычисления (т.е. иметь универсальную машину Тьюринга) можно, но, например, нельзя иметь машину, которая по записи машины M' скажет, принимает ли M' пустую строку, можно лишь вести себя как M' на пустой строке.

1.2 Классы сложности

1.2.1 Классы DTime и P

Определение 5: Конструируемая по времени функция

Функция $t: \mathbb{N} \rightarrow \mathbb{N}$ называется **конструируемой по времени**, если

- $t(n) \geq n$;
- двоичную запись $t(|x|)$ можно найти по входу x на ДМТ за $t(|x|)$ шагов.

Замечание. Все стандартные функции (многочлены, экспоненты) конструируемы по времени.

Неконструируемыми являются вырожденные, специально построенные случаи.

Определение 6: Класс DTime

Язык L принадлежит классу $\mathbf{DTime}[t(n)]$, если существует ДМТ M , принимающая L за время $\mathcal{O}(t(n))$, где t конструируема по времени.

Константа может зависеть от языка, но не от длины входа.

Определение 7: Класс P

Класс языков, распознаваемых за полиномиальное время на ДМТ —

$$\mathbf{P} = \bigcup_c \mathbf{DTime}[n^c].$$

Будем обозначать задачи, заданные отношениями волной.

Определение 8

Массовая задача R **полиномиально ограничена**, если существует полином p , ограничивающий длину кратчайшего решения:

$$\forall x \left(\exists u: (x, u) \in R \implies \exists w: ((x, w) \in R \wedge |w| \leq p(|x|)) \right).$$

Массовая задача R **полиномиально проверяема**, если существует полином q , ограничивающий время проверки решения: для любой пары (x, w) можно проверить принадлежность $(x, w) \stackrel{?}{\in} R$ за время $q(|(x, w)|)$.

Определение 9: Класс $\widetilde{\mathbf{NP}}$

$\widetilde{\mathbf{NP}}$ — класс задач поиска, задаваемых полиномиально ограниченными полиномиально проверяемыми массовыми задачами.

Определение 10: Класс $\tilde{\mathbf{P}}$

$\tilde{\mathbf{P}}$ — класс задач поиска из $\widetilde{\mathbf{NP}}$, разрешимых за полиномиальное время.

То есть класс задач поиска, задаваемых отношениями R , что для всех $x \in \{0, 1\}^*$ за полиномиальное время можно найти w , для которого $(x, w) \in R$.

Ключевой вопрос теории сложности $\tilde{\mathbf{P}} \stackrel{?}{=} \widetilde{\mathbf{NP}}$

Определение 11: Класс NP

\mathbf{NP} — класс языков (задач распознавания), задаваемых полиномиально ограниченными полиномиально

проверяемыми массовыми задачами:

$$\mathbf{NP} = \{L(R) \mid R \in \widetilde{\mathbf{NP}}\}.$$

Замечание. $L \in \mathbf{NP}$, если существует массовая п.о.п.п.¹ задача, такая, что

$$\forall x \in \{0,1\}^*: x \in L \iff \exists w: (x, w) \in R.$$

Определение 12: Класс P

P — класс языков (задач распознавания), распознаваемых за полиномиальное время.

$$\mathbf{P} = \{L(R) \mid R \in \tilde{\mathbf{P}}\}.$$

Замечание. Очевидно, $\mathbf{P} \subseteq \mathbf{NP}$.

Ключевой вопрос теории сложности $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$

Лекция 2

12 nov

1.3 Недетерминированная машина Тьюринга

Определение 13: Недетерминированная машина Тьюринга

Недетерминированная машина Тьюринга — машина Тьюринга, допускающая более одной инструкции для данного состояния $q \in Q$ и $c_1, \dots, c_k \in \Sigma$, то есть для состояния q и символа c функция δ будет многозначной.

Из такого определения получаем **дерево вычислений** вместо последовательности состояний ДМТ.

Мы говорим, что НМТ ² **принимает** вход, если существует путь в дереве вычислений, заканчивающийся принимающим состоянием.

Утверждение. В машины (ДМТ / НМТ) с заведомо ограниченным временем работы можно встроить **будильник** и считать время вычислений на входах одной длины всегда **одинаковым**. Для этого можем просто записать на дополнительную ленту $t(n)$ единиц и стирать по одной за ход.

Определение 14: Эквивалентное определение НМТ

Недетерминированная машина Тьюринга — ДМТ, у которой есть дополнительный аргумент — конечная (ограниченная временем работы) подсказка w на второй ленте.

Определённая таким образом НМТ M принимает вход x , если существует такая подсказка w , что соответствующая ДМТ примет такой вход, т.е. $M(x, w) = 1$.

□ Докажем эквивалентность.

- Представим дерево вычисления как бинарное дерево и пронумеруем ребра из каждой вершины 0 и 1. Теперь запишем нужную принимающую ветку на ленту-подсказку.

По подсказке можем построить дерево, где будет нужный путь.

- Обратно: у нас есть ДМТ, которая ожидает вход из самого входа x (который мы знаем — всегда одинаков) и подсказки w . Считаем, что мы знаем оценку верхнюю на длину подсказки. Построим теперь НМТ: бинарное дерево 0 и 1.

Отработав какой-то путь в таком дереве, мы получим возможную подсказку, которую отдадим в качестве w ДМТ. Тогда приниматься будет тогда и только тогда, когда существует подсказка.



¹полиномиально ограниченная полиномиально проверяемая

²недетерминированная машина Тьюринга

Определение 15

Еще одно определение класса **NP** — класс языков, принимаемых полиномиальными по времени НМТ.

Замечание. Это определение класса **NP**, можно считать двумя, так как у нас есть два определения НМТ.

1.4 Сведения и сводимости**Определение 16: Сведение по Карпу**

Язык L_1 **сводится по Карпу** к языку L_2 , если существует полиномиально вычислимая функция f такая, что

$$\forall x: x \in L_1 \iff f(x) \in L_2.$$

Определение 17: Сведение по Левину

Задача поиска (отношение) R_1 **сводится по Левину** к задаче R_2 , если существуют функции f, g, h такие, что для всех x_1, y_1, y_2 верно

- $R_1(x_1, y_1) \implies R_2(f(x_1), g(x_1, y_1))$;
- $R_1(x_1, h(f(x_1), y_2)) \iff R_2(f(x_1), y_2)$;
- f, g, h полиномиально вычислимы.

Замечание. Первое условие нужно для того, чтобы образы каждого входа, имеющего решение первой задачи, имели решение и второй задачи.

Теорема 1.4.1. Классы **P, NP, \tilde{P} , \widetilde{NP}** замкнуты относительно этих сведений.

То есть если $R_2 \in \tilde{P}, R_1 \rightarrow R_2 \implies R_1 \in \tilde{P}$

□ Пусть есть два языка или две задачи поиска R_1, R_2 , R_2 принадлежит классу $C \in \{\mathbf{P}, \tilde{\mathbf{P}}, \mathbf{NP}, \widetilde{\mathbf{NP}}\}$ и R_1 сводится к R_2 . Чтобы проверить принадлежность $R_1 \in C$, нам нужно:

- Проверить полиномиальную ограниченность и полиномиальную проверяемость, тогда R_2 будет в **NP** или **\widetilde{NP}** .
- Если $C \in \{\mathbf{P}, \tilde{\mathbf{P}}\}$, то еще проверить, что для R_1 есть полиномиальный алгоритм.

Задачи поиска. R_2 задано п.о.п.п. отношением. Что можно узнать про R_1 ?

Если есть решение для R_1 , то функция g дает решение R_2 , которое не на много длиннее.

Еще есть h , которая позволяет из решения R_2 обратно быстро построить решение R_1 .

- **Полиномиальная ограниченность.** Пусть есть некоторое решение y_1 для задачи R_1 . Для него можно получить некоторое решение $R_2 - g(x_1, y_1)$. Так как существует **какое-то** решение $g(x_1, y_1)$, то (по определению п.о.) есть решение y_2 , которое ограничено полиномом. В таком случае его можно перевести в $h(f(x_1), y_2)$ – решение для R_1 . Оно тоже будет ограничено полиномом, так как h – полиномиально вычислимо (и как следствие – полиномиально ограничена).
- **Полиномиальная проверяемость.** Проверяется аналогично.
- **Полиномиальный алгоритм.** Если есть алгоритм для R_2 и x_1 , сначала перегоняем $x_1 \rightarrow f(x_1)$, далее применяем алгоритм, получаем y_2 , а далее, используя h , перегоняем обратно в R_1 .

Языки. Пусть есть $L_1 \rightarrow L_2, L_2 \in C$.

- $C = \mathbf{P}$. Хотим проверить $x \in L_1$. Сначала можем за полином посчитать $f(x)$, а затем опять за полином проверить $f(x) \in L_2$.
- $C = \mathbf{NP}$. У нас есть алгоритм проверки решения для L_2 за полином. Так как $L_1 \rightarrow L_2$, если $x \in L_1$, то $f(x) \in L_2$, поэтому можем проверить решение в L_2 .



Утверждение. Если задача R_1 сводится по Левину к R_2 , то язык $L(R_1)$ сводится по Карпу к $L(R_2)$.

□ Функция f для сведения по Карпу будет взята из сведения по Левину. Разберем два случая:

- $x \in L(R_1)$. Тогда мы точно знаем, что есть некоторый y_1 , причем $(x, y_1) \in R_1$. Следовательно, по сведению по Левину есть такой $y_2 = g(x_1, y_1)$, что $(f(x_2), y_2) \in R_2$. Значит, $x \in L(R_1) \implies f(x) \in L(R_2)$.
- $x \notin L(R_1)$. Хотим показать, что тогда $f(x) \notin L(R_2)$. Пусть это не так. Тогда есть некоторый y_2 такой, что $(f(x), y_2) \in R_2$. Тогда из сведения по Левину знаем, что $(f(x), h(f(x), y_2)) \in R_1$, то есть $x \in L(R_1)$, противоречие.



Определение 18: Оракульная МТ

Оракульная МТ — МТ с доступом к оракулу, который за один шаг дает ответ на некоторый вопрос.

Обозначение. При переходе в состояние q_{in} происходит «фантастический переход» в состояние q_{out} , заменяющий содержимое некоторой ленты на ответ оракула.

M^B — оракульная машина M , которой дали оракул B .

Определение 19: Сведение по Тьюрингу

Язык или задача A **сводится по Тьюрингу** к B , если существует оракульная машина полиномиальная по времени M^\bullet такая, что M^B решает A .

Например, если A — язык, $A = L(M^B)$.

Пример 1.4.1

Классы \mathbf{P} и $\tilde{\mathbf{P}}$ замкнуты относительно сведений по Тьюрингу. А \mathbf{NP} и $\widetilde{\mathbf{NP}}$ могут быть незамкнуты: если $A = \text{UNSAT}$, $B = \text{SAT}$, $M^O(x) = \overline{(x \in O)}$, и A сводится по Тьюрингу к B , но $B \in \mathbf{NP}$, а про A не известно.

1.4.1 Трудные и полные задачи

Определение 20: Трудные и полные задачи

Задача A называется **трудной** для класса \mathbf{C} , если $\forall C \in \mathbf{C}: C \rightarrow A$.

Задача A называется **полной** для класса \mathbf{C} , если она трудная и принадлежит \mathbf{C} .

Замечание. Если не упомянуто другое, имеется ввиду сводимость по Карпу.

Теорема 1.4.2. Если A — \mathbf{NP} -трудная и $A \in \mathbf{P}$, то $\mathbf{P} = \mathbf{NP}$.

Следствие 1. Если A — \mathbf{NP} -полная, то $A \in \mathbf{P} \iff \mathbf{P} = \mathbf{NP}$.

Задача об ограниченной остановке

Определение 21: ВН

Определим задачу об ограниченной остановке $\widetilde{\text{ВН}}(\langle M, x, 1^t \rangle, w)$ ^a так: дана НМТ M и вход x , требуется найти такую подсказку w , чтобы M распознавала x не более чем за t шагов.

Соответствующая задача распознавания — ответить, существует ли такая подсказка.

^aздесь 1^t — служебные t единиц, ограничивающие время работы МТ

Теорема 1.4.3. Задача об ограниченной остановке $\widetilde{\text{НР}}$ -полная, а соответствующий язык НР -полный.

□

- Принадлежность $\widetilde{\text{НР}}$ и НР следует из существования универсальной ДМТ, которая за $\mathcal{O}(t \log t)$ промоделирует вычисление ДМТ, описание которой дано ей на вход. После этого нужно проверить, что M успевает за t шагов закончить.
- Проверим, что язык НР -трудный. Пусть язык L принадлежит НР . По определению НР существует для соответствующего отношения R машина Тьюринга $M^*(x, w)$, которая работает за $p(|x|)$ — полином. Сведем L к задаче ВН. Рассмотрим тройку $\langle M^*, x, 1^{p(|x|)} \rangle$. Пусть функция $f(x)$ будет равна этой тройке. В таком случае условие распознавание тройки равносильно тому, что x из языка $L(M^*)$
- Аналогично для задач поиска.

■

1.4.2 Булевы схемы

Определение 22: Булева схема

Булева схема — ориентированный граф без циклов, в вершинах которого записаны бинарные, унарные или нульарные операции над битами (\wedge, \vee, \oplus), при этом есть специальные вершины-входы и вершины-выходы.

Определение 23: CIRCUIT_SAT

$$\widetilde{\text{CIRCUIT_SAT}} = \{(C, w) \mid C \text{ — булева схема}, C(w) = 1\}.$$

Соответственно, $C \in \text{CIRCUIT_SAT}$, если существует такой вход w , что схема возвращает 1.

Очевидно, что $\text{CIRCUIT_SAT} \in \text{НР}$. Чтобы доказать НР -трудность, сведем $\text{ВН} \rightarrow \text{CIRCUIT_SAT}$: будем рисовать конфигурацию МТ на схемах.

- Пусть каждый этаж схемы — конфигурация ДМТ. Всего этажей будет столько же, сколько шагов в МТ, то есть t . Если в последнем этаже q_{acc} , то результат 1.
- **Пересчет конфигураций.** Заметим, что при переходе от одной конфигурации к другой меняются только гейты рядом с положением головки.

Выделим подсхему, которая должна по состоянию и элементу рядом с головкой получить новое состояние, положение головки и поменять элемент, а остальные символы скопировать. Так мы построим новый уровень с измененными элементами.

Для хранения головки будем после каждого символа с ленты хранить d_i равное единице, если головка на символе c_i перед ним.

Если $d_i = 0$, то новый $c_i^l = c_i$, иначе нужно заменить c_i на c_i^l из программы МТ:

$$(q, c_{i-1}, c_i, c_{i+1}, d_{i-1}, d_i, d_{i+1}) \mapsto (q^l, c_i^l, d_i^l).$$

- Так как входная строка x нам дана, «запаяем» ее в схему: подключим входные гейты с элементами x к нужным входам.

Тогда оставшимся входом схемы останется подсказка w для НМТ, а выходом — попадание в q_{acc} .

Таким образом, мы по $\langle M, x, t \rangle$ построили схему $C(w)$.

Лекция 3
19 nov

1.5 Теорема Кука-Левина

$$\widetilde{3\text{-SAT}} = \{(F, A) \mid F \text{ — в 3-КНФ, } F(A) = 1\}.$$

Пример 1.5.1

$$((\neg x \vee \neg y \vee \neg z) \wedge (y \vee \neg z) \wedge (z), [x = 0, y = 1, z = 1]) \in \widetilde{3\text{-SAT}}.$$

Теорема 1.5.1 (Кук-Левин). $\widetilde{3\text{-SAT}}$ — $\widetilde{\text{NP}}$ -полная задача.

- Мы уже доказали полноту задач выполнимости булевых схем, поэтому будем сводить к ней.

Пусть у нас есть некоторая схема. Для каждого гейта заведем по переменной, которая обозначает результат операции в этом гейте. Входы тоже остаются гейтами в схеме.

Запишем для гейтов клозы длины 3, которые выражают результат в зависимости от аргументов.

Например, для входов x, y и операции $\oplus = g(x, y)$,

$$\begin{aligned} (x \vee y \vee \neg g) \\ (\neg x \vee \neg y \vee \neg g) \\ (x \vee \neg y \vee g) \\ (\neg x \vee y \vee g) \end{aligned}$$

Еще для последнего гейта g (выходного) запишем клоз (g).

Значение в полученной схеме будет соответствовать результату конъюнкции всех клозов и наоборот: по формуле можем построить булеву схему и входные переменные выполняют ее. ■

Теорема 1.5.2. Если $R \in \widetilde{\text{NP}}$, и соответствующий язык $L(R)$ — NP -полон, то R сводится по Тьюрингу к $L(R)$.

- Во-первых, задача поиска из NP сводится к $\widetilde{\text{SAT}}$. При этом SAT сводится к $L(R)$.

Осталось свести $\widetilde{\text{SAT}} \rightarrow \text{SAT}$. Для этого нужно построить МТ с оракулом SAT будет решать $\widetilde{\text{SAT}}$.

Пусть нам дали формулу F и мы хотим найти выполняющий набор. Предполагается, что она выполнима.

Подставим первую переменную x_1 как 0 и как 1 в F (это тоже будут формулы) и спросим у оракула SAT про $F[x_1 = 0] \in \text{SAT}$ и $F[x_1 = 1] \in \text{SAT}$.

Так как F выполнима, хотя бы одна из полученных схем выполнима. Выберем ее и продолжим подставлять в нее. Так мы дойдем до конечной истинной формулы. Следовательно, последовательность подставляемых значений x_i и будет выполняющим набором. ■

1.5.1 Оптимальный алгоритм для $\widetilde{\text{NP}}$ -задачи

Пусть мы хотим решить задачу, заданную отношениям R , с входом x . Давайте переберем все машины (не только полиномиальные) и, если какая-то машина выдала результат y , то проверим его $R(x, y)$ за

полином.

Если ответ подошел, то просто заканчиваем работу, иначе продолжаем ждать других результатов.

Если машины были бы запущены параллельно, то мы бы нашли ответ за время самой быстрой машины на данном входе.

А мы будем делать шаги «змейкой»: выделим для l -ой машины 2^{-l} времени удельно.

На очередном этапе $2^l(1 + 2k) = 2^l + 2^{l+1}k$ будем моделировать k -ый шаг машины M_l . То есть для l -той машины первый шаг (нулевой) будет сделан на 2^l -том реальном шаге, и каждый следующий будет совершаться каждые 2^{l+1} шаг.

Так, например, будет выглядеть табличка для первых машин (указаны этапы моделирования, на которых будет выполнен соответствующий шаг):

№ МТ	0 шаг	1 шаг	2 шаг	3 шаг
0	1	3	5	7
1	2	6	10	14
2	4	12	20	28
3	8	24	40	56

Посчитаем замедление алгоритма. Мы хотим выдать ответ не сильно позже, чем самая быстрая машина. Но заметим, что такая машина одна³, поэтому l это константа, следовательно, множитель 2^l тоже константа.

Как моделировать эти машины? Если было бы быстрое обращение к каждому элементу памяти, то $t(x) \leq \text{const}_i \cdot t_i + p(|x|)$, где i - номер самой быстрой машины ($p(|x|)$ на проверку).

Но у нас ДМТ, поэтому получаем $t(x) \leq \text{const}_i \cdot p(t_i(x))$ из-за необходимости хранить на одной ленте несколько МТ и следовательно тратить время на «переключения» между ними.

Замечание. Если $\mathbf{P} = \mathbf{NP}$, то построенный алгоритм может решить SAT за полиномиальное от времени работы самой быстрой машины $p(t_i(x))$, но и оно полиномиально в случае $\mathbf{P} = \mathbf{NP}$.

Теорема 1.5.3. Если $\mathbf{P} \neq \mathbf{NP}$, то существует язык $L \in \mathbf{NP} \setminus \mathbf{P}$, не являющийся \mathbf{NP} -полным.

□ Найдем задачу, которая и не в \mathbf{P} и не является \mathbf{NP} -полной.

Занумеруем все полиномиальные ДМТ с полиномиальными будильниками⁴:

$$M_1, M_2, \dots$$

Аналогично пронумеруем полиномиальные сведения с будильниками, про которые мы будем думать, как про машины Тьюринга.

$$R_1, R_2, \dots$$

Построим следующий язык $\mathcal{K} = \{x \mid x \in \mathbf{SAT} \wedge f(|x|) \equiv 0 \pmod{2}\}$, где $f(n)$ ведет себя следующим образом:

1. за n шагов вычисляем $f(0), f(1), \dots, f(i) =: k$ (k — последнее значение, которое мы успели вычислить).

2. тоже за n шагов:

(а) если $k \equiv 0 \pmod{2}$, то проверим, что $\exists z: M_{k/2}(z) \neq \mathcal{K}(z)$, и в случае успеха вернем $k + 1$, иначе (или истратили n шагов) k ;

(б) если $k \equiv 1 \pmod{2}$, проверим, что очередное $R_{\frac{k-1}{2}}$ правильно сводит SAT к R : если $\exists z: \mathcal{K}(R_{\frac{k-1}{2}}(z)) \neq \mathbf{SAT}(z)$, возвращаем $k + 1$, в любом другом случае — k .

³Кажется, здесь важно просто, что такая есть и она имеет какой-то номер l .

⁴то есть настроенными на полиномиальное время

По факту в пункте (а) мы проверяем, что рассматриваемая машина Тьюринга не решает задачу \mathcal{K} . Такое вычисление точно остановится, т.к. ограничено число шагов, т.е., если нынешняя МТ вдруг решала бы задачу \mathcal{K} , то число k просто и вернулось бы. Здесь первый пункт проверяет $\mathcal{K} \in \mathbf{P}$, а второй — $\mathbf{SAT} \rightarrow \mathcal{K}$, то есть \mathcal{K} **NP**-трудная.

Для какого-то огромного n найдутся контр-примеры и мы получим $k + 1$.

3. $f(0) = 0$

Заметим, что $\mathcal{K} \in \mathbf{NP}$, так как выполняющий набор можно проверить принадлежность **SAT** за полином и подставить в f , которая тоже работает полином ($2n$ шагов).

- Пусть $\mathcal{K} \in \mathbf{P}$, тогда есть полиномиальная машина M , которая принимает этот язык. Поэтому в пункте (а) мы дальше этой машины не пройдем, так как контр-примера там нет. То есть с некоторого момента $f(n) \equiv 0 \pmod{2}$, по определению с некоторого места $\mathcal{K} = \mathbf{SAT}$, кроме конечного числа случаев, их можно разобрать отдельно. Тогда $\mathcal{K} \in \mathbf{NP-complete}$ и $\mathcal{K} \in \mathbf{P}$, поэтому $\mathbf{P} = \mathbf{NP}$.
- Если $\mathcal{K} \in \mathbf{NP-complete}$, то с некоторого момента мы не пройдем пункт (b), так как у нас действительно будет сводимость к \mathcal{K} . С некоторого места $f(n) \equiv 1 \pmod{2}$, а тогда $|\mathcal{K}| < \infty$. Следовательно, $\mathcal{K} \in \mathbf{P}$. Опять противоречие.



1.6 Полиномиальная иерархия

Обозначение. Пусть есть два класса языков \mathcal{C}, \mathcal{D} . Можно построить класс $\mathcal{C}^{\mathcal{D}}$, который состоит из языков вида \mathcal{C}^D , где $D \in \mathcal{D}$ и \mathcal{C} — машина для языка из \mathcal{C} .

Определение 24: Класс дополнений

$$\mathbf{co-C} = \{L \mid \bar{L} \in \mathcal{C}\}.$$

Пример 1.6.1

$\mathbf{SAT} \in \mathbf{NP}$, дополнение к **SAT**, то есть всюду ложные формулы (или записи, которые вообще формулами не являются, но их легко проверить полиномиально) $\in \mathbf{co-NP}$.

1.6.1 Полиномиальная иерархия

Самый нижний класс иерархии — **P**. Остальные классы строятся также из **NP** и **co-NP**.

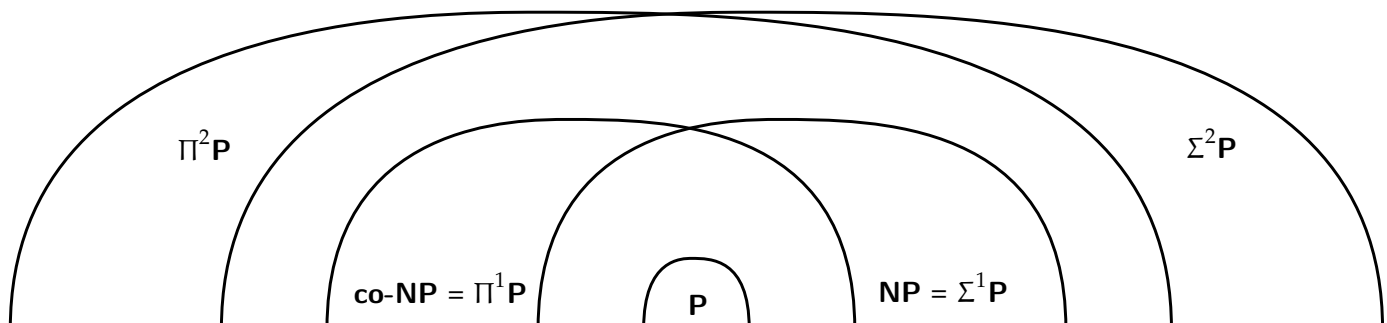


Figure 1.3: Полиномиальная иерархия

$$\Sigma^0 \mathbf{P} = \Pi^0 \mathbf{P} = \Delta^0 \mathbf{P} = \mathbf{P}$$

$$\Sigma^{i+1} \mathbf{P} = \mathbf{NP}^{\Pi^i \mathbf{P}}$$

$$\Pi^{i+1} \mathbf{P} = \mathbf{co-NP}^{\Sigma^i \mathbf{P}}$$

$$\Delta^{i+1} \mathbf{P} = \mathbf{P}^{\Sigma^i \mathbf{P}}$$

$$\mathbf{PH} = \bigcup_{i \geq 0} \Sigma^i \mathbf{P}$$

Лемма 1. $\mathbf{NP}^{\Pi^i \mathbf{P}} = \mathbf{NP}^{\Sigma^i \mathbf{P}}$

□ Пусть есть машина M с оракулом A . Рассмотрим оракул \bar{A} и построим машину M' , чтобы $M'^{\bar{A}}$ вела себя аналогично M^A . Для этого просто M' будет переворачивать любой ответ, полученный от оракула, все остальные действия повторяем за M .

Теперь докажем по индукции утверждение леммы, база очевидна. Переход такой: можно поменять оракула $\Pi^i \mathbf{P}$ на оракула $\bar{\Pi}^i \mathbf{P} = \mathbf{NP}^{\Sigma^{i-1} \mathbf{P}} = \mathbf{NP}^{\Pi^{i-1} \mathbf{P}} = \Sigma^i \mathbf{P}$. ■

Теорема 1.6.1. $L \in \Sigma^k \mathbf{P}$, тогда^a существует полиномиально ограниченное отношение $R \in \Pi^{k-1} \mathbf{P}$ такое, что для всех x :

$$x \in L \iff \exists y: R(x, y).$$

^aтогда и только тогда, когда

□

⇒ Докажем по индукции.

- **База:** по определению $\Sigma^1 \mathbf{P} = \mathbf{NP}^{\mathbf{P}} = \mathbf{NP}$.
- **Переход:** $k-1 \rightarrow k$. Пусть $L = L(M^O)$, где M — полиномиальная НМТ, $O \in \Sigma^{k-1} \mathbf{P}$.

По предположению индукции для O существует полиномиально ограниченное $S \in \Pi^{k-2} \mathbf{P}$ такое, что $\forall q: q \in O \iff \exists w: S(q, w)$.

Сконструируем из этого R :

- $R(x, y) = 1$, если y — принимающая ветвь вычисления M^O (то есть последовательность 0, 1 — идти налево или направо), при этом положительные ответы оракула должны быть снабжены сертификатами $w: S(q, w) = 1$.

То есть все переходы, основанные на ответе оракула «да», должны дополнительно содержать «доказательство».

Проверим, что это отношение из $\Pi^{k-1} \mathbf{P}$, и, что оно задает язык L .

- $R \in \Pi^{k-1} \mathbf{P}$: детерминировано проверяем корректность y , а далее проверяем, что ответы оракула были верными.

Для ответов «нет» нужно проверить, что O для него равно нулю (запускаем $\Pi^{k-1} \mathbf{P}$ вычисление), а для ответов «да» — что S равно 1, то есть проверить сертификат ($\Pi^{k-2} \mathbf{P}$ вычисление).

Нужно, чтобы все $\Pi^{k-1} \mathbf{P}$ и $\Pi^{k-2} \mathbf{P}$ (это частный случай $k-1$) вычисления вернули «да». Для этого построим схему: присоединим к большой конъюнкции все вычисления, чтобы ответ был положительным, нужно, чтобы все ветки $\Pi^{k-1} \mathbf{P}$ (то есть $\mathbf{co-NP}$) вернули одно и то же, при этом мы остаемся в $\Pi^k \mathbf{P}$.

Для машины M существует принимающее вычисление и оно может быть дано в качестве y .

Наоборот, если у нас есть корректное вычисление машины M , то оно и будет принадлежать L .

⇐ Пусть у нас есть отношение R . Возьмем машину с оракулом R .

Она будет недетерминировано выбирать y и проверять, то есть спрашивать у оракула, $R(x, y)$.
Эта машина и будет распознавать наш язык L .



Аналогично можно доказать следующую теорему

Теорема 1.6.2. $L \in \Pi^k \mathbf{P}$, тогда существует полиномиально ограниченное отношение $R \in \Sigma^{k-1} \mathbf{P}$ такое, что для всех x :

$$x \in L \iff \forall y: R(x, y).$$

Следствие 2. $L \in \Sigma^k \mathbf{P}$, тогда существует полиномиально ограниченное отношение $R \in \mathbf{P}$ такое, что для всех x :

$$x \in L \iff \exists y_1 \forall y_2 \exists y_3 \dots R(x, y_1, y_2, \dots, y_k).$$

Следствие 3. $L \in \Pi^k \mathbf{P}$, тогда существует полиномиально ограниченное отношение $R \in \mathbf{P}$ такое, что для всех x :

$$x \in L \iff \forall y_1 \exists y_2 \forall y_3 \dots R(x, y_1, y_2, \dots, y_k).$$

Лекция 4
26 nov

1.7 $\Sigma^k \mathbf{P}$ -полная задача

Определение 25: Язык \mathbf{QBF}_k

Язык \mathbf{QBF}_k — язык, состоящий из замкнутых истинных формул вида

$$\exists X_1 \forall X_2 \exists X_3 \dots X_k \varphi,$$

где φ — формула в КНФ или ДНФ, а $\{X_i\}_{i=1}^k$ — разбиение множества переменных этой формулы на непустые подмножества.

Теорема 1.7.1. \mathbf{QBF}_k — $\Sigma^k \mathbf{P}$ -полный язык.



- Во-первых, проверим принадлежность.

По следствию из прошлой теоремы достаточно найти полиномиально ограниченное отношение $R \in \mathbf{P}$ такое, что $\forall x: x \in \mathbf{QBF}_k \iff \exists y_1 \forall y_2 \exists y_3 \dots R(x, y_1, y_2, y_3, \dots)$.

Возьмем просто R , которое для R (формула с φ, x_1, x_2, \dots) выдает результат подстановки x_i в φ .

Проверка будет работать за полином, то есть $R \in \mathbf{P}$.

- Докажем, что к этой задаче сводится любой язык из $\Sigma^k \mathbf{P}$.

Пусть есть язык $L \in \Sigma^k \mathbf{P}$. Тогда существует полиномиально ограниченное отношение $R \in \mathbf{P}$:

$$\forall x: x \in L \iff \exists y_1 \forall y_2 \exists y_3 \dots R(x, y_1, y_2, \dots).$$

- Если в конце формулы идет квантор \exists , то запишем язык R в таком виде (как в теореме Кука-Левина):

$$R(z) \iff \exists w \Phi(z, w).$$

Тогда определение языка будет иметь следующий вид:

$$\forall x: x \in L \iff \exists y_1 \forall y_2 \exists y_3 \dots \exists y_k \exists w \Phi(x, y_1, y_2, \dots, y_k, w).$$

Но так как последние два квантора можно объединить в один, получаем формулу из \mathbf{QBF}_k .

- Иначе запишем \bar{R} в виде булевой формулы Ψ , как в теореме Кука-Левина:

$$\bar{R}(z) \iff \exists w \Psi(z, w).$$

Тогда определение языка будет иметь следующий вид:

$$\forall x: x \in L \iff \exists y_1 \forall y_2 \exists y_3 \dots \forall y_k \forall w \bar{\Psi}(x, y_1, y_2, \dots, y_k, w).$$

Аналогично можно объединить последние два квантора.

Следствие 4. Если в определении QBF_k заменить кванторы существования на кванторы всеобщности и наоборот, то получится $\Pi^k \mathbf{P}$ -полный язык.

1.7.1 Коллапс полиномиальной иерархии

Теорема 1.7.2. Если $\Sigma^k \mathbf{P} = \Pi^k \mathbf{P}$, то $\mathbf{PH} = \Sigma^k \mathbf{P}$.

- Достаточно показать, что следующий уровень равен текущему: $\Sigma^{k+1} \mathbf{P} = \Pi^k \mathbf{P}$.

Пусть $L \in \Sigma^{k+1} \mathbf{P}$, то есть $L = \{x \mid \exists y: R(x, y)\}$ для $R \in \Pi^k \mathbf{P} = \Sigma^k \mathbf{P}$.

Тогда существует $S \in \Pi^{k-1} \mathbf{P}$ такое, что

$$R(x, y) \iff \exists z: S(x, y, z).$$

После подстановки получаем

$$x \in L \iff \underbrace{\exists y}_{\exists t} \underbrace{\exists z: S(x, y, z)}_{S(x, t)}.$$

То есть $L \in \Sigma^k \mathbf{P}$.

Замечание. Доказали про следующие Σ классы, но из этого следует и, что Π классы тоже будут совпадать.

Следствие 5. Если существует \mathbf{PH} -полная задача, полиномиальная иерархия коллапсирует (конечна).

- Полный язык лежит в каком-то конкретном $\Sigma^k \mathbf{P}$ (потому что \mathbf{PH} есть их объединение), при этом все остальные к нему сводятся, следовательно, они тоже лежат в $\Sigma^k \mathbf{P}$.

1.8 Классы, ограниченные по времени и памяти

Определение 26: DSpace

$$\mathbf{DSpace}[f(n)] = \{L \mid L \text{ принимается ДМТ с памятью } \mathcal{O}(f(n))\},$$

где $f(n)$ должна быть неубывающей и вычислимой с памятью $\mathcal{O}(f(n))$ (на входе 1^n , на выходе двоичное представление $f(n)$)^a.

$$\mathbf{PSPACE} = \bigcup_{k \geq 0} \mathbf{DSpace}[n^k].$$

^aЭто определение конструируемой по памяти функции

1.8.1 PSPACE-полная задача

Определение 27: Язык QBF

Язык *QBF* — язык, состоящий из замкнутых истинных формул вида

$$q_1 x_1 q_2 x_2 \dots \varphi,$$

где φ — формула в КНФ, $q_i \in \{\forall, \exists\}$.

Теорема 1.8.1. Язык QBF PSPACE-полон.

□

- QBF \in PSPACE. Рассмотрим дерево перебора всех значений переменных. Когда мы дойдем до листа этого дерева и подставим значения переменных с ветки, получим значение функции. Значение

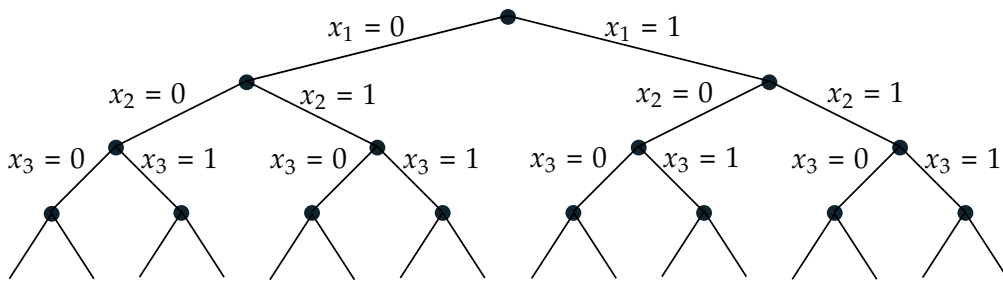


Figure 1.4: Дерево перебора

булевой формулы можем вычислить поиском в глубину.

Для этого понадобится память равная глубине дерева, то есть пропорциональная количеству кванторов, еще нужна память для вычисления значения φ , для этого тоже вполне хватит полинома.

- Сведем $L \in$ PSPACE к QBF. Для языка L есть МТ.

Модифицируем ее немного: если у нее было много принимающих состояний, например, из-за каких-то рабочих данных на других лентах, будем перед переходом в принимающее состояние очищать все, кроме q_{acc} .

Теперь построим граф, где конфигурации будут вершинами, а ребро между конфигурациями будет означать, что из одной можно получить другую.

Так как МТ детерминированная, выходить будет ровно одно ребро (кроме конечной).

Поскольку память полиномиальна, всего вершин будет $2^{p(n)}$, где $p(n)$ — некоторый полином.

Длина пути точно не более $2^{p(n)} + 1$, либо он заикливается.

Запишем теперь это в формулу. Пусть

$$\varphi_i(c_1, c_2) = \text{существует путь из } c_1 \text{ в } c_2 \text{ длины } \leq 2^i.$$

Возьмем вершину d посередине пути $c_1 \rightarrow c_2$. Заметим, что можно записать так

$$\varphi_i(c_1, c_2) = \exists d \forall x \forall y: ((x = c_1 \wedge y = d) \vee (x = d \wedge y = c_2)) \implies \varphi_{i-1}(x, y).$$

Формула, которую мы будем получать рекурсивно подставляя в $\varphi_{p(n)}(q_0, q_{acc})$, будет иметь длину $p^2(n)$ (всего шагов $p(n)$, на каждом записываем вектора длины $p(n)$), то есть полином.

$$\varphi_0(c_1, c_2) = \text{можно ли из первого состояния за один шаг получить второе.}$$

Так мы получили формулу из QBF.

Замечание. После построения формулы нужно будет привести ее к нужному виду, а именно, перенести кванторы в начало, перевести формулы в КНФ.



Следствие 6. Если $P = PSPACE$, то P коллапсирует.

1.8.2 Иерархия по памяти

Полезная ссылка: [Статья А.С.Охотина](#)

Теорема 1.8.2 (Об иерархии по памяти). $DSPACE[s(n)] \neq DSPACE[S(n)]$, где $s(n) = o(S(n))$ и для всех $n > n_0: S(n) \geq \log n$.

□ Рассмотрим следующий язык:

$$L = \left\{ x = M01^k \left| \begin{array}{l} k \in \mathbb{N} \cup \{0\}, \\ |M| < S(|x|), \\ M \text{ отвергает } x \text{ с памятью } \leq S(|x|) \end{array} \right. \right\}.$$

Заметим, что $L \in DSPACE[S(n)]$, так как можем промоделировать работу на универсальной машине Тьюринга, а памяти дополнительной она почти не использует.

Докажем, что $L \notin DSPACE[s(n)]$. Пусть M_* распознает L с памятью $s_*(|x|) = \mathcal{O}(s(|x|))$.

Выберем N_1 таким, что $\forall n > N_1: s_*(n) < S(n)$.

Еще найдем такое $N_* > \max(N_1, 2^{|M_*|})$.

Теперь посмотрим как себя ведет M_* . Пусть $x = M_*01^{N_* - |M_*| - 1}$.

Заметим, что M_* использует не более $s_*(|x|) < S(|x|)$ памяти. Кроме этого по построению $|x| = N_*$.

- Если $M_*(x) = 0$, третье условие выполнено, первое тоже. Пусть второе условие не выполнено. Тогда: $N_* > 2^{|M_*|} \geq 2^{S(|x|)} \geq |x| = N_*$, так как $S(n) \geq \log n$. Противоречие.
- Если $M_*(x) = 1$, то не выполняется третье условия языка L .

Противоречие. Получили, что $DSPACE[s(n)] \subsetneq DSPACE[S(n)]$.



1.8.3 Если памяти совсем мало...

Две теоремы без доказательств для общего развития.

Теорема 1.8.3. $DSPACE[\log \log n] \neq DSPACE[\mathcal{O}(1)]$

Теорема 1.8.4. $DSPACE[(\log \log n)^{1-\varepsilon}] = DSPACE[\mathcal{O}(1)]$

1.8.4 Иерархия по времени

Теорема 1.8.5 (Об иерархии по времени). $\mathbf{DTime}[t(n)] \neq \mathbf{DTime}[T(n)]$, где $t(n) \log t(n) = o(T(n))$, $T(n) = \Omega(n)$.

□ Аналогично ситуации с памятью рассмотрим следующий язык:

$$L = \left\{ x = M01^k \mid \begin{array}{l} k \in \mathbb{N} \cup \{0\}, \\ |M| < T(|x|), \\ M \text{ отвергает } x \text{ за время } \leq \frac{T(|x|)}{\log T(|x|)} \end{array} \right\}.$$

Также за счет универсальной МТ получаем принадлежность $\mathbf{DTime}[T(|x|)]$ (именно из-за этого нам нужен логарифм), она $f(n)$ шагов произвольной МТ моделирует за $O(f(n) \log f(n))$ шагов.

Остальное полностью аналогично иерархии по памяти. ■

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \dots \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP} = \bigcup_{k \geq 0} \mathbf{DTime}[2^{n^k}].$$

Про \mathbf{P} и \mathbf{EXP} мы знаем (по теореме об иерархии по времени), что есть неравенство. Следовательно, в каком-то из этих включений тоже должно быть строгое, но пока не известно какое.

1.9 Полиномиальные схемы

Определение 28: Size

$L \in \mathbf{Size}[f(n)]$, если существует семейство булевых схем $\{C_n\}_{n \in \mathbb{N}}$ такое, что

- $\forall n: |C_n| \leq f(n)$;
- $\forall x: x \in L \iff C_{|x|}(x) = 1$.

Определение 29: Полиномиальные схемы

$$\mathbf{P/poly} = \bigcup_{k \in \mathbb{N}} \mathbf{Size}[n^k].$$

Утверждение. $\mathbf{P} \not\subseteq \mathbf{P/poly}$.

□ Включение очевидно: любой ДМТ соответствует схема (этажи и всё такое, как мы делали раньше). Почему нет равенства точно? Например, $L = \{1^n \mid n \text{ — номер останавливающейся МТ}, M_n(M_n) = 1\}$. $L \notin \mathbf{P}$, но легко вычисляется схемами

$$C_n(x) = \begin{cases} 0, & 1^n \notin L \\ 0, & x \neq 1^n \\ 1, & 1^n \in L \end{cases}$$

■

Определение 30: Альтернативное определение $\mathbf{P/poly}$

$L \in \mathbf{P/poly}$, если имеются $R \in \mathbf{P}$ и последовательность строк (подсказок) $\{y_n\}_{n \in \mathbb{N}}$ полиномиальной длины (по одной, для каждой длины входа) таких, что

$$\forall x: x \in L \iff R(x, y_{|x|}) = 1.$$

Замечание (связь определений).

$1 \Rightarrow 2$ $\{C_n\}$ — подсказки, $R(x, C_{|x|}) = C_{|x|}(x)$.

$2 \Rightarrow 1$ Берем R , строим $C_n = C'_n(\dots, y_n)$ — подставили y_n (можем по Куку-Левину).

Лекция 5

3 dec

Теорема 1.9.1 (Карп-Липтон). $\mathbf{NP} \subseteq \mathbf{P/poly} \Rightarrow \mathbf{PH} = \Sigma^2\mathbf{P}$.

□ Покажем, что $\Sigma^3\mathbf{P}$ -полный язык лежит в $\Sigma^2\mathbf{P}$. Тогда все следующие классы тоже схлопываются. Мы знаем $\Sigma^3\mathbf{P}$ -полный язык

$$\mathbf{QBF}_3 = \{F \text{ — формула в КНФ} \mid \exists x \forall y \exists z F(x, y, z)\}.$$

Докажем, что он лежит в $\Sigma^2\mathbf{P}$.

Заметим, что корректность некоторой схемы $G \in \mathbf{SAT}$ можно проверить так: сначала подставим в первую переменную 0, а потом 1 и проверим полученные схемы. Исходная корректна, тогда хотя бы одна из полученных корректна.

Можно записать это так:

$$\forall G: C_{|G|} = C_{|G[x_1:=0]|} (G[x_1 := 0]) \vee C_{|G[x_1:=1]|} (G[x_1 := 1]).$$

Хотим доказать, что \mathbf{QBF}_3 можно уложить в два квантора. Сделаем это следующим образом:

$$\begin{aligned} (\exists x \forall y \exists z F) \in \mathbf{QBF}_3 &\iff \\ &\exists \text{ схемы } C_1, \dots, C_{|F|} \text{ размера, ограниченного полиномом} \\ &\exists x \\ &\forall y \\ &\forall G \text{ — булева формула длина не более } |F| \\ &(\text{семейство } \{C_i\} \text{ корректно для } G) \wedge C_{|F|}(F(x, y, z)) = 1 \end{aligned}$$

Такая запись принадлежит $\Sigma^2\mathbf{P}$.

Замечание. $\mathbf{NP} \subseteq \mathbf{P/poly}$ используем, когда проверяем схему из \mathbf{SAT} за полином.



1.9.1 Схемы фиксированного полиномиального размера

Теорема 1.9.2. $\forall k: \Sigma^4\mathbf{P} \not\subseteq \mathbf{Size}[n^k]$

□ Заметим, что существует функция $f: \{0, 1\}^n \rightarrow \{0, 1\}$, зависящая только от первых $c \cdot k \cdot \log n$ битов, для которой нет булевой схемы размера n^k , так как всего функций с ограничением на биты $2^{c \cdot k \cdot \log n}$, а схем $\mathcal{O}(2^{n^k \log n})$ (для каждого из n^k гейтов нужно задать два числа, каждое задается $\log n^k = \mathcal{O}(\log n)$).

Найдем такую в $\Sigma^4\mathbf{P}$:

$$\begin{aligned} y \in L &\iff \exists f \forall c (\text{схемы размера } n^k) \forall f' \exists x \exists c' (\text{схема размера } n^k) \forall x': \\ &\underbrace{f(x) \neq c(x)}_{\text{не принимается схемой}} \wedge \underbrace{((f < f') \vee f'(x') = c'(x'))}_{\text{лексикографически первая } f} \wedge \underbrace{(f(y) = 1)}_{\text{значение}} \end{aligned}$$

Все кванторы имеют полиномиальные размеры.



Следствие 7. $\forall k: \Sigma^2\mathbf{P} \cap \Pi^2\mathbf{P} \notin \mathbf{Size}[n^k]^a$.

^aЗдесь берется пересечение, так как схемам все равно, выдавать 0 или 1

□ Пусть $\Sigma^2\mathbf{P} \cap \Pi^2\mathbf{P} \subseteq \mathbf{Size}[n^k]$. Тогда $\mathbf{NP} \subseteq \mathbf{P}/\mathbf{poly}$, поэтому можно применить теорему Карпа-Липтона:

$$\mathbf{PH} = \Sigma^2\mathbf{P} \cap \Pi^2\mathbf{P} \subseteq \mathbf{Size}[n^k].$$

Но $\Sigma^4\mathbf{P} \subseteq \mathbf{PH} \subseteq \mathbf{Size}[n^k]$. Противоречие. ■

1.9.2 Класс NSpace

Определение 31: NSpace

$\mathbf{NSpace}[f(n)] = \{L \mid L \text{ принимается НМТ с памятью } \mathcal{O}(f(n))\}$.

Замечание.

- $f(n)$ должна быть конструируемая по памяти;
- входная лента read-only, выходная лента write-only, память на них не учитывается;
- **ленту подсказки** можно читать только слева направо.

$$\mathbf{NPSpace} = \bigcup_{k \geq 0} \mathbf{NSpace}[n^k].$$

1.10 Логарифмическая память

Определение 32

$\mathbf{STCON} = \{(G, s, t) \mid G \text{ — ориентированный граф, } \exists s \leadsto t\}$.

Лемма 2. $\mathbf{STCON} \in \mathbf{DSpace}[\log^2 n]$.

□ Будем делить путь пополам и искать путь от начала до середины и от середины до конца.

Пусть $\mathbf{PATH}(x, y, i)$ равно 1, если есть путь из x в y длины не более 2^i .

$$\mathbf{PATH}(x, y, 0) = (x, y) \in E$$

$$\mathbf{PATH}(x, x, 1) = 1$$

$$\mathbf{PATH}(x, y, i) = \bigvee_z (\mathbf{PATH}(x, z, i-1) \wedge \mathbf{PATH}(z, y, i-1))$$

Перебираем промежуточную вершину. Будем хранить «задания» (пары, для которых проверяем путь, и еще логарифм длины пути) в стеке. Достаем оттуда одно «задание» и заменяем его на два меньших. Когда-то мы либо найдем 1, тогда нужно вернуться к проверке второй половины, либо 0, тогда нужно перейти к следующей промежуточной вершине, так как с текущей пути нет.

Чтобы получить ответ, посчитаем $\mathbf{PATH}(s, t, \log |V|)$.

Оценим память: на счетчик промежуточных вершин и на элемент стека нужен логарифм, максимальная глубина стека тоже $\log n$. Тогда всего не более $\log^2 n$. ■

Теорема 1.10.1. $\mathbf{NSpace}[f(n)] \subseteq \mathbf{DSpace}[f^2(n)]$ для $f(n) = \Omega(\log n)$.

□ Пусть язык $L \in \mathbf{NSpace}[f(n)]$. Это значит, что есть НМТ, которая принимает его с памятью $\mathcal{O}(f(n))$. Построим детерминированный алгоритм, который выполнит ту же работу за $\mathcal{O}(f^2(n))$.

Построим граф псевдоконфигураций: вершина — состояние, содержащее рабочие ленты и положение всех головок.

Всего памяти для хранения псевдоконфигураций требуется $2^{f(n) \cdot k}$ памяти: всего состояний конечное число, положение занимает $\log n$, остается содержимое.

Считаем, что у нас только одно принимающее состояние.

Заметим, что граф нам нужен, когда хотим выяснить достижимость за один шаг.

Делаем такой же стек, как и выше, на него уходит порядка $f(n)$ памяти (на каждую «задачу» три числа: начало, конец и длина).

Теперь посмотрим на «задачу» $(z, t, 0)$, до которой мы в какой-то момент дошли. Мы хотим выяснить, есть ли переход из z в t , но явного графа у нас нет. Для этого сравниваем конфигурации посимвольно, кроме состояния (и, возможно, текущего символа).

Вспоминаем, что наша задача — проверить, что НМТ M принимает данный вход.

Поэтому для сравнения z и t осталось посмотреть табличку перехода M (константного размера) и на текущий символ на входной ленте (его индекс хранится вместе с рабочими индексами) и понять, можно ли из z получить t .

Так как один стек занимает $\mathcal{O}(f(n))$ памяти, то для всей работы достаточно $\mathcal{O}(f^2(n))$.

Итого, мы построили нужный детерминированный алгоритм. ■

Следствие 8. $\mathbf{PSPACE} = \mathbf{NPSPACE}$

Определение 33: \mathbf{L} и \mathbf{NL}

Класс языков \mathbf{L} — множество языков, разрешимых на детерминированной машине Тьюринга с использованием $\mathcal{O}(\log n)$ дополнительной памяти для входа длиной n .

Класс языков \mathbf{NL} — множество языков, разрешимых на недетерминированной машине Тьюринга с использованием $\mathcal{O}(\log n)$ дополнительной памяти для входа длиной n .

Лемма 3. \mathbf{STCON} является \mathbf{NL} -полной относительно $\log\text{space}$ -сведений^a.

^aТо есть сводящая функция использует логарифмическую память.

□

- Принадлежность \mathbf{NL} : храним только текущую вершину и угадываем следующее состояние, пока не дойдем до конца.
- Полнота: если есть задача из \mathbf{NL} , то есть НМТ M , которая использует логарифмическую память. M принимает вход, если в дереве вычислений есть путь от стартовой вершины до принимающей:

$$M(x) = 1 \iff \text{start} \rightsquigarrow \text{accept}.$$

Поэтому функция сведения должна построить граф по машине Тьюринга для данного входа x . ■

Утверждение. Для неориентированного графа алгоритм гораздо сложнее: $\mathbf{USTCON} \in \mathbf{L}$ (Reingold, 2004).

Утверждение. Все задачи из \mathbf{L} являются \mathbf{L} -полными (относительно $\log\text{space}$ сводимости).

1.11 Равномерные полиномиальные схемы

Определение 34

Семейство схем $\{C_n\}_{n \in \mathbb{N}}$ **равномерно**, если существует полиномиальный алгоритм A такой, что $A(1^n) = C_n$.

Лемма 4. Равномерные схемы задают класс P .

□ Полиномиальная МТ на входе x сначала запускает $A(|x|)$, потом запускает полученную схему $C(x)$.

Наоборот, пусть дана полиномиальная МТ, про вход мы знаем, что $|x| = n$, тогда можно нарисовать схему из состояний, которая будет работать как МТ. Каждый уровень можно породить алгоритмом, который будет их пересчитывать по функции перехода. ■

Определение 35

Семейство схем $\{C_n\}_{n \in \mathbb{N}}$ **logspace-равномерно**, если существует полиномиальный алгоритм A , который использует $\mathcal{O}(\log n)$ памяти и $A(1^n) = C_n$.

Глубина булевой схемы — время параллельного вычисления.

Определение 36: Nick's class

Класс $NC^i = \{L \mid \text{для } L \text{ есть logspace-равномерные схемы глубины } \mathcal{O}(\log^i n)\}$.

Класс $NC = \bigcup_i NC^i \subseteq P$.

Замечание. Еще есть класс AC^i , который определяется аналогично NC^i , только разрешает степень входа гейта больше двух.

Лемма 5. Композиция двух logspace-функций $f_2(f_1(x))$ принадлежит logspace.

□ Будем запускать⁵ и f_1 и f_2 . Храним счетчики позиций на лентах: для f_1 на выходе и для f_2 на входе.

Пусть f_2 нужен очередной бит входа с номером y_i . Запускаем f_1 , как только ее счетчик достигает y_i , останавливаем f_1 , продолжаем работу f_2 .

Если f_2 требуется новый y_j (возможно $j < i$), то мы запускаем f_1 снова (с самого начала).

В итоге у нас хранится только вход для f_1 и индексы, а он не входит в рабочую память.

И так пока f_2 не закончит. Во время работы f_1, f_2 запущено только два экземпляра, поэтому память логарифмическая (для счетчиков – логарифм размера ленты).

Замечание. f_1 мы моделируем без выходной ленты, помним только последний символ. ■

Теорема 1.11.1. Если L — P -полный (относительно logspace-сводимости), то $L \in L \iff P = L$.

□

- Докажем, что $L \in L \Rightarrow P = L$. Если $L' \in P$, то он сводится к L , так как второй P -полный. А L решается logspace-алгоритмом. Получили две функции, обе logspace, значит, по доказанной лемме их композиция тоже logspace, поэтому $L' \in L$.

С другой стороны, $L \subseteq P$, так как $L \in P$ и L является L -полной.

- В обратную сторону. Пусть $P = L$, по условию L — P -полный, поэтому $L \in L = P$.

⁵считаем, что функции — МТ, которые принимают аргумент на входной ленте и возвращают результат на выходной

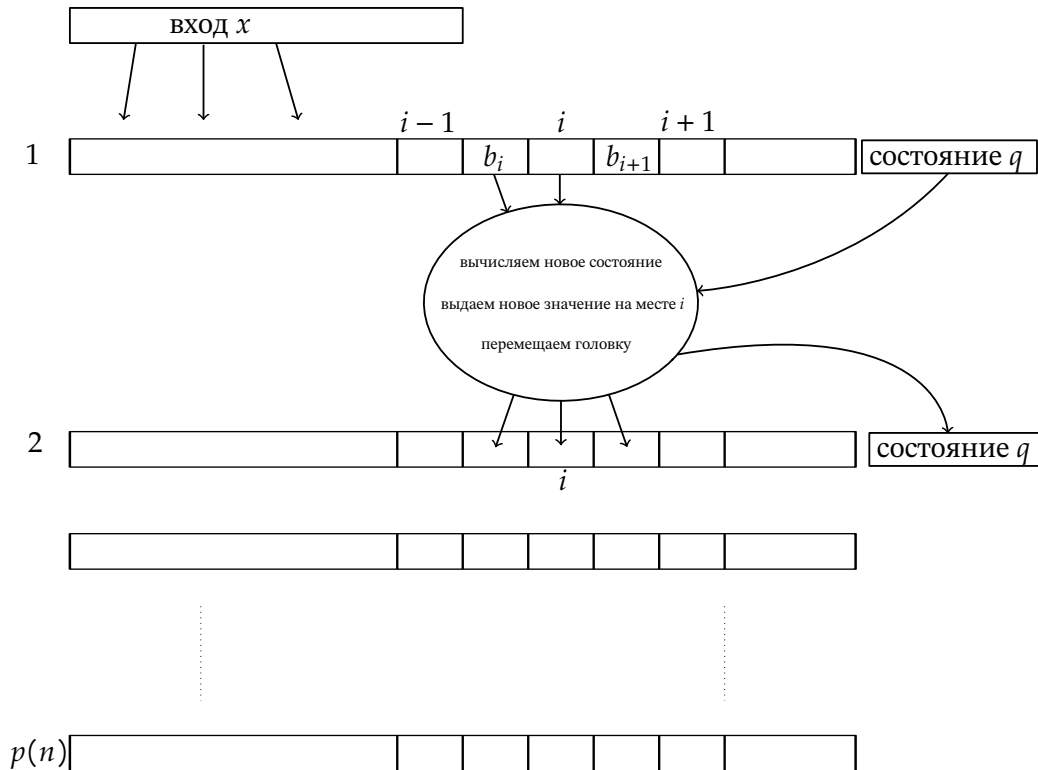


Figure 1.5: Построение схемы по ДМТ

Пример 1.11.1: Р-полный язык

Приведем пример **Р**-полного языка.

$$\text{CIRCUIT_EVAL} = \{(\text{схема } C, \text{ вход } x) \mid C(x) = 1\}.$$

Нам уже дан вход, остается только его проверить. Это легко и быстро сделать.

Докажем полноту. Вспомним, как строили схему по недетерминированной МТ в доказательстве теоремы Кука-Левина. Построим аналогично по детерминированной: каждый слой отвечает за состояние, переход к следующему – маленькие подсхемки, на вход только x . Перед каждым символом запишем 1, если головка сейчас на нем, иначе – 0.

Единственное, что нужно проверить — что нарисовать можно с логарифмической памятью.

Описание машины — константа. От x нам нужна только длина, чтобы узнать количество этажей.

На каждом уровне, чтобы построить следующий, нам нужно знать положение головки, три бита прошлой строки и состояние. Будем хранить t — номер уровня, i — положение головки, логарифма для этого хватит.

Теорема 1.11.2. $\text{NC}^1 \subseteq \text{L} \subseteq \text{NL} \subseteq \text{NC}^2$.

□

1. $\text{NL} \subseteq \text{NC}^2$. Начнем с последнего включения. У нас есть недетерминированная машина, которая использует логарифмическую память, хотим ее параллелизовать.

Вспомним про граф псевдоконфигураций нашей детерминированной машины. Так как память логарифмическая, конфигураций полиномиальное число.

Можем задать допустимые переходы НМТ матрицей смежности A ($k \times k$). Чтобы найти пути длины k (а больше нам не нужно), возведем булево в степень k .

Это $\log k$ последовательных умножений: $A^2, (A^2)^2, \dots$, причем каждое умножение пары булевых матриц вычисляется схемой глубины $\mathcal{O}(\log k)$.

Тогда общая глубина $\log^2 k$, из чего следует последнее включение.

2. $\mathbf{L} \subseteq \mathbf{NL}$. Среднее включение тривиально.

3. $\mathbf{NC}^1 \subseteq \mathbf{L}$. Докажем первое. Мы знаем, что \logspace функции можно комбинировать и получится тоже \logspace .

Сейчас находимся в \mathbf{NC}^1 , поэтому у нас есть семейство схем $\{C_n\}$, каждая вычисляется с логарифмической памятью.

Построим композицию трех функций:

- Первая функция вычисляет схему. Заметим, что эта схема будет логарифмической глубины (так как мы в \mathbf{NC}^1).
- Докажем, что эту схему можно преобразовать в формулу тоже с логарифмической памятью. Пусть есть некоторая схема. Склонируем все гейты, которые используются два раза в следующих гейтах. Так как глубина логарифмическая, больше полинома вершин в полученном дереве не окажется. Научимся делать это формально с \logspace . Запустим dfs от выхода схемы и

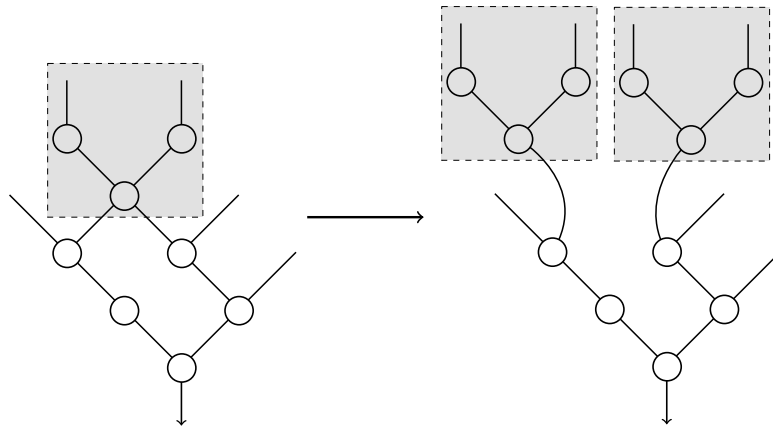


Figure 1.6: Схема \rightarrow дерево

будем записывать все пути. Заметим, что каждый путь представляет из себя как раз одну из полученных веток с клонированными уникальными гейтами.

После построения одного пути, начинаем из корня, храним только текущую вершину. «Сам путь нам подскажет, где мы ходили налево, а где еще нет»⁶.

- Теперь вычисляем значение формулы на входе x . Аналогично запускаем dfs, храним текущую вершину и результаты в поддеревьях, когда обошли оба поддерева, поднимемся выше и передадим результат операции гейта.

■

Теорема 1.11.3. Если L — \mathbf{P} -полный, то все параллелизуется

$$L \in \mathbf{NC} \iff \mathbf{P} = \mathbf{NC}.$$

□ Если $L' \in \mathbf{P}$, его можно свести к L (\logspace), а $L \in \mathbf{NC}$. То есть $\exists i: L \in \mathbf{NC}^i$.

Логарифмические вычисления мы умеем превращать в параллельные с \log^2 , поэтому $L' \in \mathbf{NC}^{i+2}$. ■

⁶Тогда уж «сила подскажет», можно сделать так: храним еще одну примерно $\log n$ -битную переменную, в которой i -й бит равен 1, если на i -ом уровне мы уже ходили в левое поддерево.

1.11.1 Замкнутость NSpace относительно дополнения

Теорема 1.11.4 (Immerman, Szelepcsényi). $\text{STCON} \in \mathbf{co-NL}$. Альтернативная формулировка $\overline{\text{STCON}} \in \mathbf{NL}$.

Хотим: недетерминированно с логарифмической памятью принимать те, где нет пути в орграфе

□ Придумаем такие подсказки, которые будут нас убеждать, что пути между s и t нет.

Пусть S_i — множество вершин на расстоянии не более i от s .

Сертификат того, что $x \in S_i$ — последовательность вершин от s до x .

Так как подсказки будут в недетерминированной подсказке, память занимать они не будут.

Теперь посмотрим на такой сертификат *непринадлежности* ($x \notin S_i$):

- все вершины S_i с сертификатами принадлежности,
- размер S_i ⁷

Теперь нужно проверить все вершины из S_i , что ни одна из них не совпала с x , и, что их столько, сколько нужно.

Будем строить сертификат $|S_i|$ индуктивно так:

- $|S_{i-1}|$ мы знаем,
- переберем все u и проверим, что $u \in S_i$ так:
 - переберем все v (нам интересно, могли ли они быть предыдущими на пути), проверяя $(v, u) \in E$, и, если такое ребро есть, проверяем сертификат принадлежности (то есть путь) для $v \in S_{i-1}$,
 - в процессе проверки, подсчитываем количество правильных сертификатов, в итоге должно сойтись с $|S_{i-1}|$.

Теперь мы знаем общее число вершин из S_i .

Ответом будет $t \notin S_{|V|-1}$.

Замечание. Что мы храним? Храним номера вершин и счетчики, поддерживаем простые арифметические операции. Еще нужно хранить обращение к входу (несколько счетчиков для поиска ребер), считаем, что граф записан в нормальном виде, например, как перечисление ребер или матрица смежности.



Следствие 9. Если $s(n) = \Omega(\log n)$, то $\mathbf{NSpace}[s(n)] = \mathbf{coNSpace}[s(n)]$.

□ Пусть есть МТ M для языка из $\mathbf{NSpace}[s(n)]$. Тогда нас интересует достижимость в графе конфигураций из $2^{s(n)}$ состояний.

Только что мы научились проверять достижимость в классе $\mathbf{co-NL}$, то есть доказывать недостижимость в классе \mathbf{NL} .

$$\log 2^{s(n)} \approx s(n).$$

Поэтому теперь мы можем запустить алгоритм из прошлой теоремы, единственное, что нужно изменить — теперь не нужно читать граф с входной ленты, а нужно проверять смежность двух конфигураций. ■

⁷Это тоже нужно сертифицировать и об этом будет написано ниже.

1.12 Вероятностные вычисления

1.12.1 Классы с односторонней ошибкой

Определение 37: Класс RP

$L \in \mathbf{RP}$, если существует п.о.п.п. отношение R такое, что $\forall x \in \{0, 1\}^*$:

$$\begin{aligned} x \notin L &\implies \forall w: (x, w) \notin R \\ x \in L &\implies \frac{|\{w \mid (x, w) \in R\}|}{|\{\text{все } w\}|} > \frac{1}{2} \end{aligned}$$

Определение 38: Классы без ошибки

$\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{co-RP}$.

Определение 39: Классы с двусторонней ошибкой

$L \in \mathbf{BPP}$, если существует п.о.п.п. отношение R такое, что $\forall x \in \{0, 1\}^*$:

$$\begin{aligned} x \notin L &\implies \frac{|\{w \mid (x, w) \in R\}|}{|\{\text{все } w\}|} < \frac{1}{3} \\ x \in L &\implies \frac{|\{w \mid (x, w) \in R\}|}{|\{\text{все } w\}|} > \frac{2}{3} \end{aligned}$$

Лемма 6. Пусть некоторый язык принадлежит \mathbf{RP} . Запустим соответствующий алгоритм k раз. Тогда

$$\Pr[k \text{ неудач}] \leq \frac{1}{2^k}.$$

□ Очевидно



Лемма 7. Пусть некоторый язык принадлежит \mathbf{BPP} . Запустим соответствующий алгоритм k раз, вернем самый частый ответ. Тогда

$$\Pr[\text{ошибок более } \frac{k}{2}] \leq \frac{1}{2^{\Omega(k)}}.$$

□

Утверждение (Неравенство Чернова, без доказательства).

$$\Pr[X > (1 + \varepsilon)pk] < \left(\frac{e^\varepsilon}{(1 + \varepsilon)^{1+\varepsilon}} \right)^{pk} \leq e^{-\frac{pk\varepsilon^2}{4}},$$

где $X = \sum_{i=1}^k x_i$, x_i — независимые случайные величины, принимающие 1 с вероятностью p и 0 с вероятностью $1 - p$.

В данной задаче x_i — наличие ошибки при i -ом вычислении, $p = \frac{1}{3}$, $\varepsilon = \frac{1}{2}$.



Определение 40: Альтернативное определение ZPP

\mathbf{ZPP} — алгоритмы без ошибок с полиномиальным матожиданием времени работы.

□

- Пусть есть два алгоритма: один **RP**, второй **co-RP**. Пока первый выдает 0, мы не уверены в ответе, и, пока второй выдает 1 мы не уверены.

Когда-то один из них выдаст правильный ответ. Тогда матожидание равно

$$\mathbb{E} = \mathcal{O}\left(\left(\frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + 4 \cdot \frac{1}{16} + \dots\right)t(n)\right) = \mathcal{O}(t(n)),$$

где $t(n)$ — время самого долгого из алгоритмов.

- Пусть есть алгоритм A , для которого $\mathbb{E}t_A(n) \leq p(n)$.

Запустим его $k \cdot p(n)$ раз и прервем. Тогда вероятность неверного ответа по неравенству МАРКОВА, меньше $\frac{1}{k}$, а время на выполнения все еще полиномиально.



1.12.2 Связь с другими классами

Теорема 1.12.1. $BPP \subset P/poly$.



- Вероятностный алгоритм дополнительно получает случайную строку (подсказку)⁸.

Назовем ее *хорошей* подсказкой для входа x , если она приводит к правильному ответу, и *плохой* иначе.

Давайте уменьшим ошибку до $\frac{1}{4^n}$.

- Хотелось бы найти хорошую строку для всех входов x . Входов длины n всего 2^n штук.

Заметим, что $\frac{1}{4^n} \cdot 2^n < 1$, поэтому доля плохих строк для всех входов точно меньше единицы, поэтому есть хорошая для всех входов.

- Построим булеву схему по нашему алгоритму.

BPP задается полиномиальным алгоритмом, у которого два аргумента: вход и случайная строка. Тогда это просто НМТ. По теореме Кука-Левина мы можем построить булеву схему, соответствующую НМТ. У этой схемы есть часть входов для x и часть для случайной строки w . Просто зашьем вместо

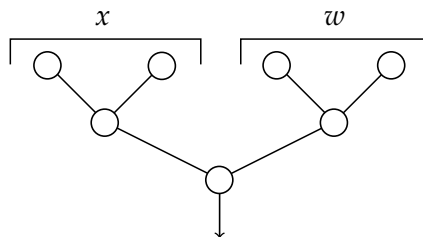


Figure 1.7: Булева схема для **BPP**

w одну из хороших для всех входов строк.



⁸Всегда можно считать, что ее длина не больше времени работы, так как дальше мы ее просто не прочитаем. Еще можно считать, что длины всех случайных строк равны.

Теорема 1.12.2. $\text{BPP} \subseteq \Sigma^2\text{P}$.

□ Пусть есть язык $R \in \text{BPP}$. Пусть вероятность ошибки мы уже уменьшили до 2^{-n} .

$$A_x = \{w \in \{0,1\}^{p(n)} \mid R(x, w) = 1\}.$$

Построим формулу с двумя кванторами.

- $x \in L$. Будем покрывать все возможные случаи строки $U = \{0,1\}^{p(n)}$ копиями A_x .
Если мы покроем полиномиальным количеством копий, то получим следующее:

$$\exists \{t_i\}_{i=1}^k \forall r \in U \bigvee_{i=1}^k (r \in A_x \oplus t_i).$$

После первого квантора полином битов, после второго тоже, поэтому, чтобы данная схема была из

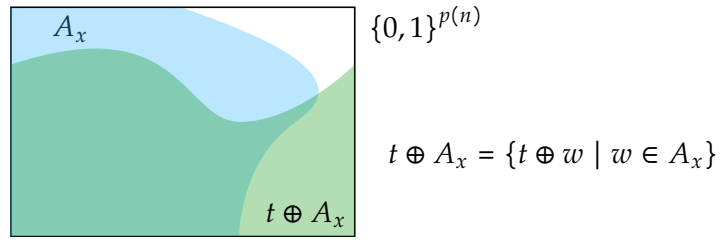


Figure 1.8: Покрытие случайных строк

$\Sigma^2\text{P}$, нужно проверить, что все вычисления работают полином.

- $x \notin L$. В этом случае за полином A_x мы не покроем все U , так как их доля $\frac{1}{2^n}$.

Для первого случая осталось доказать, что, во-первых, можно найти полиномиальное множество $\{t_i\}$, а, во-вторых, вычисления можно провести за полиномиальное время.

1. Выберем случайно множество $\{t_i\}_i$ и убедимся, что вероятность полного покрытия больше нуля.

Запишем вероятность того, что мы не покрыли все строки:

$$\begin{aligned} \Pr \left[\neg \left(\forall r \in U \bigvee_{i=1}^k (r \in A_x \oplus t_i) \right) \right] &= \Pr \left[\exists r \in U \bigwedge_{i=1}^k (r \notin A_x \oplus t_i) \right] \leq \\ &\leq \sum_{r \in U} \Pr \left[\bigwedge_{i=1}^k (r \notin A_x \oplus t_i) \right] \stackrel{\text{выбирали независимо}}{=} \sum_{r \in U} \prod_{i=1}^k \Pr[r \notin A_x \oplus t_i] \leq \\ &\leq \frac{1}{2^{nk}} \cdot 2^{p(n)} \end{aligned}$$

Чтобы сделать $\frac{2^{p(n)}}{2^{nk}} < 1$, возьмем, например, $k = p(n)$.

2. Посчитать \forall для k значений легко. Заметим, что

$$r \in A_x \oplus t_i \iff r \oplus t_i \in A_x.$$

Чтобы выяснить последнее нужно запустить $R(x, r \oplus t_i)$, так как A_x — как раз строки, на которых R выдает единицу.

R работает полином, следовательно, все вычисления будут работать тоже полином.

Замечание. Так как $\text{BPP} = \text{co-BPP}$, $\text{BPP} \subseteq \Pi^2\text{P}$, а тогда $\text{BPP} \subseteq \Sigma^2\text{P} \cap \Pi^2\text{P}$.

1.13 Общая картина

Ничего не известно про связь **BPP** с **NP** и **co-NP**

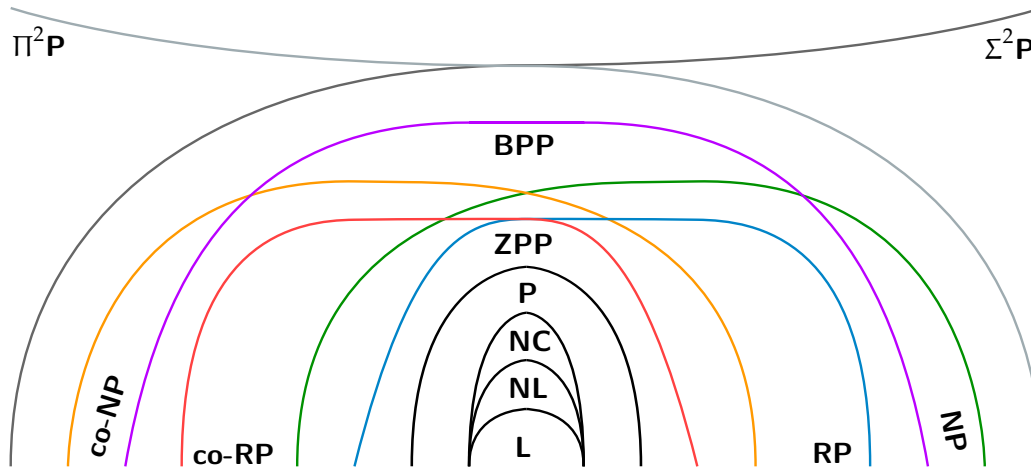


Figure 1.9: Общая картина иерархии

1.14 Квантовые вычисления

Недетерминированные МТ

Посмотрим на вычисления как на матрицы: $A(u, v) = 1 \iff u \xrightarrow{\text{шаг МТ}} v$.

Заметим, что не любая матрица подойдет под описание МТ. Это объясняется тем, что при переходе от одной конфигурации меняется константное число битов и один символ на ленте. И поэтому много участков памяти остаются прежними.

Детерминированные МТ

Мы аналогично можем записать все в матрицу, но в каждой строке будет ровно одна единица ($\sum_j a_{ij} = 1$), так как мы точно должны знать, куда попадем на следующем шаге.

Текущее состояние можно записать как строку $x = (0, \dots, 0, 1, 0, \dots, 0)$, где 1 на i -ом месте говорит, что мы сейчас в i -ом состоянии.

Теперь мы можем умножать транспонированный вектор на матрицу и получать следующее состояние.

$$x \mapsto xA \mapsto xA^2 \mapsto \dots$$

Вероятностные МТ

С вероятностными МТ можно в матрице писать в возможных переходах писать вероятность $\frac{1}{2}$ (так как у нас случайные биты, всего два варианта развития событий).

Аналогично, $\sum_j a_{ij} = 1$. Такая матрица называется **стохастической**.

$$\delta(q, a, 0) = (q', b, \rightarrow / \leftarrow)$$

$$\delta(q, a, 1) = (q'', c, \rightarrow / \leftarrow)$$

Теперь можно думать о состоянии, как о «сумме состояний»: сумма произведений вероятности на прошлое состояние.

Квантовые МТ

Теперь в матрице записаны некоторые комплексные числа, которые называются **амплитудами**.

Матрица должна быть унитарной, $AA^* = A^*A = E$.

Наше текущее положение дел также записано в строку матрицы.

Смешанное состояние — линейная комбинация чистых конфигураций с амплитудами

$$\sum_i x_i \cdot s_i.$$

Вероятность — $|x_i|^2$, это примерно длина вектора.

Эволюция происходит аналогично, домножаем строку на матрицу. Эволюция происходит пока мы не захотим получить ответ.

Матрица все равно должна быть получена из МТ или другого устройства, мы не можем осуществлять какие-то нелогичные переходы, не понятна их реализация.

Преобразования тоже должны быть унитарными с небольшим числом бит.

О квантовых МТ можно думать, как о булевых схемах с маленькими унитарными преобразованиями.

Пример 1.14.1

Пусть таким переходом будет матрица 2×2 , применяем к $(1, 0)$.

$$\begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

Результатом будет 1 с вероятностью $\frac{1}{2}$ и 0 с вероятностью $\frac{1}{2}$.

Если же применить дважды, так как матрица обратная к себе, получим только 0.

Определение 41

BQP — класс решаемых квантовым компьютером за полиномиальное время с ограниченной ошибкой.

Part II

Теория вычислимости

Chapter 1

Вычислимость. Система вычислимости по Клини

1.1 Рекурсивные функции

Лекция 1
11 feb

Определение 42

Пусть функция $f: \mathbb{N}^k \rightarrow \mathbb{N} \cup \{\uparrow\}$, $k \in \mathbb{N}$, где $\mathbb{N} = \{0, 1, 2, \dots\}$ ^a. Такая функция называется ***k-местной частичной функцией***. Если $k = 0$, то $f = \text{const}$.

^aМы здесь считаем ноль натуральным числом

1.1.1 Простейшие функции

Простейшими будем называть следующие функции:

- Нуль местный нуль — функция без аргументов, возвращающая 0;
- Одноместный нуль — $0(x) = 0$;
- Функция следования — $s(x) = x + 1$;
- Функция выбора (проекция) — $I_n^m(x_1, \dots, x_n) = x_m$

1.1.2 Операторы

Определим три оператора:

Определение 43

- Функция f **получается оператором суперпозиции** из функций h и g_i , где

$$h(y_1, \dots, y_m), g_i(x_1, \dots, x_n); 1 \leq i \leq m,$$

если

$$f(x_1, \dots, x_n) = h(g_1(x_1, \dots, x_n), \dots, g_m(x_1, \dots, x_n)).$$

Оператор обозначается **S**.

- Функция $f^{(n+1)}_a$ **получается оператором примитивной рекурсии** из $g^{(n)}$ и $h^{(n+2)}$, если

$$\begin{cases} f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n) \\ f(x_1, \dots, x_n, y + 1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)) \end{cases}$$

Оператор обозначается **R**.

- Функция f задается **оператором минимизации (M)**, если она получается из функции g :

$$\begin{aligned} f(x_1, \dots, x_n) &= \mu y [g(x_1, \dots, x_n, y) = 0] = \\ &= \begin{cases} y & g(x_1, \dots, x_n, y) = 0 \wedge g(x_1, \dots, x_n, i)^b \neq 0 \forall i < y \\ \uparrow^c & \text{else} \end{cases} \end{aligned}$$

^aЗдесь и далее $f^{(n)}$ обозначается функция, принимающая n аргументов, то есть n -местная

^bподразумевается, что функция определена в этих точках

^cне определена

Пример 1.1.1

$$x - y = \begin{cases} x - y, & x \geq y \\ \uparrow, & x < y \end{cases}$$

Можно задать, используя оператор минимизации:

$$x - y = \mu z [|(y + z) - x| = 0].$$

1.1.3 Функции

Определение 44: Примитивно рекурсивная функция

Функция f называется **примитивно рекурсивной (ПРФ)**, если существует последовательность таких функций f_1, \dots, f_k , что все f_i либо простейшие, либо получены из предыдущих f_1, \dots, f_{i-1} с помощью одного из операторов **S** и **R** и $f = f_k$.

Пример 1.1.2

Докажем, что $f(x, y) = x + y$ — **ПРФ**. По **R** можем получить f так:

$$\begin{cases} f(x, 0) &= x = I_1^1(x) \\ f(x, y + 1) &= (x + y) + 1 = s(f(x, y)) = s(I_3^3(x, y, f(x, y))) \end{cases}$$

Теперь построим последовательность функций f_i , где последним элементом будет f , полученный с помощью **R**:

$$g = I_1^1, s, I_3^3, h = S(s, I_3^3), f = R(g, h).$$

Определение 45: Частично рекурсивная функция

Функция f называется **частично рекурсивной функцией (ЧРФ)**, если существует последовательность функций f_1, \dots, f_k , таких что f_i либо простейшая, либо получается из предыдущих с помощью одного из операторов **S**, **R**, **M**.

Замечание. Частично рекурсивная функция может быть не везде определена. Примитивно рекурсивная определена везде.

Замечание. Существуют частично рекурсивные функции, которые всюду определены, но при этом не являются **ПРФ**.

Определение 46

Общерекурсивная функция — всюду определенная частично рекурсивная.

Пример 1.1.3

$\mu y[x + y + 1 = 0]$ — нигде не определена, но получается из последовательности других функций с помощью операторов.

Лемма 8. Следующие функции являются **ПРФ**:

1. $\text{const}^{(n)}$

2. $x + y$

3. $x \cdot y$

4. x^y , где 0^0 можем определить, как хотим

5. $\text{sg}(x) = \begin{cases} 0 & x = 0 \\ 1 & x \neq 0 \end{cases}$

6. $\overline{\text{sg}}(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0 \end{cases}$

7. $x \div 1 = \begin{cases} 0 & x = 0 \\ x - 1 & x > 0 \end{cases}$

8. $x \div y = \begin{cases} 0 & x < y \\ x - y & \text{else} \end{cases}$

9. $|x - y|$



1. Сначала можем получить нужное число последовательной суперпозицией функции следования (получили константу от одной переменной), затем проецируем I_1^{n+1} , чтобы получить n переменных (первая - наша константа).

2. Доказали выше в примере 1.1.2.

3. $f(x, y) = xy$ определим так:

$$\begin{cases} f(x, 0) & = 0 \\ f(x, y + 1) & = f(x, y) + x \end{cases}$$

а складывать мы умеем.

4. $f(x, y) = x^y$:

$$\begin{cases} f(x, 0) & = 1 = s(0) \\ f(x, y + 1) & = f(x, y) \cdot x \end{cases}$$

Умножать тоже можно по третьему пункту.

5. $\text{sg}(x) = \begin{cases} 0 & x = 0 \\ 1 & x \neq 0 \end{cases}$

$$\begin{cases} \text{sg}(0) & = 0 \\ \text{sg}(x + 1) & = 1 = s(0) \end{cases}$$

6. Аналогично

$$7. f(x) = x \div 1$$

$$\begin{cases} f(0) &= 0 \\ f(x+1) &= x = I_1^1(x) \end{cases}$$

$$8. f(x, y) = x \div y$$

$$\begin{cases} f(x, 0) &= x = I_1^1(x) \\ f(x, y+1) &= f(x, y) \div 1 \end{cases}$$

$$9. f(x, y) = |x - y| = (x \div y) + (y \div x)$$

Замечание. Обычное вычитание не является **ПРФ**, так как не везде определено на \mathbb{N} .

1.1.4 Оператор ограниченной минимизации

Определение 47: Оператор ограниченной минимизации

Функция $f^{(n)}$ задается **оператором ограниченной минимизации** из функций $g^{(n+1)}$ и $h^{(n)}$, если

$$\mu y \leq h(\bar{x}) [g(\bar{x}, y) = 0]^a.$$

Это означает, что

$$f(\bar{x}) = \begin{cases} y & g(\bar{x}, y) = 0 \wedge y \leq h(\bar{x}) \wedge g(\bar{x}, i) \neq 0^b \forall i < y \\ h(\bar{x}) + 1 & \text{else} \end{cases}$$

^aЗдесь и далее $\bar{x} = x_1, \dots, x_n$.

^bАналогично, подразумевается, что функция определена в этих точках

Утверждение. Пусть $g^{(n+1)}, h^{(n)}$ — примитивно рекурсивные функции, и $f^{(n)}$ получается из g и h с помощью ограниченной минимизации, то f тоже **ПРФ**.

□ Заметим, что f можно получить следующим образом:

$$f(\bar{x}) = \sum_{y=0}^{h(\bar{x})} \prod_{i=0}^y \text{sg}(g(\bar{x}, i)).$$

Внутреннее произведение равно единице только тогда, когда все $g(\bar{x}, i) \neq 0$. Если для некоторого y обнуляется $g(\bar{x}, y)$, то все произведения, начиная с $y+1$, будут равны нулю, поэтому просуммируем только y единиц. Если же такого y нет, получим сумму из $h(\bar{x}) + 1$ единицы. Именно это и нужно.

Проверим, что можно получить

$$a(\bar{x}, y) = \sum_{i=0}^y g(\bar{x}, i), \quad m(\bar{x}, y) = \prod_{i=0}^y g(\bar{x}, i)$$

с помощью примитивной рекурсии:

$$\begin{cases} a(\bar{x}, 0) &= g(\bar{x}, 0) \\ a(\bar{x}, y+1) &= a(\bar{x}, y) + g(\bar{x}, y+1) \end{cases} \quad \begin{cases} m(\bar{x}, 0) &= g(\bar{x}, 0) \\ m(\bar{x}, y+1) &= m(\bar{x}, y) \cdot g(\bar{x}, y+1) \end{cases}$$

Замечание. $0(x)$ можно исключить из определения простейших функций, так как ее можно получить с помощью оператора \mathbf{R} для нульместного 0 и $I_2^2(x, y)$:

$$0(y) = \begin{cases} 0(0) & = 0 \\ 0(y+1) & = I_2^2(y, 0) \end{cases}$$

1.1.5 Предикаты

Определение 48

Предикат — условие задающее подмножество: $R \subset \mathbb{N}^k$.

Предикат называется **примитивно рекурсивным (общерекурсивным)**, если его характеристическая функция примитивно рекурсивная (общерекурсивная).

$$\chi_R(\bar{x}) = \begin{cases} 1, & \bar{x} \in R \\ 0, & \bar{x} \notin R \end{cases}$$

Утверждение.

- Если R, Q — примитивно рекурсивные (общерекурсивные) предикаты, то предикаты $P \vee Q, P \wedge Q, P \rightarrow Q, \neg P$ тоже примитивно рекурсивные (общерекурсивные).
- Предикаты $=, \leq, \geq, <, >$ тоже примитивно и общерекурсивны.

□

- Проверим, что характеристические функции примитивно / общерекурсивны:

$$\begin{aligned} \chi_{P \wedge Q}(\bar{x}) &= \chi_P(\bar{x}) \cdot \chi_Q(\bar{x}) \\ \chi_{P \vee Q}(\bar{x}) &= \text{sg}(\chi_P(\bar{x}) + \chi_Q(\bar{x})) \\ \chi_{P \rightarrow Q}(\bar{x}) &= \text{sg}(\overline{\text{sg}}(\chi_P(\bar{x})) + \text{sg}(\chi_Q(\bar{x}))) \\ \chi_{\neg P}(\bar{x}) &= \overline{\text{sg}}(\chi_P(\bar{x})) \end{aligned}$$

- Аналогично выразим, через простейшие:

$$\begin{aligned} \chi_{=}(x, y) &= \overline{\text{sg}}(|x - y|) = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases} \\ \chi_{<}(x, y) &= \overline{\text{sg}}(y \div x) \end{aligned}$$

Остальные можем выразить также или через уже проверенные $<$ и \neg .

■

Лемма 9. Следующие функции являются примитивно рекурсивными:

1. $\left\lfloor \frac{x}{y} \right\rfloor$, считаем, что $\left\lfloor \frac{x}{0} \right\rfloor = x$
2. $\text{Div}(x, y) = \begin{cases} 1, & y \mid x \\ 0, & \text{else} \end{cases}$
3. $\text{Prime}(x) = \begin{cases} 1, & x \in \mathbb{P} \\ 0, & \text{else} \end{cases}$
4. $f(x) = p_x$, где p_x — x -тое простое число, $p_0 := 2$
5. $\text{ex}(i, x)$ — степень простого числа p_i разложении x , $\text{ex}(i, 0) := 0$



1. $f(x, y) = \left\lfloor \frac{x}{y} \right\rfloor$. Найдем минимальное k , что $f'(x, y, k) = yk > x$. Чтобы получить

$$f(x, y) = \min(k \mid f'(x, y, k)) - 1,$$

используем оператор ограниченной минимизации, где $h(x) = x$:

$$f(x, y) = (\mu k \leq h(x))[\neg f'(x, y, k) = 0] - 1.$$

2. $\text{Div}(x, y) = \left(\left\lfloor \frac{x}{y} \right\rfloor \cdot y == x \right)$

3. Определим $\text{Div}'(x, y) = (y \leq 1) \vee (\neg \text{Div}(x, y))$, эта функция проверяет, что число y не является нетривиальным делителем x .

Теперь, используя ограниченную минимизацию, выразим $\text{Prime}(x)$:

$$\text{Prime}(x) = (\mu y \leq h(x))[\text{Div}'(x, y) = 0] = x, \text{ где } h(x) = x - 1.$$

То есть мы посмотрели на все меньшие числа, если среди них найдется нетривиальный делитель, то число не простое.

4. Пусть $f'(y) =$ количество простых $\leq y$.

$$\begin{cases} f'(0) &= 0 \\ f'(y+1) &= \text{Prime}(y+1) + f'(y) \end{cases}$$

Теперь можно вычислить $f(x)$: для этого определим функцию $g(x, y) = (f'(y) = x)$,

$$f(x) = (\mu y \leq h(x))[\neg g(x, y) = 0].$$

Можно ограничить какой-нибудь функцией h .

5. Чтобы найти степень вхождения простого числа p_i в x , сначала находим это простое число по номеру, затем находим минимальное k , что x не делится на p_i^k и вычитаем единицу.



1.1.6 Канторовская нумерация

Теорема 1.1.1 (Канторовская нумерация). Пусть $\pi: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$:

$$\pi(x, y) = \frac{1}{2}(x+y)(x+y+1) + y.$$

- Тогда для любого z существует единственное представление $z = \pi(x, y)$.
- Причем функции $x(z), y(z)$ примитивно рекурсивные.



Можно по-честному все посчитать и выразить $x(z), y(z)$. Пусть

$$w = x + y$$

$$t = \frac{1}{2}w(w+1) = \frac{w^2 + w}{2}$$

$$z = t + y$$

Решим квадратное уравнение, чтобы выразить w через t ¹:

$$w = \frac{-1 + \sqrt{8t + 1}}{2}.$$

Запишем неравенство:


$$t \leq z = t + y < t + (w + 1) = \frac{(w + 1)^2 + (w + 1)}{2}.$$

Аналогично выразим $w + 1$ через z : имеем $z < \frac{(w+1)^2 + (w+1)}{2}$, решаем неравенство, а далее вспоминаем, что все числа положительные и можно забыть про отрицательные корни. Отсюда

$$w = \frac{-1 + \sqrt{8t + 1}}{2} \leq \frac{-1 + \sqrt{8z + 1}}{2} < w + 1.$$

Тогда

$$\begin{aligned} w &= \left\lfloor \frac{-1 + \sqrt{8z + 1}}{2} \right\rfloor \\ t &= \frac{w^2 + w}{2} \\ y &= z - t \\ x &= w - y \end{aligned}$$

Таким образом, мы выразили через z обе координаты. Единственный момент — нужно извлекать корень, в натуральную степень возводить мы умеем, поэтому можем с помощью ограниченной минимизации перебрать все меньшие числа, возвести их в квадрат и сравнить с нашим числом. 

Альтернативное доказательство.

□ Понятно, что для любого z существуют x, y : $\pi(x, y) = z$, докажем единственность представления $z = \pi(x, y)$.

Сначала покажем, что если $z = \pi(x_1, y_1)$ и $z = \pi(x_2, y_2)$, где $(x_1, y_1) \neq (x_2, y_2)$, то $x_1 + y_1 = x_2 + y_2$.

Действительно, если $x_1 + y_1 = x_2 + y_2 + c$, где $c > 0$, то


$$z = \frac{1}{2}(x_2 + y_2)(x_2 + y_2 + 1) + y_2 = \frac{1}{2}(x_1 + y_1)(x_1 + y_1 + 1) + y_1 = \frac{1}{2}(x_2 + y_2 + c)(x_2 + y_2 + c + 1) + y_1 =$$

$$\frac{1}{2}((x_2 + y_2)(x_2 + y_2 + 1) + 2cx_2 + 2cy_2 + c + c^2) + y_1 > \frac{1}{2}(x_2 + y_2)(x_2 + y_2 + 1) + x_2 + y_2 \geq$$

$$\frac{1}{2}(x_2 + y_2)(x_2 + y_2 + 1) + y_2 = z,$$

что есть противоречие.

Пусть $z = \pi(x, y)$ и знаем корректное значение $x + y$ (которое как мы поняли единственное). Тогда $y = z \div \frac{1}{2}(x + y)(x + y + 1)$, затем $x = (x + y) \div y$. Значит представление $z = \pi(x, y)$ единственно.

Теперь посчитаем $x(z), y(z)$. Перебираем $x + y$ (ограниченная минимизация), по $x + y$ однозначно восстанавливаем x, y и проверяем, что $\pi(x, y) = z$. Обе функции очевидно примитивно рекурсивны. 

¹отрицательный корень можем сразу отбросить

1.1.7 Теоремы про рекурсии

Теорема 1.1.2 (Возвратная рекурсия). Зафиксируем s . Пусть

$$\begin{cases} f(\bar{x}, 0) &= g(\bar{x}) \\ f(\bar{x}, y + 1) &= h(\bar{x}, y, f(\bar{x}, t_1(y)), \dots, f(\bar{x}, t_s(y))) \end{cases}$$

где $\forall 1 \leq i \leq s \ t_i(y) \leq y, g^{(n)}, h^{(n+1+s)}, t_i^{(1)}$.

Тогда, если g, h, t_i — примитивно / общерекурсивные, то и f тоже.

Основная идея этой теоремы — можем использовать все ранее вычисленные значения функции, а не только предыдущее.

□ Построим с помощью примитивной рекурсии функцию $m(\bar{x}, y)$, которая возвращает закодированную последовательность $f(\bar{x}, i)$, $0 \leq i \leq y$.

Кодировать будем так: каждому $f(\bar{x}, i)$ будет соответствовать p_i (i -ое простое число) в степени $1 + f(\bar{x}, i)$.

Если мы построим эту функцию, то $f(\bar{x}, y)$ — уменьшенная на 1 степень y -ого простого, обозначим функцию, которая это делает:

$$f(\bar{x}, y) = \text{ith}(y, m(\bar{x}, y)).$$

Вернемся к построению m :

$$\begin{cases} m(\bar{x}, 0) &= 2^{1+g(\bar{x})} \\ m(\bar{x}, y + 1) &= m(\bar{x}, y) \cdot p_{y+1}^{1+h(\bar{x}, y, f(\bar{x}, t_1(y)), \dots, f(\bar{x}, t_s(y)))} \\ &= m(\bar{x}, y) \cdot p_{y+1}^{1+h(\bar{x}, y, \text{ith}(t_1(y), m(\bar{x}, y)), \dots, \text{ith}(t_s(y), m(\bar{x}, y)))} \end{cases}$$

Теорема 1.1.3 (Совместная рекурсия). Пусть $f_i^{(n+1)}, 1 \leq i \leq k$,

$$\begin{cases} f_i(\bar{x}, 0) &= g_i(\bar{x}) \\ f_i(\bar{x}, y + 1) &= h_i(\bar{x}, y, f_1(\bar{x}, y), \dots, f_k(\bar{x}, y)) \end{cases}$$

Если $g_i^{(n)}, h_i^{(k+2)}, 1 \leq i \leq k$ — примитивно / общерекурсивные, то f_i тоже.

Основная идея этой теоремы — можем использовать y -е значение каждой из k функций.

□ Заметим, что канторовскую функцию можно, последовательно применив несколько раз, расширить до k -местной. Обозначим полученную функцию за c , а обратные за c_1, \dots, c_k .

Давайте просто объединим все f_i в одну функцию

$$m(\bar{x}, y) = c(f_1(\bar{x}, y), \dots, f_k(\bar{x}, y)).$$

Теперь каждую f_i можно вычислить

$$f_i(\bar{x}, y) = c_i(m(\bar{x}, y)).$$

Чтобы получить m достаточно использовать примитивную рекурсию:

$$\begin{cases} m(\bar{x}, 0) &= c(g_1(\bar{x}), \dots, g_k(\bar{x})) \\ m(\bar{x}, y + 1) &= c(\\ &\quad h_1(\bar{x}, y, c_1(m(\bar{x}, y)), \dots, c_k(m(\bar{x}, y))), \\ &\quad \vdots \\ &\quad h_k(\bar{x}, y, c_1(m(\bar{x}, y)), \dots, c_k(m(\bar{x}, y))) \\ & \end{cases}$$

Теорема 1.1.4 (Кусочное задание функции). Пусть R_0, \dots, R_k — отношения^a, такие что $\bigcup_{i=0}^k R_i = \mathbb{N}^n$ ^b. Для $|\bar{x}| = n$ кусочно зададим функцию $f^{(n)}$:

$$f(\bar{x}) = \begin{cases} f_0(\bar{x}), & \text{если } R_0(\bar{x}) \\ f_1(\bar{x}), & \text{если } R_1(\bar{x}) \\ \vdots & \vdots \\ f_k(\bar{x}), & \text{если } R_k(\bar{x}) \end{cases}$$

Если $f_i^{(n)}, R_i$ — примитивно / общерекурсивны, то и f тоже.

^aНабор предикатов

^bТо есть для $i \neq j$ верно $R_i \cap R_j = \emptyset$.

□ Рассмотрим характеристические функции χ_{R_i} для R_i . Тогда

$$f(\bar{x}) = \sum_{i=0}^k f_i(\bar{x}) \cdot \chi_{R_i}(\bar{x}).$$

А это просто сумма произведений, которые мы можем вычислять. ■

1.2 Равносильность МТ и ЧРФ

Теорема 1.2.1. Функция вычисляется машиной Тьюринга тогда и только тогда, когда она частично рекурсивная (то есть вычислима по Клини).

□

2 \Rightarrow 1 Если $f(x_1, \dots, x_n) = y$, то считаем, что МТ получает $1^{x_1}01^{x_2}0 \dots 01^{x_n}$ и должна выдать 1^y ; если f не определена, МТ должна заикливиться и наоборот.

- Для простых функций можем построить МТ напрямую:
 - Если мы хотим выдавать нуль, просто стираем вход.
 - Если нужно увеличить число на один, приписываем 1 в конец справа.
 - Если нужно вернуть k -ую проекцию, стираем все до начала k -ого числа (то есть нужно отсчитать $k - 1$ нуль на входе), далее стереть все после.
- Для операторов **S, R, M**:

S: Пусть есть набор функций $h^{(n)}, g_1^{(m)}, \dots, g_n^{(m)} \rightarrow f^{(m)}$, для каждой из которых есть машина Тьюринга M_h и M_{g_i} .

Хотим построить МТ M_f для вычисления f .

Сделаем это так:

- Копируем весь вход n раз:

$$(1^{x_1}01^{x_2} \dots 01^{x_n} *)^n.$$

- Запускаем M_{g_i} на соответствующей части полученного входа.

Если нужно что-то записать, то будем сдвигать всю правую часть на нужное число клеток, чтобы освободить место.

МТ запускаем псевдопараллельно (по очереди даем поработать).

В каждой части после окончания работы оставляем только ответ:

$$1^{y_1} * 1^{y_2} \dots * 1^{y_n},$$

где $y_i = g_i(x_1, \dots, x_m)$.

- Запускаем на этом результате M_h , предварительно, конечно, поменяв $*$ на 0.

R: Пусть рекурсия задает $f^{(m+1)}(x_1, \dots, x_m, y)$ из $g^{(m)}$ и $h^{(m+2)}$.

Лекция 2
18 feb

$$\begin{cases} f(\bar{x}, 0) &= g(\bar{x}) \\ f(\bar{x}, y + 1) &= h(\bar{x}, y, f(\bar{x}, y)) \end{cases}$$

Считаем, что для g, h уже есть МТ (M_g и M_h), и мы хотим построить M_f , которая будет вычислять f .

Построим вспомогательные МТ:

- M_1 : для входа $1^{x_1}0 \dots 01^{x_m}01^y$ построим $1^y01^{x_1}0 \dots 01^{x_m}0(1^0)01^{g(x_1, \dots, x_m)}$. Для этого просто запустим M_g на входе, но не будем стирать его, а результат просто припишем после двух нулей справа. Далее мы будем накапливать значение u между этими нулями, а сейчас там ничего нет, то есть $u = 0$.
- M_2 : для входа $1^y01^{x_1}0 \dots 01^{x_m}01^u01^z$ построим $1^y01^{x_1}0 \dots 01^{x_m}01^{u+1}01^{h(x_1, \dots, x_m, u, z)}$. Для этого, используя M_h , допишем в конец вместо z результат h и допишем единицу к 1^u . Здесь $u + 1$ обозначает текущее значение y' , а значение h — значение $f(y')$.
- M_3 : для входа $1^y01^{x_1}0 \dots 01^{x_m}01^u01^z$ оставим только 1^z .
- Φ : для входа $1^y01^{x_1}0 \dots 01^{x_m}01^u01^z$ проверим, что $u \neq y$.

Теперь соберем все вместе: сначала запустим M_1 , далее пока Φ возвращает неравенство, запускаем M_2 (увеличиваем u на один, вычисляем следующее значение функции), и в конце стираем лишнее, запустив M_3 .

M: Хотим по МТ M_g построить M_f , вычисляющую

$$f(\bar{x}) = \begin{cases} y & g(\bar{x}, y) = 0 \wedge g(\bar{x}, z) \neq 0 \quad \forall z < y \\ \uparrow & \text{else} \end{cases}$$

Аналогично построим несколько вспомогательных МТ:

- N_1 : приписывает $0(1^0)$ ко входу, это инициализация $y = 0$:

$$1^{x_1}0 \dots 01^{x_m} \longrightarrow 1^{x_1}0 \dots 01^{x_m}0(1^0).$$

- N_2 : дублирует вход, разделяя решеткой: $w \longrightarrow w\#w$
- N_3 : в продублированном входе меняет вторую половину на результат M_g

$$1^{x_1}0 \dots 01^{x_m}01^y\#1^{x_1}0 \dots 01^{x_m}01^y \xrightarrow{M_g} 1^{x_1}0 \dots 01^{x_m}01^y\#1^{g(x_1, \dots, x_m, y)}.$$

- N_4 : очищает все после решетки и дописывает единицу в конец

$$1^{x_1}0 \dots 01^{x_m}01^y\#w \longrightarrow 1^{x_1}0 \dots 01^{x_m}01^{y+1}.$$

- N_5 : стирает все, кроме ответа

$$1^{x_1}0 \dots 01^{x_m}01^y\#w \longrightarrow 1^y.$$

- Φ : проверяет, что после решетки что-то еще есть $w\#v \longrightarrow v \neq \varepsilon$.

Теперь можем построить M_f так:

$$N_1; N_2; N_3; \text{while } \Phi \text{ do } N_4, N_2, N_3; N_5.$$

1 \Rightarrow 2 Теперь мы хотим промоделировать работу МТ с помощью частично рекурсивной функции. На вход должны либо выдать результат, либо заиклиться. Так как машины Тьюринга работают со строками, а функции с натуральными числами, нужно придумать правила кодирования.

Пусть есть конфигурация МТ

$$\alpha q_i a_j \beta,$$

где α — строка слева от головки, q_i — состояние, a_j — текущий символ, β — строка справа от головки.

Пронумеруем рабочий алфавит $\Gamma = \{a_0, \dots, a_{m-1}\}$, где a_0 — пустой символ ($_$).

Кодирование конфигураций Теперь можем конфигурацию записать как

$$\tilde{\alpha}, \tilde{q}_i, \tilde{a}_j, \tilde{\beta},$$

где $\tilde{\alpha}$ — число, соответствующее α в m -ичной записи, \tilde{q}_i — просто номер состояния, \tilde{a}_j — номер в алфавите (j), $\tilde{\beta}$ — число, соответствующее β в m -ичной записи, записанное справа налево.

Сдвиги обозначать будем d : вправо $d = 1$, влево $d = 2$.

Терминальное состояние — z . Множество состояний тоже пронумеруем и получим множество состояний $\tilde{Q} = \{0, 1, \dots, |Q| - 1\}$.

Пример 1.2.1

Рассмотрим небольшой пример. Пусть $\Gamma = \{a_0, a_1\}$, тогда следующее состояние будет записано как (22, 3, 1, 13):

$$\underbrace{a_1 a_0 a_1 a_1 a_0}_{\alpha} q_3 a_1 \underbrace{a_1 a_0 a_1 a_1}_{\beta}$$

Кодирование команд Пусть есть переход $(q, a) \rightarrow (p, b, d) = (p - \text{новое состояние}, b - \text{новый символ}, d - \text{направление движения головки})$. Сопоставим p, b, d тройку функций $\varphi_q, \varphi_a, \varphi_d$:

$$\begin{aligned}\varphi_q: \tilde{Q} \times \tilde{\Gamma} &\rightarrow \tilde{Q} \\ \varphi_a: \tilde{Q} \times \tilde{\Gamma} &\rightarrow \tilde{\Gamma} \\ \varphi_d: \tilde{Q} \times \tilde{\Gamma} &\rightarrow \{1, 2\}\end{aligned}$$

Эти функции будут примитивно рекурсивными, так как заданы на конечном множестве, на остальных можем доопределить нулем.

Преобразование конфигураций Пусть у нас есть переход между двумя конфигурациями:

$$K = \alpha q_i a_j \beta \rightarrow \alpha' q'_i a'_j \beta' = K'.$$

Зададим функцию на числах, которая проделает этот переход $\Phi: K \rightarrow K'$. На самом деле эта функция состоит из четырех, которые мы сейчас и определим.

Пусть

$$\begin{aligned}\tilde{q}'_i(\tilde{\alpha}, \tilde{q}_i, \tilde{a}_j, \tilde{\beta}) &= \varphi_q(\tilde{q}_i, \tilde{a}_j) \\ \tilde{\alpha}'(\tilde{\alpha}, \tilde{q}_i, \tilde{a}_j, \tilde{\beta}) &= \begin{cases} \tilde{\alpha} \cdot m + \varphi_a(\tilde{q}_i, \tilde{a}_j), & \varphi_d(\tilde{q}_i, \tilde{a}_j) = 1 \\ \left\lfloor \frac{\tilde{\alpha}}{m} \right\rfloor, & \varphi_d(\tilde{q}_i, \tilde{a}_j) = 2 \end{cases} \\ \tilde{\beta}'(\tilde{\alpha}, \tilde{q}_i, \tilde{a}_j, \tilde{\beta}) &= \begin{cases} \tilde{\beta} \cdot m + \varphi_a(\tilde{q}_i, \tilde{a}_j), & \varphi_d(\tilde{q}_i, \tilde{a}_j) = 2 \\ \left\lfloor \frac{\tilde{\beta}}{m} \right\rfloor, & \varphi_d(\tilde{q}_i, \tilde{a}_j) = 1 \end{cases} \\ \tilde{a}'_j &= \begin{cases} \tilde{\beta} \bmod m, & \varphi_d(\tilde{q}_i, \tilde{a}_j) = 1 \\ \tilde{\alpha} \bmod m, & \varphi_d(\tilde{q}_i, \tilde{a}_j) = 2 \end{cases}\end{aligned}$$

Заметим, что все эти формулы примитивно рекурсивны².

Общая работа МТ Пусть $K(0) = (\tilde{\alpha}_0, \tilde{q}_0, \tilde{a}_0, \tilde{\beta}_0)$ — начальная конфигурация. Чтобы получить новую конфигурацию для шага t , посчитаем все четыре параметра:

$$\begin{aligned}K(t) = (& \\ & K_\alpha(\tilde{\alpha}_0, \tilde{q}_0, \tilde{a}_0, \tilde{\beta}_0, t) \\ & K_q(\tilde{\alpha}_0, \tilde{q}_0, \tilde{a}_0, \tilde{\beta}_0, t) \\ & K_a(\tilde{\alpha}_0, \tilde{q}_0, \tilde{a}_0, \tilde{\beta}_0, t) \\ & K_\beta(\tilde{\alpha}_0, \tilde{q}_0, \tilde{a}_0, \tilde{\beta}_0, t) \\ &)\end{aligned}$$

²Единственное, чего нет явно в лемме 8 выше, это остаток по модулю, но его легко получить из деления нацело.

Теперь запишем совместную рекурсию для $K_\alpha, K_q, K_a, K_\beta$:

$$\begin{cases} K_\alpha(\tilde{\alpha}_0, \tilde{q}_0, \tilde{a}_0, \tilde{\beta}_0, 0) &= \tilde{\alpha}_0 \\ K_\alpha(\tilde{\alpha}_0, \tilde{q}_0, \tilde{a}_0, \tilde{\beta}_0, t+1) &= \tilde{\alpha}'(K_\alpha(\dots, t), K_q(\dots, t), K_a(\dots, t), K_\beta(\dots, t)) \end{cases}$$

Для остальных точно также.

Результат Пусть начальное состояние $q_0 a_0 \beta_0$ (стоим на самом левом символе), конечное — $q_z a_z \beta_z$, причем его встречаем впервые. То есть нам нужно вычислить функцию, которая переводит

$$x = \tilde{a}_0 + \tilde{\beta}_0 t \longrightarrow \tilde{a}_z + \tilde{\beta}_z t,$$

если машина Тьюринга пришла сюда, и не определена, если МТ зацикливается:

$$t_z = \mu t[K_q(t) = z].$$

Тогда результатом работы МТ будет

$$\begin{aligned} \varphi(x) = m \cdot K_\beta \Big(0, 0, x \bmod m, \lfloor \frac{x}{m} \rfloor, \mu t[K_q(0, 0, x \bmod m, \lfloor \frac{x}{m} \rfloor, t) = z] \Big) \\ + K_a \Big(0, 0, x \bmod m, \lfloor \frac{x}{m} \rfloor, \mu t[K_q(0, 0, x \bmod m, \lfloor \frac{x}{m} \rfloor, t) = z] \Big) \end{aligned}$$

Следствие 10. Любую частично рекурсивную функцию можно представить так, чтобы минимизация использовалась только один раз.

□ Сначала запишем для нее МТ, а потом постоим обратно функцию. В итоге получим эквивалентную функцию, причем по построению оператор минимизации использовался лишь один раз.

Следствие 11. Функция, вычислимая за примитивно рекурсивное время ^a, тоже является примитивно рекурсивной.

^aвремя, ограниченное примитивно рекурсивной функцией

□ В построении функции использовали минимизацию по числу шагов МТ, поэтому, если работаем примитивно рекурсивное время, можем применить ограниченную минимизацию.

1.3 Функция Аккермана

Можно построить общерекурсивную функцию, которая растет быстрее любой примитивно рекурсивной. Из этого следует, что ПРФ не совпадает с ОРФ.

Определение 49: Функция Аккермана

Функция Аккермана — функция от двух аргументов $\alpha_n(x)$, определенная следующим образом:

$$\begin{cases} \alpha_0(x) &= x + 1 \\ \alpha_{n+1}(x) &= \alpha_n^{[x+2]}(x) = \underbrace{\alpha_n(\alpha_n(\dots(x)))}_{x+2 \text{ раза}} \end{cases}$$

Лемма 10. $\alpha_n(x) \geq x + n + 1$. В частности, $\alpha_n(x) > x$.

□ Докажем по индукции по n .

- База: $n = 0$. $\alpha_0(x) = x + 1 = x + 0 + 1$.
- Переход: $n - 1 \rightarrow n$.

$$\begin{aligned}\alpha_n(x) &= \alpha_{n-1}^{[x+2]}(x) \geq \alpha_{n-1}^{[x+1]}(x) + 1 > && \text{(т.к. } \alpha_{n-1}(t) > t \text{ по предположению индукции)} \\ &> \alpha_{n-1}^{[x]}(x) + 1 > \alpha_{n-1}^{[x-1]}(x) + 1 > \dots > && \text{(продолжаем до кратности 1)} \\ &> \alpha_{n-1}(x) + 1 \geq x + (n - 1) + 1 + 1 = x + n + 1\end{aligned}$$

Лемма 11. Если $x > y$, то $\alpha_n(x) > \alpha_n(y)$.

□ Индукция по n .

- База: $n = 0$. $\alpha_0(x) = x + 1 > y + 1 = \alpha_0(y)$.
- Переход: $n - 1 \rightarrow n$.

$$\begin{aligned}\alpha_n(x) &= \alpha_{n-1}^{[x+2]}(x) > \alpha_{n-1}^{[x+2]}(y) && \text{(по предположению для } n - 1) \\ &> \alpha_{n-1}^{[x+1]}(y) > \dots > \alpha_{n-1}^{[y+2]}(y) && \text{(по прошлой лемме 10)} \\ &= \alpha_n(y)\end{aligned}$$

Лемма 12. Если $n > m$, то $\alpha_n(x) > \alpha_m(x)$.

□ Индукция по m .

- База: $m = 0$. По лемме 10 $\alpha_n(x) \geq x + n + 1 \geq x + 1 = \alpha_0(x)$.
- Переход: $m - 1 \rightarrow m$. По определению $\alpha_n(x) = \alpha_{n-1}^{[x+2]}(x)$. Применим индукционное предположение ко всем α_{n-1} и заменим на α_{m-1} , после каждой замены значения будут уменьшаться:

$$\alpha_{n-1}^{[x+2]}(x) > \alpha_{n-1}^{[x+1]}(\alpha_{m-1}(x)) > \dots > \alpha_{n-1}(\alpha_{m-1}^{[x+1]}(x)) > \alpha_{m-1}^{[x+2]}(x).$$

Лемма 13. Если $n > 1$, то $\alpha_n(x) \geq \alpha_{n-1}^{[2]}(x)$.

□ Рассмотрим 2 случая:

- Если $x = 0$, то $\alpha_n(x) = \alpha_{n-1}^{[x+2]}(x) = \alpha_{n-1}^{[2]}(x)$.
- Иначе $\alpha_n(x) = \alpha_{n-1}^{[x+2]}(x) > \alpha_{n-1}^{[x+1]}(x) \geq \alpha_{n-1}^{[2]}(x)$

Лемма 14. Для любой ПРФ $f(x_1, \dots, x_n)$ существует константа k , что $f(x_1, \dots, x_n) \leq \alpha_k(\max\{x_1, \dots, x_n\})$. Если f имеет 0 аргументов, подставим 0.

□

- Для простейших функций подойдет $k = 0$: для нульместного и одноместного нулей, функцию $s(x)$ и проекции $I_a^b(x_1, \dots, x_a)$ неравенство верно, так как $\alpha_0(\max \bar{x}) = \max \bar{x} + 1$.
- Если применяется оператор суперпозиции для других функций с найденными k_i , можно взять наибольшее из них (пусть k), т.е. $h(\bar{x}), g_i(\bar{x}) \leq \alpha_k(\max \bar{x}) \implies \max g_i(\bar{x}) \leq \alpha_k(\max \bar{x})$. Докажем, что подойдет $\alpha_{k+1}(x)$.

Пусть суперпозиция применяется к $h(x_1, \dots, x_m)$ и $g_i(x_1, \dots, x_n)$.

$$\begin{aligned}
 h(g_1(\bar{x}), \dots, g_m(\bar{x})) &\leq \alpha_k(\max_{i \in [1, m]} g_i(\bar{x})) && \text{(используем } \alpha_k \text{ для } h) \\
 &\leq \alpha_k(\alpha_k(\max \bar{x})) && \text{(используем } \alpha_k \text{ для } g_i) \\
 &= \alpha_k^{[2]}(\max \bar{x}) \\
 &\leq \alpha_{k+1}(\max \bar{x}) && \text{(лемма 13)}
 \end{aligned}$$

- Если функция $f(\bar{x}, n)$ получена из $g(\bar{x})$ и $h(\bar{x}, n, t)$ с помощью примитивной рекурсии, сначала найдем k , чтобы $g(\bar{x}) \leq \alpha_k(\max \bar{x})$ и $h(\bar{x}, n, t) \leq \alpha_k(\max(\bar{x}, n, t))$.

Оценим функцию f .

$$\begin{cases} f(\bar{x}, 0) = g(\bar{x}) \\ f(\bar{x}, n + 1) = h(\bar{x}, n, f(\bar{x}, n)) \end{cases}$$

Докажем, что $f(\bar{x}, n) \leq \alpha_k^{[n+1]}(\max(\bar{x}, n))$.³

- База: $n = 0$. Верно, по определению f .
- Переход: $n \rightarrow n + 1$.

$$\begin{aligned}
 f(\bar{x}, n + 1) &\leq \alpha_k(\max(\bar{x}, n, f(\bar{x}, n))) \leq \alpha_k(\max(\bar{x}, n, \alpha_k^{[n+1]}(\max(\bar{x}, n)))) \\
 &\quad \text{(индукционное предположение)} \\
 &= \alpha_k(\alpha_k^{[n+1]}(\max(\bar{x}, n))) = \alpha_k^{[n+2]}(\max(\bar{x}, n))
 \end{aligned}$$

То есть

$$f(\bar{x}, n) \leq \alpha_k^{[n+1]}(\max(\bar{x}, n)) < \alpha_k^{[\max(\bar{x}, n)+2]}(\max(\bar{x}, n)) = \alpha_{k+1}(\max(\bar{x}, n))$$

Лемма 15. Для любой ПРФ $f(n)$ найдется такое $N \in \mathbb{N}$, что при $n > N$ выполнено $\alpha_n(n) > f(n)$.

- По лемме 14 для $f(n) + 1$ найдется такое N , что $\alpha_N(n) \geq f(n) + 1 > f(n)$. А тогда и для всех больших $m \geq N$ верно неравенство $\alpha_m(n) > f(n)$, в том числе и $\alpha_m(m) > f(m)$.

Теорема 1.3.1. $\alpha_n(n): \mathbb{N} \rightarrow \mathbb{N}$ растет быстрее любой примитивно рекурсивной.

- По лемме 15 для любой примитивно рекурсивной функции есть константа N , начиная с которой $\alpha_n(n) > f(n)$, то есть она будет расти быстрее.

Из этого также следует, что $\alpha_n(n)$ не ПРФ.

Лемма 16. $\alpha_n(x)$ является общерекурсивной функцией двух аргументов. В частности, $\alpha_n(n)$ — одного аргумента.

- Построим машину Тьюринга A , вычисляющую $\alpha_n(x)$ по n и x : $x + 2$ раза повторяем рекурсивно вызывать себя для $n - 1$ и результата рекурсивных вызовов. Когда доходим до $n = 0$, возвращаем число, увеличенное на один.

В итоге мы построили МТ строго по определению.

³Здесь под $\max(\bar{x}, \text{что-то еще})$ подразумевается максимум по всем координатам и n : $\max(x_1, \dots, x_m, \text{что-то еще})$.

Chapter 2

Разрешимые и перечислимые множества

2.1 Определения

Определение 50: Разрешимое множество

Множество $X \subseteq \mathbb{N}^k$ называется **разрешимым**, если его характеристическая функция вычислима^a.

^aЭто может быть частично рекурсивная функция, машина Тьюринга, λ -функция...

Замечание. Любое конечное множество разрешимо. Пересечение, объединение, разность разрешимых тоже разрешимо.

Теорема 2.1.1. Множество $X \subseteq \mathbb{N}$ разрешимо тогда и только тогда, когда X — множество значений всюду определенной вычислимой неубывающей функции (или пустое множество).

□

1 \implies 2 Можем в характеристической функции $\chi_X(n)$ возвращать n вместо 1, а в остальных значениях прошлое выданное. Эта функция подходит под описание.

2 \implies 1 Если множество конечно, то оно разрешимо, так как можем задать функцию χ_X на конечном числе точек. Если X бесконечно будем действовать, как описано далее.

Пусть есть функция f . Из нее хотим построить χ_X . Посчитаем $\chi_X(n)$ так: начнем с $i = 0$

- вычислим $f(i)$;
- если значение больше n , то в следующих входах, значения будут еще больше, поэтому можем сразу вернуть 0;
- если меньше, то посчитаем $f(i + 1)$ и вернемся к предыдущему пункту;
- так как функция неубывающая и достигает всех значений из X (причем их бесконечно много, поэтому есть элемент больше n), мы либо найдем значение больше n (тогда вернем 0), либо равное (тогда вернем единицу).



2.2 Перечислимые множества

Определение 51: Перечислимое множество

Множество $X \subseteq \mathbb{N}^k$ называется **перечислимым**, если

- его *полухарактеристическая* функция вычислима:

$$\chi_X(n) = \begin{cases} 1, & n \in X \\ \uparrow, & n \notin X \end{cases}$$

- или, если существует алгоритм, который выводит все его элементы в некотором порядке.

Теорема 2.2.1 (Об эквивалентных определениях). Следующие утверждения эквивалентны:

0. **Перечислимость:** существует алгоритм, который выводит все элементы X в некотором порядке
1. X — область определения вычислимой функции
2. X — область значений вычислимой функции
3. полухарактеристическая функция X вычислима
4. X — область значений всюду определенной вычислимой функции.

□

$0 \Rightarrow 3$ Чтобы посчитать $\chi_X(n)$, запускаем алгоритм, перечисляющий элементы множества, ждем n .

Если вывелось n , то выводим $\chi_X(n) = 1$, а иначе мы зациклились, то есть получили расходимость.

$3 \Rightarrow 1$ Действительно, множество X будет областью определения χ_X , а она вычислима.

$1 \Rightarrow 0$ Пусть область определения вычисляется алгоритмом B , который просто считает функцию, если вход принадлежит области определения, алгоритм завершается, иначе зацикливается.

Построим алгоритм A следующим образом: будем запускать B по шагам и выводить элементы множества

- 1 шаг на входе 0
- 2 шага на 0, 2 шага на 1
- 3 шага на 0, 1, 2
- и так далее
- как только B закончил работу на некотором элементе, выводим его.

Этот алгоритм A перечисляет наше множество, так как для алгоритма B требуется конечное время работы на элементах области определения.

$2 \Rightarrow 0$ Аналогично, но выводим значение функции на элементе, на котором мы останавливаемся.

$1 \Rightarrow 2$ Пусть X — область определения функции, которая вычисляется алгоритмом A . Рассмотрим следующую функцию:

$$b(n) = \begin{cases} n, & \text{если } A \text{ заканчивает работать на } n \\ \uparrow, & \text{если } A \text{ зацикливается на } n \end{cases}$$

Теперь X — область значений $b(n)$, а она вычислима.

0 \Rightarrow 4 Пусть A — алгоритм, перечисляющий X . Рассмотрим любой $n_0 \in X$.

Построим функцию f , которая всюду определена и X — ее область значений.

$$f(n) = \begin{cases} t, & \text{если на } n\text{-ом шаге работы } A \text{ появляется } t \\ n_0, & \text{если ничего не появляется} \end{cases}$$

4 \Rightarrow 2 Очевидно



Замечание. Все области значений и определений не применимы к пустому множеству, которое тоже перечислимое.

Задача. В определении перечислимого множества можно выводить элементы с повторениями. Это будет эквивалентно определению без повторений.

Теорема 2.2.2. Если считать перечислимыми только те множества, для которых существует машина Тьюринга, выводющая каждый элемент множества *ровно по разу*, то их класс не поменяется.

□ Увеличиться класс точно не может, так как мы накладываем более строгое условие.

Проверим, что по обычной МТ M , перечисляющей множество A , можно построить МТ M' , которая будет выводить все элементы ровно один раз.

Пусть машина M' работает почти как M , но записывает на ленту все числа, которые она уже выводила.

Теперь, если M должна вывести число, M' проверяет, что еще не возвращала его ранее, записывает и выдает. Если число уже записано, возвращать не будем.



Теорема 2.2.3. Объединение и пересечение перечислимых множеств тоже перечислимое.

□ По определению перечислимости для первого множества есть алгоритм A , который завершается на всех элементах этого множества. Аналогично для второго — B .

- Хотим проверить, что n принадлежит объединению. Будем давать алгоритмам A и B поработать по шагу. Ждем шага, на котором завершает работу хотя бы один алгоритм. Значит, n лежит в одном из множеств.
- Чтобы проверить принадлежность пересечению запустим сначала A , если он завершит работу, то запустим B . Если и B остановится, n лежит в обоих множествах.

Если элемент не принадлежит объединению или пересечению, получим расходимость.



Теорема 2.2.4 (критерий Поста). A разрешимо тогда и только тогда, когда A и \bar{A} перечислимые.

□

1 \Rightarrow 2 Так как A разрешимо, можем рассмотреть вычислимую характеристическую функцию $\chi_A(n)$.

Построим полухарактеристические для A и \bar{A} :

- для A : если $n \in A$, то $\chi'_A = 1$, иначе $\chi'_A(n)$ расходится
- для \bar{A} аналогично, только результаты инвертированы.

$2 \Rightarrow 1$ Пусть мы хотим проверить $n \stackrel{?}{\in} A$.

Запускаем одновременно по шагам алгоритмы, перечисляющие A и \bar{A} , ждем появления n . Рано или поздно должно появиться, так как в объединении A и \bar{A} дают все множество.

Если его выдал алгоритм для A , то $\chi_A(n) = 1$, а если для \bar{A} , то $\chi_A(n) = 0$.



Определение 52: Проекция

Подмножество $P \subseteq \mathbb{N}$ называется **проекцией** $Q \subseteq \mathbb{N}^2$, если

$$\forall x: x \in P \iff \exists y: (x, y) \in Q.$$

Теорема 2.2.5 (О проекции). Множество P перечислимо тогда и только тогда, когда P — проекция некоторого разрешимого множества Q .

□

$1 \Rightarrow 2$ Пусть A — алгоритм, перечисляющий P . Тогда подойдет

$$Q := \{(n, t) \mid n \text{ появляется в течение } t \text{ шагов работы } A\}.$$

$2 \Rightarrow 1$ Проекция перечислимого перечислима: берем алгоритм, которые перечисляет Q , но оставляем только первую координату.



Теорема 2.2.6 (О графике). Частичная функция $f: \mathbb{N} \rightarrow \mathbb{N}$ вычислима тогда и только тогда, когда перечислим ее график

$$F := \{(x, y) \mid f(x) \text{ определена и } f(x) = y\}.$$

□

$1 \Rightarrow 2$ Чтобы перечислить все все точки графика будем по очереди запускать алгоритм для f на $x \in \mathbb{N}$, причем будем давать ему поработать i шагов на шаге i .

Если $f(x)$ в некоторый момент вычислено, выводим $(x, f(x))$.

$2 \Rightarrow 1$ Построим алгоритм вычисляющий f .

Пусть нам на вход дан элемент x , запустим алгоритм, перечисляющий элементы F .

Если $(x, f(x)) \in F$, то есть f определена в точке x , и в какой-то момент мы выпишем значение.

Если же функция не определена в x , то пары $(x, f(x))$ в F нет и мы закикливаемся.



Замечание. Различные названия типов множеств в других источниках:

перечислимое	разрешимое
полурешимое	
вычислимо перечислимое	вычислимое
полурекурсивное	рекурсивное
semi-decidable	decidable
semi-recursive	recursive
enumerable	

2.3 Универсальные функции

Определение 53

Сечением функции $U^{(m+1)}(n, \bar{x})$ назовем функцию $U_n(\bar{x})$ от m аргументов, которая получается из U фиксацией первого аргумента.

Определение 54: Универсальная функция

$U(n, \bar{x})$ — **универсальная** для класса K функций от m аргументов, если

- $\forall n: U_n(\bar{x}) \in K$
- $\forall f \in K \exists n: f = U_n$

То есть множество ее сечений совпадает с K .

Замечание. Универсальная функция существует только для не более чем счетных K .

Замечание. Все рассматриваемые функции частичные.

Обозначение. \mathcal{F}^m — вычислимые функции от m аргументов.

\mathcal{F}_*^m — всюду определенные вычислимые функции от m аргументов.

Без верхнего индекса по умолчанию подразумевается единица.

Теорема 2.3.1. Существует вычислимая функция 2-х аргументов $U \in \mathcal{F}^2$, универсальная для класса вычислимых функций 1-ого аргумента \mathcal{F}^1 .

□ Запишем все коды МТ, вычисляющих функции из \mathcal{F}^1 , в порядке возрастания (сначала по длине, затем в алфавитном).

Пусть $U(i, x)$ — функция, которая находит запись i -ой МТ M_i , запускает ее на входе x и возвращает результат.

Во-первых, U вычислима, так как вычисляется описанным выше алгоритмом.

Во-вторых, сечение U_i соответствует МТ M_i , поэтому U универсальна для \mathcal{F}^1 . ■

Следствие 12. Существует $U' \in \mathcal{F}^{m+1}$, универсальная для \mathcal{F}^m .

Замечание. Здесь мы будем использовать m -местную канторовскую нумерацию $c(x_1, \dots, x_m)$, которую можно построить, например, последовательным сворачиванием пар. Обозначим обратные проекции на i координату $c_i(y) = x_i$.

□ Проверим, что универсальной функцией будет

$$U'(n, \bar{x}) := U(n, c(\bar{x})),$$

где U — универсальная для \mathcal{F}^1 .

Во-первых, заметим, что все сечения вычислимы.

Далее рассмотрим произвольную функцию $f(\bar{x}) \in \mathcal{F}^m$. Найдем для нее одно из сечений U' .

Определим

$$g(y) := f(c_1(y), \dots, c_m(y)).$$

g вычислима, U универсальна, поэтому

$$\exists n: U_n(y) = g(y).$$

$$\begin{aligned}
U'(n, \bar{x}) &= && \text{(по определению } U) \\
= U(n, c(\bar{x})) &= && (n - \text{номер } g) \\
= g(c(\bar{x})) &= && \text{(по определению } g) \\
= f(c_1(c(\bar{x})), \dots, c_m(c(\bar{x}))) &= f(\bar{x})
\end{aligned}$$

То есть U' действительно универсальная. ■

Теорема 2.3.2. Не существует $U \in \mathcal{F}_*^2$ универсальной для \mathcal{F}_* .

□ Предположим, что такая функция $U \in \mathcal{F}_*^2$ существует.

Рассмотрим диагональную функцию:

$$d'(n) = U(n, n) + 1 \in \mathcal{F}_*.$$

С одной стороны, $d'(n)$ — общерекурсивная функция, поэтому из универсальности U следует, что существует сечение $U_n = d'$.

С другой стороны, $d'(n)$ отличается от всех сечений U : если $\forall x: U(n, x) = d'(x)$, подставим $x = n$, получим $U(n, n) = U(n, n) + 1$. Противоречие. ■

Замечание. Для класса частичных функций такое рассуждение не проходит, так как они могут быть не определены и прибавление единицы ничего не меняет для неопределенности.

Теорема 2.3.3. Существует вычислимая частичная функция, которая не имеет всюду определенного вычислимого продолжения ^a.

^aто есть нельзя доопределить до всюду определенной вычислимой

□ Подходит функция $d'(n) = U(n, n) + 1$, где U — универсальная вычислимая функция ¹.

Пусть ее можно доопределить до вычислимой d'' :

$$d'' = \begin{cases} U(n, n) + 1, & \text{такие } n, \text{ где } U(n, n) \text{ определена} \\ \text{определена,} & \text{где } U(n, n) \text{ не определена} \end{cases}$$

Поэтому d'' отличается от всех сечений универсальной функции. Противоречие. ■

2.3.1 Перечислимое неразрешимое множество

Определение 55

Универсальное множество $U \subseteq \mathbb{N}^{n+1}$ для класса α — такое множество, что для любого $s \in \alpha$ существует такое i , что

$$s = \{\bar{x} \mid (i, \bar{x}) \in U\}.$$

Определение 56: Сечение множества

Сечением множества $W \subseteq \mathbb{N}^k$ назовем $W_n = \{\bar{x} \mid (n, \bar{x}) \in W\}$.

Докажем аналог прошлой теоремы для множеств.

Теорема 2.3.4. Существует перечислимое множество $W^{(m+1)} \subset \mathbb{N}^{m+1}$, являющееся универсальным для

¹далее это обозначение по умолчанию определяет универсальную вычислимую частичную функцию $U^{(m+1)}$ для вычислимых частичных функций m аргументов

всех перечислимых подмножеств \mathbb{N}^m .

□ Рассмотрим универсальную функцию $U^{(m+1)}$. Пусть множество $W^{(m+1)}$ — ее область определения, оно будет перечислимым.

Пусть у нас есть перечислимое множество $X \in \mathbb{N}^m$.

Найдем такую функцию $f \in \mathcal{F}^m$, для которой X — область определения, и такое n , что $U_n = f$.

Тогда $W_n = X$. ■

Теорема 2.3.5. Существует перечислимое неразрешимое множество.

□ Рассмотрим вычислимую $f(x)$, не имеющую всюду определенного вычислимого продолжения.

Пусть F — ее область определения, она перечислима и неразрешима:

- F перечислимо, потому что оно является областью определения вычислимой функции;
- F неразрешимо, так как в противном случае можно рассмотреть общерекурсивное доопределение f :

$$g(x) = \begin{cases} f(x), & x \in F \\ 0, & x \notin F \end{cases}$$

Эта функция всюду определена, вычислима, является продолжением f . Противоречие.

Замечание. $F = \{n \mid U(n, n) \text{ определено}\}$ — переформулировка класса L_1 (останавливающиеся на своем входе МТ).

Замечание. \bar{F} — пример непечислимого множества, потому что выше мы доказали неразрешимость F , а если бы дополнение было перечислимым, то из критерия Поста бы следовала разрешимость.

Замечание. Область определения универсальной функции перечислимое, но не разрешимое множество, так как область определения $U(n, n)$ — частично определенная неразрешимая, потому что, если допустить, что область определения разрешимо, то F разрешимо, противоречие.

Замечание. «Проблема остановки»² — переформулировка принадлежности данной функции к области определения универсальной функции. ■

Теорема 2.3.6. Существует частичная вычислимая функция $f: \mathbb{N} \rightarrow \{0, 1\}$, которая не имеет всюду определенного вычислимого продолжения.

□ Определим

$$d'''(n) = \begin{cases} 1, & U(n, n) = 0 \\ 0, & U(n, n) > 0 \\ \uparrow, & U(n, n) \uparrow \end{cases}$$

Любое доопределение будет отличаться от $U(n, n)$, так как для всех n значения $d'''(x)$ и $U_n(x)$ различны: здесь либо обе функции расходятся, либо первое не равно второму. ■

Определение 57

Непересекающиеся множества называются **отделимыми разрешимым**, если существует разрешимое множество, содержащее одно из них и непересекающееся с другим.

Следствие 13. Существуют перечислимые непересекающиеся неотделимые никаким разрешимым множеством множества.

²останавливается ли МТ M на входе x

□ Подойдут следующие множества: $X = \{n \mid d'''(n) = 0\}$ и $Y = \{n \mid d'''(n) = 1\}$. Пусть существует разделяющее их разрешимое C , содержащее Y и непересекающееся с X .

Тогда χ_C — общерекурсивное дополнение d''' . Противоречие.

Лекция 4
4 march

2.3.2 Главные универсальные функции

Определение 58

Пусть $U^{(n+1)} \in \mathcal{F}^{n+1}$ универсальная нумерация для функций \mathcal{F}^n .

U называется **главной нумерацией** или **главной универсальной функцией**, если для любой вычислимой функции $V \in \mathcal{F}^{n+1}$ существует **транслятор** $s \in \mathcal{F}_*^a$, такой что

$$\forall m \in \mathbb{N}, \bar{x} \in \mathbb{N}^n: V(m, \bar{x}) = U(s(m), \bar{x}).$$

^aпо номеру сечения V находит какой-то номер такого же сечения U

Теорема 2.3.7. Существует главная универсальная функция $U^{(n+1)} \in \mathcal{F}^{n+1}$ для класса \mathcal{F}^n .

□ По [следствию 12](#) существует $T \in \mathcal{F}^{n+2}$, универсальная для \mathcal{F}^{n+1} . Построим U :

- Определим $U(x, \bar{y}) := T(l(x), r(x), \bar{y})$.

Здесь, как обычно, c — канторовская нумерация, l, r — левая и правая обратные функции.

- Докажем, что U главная. Пусть $V \in \mathcal{F}^{n+1}$ — некоторая функция. Так как T универсальная для содержащего V класса, существует m (номер функции V среди сечений T), такой что

$$\forall x, \bar{y}: V(x, \bar{y}) = T(m, x, \bar{y}).$$

Проверим, что транслятор $s(x) = c(m, x)$ подойдет, то есть $V(x, \bar{y}) = U(s(x), \bar{y})$.

$$\begin{aligned} U(s(x), \bar{y}) &= && \text{(По определению } U) \\ &= T(l(s(x)), r(s(x)), \bar{y}) && = T(l(c(m, x)), r(c(m, x)), \bar{y}) = \\ &= T(m, x, \bar{y}) && = V(x, \bar{y}) \end{aligned}$$

Значит, U главная.

Второе доказательство:

□ Аналогично мы строили универсальную функцию выше ([теорема 2.3.1](#) и [следствие 12](#)).

Докажем для двумерного случая. Пусть есть U — универсальная функция, которую мы построили в [теореме 2.3.1](#). Покажем, что U — главная. Пусть V — некоторая вычислимая. Нужно построить транслятор s .

Определим его так. У нас имеется некоторая двухместная МТ M , которая вычисляет V . Зафиксируем у неё первый аргумент n и получим одноместную МТ M' . При этом, за конечное время мы можем найти номер M' в нумерации U — пусть i . Тогда, положим $s(n) = i$. Получили алгоритм для s , что и требовалось.

Теорема 2.3.8 (О вычислимости номера композиции). $U \in \mathcal{F}^2$ — универсальная функция для класса \mathcal{F} . U — **главная** универсальная тогда и только тогда, когда существует $f \in \mathcal{F}_*^2$, такая что

$$U_p \circ U_q = U_{f(p,q)}.$$

То есть $\forall p, q, x \in \mathbb{N}: U(p, U(q, x)) = U(f(p, q), x)$.

□

1 \Rightarrow 2 Пусть U — главная.

Рассмотрим $V(n, x) = U(l(n), U(r(n), x))$, то есть $V(c(p, q), x) = U(p, U(q, x))$.

Фактически V — это $U_p \circ U_q$. Так как U — главная универсальная,

$$\exists s \in \mathcal{F}_*: V(n, x) = U(s(n), x).$$

Тогда

$$\begin{aligned} U_p \circ U_q &= U(p, U(q, x)) = V(c(p, q), x) = \\ &= U(s(c(p, q)), x) = U_{s(c(p, q))}(x) \end{aligned} \quad (\text{По определению транслятора})$$

Теперь обозначим $f(p, q) = s(c(p, q))$ и получим нужное равенство.

2 \Rightarrow 1

□ Пусть есть такая функция $f(p, q) \in \mathcal{F}_*^2$, что $U_p \circ U_q = U_{f(p, q)}$. Хотим доказать, что U главная универсальная.

- Построим функцию (транслятор) $t \in \mathcal{F}$ такую, что $\forall n, x \in \mathbb{N} \quad U(t(n), x) = c(n, x)$.
Для этого рассмотрим две вспомогательные функции

$$\begin{aligned} k(z) &= c(0, z) \\ g(z) &= c(l(z) + 1, r(z)) \end{aligned}$$

Так как U универсальная, эти функции имеют номера, пусть n_k и n_g соответственно. Тогда можно определить функцию t так:

$$\begin{cases} t(0) = n_k \\ t(n+1) = f(n_g, t(n)) \end{cases}$$

Проверим по индукции по n , что $U(t(n), x) = c(n, x)$:

- База: $n = 0$. $U(t(0), x) = U(n_k, x) = U_{n_k}(x) = k(x) = c(0, x)$, что и требовалось.
- Переход: $n \rightarrow n+1$.

$$\begin{aligned} U(t(n+1), x) &= U(f(n_g, t(n)), x) = U_{n_g} \circ U_{t(n)}(x) && (\text{определение } t \text{ и свойство } f) \\ &= g \circ c_n(x) = g(c(n, x)) && (\text{предположение индукции}) \\ &= c(n+1, x) \end{aligned}$$

Теперь можем пользоваться t .

- Построим транслятор для любой функции $V \in \mathcal{F}^2$. Пусть $h(y) = V(l(y), r(y))$. Так как U универсальная, есть сечение $U_a = h$. Подставим $c(n, x)$:

$$U_a(c(n, x)) = h(c(n, x)) = V_n(x).$$

Выразим $c(n, x)$ через U с помощью t :

$$U_{f(a, t(n))}(x) = U_a \circ U_{t(n)}(x) = U_a(U(t(n), x)) = U_a(c(n, x)) = V_n(x)$$

Тогда $s(n) = f(a, t(n))$ — нужный транслятор для V .



2.3.3 Теорема Райса

Определение 59: Свойство функций

Свойство \mathcal{A} функций класса \mathcal{C} — подмножество функций, удовлетворяющих этому свойству, то есть лежащих в \mathcal{A} .

Нетривиальное свойство — не пустое и не совпадающее со всем классом: $\emptyset \subsetneq \mathcal{A} \subsetneq \mathcal{C}$.

Теорема 2.3.9 (Райса / Успенского). Пусть $\mathcal{A} \subset \mathcal{F}$ — некоторое нетривиальное свойство вычислимой функции, U — главная универсальная функция для всех вычислимых функций класса \mathcal{F} .

Тогда не существует алгоритма, который по U -номеру вычислимой функции проверяет \mathcal{A} .

То есть множество $A = \{n \mid U_n \in \mathcal{A}\}$ неразрешимо.

□ Покажем, что, если свойство \mathcal{A} можно алгоритмически проверить, то любые два непересекающихся перечислимых множества можно отделить некоторым разрешимым.

Предположим, что A разрешимо.

Это равносильно тому, что по U -номеру n , мы проверяем принадлежность функции к \mathcal{A} (то есть проверяем принадлежность номера к A).

Пусть P и Q — произвольные непересекающиеся *перечислимые* множества.

И ζ — какая-нибудь функция из \mathcal{A} , а η — какая-нибудь не из \mathcal{A} .

Рассмотрим следующую функцию:

$$V(n, x) = \begin{cases} \zeta(x), & n \in P \\ \eta(x), & n \in Q \\ \uparrow, & n \notin P \cup Q \end{cases}$$

Заметим, что V вычислима, так как можем запустить по шагам алгоритмы для перечисления P и Q , если один выводит n , то остается вычислить соответствующую функцию (или ζ , или η), а иначе значение не определено, зацикливаемся.

Возьмем s — транслятор для U . Тогда $V_n(x) = U_{s(n)}(x)$.

Определим $\varphi(n) := \chi_A(s(n))$, характеристическая функция есть, так как A разрешимо.

С помощью φ можно отделить P и Q : если принадлежит A , значит из P , иначе из Q . Получаем противоречие со [следствием 13](#). ■

Следствие 14. Множество номеров некоторой заданной функции φ в главной нумерации неразрешимо.

В частности, в главной нумерации множество МТ, вычисляющих одну функцию, бесконечно много.

Следствие 15 (Пример универсальной неглавной функции). Существует универсальная неглавная для класса \mathcal{F}^n функция $V \in \mathcal{F}^{n+1}$.

□ Пусть $U(n, x)$ — произвольная главная универсальная функция для \mathcal{F}^n и D — множество номеров функций в нумерации U с непустой областью определения.

Заметим, что D перечислимое, так как можно построить следующий алгоритм: на шаге k будем для $i \in \{0, 1, \dots, k-1\}$ и $\bar{j} \in \{0, 1, \dots, k-1\}^n$ считать $U_i(\bar{j})$. Для этого даем на каждой из k^{n+1} машин Тьюринга (для $U_i(\bar{j})$) поработать k шагов. Теперь для всех пар (i, \bar{j}) , на которых был выдан результат, выводим i .

Для любой функции U_i с непустой областью определения рано или поздно найдется k , которое больше номера функции и координат какой-то точки из области определения и количества действий, требуемых для вычисления значения в ней, так как мы, во-первых, мы перебираем все точки, а, во-вторых, постоянно увеличиваем количество шагов.

И когда мы дойдем до этого k номер i будет выведен. Поэтому алгоритм действительно перечисляет номера функций, но, естественно, только с непустой областью определения.

Определим функцию $f(n)$, для n равную n -ому элементу D в порядке возвращения построенным алгоритмом.

Можно заметить, что по теореме Райса, D неразрешимо (так как свойство «иметь непустую область определения» очевидно непустое, и D — множество номеров таких функций).

Поэтому D бесконечно, следовательно, для всех n когда-то будет выведен n -ый элемент D .

Теперь рассмотрим функцию

$$V(i, \bar{x}) = \begin{cases} \uparrow, & i = 0 \\ U(f(i-1), \bar{x}), & i \neq 0 \end{cases}$$

Эта функция вычислима, универсальна. При этом единственный номер «нигде не определенного сечения» — только 0, это множество конечно, следовательно разрешимо. Поэтому V неглавная по теореме Райса.

Также любая где-то определенная функция будет получена для какого-то V_n , поэтому V универсальна.

Следствие 16 (Переформулировка следствия 14). Для любой главной нумерации U и любой вычислимой функции f множество $\{n \mid U_n = f\}$ неразрешимо.

2.4 Иммуные и простые множества

Определение 60

Множество называется **иммунным**, если оно бесконечно и не содержит ни одного бесконечного перечислимого множества.

Определение 61

Множество называется **простым**, если оно перечислимо, а его дополнение иммуно.

Теорема 2.4.1. Существуют простые подмножества \mathbb{N}

□ Множество P простое, если \bar{P} иммуно, то есть $\forall A$ - бесконечного перечислимого, выполняется $A \not\subseteq \bar{P}$, другими словами $|\bar{P}| = \infty$ и $A \cap P \neq \emptyset$.

Зафиксируем некоторую главную нумерацию и явно построим алгоритм, перечисляющий P :

```
for steps in [1, inf):
    for i in [1, steps]:
        # running i-th Turing Machine on `steps` steps
        x = # result of TM
        if x != None and x > 2 * i:
            print(x)
```

По построению верно, что \forall бесконечного перечислимого $A \cap P \neq \emptyset$. Также, благодаря условию $x > 2 * i$, среди первых n натуральных чисел, алгоритм выведет $< \frac{n}{2}$, значит оставшиеся $\in \bar{P}$, то есть $|\bar{P}| = \infty$.

Теорема 2.4.2. Любое бесконечное подмножество \mathbb{N} , не содержащее бесконечных разрешимых подмножеств, иммуно.

□ Пусть множество A удовлетворяет условию теоремы, хотим доказать, что оно иммуно. Предположим, что \exists бесконечное перечислимое E такое, что $E \subseteq A$. Если научимся строить бесконечное разрешимое

$R \subseteq E$, то сразу получим противоречие ($R \subseteq E \subseteq A$).

Будем строить такое R явно. А именно, E пересчитимо, поэтому можем запустить перечисляющий его алгоритм. Пусть он выводит последовательность e_1, e_2, \dots — элементы множества E . Выделим бесконечную возрастающую подпоследовательность и положим её элементы в R : во-первых, $e_1 \in R$, далее первый e_k такой, что $e_k > e_1$ тоже лежит в R и так далее. Последовательность $\{e_i\}$ бесконечна, поэтому и $|R| = +\infty$. Покажем, что заданное таким образом R разрешимо. Чтобы проверить принадлежность x к R , запустим построенный алгоритм и будем ждать, пока не появится $y \geq x$. Если $y = x$, то $x \in R$, иначе $x \notin R$. ■

2.5 Теорема о неподвижной точке

Лемма 17. Пусть \equiv — отношение эквивалентности на \mathbb{N} .

Тогда следующие утверждения *не* выполняются одновременно:

1. Для любой $f \in \mathcal{F}$ существует \equiv -продолжение $g \in \mathcal{F}_*$ ^a.
2. Найдется $h \in \mathcal{F}_*$, не имеющая \equiv -неподвижной точки, то есть $\forall n: n \not\equiv h(n)$.

^aТо есть, если $f(x)$ определена, то $g(x)$ тоже определена и $g(x) \equiv f(x)$

□ Рассмотрим $f \in \mathcal{F}$, от которой никакая вычислимая функция не может отличаться всюду, например, $f(x) = U(x, x)$.

Пусть выполняются оба пункта.

1. По первому существует \equiv -продолжение f функция $g \in \mathcal{F}_*$.
2. По второму существует такая $h \in \mathcal{F}_*$, что $\forall n: h(n) \not\equiv n$.

Рассмотрим $t(x) := h(g(x))$ и проверим, что она всюду отличается от f :

- Если f определена, то $f(x) \equiv g(x) \not\equiv h(g(x)) = t(x)$
- Если f не определена, то t определена

Но от f никакая вычислимая функция не может отличаться всюду. ■

Теорема 2.5.1 (О неподвижной точке). Если U — главная универсальная вычислимая функция для класса \mathcal{F} , а $h \in \mathcal{F}_*$, то $\exists n: U_n = U_{h(n)}$.

□ Возьмем в качестве отношения эквивалентности следующее: $x \equiv y \iff U_x = U_y$.

Покажем, что выполняется первый пункт из [леммы 17](#).

Пусть $f \in \mathcal{F}$. Тогда можем рассмотреть $V(n, x) := U(f(n), x)$.

U главная, поэтому существует транслятор:

$$\exists s \in \mathcal{F}_*: \forall n, x \ V(n, x) = U(s(n), x).$$

Проверим, что s и есть \equiv -продолжение f

- если $f(n)$ определена, то $U_{s(n)} = U_{f(n)}$, то есть $s(n) \equiv f(n)$.
- если не определена, то и $U_{s(n)}$ нигде не определена.

В итоге первый пункт [леммы](#) выполняется, поэтому второй не выполняется. ■

Следствие 17. $U(n, x)$ — главная универсальная вычислимая функция. Тогда

$$\exists p \in \mathbb{N}: \quad \forall x \quad U(p, x) = p.$$

- Рассмотрим $V \in \mathcal{F}^2$, такую что $V(n, x) = n$. Тогда существует $s(n)$, такое что $U_{s(n)} = V_n = n$.
Теперь применим теорему о неподвижной точке к $s(n)$:

$$\exists p: \quad U_p = U_{s(p)} = V_p = p.$$



2.6 m -сводимость

Определение 62: m -сводимость

Множество $A \subset \mathbb{N}$ **m -сводится** ($A \leq_m B$) к $B \subset \mathbb{N}$, если существует $f \in \mathcal{F}_*$, такая что

$$\forall x \in \mathbb{N}: \quad x \in A \iff f(x) \in B.$$

Свойства.

- Если $A \leq_m B$ и B разрешимо, то A разрешимо.
- Если $A \leq_m B$ и B перечислимо, то A перечислимо.
- Отношение \leq_m рефлексивно и транзитивно.
- Если $A \leq_m B$, то $\mathbb{N} \setminus A \leq_m \mathbb{N} \setminus B$.



- Чтобы проверить $x \stackrel{?}{\in} A$, проверим $f(x) \in B$. Так как B разрешимо, вторая проверка выдаст какой-то ответ и мы можем его вернуть.
- Аналогично, если $f(x) \in B$, то $x \in A$, а если расходится, то $x \notin A$.
- Для рефлексивности подойдет $f = \text{id}$, для транзитивности берем композицию.
- Инвертируем результат:

$$x \in \mathbb{N} \setminus A \iff x \notin A \iff f(x) \notin B \iff f(x) \in \mathbb{N} \setminus B.$$



Замечание. Разрешимое множество сводится к любому $B \notin \{\emptyset, \mathbb{N}\}$

- Пусть дано разрешимое A . Рассмотрим два элемента $b \in B, b' \notin B$. Тогда $f(x) = b$, если $x \in A$, $f(x) = b'$ иначе. Такая функция f будет вычислима и всюду определена.



Замечание. К пустому множеству сводится только пустое. К \mathbb{N} сводится только \mathbb{N} .

Определение 63: m -полнота

Перечислимое множество A называется **m -полным** (в классе перечислимых множеств), если любое перечислимое B m -сводится к A .

Теорема 2.6.1. Существует m -полное перечислимое множество.

□ Рассмотрим универсальное множество $U \subset \mathbb{N} \times \mathbb{N}$.

Пусть $V \subset \mathbb{N}$ — множество номеров пар из U :

$$c(x, y) \in V \iff (x, y) \in U.$$

Проверим, что это множество подходит.

Пусть T — произвольное перечислимое множество. Так как U универсальное, то

$$\exists n: T = U_n.$$

Тогда

$$x \in T \iff x \in U_n \iff (n, x) \in U \iff c(n, x) \in V.$$

То есть $f(x) = c(n, x)$ сводит T к V . ■

Замечание. Если K — m -полное, $K \leq_m A$ — перечислимое, то A тоже m -полное.

Теорема 2.6.2. $U \subset \mathbb{N} \times \mathbb{N}$ — главное универсальное перечислимое множество. Тогда его диагональ $D = \{x \mid (x, x) \in U\}$ будет m -полной.

□ Во-первых, D перечислимое. Далее предположим, что K — произвольное перечислимое множество. Тогда $V = K \times \mathbb{N}$ перечислимо.

$$V_n = \begin{cases} \emptyset, & n \notin K \\ \mathbb{N}, & n \in K \end{cases}$$

Так как U главная

$$\exists s \in \mathcal{F}_*: V_n = U_{s(n)} = \begin{cases} \mathbb{N}, & n \in K \\ \emptyset, & n \notin K \end{cases}$$

Тогда

$$n \in K \iff s(n) \in U_{s(n)} \iff s(n) \in D.$$

То есть $s(n)$ сводит K к D . ■

Определение 64

Зафиксируем некоторое главное универсальное перечислимое множество W (число n будет номером множества W_n). Будем говорить, что множество A является **эффективно неперечислимым**, если существует такая всюду определенная вычислимая функция f , что $f(z) \in A \Delta W_z$ ^a.

^aЗдесь Δ означает симметрическую разность; другими словами, $f(z)$ является точкой, где A отличается от W_z

Чтобы доказать существование перечислимых неразрешимых множеств, не являющихся m -полными, нужно доказать следующие теоремы (которые хорошо доказаны в книге Шеня). Спойлер: любое простое множество является примером.

Теорема 2.6.3. $A \leq_m B$. A — эффективно неперечислимо, тогда B — эффективно неперечислимо.

Теорема 2.6.4. Существуют перечислимые множества с эффективно неперечислимыми дополнениями.

□ В качестве примера подойдет $D = \{n \mid (n, n) \in W\}$. ■

Теорема 2.6.5. Всякое m -полное перечислимое множество имеет эффективно неперечислимое дополнение.

□ Прямое следствие двух предыдущих теорем. ■

Теорема 2.6.6. Пусть K — перечислимое множество, а A эффективно неперечислимо. Тогда $\mathbb{N} \setminus K \leq_m A$ (что равносильно $K \leq_m \mathbb{N} \setminus A$).

□ Рассмотрим множество $V = K \times \mathbb{N}$. Его сечения V_n либо пусты (при $n \notin K$), либо совпадают со всем натуральным рядом (при $n \in K$). Пользуясь тем, что множество W является главным, мы находим всюду определённую функцию s , для которой $W_{s(n)} = \emptyset$ при $n \notin K$ и $W_{s(n)} = \mathbb{N}$ при $n \in K$. Пусть f — функция, обеспечивающая эффективную неперечислимость множества A . Тогда $f(s(n)) \in A$ при $n \notin K$ и $f(s(n)) \notin A$ при $n \in K$. Другими словами, композиция функций f и s сводит $\mathbb{N} \setminus K$ к множеству A , что и требовалось. ■

Теорема 2.6.7. Перечислимое множество является m -полным тогда и только тогда, когда его дополнение эффективно неперечислимо. Это прямое следствие двух предыдущих теорем.

Теорема 2.6.8. Множество эффективно неперечислимо тогда и только тогда, когда к нему m -сводится дополнение некоторого (вариант: любого) m -полного множества.

Теорема 2.6.9. Любое эффективно неперечислимое множество содержит бесконечное перечислимое подмножество (т. е. не является иммунным).

Утверждение. Существуют перечислимые неразрешимые множества, которые не являются m -полными в классе перечислимых.

□ Примером будут являться простые множества. Они перечислимы, неразрешимы (так как тогда их дополнения были бы разрешимы). Если какое-то простое множество A — m -полное, то его дополнение эффективно неперечислимо, а значит, содержит бесконечное перечислимое подмножество.

Но это противоречит определению простоты, так как дополнение должно быть иммунным. ■

2.7 Проблема соответствия Поста (PCP)

Это одна из самых известных неразрешимых задач.

Даны два конечных списка строк одинаковой мощности над алфавитом A :

$$\begin{aligned} L_1 &: w_1, \dots, w_k \quad x_i, w_i \in A^* \\ L_2 &: x_1, \dots, x_k \quad |L_1| = |L_2| \end{aligned}$$

Хотим проверить, существуют ли конечные последовательности $i_1, \dots, i_m \in \{1, \dots, k\}^+$, такие что

$$w_{i_1} \dots w_{i_m} = x_{i_1} \dots x_{i_m}.$$

Пример 2.7.1

Пусть $L_1 = a^2, b^2, ab^2$, $L_2 = a^2b, ba, b$. Решением будет 1213:

$$w_1 w_2 w_1 w_3 = aabbaaabb = x_1 x_2 x_1 x_3.$$

Пример 2.7.2

Теперь возьмем $L_1 = a^2b, a$, $L_2 = a^2, ba^2$. Несложный перебор приводит к тому, что решений нет.

Переформулировки

- Через гомоморфизмы. Даны два гомоморфизма $h_1, h_2: \Delta^* \rightarrow A^*$. Проверяем, есть ли строка $u \in \Delta^*$: $h_1(u) = h_2(u)$
- Через доминошки. Есть k типов доминошек (w_i, x_i) , каждого типа бесконечно много. Проверяем, можно ли составить последовательно доминошки, чтобы строка сверху совпала со строкой снизу.

Теорема 2.7.1 (Пост, 1946). Проблема соответствия Поста неразрешима.

□ Сведем задачу об остановке односторонней одноленточной МТ к РСР.

Отметим начало ленты символом \triangleright, \sqcup — вместо ε . Начальное состояние q_0 не входит в правые части команд. Также договоримся, что изначально головка МТ указывает на первый символ строки.

По МТ $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{term})$ построим пример для задачи Поста. Считаем, что в конце M стирает все.

Хотим записать историю вычисления МТ.

Пусть $A = \{\triangleright, \# \} \cup \Gamma \cup Q, \#$ — для разделения конфигураций.

1. Начальные доминошки $(\varepsilon, \triangleright q_0 x \#)$, $x \in \Sigma \cup \{\sqcup\}$
2. Доминошки копирования (c, c) , $c \in A$
3. Для всех правил $(q, c) \rightarrow (q', c', +1) : (qc, c'q')$, если $c = \sqcup$, добавляем еще доминошку $(q\#, c'q'\#)$.
4. Для всех правил $(q, c) \rightarrow (q', c', -1) : (aqc, q'ac')$, если $c = \sqcup$, добавляем $(aq\#, q'ac'\#)$
5. Конец строки, для всех $c \in A \setminus \triangleright$, три доминошки $(cq_{term}, q_{term}), (q_{term}c, q_{term}), (\triangleright q_{term}q_{term}, q_{term})$

Замечание. Доминошки копирования дают решение задачи Поста, далее это поправим, а пока считаем, что начинаем с доминошки типа 1.

Шаг 1 Сверху пусто, снизу начальная конфигурация.

Шаг 2 Обязательно доминошка $(\triangleright, \triangleright)$

Шаг i Сверху $i - 1$ конфигурация МТ, снизу i -ая.

Доминошка копирования, доминошка команды, доминошка копирования

Если МТ не останавливается, то последовательность доминошек не совпадет.

Если МТ останавливается, то достаточно добавить доминошку 5 в конец.

Добьемся того, чтобы нельзя было начинать с доминошки копирования. Например, можно сделать две копии алфавита A и A' , дальше раздвоить каждую доминошку: (сверху A , снизу A'), (A', A) . Теперь на первое место можно поставить лишь доминошку типа 1, ведь у остальных снизу и сверху первый символ из разных алфавитов, а значит точно не совпадает. ■

Следствие 18. Задача о пустом пересечении 2-х грамматик алгоритмически неразрешима.

□ Пусть разрешима. Тогда РСР разрешима следующим образом:

$$\begin{aligned} Gr_1: S &\rightarrow x_i S w_i^R / \# \quad \forall i \in \{1, \dots, k\} \\ Gr_2: S &\rightarrow a S a / a \# a \quad \forall a \in A \end{aligned}$$

То есть Gr_1 — строка вида $x_{i_1} \dots x_{i_n} \# (w_{i_1} \dots w_{i_n})^R$, а Gr_2 — $v \# v^R$, где $v \in A^+$

$$L(Gr_1) \cap L(Gr_2) \neq \emptyset \iff \text{РСР имеет решение.}$$

■

Замечание. РСР неразрешима даже для бинарного алфавита, так как можем взять инъективное кодирование бинарными словами.

Замечание. Можно доказать, что РСР неразрешима для списков длины $k = 5$ (без доказательства)

Замечание. Для $k = 2$ разрешима, для $k \in \{3, 4\}$ неизвестно.

2.8 Т-сводимость (по Тьюрингу)

Определение 65: Сводимость по Тьюрингу

Пусть $A, B \subseteq \mathbb{N}$. Тогда B **сводится по Тьюрингу** к A ($B \leq_T A$), если существует алгоритм с оракулом A , отвечающий на вопрос о принадлежности n множеству B .

Свойства.

- $B \leq_m A \implies B \leq_T A$
- $\forall A: A \leq_T \mathbb{N} \setminus A$
- сводимость по Тьюрингу транзитивна и рефлексивна
- $A \leq_T B$ и B разрешимо, то A тоже разрешимо

□

- Хотим научиться проверять принадлежность $n \in B$. По m -сходимости верно, что $f \in \mathcal{F}_* : x \in B \iff f(x) \in A$. Тогда оракул A подойдет, поскольку достаточно спросить принадлежность $f(n)$ к A .
- Очевидно, чтобы проверить принадлежность n к A , имея оракул $\mathbb{N} \setminus A$, достаточно инвертировать ответ оракула.
- Рефлексивность очевидна. Покажем, что $A \leq_T B, B \leq_T C \implies A \leq_T C$. Нужно построить алгоритм, который с оракулом C умеет определять $x \in A$. У нас есть алгоритмы A_B и B_C . Пусть надо проверить, что $x \in A$. Думаем, что алгоритм B_C и есть оракул B . Тогда запускаем алгоритм A_B и вместо оракула пользуемся алгоритмом B_C . Итого справились с оракулом C .
- Просто не пользуемся оракулом B , а вместо него используем алгоритм для проверки принадлежности B , ведь B разрешим.

■

Замечание. Если A перечислимо, $B \leq_T A$, то B может быть неперечислимым.

□ В [теореме 2.3.5](#) про существование перечислимого неразрешимого множества мы привели в качестве примера множество F — область определения $U(n, n) + 1$. Тогда возьмем $A = F, B = \bar{F}$. По второму свойству есть Тьюринг сводимость.

■

Определение 66: Функция с оракулом

Аналогично можно определить вычисления с оракулом f (это всюду определенная функция). Функция вычисляется алгоритмом, который в любой момент может обратиться к оракулу — попросить оракула вычислить $f(n)$.

Лекция 6

18 march

2.9 Арифметическая иерархия

Вспомним следующее свойство: $A \subset \mathbb{N}$ перечислимо тогда и только тогда, когда $\exists B \subset \mathbb{N} \times \mathbb{N}$ разрешимое, такое что A — проекция B , или

$$x \in A \iff \exists y (x, y) \in B \quad (2.9.1)$$

Можем считать A, B свойствами (предикатами), то есть $A(x) \leftrightarrow x \in A$. Тогда можем переписать 2.9.1 так

$$A(x) \iff \exists y B(x, y).$$

Какие множества представимы в виде $\forall y: B(x, y)$? Это равносильно

$$\neg (\exists y \neg B(x, y)).$$

Это **коперечислимые** (то есть дополнение перечислимого)

Определение 67

$A \in \Sigma_n$, если его можно представить в виде

$$A(x) \iff \exists y_1 \forall y_2 \exists y_3 \dots B(x, y_1, \dots, y_n),$$

где B разрешимо.

$A \in \Pi_n$, если

$$A(x) \iff \forall y_1 \exists y_2 \forall y_3 \dots B(x, y_1, \dots, y_n).$$

Свойства.

1. Определение не изменится, если разрешить несколько одинаковых кванторов подряд, так как можем заменить повторные на один с помощью канторовой нумерации
2. $A(x) \in \Sigma_n \iff \neg A(x) \in \Pi_n$

Теорема 2.9.1. Если $A(x), B(x) \in \Sigma_n$ (или Π_n), то

$$A(x) \cup B(x) \in \Sigma_n \text{ (или } \Pi_n)$$

$$A(x) \cap B(x) \in \Sigma_n \text{ (или } \Pi_n).$$

□ Запишем определения

$$A(x) \iff \exists y_1 \forall y_2 \exists y_3 \dots P(x, y_1, \dots, y_n)$$

$$B(x) \iff \exists z_1 \forall z_2 \exists z_3 \dots Q(x, z_1, \dots, z_n)$$

Скомбинируем кванторы

$$A(x) \cap B(x) \iff \exists y_1 \exists z_1 \forall y_2 \forall z_2 \dots P(x, y_1, \dots, y_n) \cap Q(x, z_1, \dots, z_n).$$

$$A(x) \cap B(x) \iff \underbrace{\exists s_1}_{c(y_1, z_1)} \forall s_2 \dots \underbrace{S(x, s_1, \dots, s_n)}_{P(x, l(s_1), \dots, l(s_n)) \cap Q(x, r(s_1), \dots, r(s_n))}.$$

Так как P и Q разрешимы их объединения и пересечения тоже разрешимы. ■

Замечание. Аналогично можно определить Σ_n, Π_n для подмножеств \mathbb{N}^m .

Свойства.

- $\Sigma_n, \Pi_n \subseteq \Sigma_{n+1}, \Pi_{n+1}$
- $\Sigma_n \cup \Pi_n \subseteq \Sigma_{n+1} \cap \Pi_{n+1}$

□ Аналогично прошлой теореме сворачиваем кванторы в группы, добавляем кванторы.
В итоге получается следующая картина

$$\begin{array}{c} \Sigma_0 \subseteq \Sigma_1 \subseteq \Sigma_2 \subseteq \Sigma_3 \subseteq \dots \\ \parallel \quad \subsetneq \quad \subsetneq \\ \Pi_0 \subseteq \Pi_1 \subseteq \Pi_2 \subseteq \Pi_3 \subseteq \dots \end{array}$$

Теорема 2.9.2. $A \leq_m B, B \in \Sigma_n$, то $A \in \Sigma_n$ (аналогично с Π_n)

□ Распишем согласно определениям

$$A \leq_m B \stackrel{\text{def}}{\iff} \exists f \in \mathcal{F}_*: x \in A \iff f(x) \in B$$

и

$$B \in \Sigma_n \stackrel{\text{def}}{\iff} x \in B \iff \exists y_1 \forall y_2 \dots R(x, y_1, y_2, \dots, y_n).$$

Тогда

$$x \in A \iff f(x) \in B \iff \exists y_1 \forall y_2 \dots R(f(x), y_1, \dots, y_n).$$

Так как $f \in \mathcal{F}_*$, то и $R(f(x), y_1, \dots, y_n)$ разрешимо.

Определение 68: Универсальное множество

Множество $W \subset \mathbb{N} \times \mathbb{N}$ **универсальное** для перечислимых подмножеств \mathbb{N} , если

- W перечислимое;
- для всех перечислимых $B \subset \mathbb{N}$ существует номер n , такой что $W_n = B$.

Определение 69: Главное универсальное множество

Множество $W \subset \mathbb{N} \times \mathbb{N}$ **главное универсальное** для всех перечислимых подмножеств \mathbb{N} , если для любого перечислимого $V \in \mathbb{N} \times \mathbb{N}$ существует функция $s: \mathbb{N} \rightarrow \mathbb{N}$:

- s вычислима и всюду определена;
- $\forall x, n: (n, x) \in V \iff (s(n), x) \in W$

Пример 2.9.1

Например, таким множеством будет область определения главной универсальной функции.

Теорема 2.9.3. Для любого $n > 0$ в классе Σ_n (соответственно Π_n) существует множество, универсальное для всех множеств в Σ_n (соответственно Π_n)

Замечание. Если A — универсальное в Σ_n , то \bar{A} — универсальное в Π_n

□ Докажем по индукции.

Для Σ_1 – перечислимое, уже строили. Для Π_2

$$\forall y \underbrace{\exists z R(x, y, z)}_{\substack{\text{разрешимо} \\ P(x, y)}} \implies \forall y \underbrace{P(x, y)}_{\text{перечислимое свойство}}.$$

Рассмотрим $U(n, x, y)$ — универсальное множество для перечислимых. Тогда

$$T(n, x) := \forall y U(n, x, y).$$

$T(n, x)$ получается универсальным для Π_2 .

Следовательно, существует универсальное для Σ_2 — дополнение до $T(n, x)$.

Продолжаем далее по индукции: для 3 начинаем с Σ_3

$$\Sigma_3: \exists y \forall z \exists t R(x, y, z, t) \iff \exists y \forall z P(x, y, z).$$

Универсальное для Σ_3 :

$$T(n, x) = \exists y \forall z U(n, x, y, z), \text{ где } U \text{ — универсальное перечислимое.}$$

И так далее

Теорема 2.9.4. Универсальное множество для Σ_n не принадлежит Π_n и наоборот.

□ Пусть $T(m, x)$ — универсальное Σ_n -свойство. И предположим, что $T(m, x) \in \Pi_n$.

Рассмотрим $D(x) = T(x, x) \in \Pi_n$, так как $D \leq_m T$.

Поэтому $\neg D(x) \in \Sigma_n$, но оно отличается от всех сечений $T(m, x)$. Противоречие.

Следствие 19. $\Sigma_n \not\subseteq \Sigma_{n+1}$ и $\Pi_n \not\subseteq \Pi_{n+1}$.

□ Знаем, что $\exists x \in \Pi_n \setminus \Sigma_n$, а также что $\Pi_n \subseteq \Sigma_{n+1}$, то есть $x \notin \Sigma_n$ и $x \in \Sigma_{n+1}$.

2.10 Еще про Т-сводимость

Зафиксируем некоторую функцию-оракула α . Вся теория вычислимости может быть «релятивизована» относительно вычислений с оракулом α . То есть теперь все вычисления просто выполняются с оракулом α .

Определение 70

Функцию, вычислимую с оракулом α будем называть α -вычислимой.

Обозначение. \mathcal{F}_α^m — класс α -вычислимых функций от m аргументов.

Теорема 2.10.1. Пусть α — всюду определенная функция. Тогда $\exists U_\alpha(n, x) \in \mathcal{F}_\alpha^2$ — универсальная для класса \mathcal{F}_α^1

□ Пусть $U_\alpha(i, x)$ — результат применения i -ой МТ с оракулом α к x . Как устроен оракул нам не важно, поэтому можем просто вшить обращение к нему во входные данные.

Аналогично перечислимым множествам можем определить α -перечислимые множества как:

- область определения α -вычислимой функции
- область значений α -вычислимой функции
- проекция α -разрешимого множества

Теорема 2.10.2.

1. Для любого $X \in \mathbb{N}$ существует универсальное X -перечислимое множество для X -перечислимых.
2. Это множество будет m -полным в классе X -перечислимых.

□ Доказательство полностью аналогично такой же теореме для обычной перечислимости. ■

Определение 71

Множества P и Q являются **m -эквивалентными** ($P \equiv_m Q$), если $P \leq_m Q$ и $Q \leq_m P$.

m -степень — $\deg_m(P) = \{Q \mid Q \equiv_m P\}$.

Определение 72

Множества P и Q являются **T -эквивалентными** ($P \equiv_T Q$), если $P \leq_T Q$ и $Q \leq_T P$.

T -степень — $\deg_T(P) = \{Q \mid Q \equiv_T P\}$.

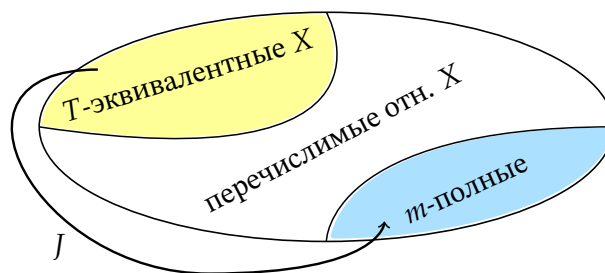
Замечание. Если $P, Q \in \deg_T(X)$, то $\mathcal{F}_P = \mathcal{F}_Q$ и P -перечислимость эквивалентна Q -перечислимости.

Поэтому можем говорить о $\deg_T X$ -перечислимости.

Определение 73: Операция скачка

Операция скачка $J: \deg_T \rightarrow \deg_T$ (или $\deg_T \rightarrow \deg_m$) выбирает для X какое-то m -полное относительно X -перечислимости.

То есть по всем эквивалентным $\deg_T(X)$ получаем X -перечислимые и среди них выбираем полные.

 **T -степени**

- \mathcal{O} — степень, содержащая все разрешимые.
- $\mathcal{O}' = J(\mathcal{O})$ — степень m -полного перечислимого неразрешимого. Один из представителей — область определения универсальной функции.
- $\mathcal{O}^{(n+1)} = (\mathcal{O}^{(n)})'$

Можно считать, что \mathcal{O}' — множества, перечислимые с оракулом проблемы остановки.

2.11 Теорема об арифметической иерархии

Теорема 2.11.1 (Об арифметической иерархии). $\forall n \geq 1: \Sigma_n = \{\mathcal{O}^{(n-1)}\text{-перечислимые множества}\}$

□ Для $n = 1$ уже знаем, это перечислимые множества.

□ Рассмотрим $X \in \Sigma_2$, тогда

$$x \in X \iff \exists y \forall z \underbrace{R(x, y, z)}_{\text{разрешимо}}.$$

Навешиваем отрицание

$$X' := \neg (\forall z R(x, y, z)) \in \Sigma_1.$$

Принадлежность Σ_1 дает m -сводимость к m -полному перечислимому множеству. А так как m -сводимость влечет T -сводимость, то по определению \mathcal{O}' множество X' будет \mathcal{O}' -разрешимо.

Следовательно, его дополнение тоже \mathcal{O}' -разрешимо.

А значит проекция $\neg X' (X)$ будет \mathcal{O}' -перечислима, так как можно перебрать y .

Аналогично действуем для больших n :

$$x \in \Sigma_n: x \in X \iff \exists y \underbrace{R(x, y)}_{\in \Pi_{n-1}}$$

Тогда $\neg R \in \Sigma_{n-1}$.

На предыдущем слое доказали, что тогда $\neg R$ будет $\mathcal{O}^{(n-2)}$ -перечислимо, а тогда оно $\mathcal{O}^{(n-1)}$ -разрешимо. Тогда его проекция $\mathcal{O}^{(n-1)}$ -перечислима.

□ См. далее (страница 71).



2.11.1 Утверждения для доказательства в обратную сторону

Определение 74

Рассмотрим c — некоторую вычислимую нумерацию конечных множеств.

Пусть D_x — множество с номером x .

Возьмем $A \subset \mathbb{N}$ (не обязательно конечное).

Определим $\text{Subset}(A) = \{x \mid D_x \subset A\}$ — множество номеров конечных подмножеств A .

Аналогично $\text{Disjoint}(A) = \{x \mid D_x \cap A = \emptyset\}$.

Лемма 18 (о Subset). Если $A \in \Sigma_n$ (или Π_n), то $\text{Subset}(A) \in \Sigma_n$ (или Π_n соответственно).

□ Пусть $A \in \Sigma_3$,

$$x \in A \iff \exists y \forall z \exists t \underbrace{R(x, y, z, t)}_{\text{разрешимо}}.$$

Для конечного набора

$$\{x_1, \dots, x_m\} \subset A \iff \exists (y_1, \dots, y_m) \forall (z_1, \dots, z_m) \exists (t_1, \dots, t_m) \bigwedge_{i=1}^m R(x_i, y_i, z_i, t_i).$$

А это равносильно $c(x_1, \dots, x_m) \in \text{Subset}(A)$.



Лемма 19 (о Disjoint). Если $A \in \Sigma_n$ (или Π_n), то $\text{Disjoint}(A) \in \Pi_n$ (или Σ_n соответственно).

□

$$D_x \cap A = \emptyset \iff D_x \subset \bar{A} \iff x \in \text{Subset}(\bar{A})$$

То есть $\text{Disjoint}(A) = \text{Subset}(\bar{A})$. Тогда

$$\begin{aligned} A \in \Sigma_n &\iff \bar{A} \in \Pi_n \implies && \text{(по лемме о Subset)} \\ \text{Subset}(\bar{A}) &\in \Pi_n \iff \\ \text{Disjoint}(A) &\in \Pi_n \end{aligned}$$

■

2.11.2 Относительная вычислимость: эквивалентные определения

Определение 75: Образец

Образец — функция $\mathbb{N} \rightarrow \mathbb{N}$, область определения которой конечна. Задается конечным множеством пар, то есть можем вычислимо пронумеровать образцы.

Образцы **совместны**, если объединение их графиков есть график функции. Т.е. если оба определены для какого-то x , то значение на этом x должно совпадать.

Определение 76

M — множество троек (x, y, t) , где $x, y \in \mathbb{N}$, а t — образец.

Две тройки $(x_1, y_1, t_1), (x_2, y_2, t_2)$ **противоречат друг другу**, если $x_1 = x_2, y_1 \neq y_2$ и t_1 и t_2 совместны.

Множество M **корректно**, если оно не содержит противоречащих троек.

Определение 77

Пусть M — корректное множество троек, α — функция.

$$M_1 = \{(x, y, t) \mid (x, y, t) \in M, t \text{ — подмножество графика } \alpha\}$$

$$M_2 = \{(x, y) \mid \exists t: (x, y, t) \in M_1\}$$

Тогда M_2 определяет график некоторой функции $M[\alpha]$. Причем определение корректно, так как для всех x не больше одного значения.

Теорема 2.11.2. Частичная функция f лежит в \mathcal{F}_α^1 , где $\alpha: \mathbb{N} \rightarrow \mathbb{N}$ всюду определена, тогда и только тогда, когда существует корректное перечислимое множество троек M , такое что $f = M[\alpha]$.

□

1 \implies 2 У нас есть алгоритм с оракулом α , вычисляющий f . Построим M .

Построим дерево, моделирующее работу алгоритма на входе x по всем значениям оракула. Скорее всего, дерево будет бесконечным, с бесконечным числом веток.

Если на входе x возможен ответ y , запишем тройку (x, y, t) в M , где t — образец, содержащий все ответы оракула на этой ветке.

- M перечислимо, так как можем стандартным образом давать поработать k первым входам по k шагов, а k увеличивать от 0 до ∞ .

- M корректно. Пусть $(x, y_1, t_1), (x, y_2, t_2) \in M$ и $y_1 \neq y_2$. Эти тройки соответствуют разным путям в дереве (так как $y_1 \neq y_2$), найдем вершину, где они разделились.

В этом месте рассматриваются два разных ответа оракула, причем первый содержится в t_1 , а второй — в t_2 , поэтому t_1 и t_2 несовместны.

Поэтому противоречивых троек нет.

Следовательно, M корректно.

Проверим, что $f = M[\alpha]$. Пусть $f(x) = y$. Это соответствует ветке дерева, начинающейся с входа x и заканчивающейся y , возможно с запросами к α .

Рассмотрим t — образец, содержащий пары вопрос-ответ в этой ветке.

t является частью α , а $(x, y, t) \in M$. Значит, $M[\alpha](x)$ определено и равно y .

2 \Rightarrow 1 Пусть есть корректное перечислимое множество троек M , такое что $f = M[\alpha]$.

Нужно построить алгоритм с оракулом α , считающий функцию f .

На входе x запускаем алгоритм, перечисляющий M . Выбираем тройку, в которой первый элемент равен x .

Обозначим тройку (x, y, t) . Так как t конечное, по каждому элементу можем задать вопрос оракулу α , тем самым проверим, что t является частью α .

Если да, то $f(x) := y$, иначе продолжаем спрашивать про следующий элемент. Если «да» никогда не получаем, то функция не определена в этой точке.

В итоге мы построили алгоритм, который вычисляет $M[\alpha]$.



Дадим аналогичное описание для α -перечислимых множеств.

Определение 78

Рассмотрим произвольное множество E пар (x, t) , где $x \in \mathbb{N}$ и t — образец.

$$E[\alpha] := \{x \mid \exists (x, t) \in E, t \text{ является частью } \alpha\}.$$

Теорема 2.11.3 (о характеристике относительной вычислимости). X — α -перечислимое тогда и только тогда, когда существует перечислимое множество пар E , такое что $X = E[\alpha]$.

□

1 \Rightarrow 2 Если X — α -перечислимое, то X — область определения α -вычислимой функции f .

По [предыдущей теореме 2.11.2](#) $f = M[\alpha]$, для некоторого перечислимого корректного M .

Можем получить E выкалыванием второй координаты из M .

Так как $E[\alpha]$ является областью определения $M[\alpha] = f$, $E[\alpha] = X$.

2 \Rightarrow 1 Пусть $X = E[\alpha]$ для некоторого E .

Рассмотрим $M = \{(x, 0, t) \mid (x, t) \in E\}$, оно корректно, поэтому соответствующая функция $M[\alpha]$ будет определена на $X = E[\alpha]$ и принимает только значение 0. Соответственно X — область определения вычислимой $M[\alpha]$, а значит перечислимо.



Продолжим доказательство [главной теоремы](#) (страница 69).

□ Сначала докажем, что любое \mathcal{O}' -перечислимое множество лежит в Σ_2 .

Пусть A является O' -перечислимым. По определению, A перечислимо с помощью оракула для какого-то перечислимого B . То есть оно перечислимо относительно $\mathbb{1}_B$.

По теореме о характеристике относительной вычислимости, существует перечислимое множество Q пар вида (x, t) , где $x \in \mathbb{N}$ и t — образец, такое что

$$x \in A \iff \exists t: (x, t) \in Q, \mathbb{1}_B \text{ продолжает } t.$$

Можем считать, что

- « $\mathbb{1}_B$ продолжает t » означает, что B содержит множество, на котором $t = 1$, и не пересекается с множеством, на котором $t = 0$
- функция, соответствующая t , принимает только 0 и 1, так как иначе $\mathbb{1}_B$ не сможет ее продолжить

Поэтому вместо образцов в данном случае можно рассматривать пары конечных множеств и вместо множества пар Q — множество P троек (x, u, v) , где u — номер конечного множества, где t принимает значение 1, v — номер конечного множества, где t принимает значение 0 (u, v однозначно задают t)

$$x \in A \iff \exists u \exists v \left((x, u, v) \in P \wedge \underbrace{D_u \subset B}_{u \in \text{Subset}(B)} \wedge \underbrace{D_v \cap B = \emptyset}_{v \in \text{Disjoint}(B)} \right).$$

Так как P перечислимо, $P \in \Sigma_1$, второе свойство ($u \in \text{Subset}(B)$) по лемме о Subset принадлежит Σ_1 , а третье $v \in \text{Disjoint}(B)$ принадлежит Π_1 по лемме о Disjoint.

То есть все условие в скобках принадлежит Σ_2 , поэтому и вся правая часть из Σ_2 . Доказали для $n = 2$.

Дальше действуем аналогично, заменив 2 на n . ■

Следствие 20. $\Sigma_n \cap \Pi_n = \{O^{(n-1)}\text{-разрешимые}\}$

□ Релятивизованная теорема Поста. ■

Следствие 21. $\Sigma_n \cup \Pi_n \not\subseteq \Sigma_{n+1} \cap \Pi_{n+1}$ для $n > 0$

□ По определению $O^{(n)}$ это степень m -полного множества X в классе $O^{(n-1)}$ -перечислимых.

Если X является m -полным, то оно не $O^{(n-1)}$ -разрешимо в этом классе (потому что иначе иерархия бы схлопнулась), поэтому \bar{X} не является $O^{(n-1)}$ -перечислимым в этом классе.

По теореме об арифметической иерархии $X \in \Sigma_n$, $\bar{X} \in \Pi_n$ и $\bar{X} \notin \Sigma_n$, $X \notin \Pi_n$.

Рассмотрим $A = \{2n \mid n \in X\} \cup \{2n + 1 \mid n \notin X\}$. Так как X и \bar{X} m -сводится к A , то $A \notin \Sigma_n$ и $A \notin \Pi_n$ (по свойству m -сводимости: X m -сводится к A и $A \in K \implies X \in K$).

При этом $A \in \Sigma_{n+1} \cap \Pi_{n+1}$, так как оно разрешимо с оракулом из $O^{(n)}$. ■

2.12 Классификация множеств в иерархии

Теорема 2.12.1. Множество номеров нигде не определенной функции в главной нумерации Π_1 -полное

□ Достаточно доказать, что его дополнение Σ_1 -полное.

Пусть U — главная универсальная вычислимая функция, $A = \{n : \exists x U_n(x) \neq \uparrow\}$,

U' — область определения U (главное универсальное перечислимое множество).

Заметим, что $A = \{n : \exists x U'(n, x)\}$. $A \in \Sigma_1$, поскольку U' перечислимо и квантор существования ничего не портит.

Пусть $X \in \Sigma_1$.

По определению $n \in X \Leftrightarrow \exists x R_X(n, x)$, где R_X - некоторое разрешимое множество.

Тогда $n \in X \Leftrightarrow \exists x R_X(n, x) \Leftrightarrow \exists x U'(s(n), x) \Leftrightarrow s(n) \in A$, где s - транслятор для R_X .

Тогда подойдет $f = s$. ■

Теорема 2.12.2. 1. Пусть $U \in \mathcal{F}^2$ — универсальная для \mathcal{F} . Тогда

$$\{n \mid U_n \text{ всюду определено тождественно равным } 0\} \in \Pi_2$$

2. Если U — главная, то это множество Π_2 -полное.

□

1. Пусть

$$A = \{n \mid \forall k \exists t U(n, k) \text{ заканчивает работу за } t \text{ шагов и выдаёт } 0\}.$$

Предикат $R(n, k, t)$, который определен как « $U(n, k)$ заканчивает работу за t шагов и выдаёт 0», можно проверить, запустив МТ на t шагов. Значит он разрешим, то есть $A \in \Pi_2$.

2. Докажем, что к нему сводится произвольное множество $P \in \Pi_2$, то есть

$$x \in P \Leftrightarrow \forall y \exists z \underbrace{R(x, y, z)}_{\text{разрешимо}}.$$

Рассмотрим $S(x, y)$, которая перебирает z и ищет такое, что $R(x, y, z)$ выполнено. Если нашли, возвращаем 0.

То есть $S_x \equiv 0 \Leftrightarrow x \in P$.

Так как U главная нумерация, $\exists s: U_{s(x)} = S_x$. То есть s сводит P к множеству номеров, которое тождественно нулевой функции. ■

Пример на третьем слое — множества с конечными дополнениями.

Задача. Пусть f — вычислимая функция, U — главная нумерация. $A = \{n \mid U_n = f\}$.

Найти минимальный класс для множества A (он зависит от функции).

Part III

Теории информации

Chapter 1

Информация по Хартли

1.1 Базовые свойства

Лекция 1
1 April

Пусть у нас есть конечное множество объектов A . Выдернем какой-то элемент.

Мы хотим придумать описание этого элемента, которое будет отличать его от всех остальных. Хотим ввести некоторую *меру информации*, то есть сколько информации мы узнаем, когда вытягиваем некоторый элемент из A .

Самый простой вариант — число битов требуемое для записи объекта.

Свойства, которые мы хотим получить от меры информации $\chi(A)$:

1. χ дает нам оценку на длину описаний
2. $\chi(A \cup B) \leq \chi(A) + \chi(B)$
3. Если $A := B \times C$, то чтобы описать элемент из A , достаточно описать элемент из B и из C :

$$\chi(A) \leq \chi(B) + \chi(C).$$

Определение 79: Информация по Хартли

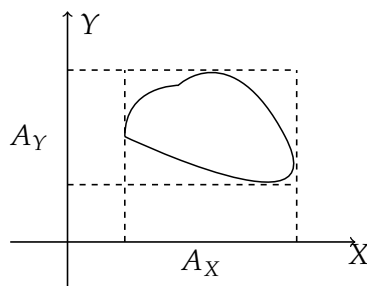
Пусть A — некоторое конечное множество. За **информацию в множестве** A будем принимать следующую величину:^a $\chi(A) := \log |A|$

^aЗдесь и далее под $\log x$ подразумевается двоичный логарифм.

Замечание. Очевидно, второе свойство выполнено для такого определения (проблемы есть, только когда одно из множеств одноэлементно и не содержится во втором). В третьем даже равенство.

Описание — например, битовая строка. Если логарифм нецелый, округляем вверх.

Пусть $A \subseteq X \times Y$. Обозначим **проекции** A_X и A_Y .



Утверждение. Пусть $A \subseteq X \times Y$ и конечно. Тогда для $\chi(A)$ выполнены следующие свойства:

1. $\chi(A) \geq 0$
2. $\chi(A_X) \leq \chi(A)$ и $\chi(A_Y) \leq \chi(A)$
3. $\chi(A) \leq \chi(A_X) + \chi(A_Y)$

□

1. Очевидно
2. $|A_X| \leq |A|$
3. $\chi(A) \leq \chi(A_X \times A_Y) \leq \chi(A_X) + \chi(A_Y)$

■

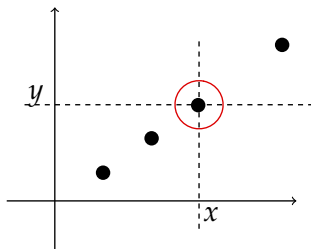


Figure 1.1: Диагональное множество

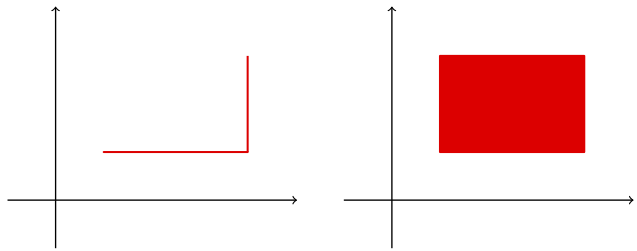


Figure 1.2: Неотличимые множества

Пример 1.1.1

Рассмотрим такой пример (см. рисунок 1.1): множество из точек на диагонали. Здесь, зная первую координату, можно сразу понять вторую.

Попробуем усилить третье свойство:

- 3'. $\chi(A) \leq \chi(A_X) + \chi_{Y|X}(A)$, где $\chi_{Y|X}(A)$ — описание Y при условии X .

Как будем определять $\chi_{Y|X}(A)$? Можно взять $\max_{x \in X} \log(|A_x|)$. Теперь для диагонального множества $\chi_{Y|X}$ просто обнуляется и неравенство переходит в равенство.

Пример 1.1.2

Но если взять одно множество из двух отрезков, а второе из всего прямоугольника, возникнут проблемы (см. рисунок 1.2).

Во-первых, на первой картинке, передав x -координату столбца, придется передавать и y тоже. Во-вторых, мы не сможем отличить эти множества по одной координате.

Лемма 20. Пусть $A \subseteq A_1 \times A_2 \times A_3$ конечно. Тогда^a

$$2\chi(A) \leq \chi(A_{12}) + \chi(A_{13}) + \chi(A_{23}).$$

^aЗдесь $A_{ij} = \{(x, y) \mid x \in A_i \wedge y \in A_j \wedge \exists z: (x, y, z) \in A\}$

□ Перепишем неравенство из условия:

$$2 \log |A| \leq \log |A_{12}| + \log |A_{13}| + \log |A_{23}|$$

$$\log |A|^2 \leq \log(|A_{12}| \cdot |A_{13}| \cdot |A_{23}|)$$

$$|A|^2 \leq |A_{12}| \cdot |A_{13}| \cdot |A_{23}|$$

(log — возрастающая функция)

Пусть $A = \{(x, y, z) \mid x \in A_1, y \in A_2, z \in A_3\}$

Сгруппируем элементы A следующим образом: $A = \bigsqcup B_{ij}$, где

$$B_{ij} = \{(x_i, y_j, z_k) \mid 1 \leq k \leq K_{ij}, x_i \in A_1, y_j \in A_2, z_k \in A_3\}.$$

То есть фиксируем x_i, y_j , а z_k меняется в допустимых пределах. При этом $\sum_{i,j} K_{ij} = |A|$, так как это сумма размеров B_{ij} , которая равна размеру A .

Теперь рассмотрим множество

$$A_{13,23} = \{(x_i, z_s, y_j, z_l) \mid (x_i, z_s) \in A_{13}, (y_j, z_l) \in A_{23}\}.$$

Если внимательно посмотреть на определение, можно понять, что $|A_{13,23}| = |A_{13}| \cdot |A_{23}|$.

Теперь опишем элементы, которые заведомо есть в $A_{13,23}$, чтобы оценить его размер снизу. Для этого зафиксируем x_i, y_j , тогда все z_k из B_{ij} точно сюда подходят в качестве z_s или z_l . То есть таких четверок не меньше чем $|B_{ij}|^2 = K_{ij}^2$. А по всем i, j получается, что $|A_{13,23}| \geq \sum K_{ij}^2$.

Таким образом,

$$|A_{13}| \cdot |A_{23}| = |A_{13,23}| \geq \sum K_{ij}^2.$$

При этом заметим, что $|A_{12}| = |\{B_{ij} \neq \emptyset\}|$, поскольку если $B_{ij} \neq \emptyset$, то пара (x_i, y_j) лежит в A_{12} . В таком случае

$$|A_{12}| = |\{B_{ij} \neq \emptyset\}| = |\{K_{ij} \neq 0\}| = n$$

Получим итоговое неравенство

$$\begin{aligned} |A_{12}| \cdot (|A_{13}| \cdot |A_{23}|) &\geq n \cdot \sum K_{ij}^2 \geq \\ &\geq \left(\sum K_{ij}\right)^2 = |A|^2 \end{aligned} \quad \text{(неравенство о средних)}$$

А это то, что мы и хотели доказать.

Пояснение неравенства о средних:

$$\begin{aligned} n \cdot \sum K_{ij}^2 &\geq \left(\sum K_{ij}\right)^2 \\ \sqrt{\left(\sum K_{ij}\right)^2} &\leq \sqrt{n \cdot \sum K_{ij}^2} \\ \frac{\sum K_{ij}}{n} &\leq \frac{\sqrt{n \cdot \sum K_{ij}^2}}{n} \\ \frac{\sum K_{ij}}{n} &\leq \sqrt{\frac{\sum K_{ij}^2}{n}} \end{aligned}$$

Это неравенство между средним арифметическим и средним квадратичным. ■

1.2 Угадывание числа

Обозначим $[n] := \{1, \dots, n\}$.

Симметричный вариант Пусть есть два игрока, первый загадывает число от 1 до n , а второй должен его угадать. Сколько вопросов необходимо задать, чтобы угадать число?

Есть два варианта игры:

- *Адаптивная*, когда второй игрок получает ответ на вопрос сразу
- *Неадаптивная*, когда он пишет сначала все вопросы, а потом получает на них все ответы.

Очевидно, что нам потребуется не менее логарифма запросов: нарисует дерево, где вершины – запросы, по двум ребрам из которых можно перейти в зависимости от ответа. Листья должны содержать $[n]$, поэтому глубина дерева не менее логарифма.

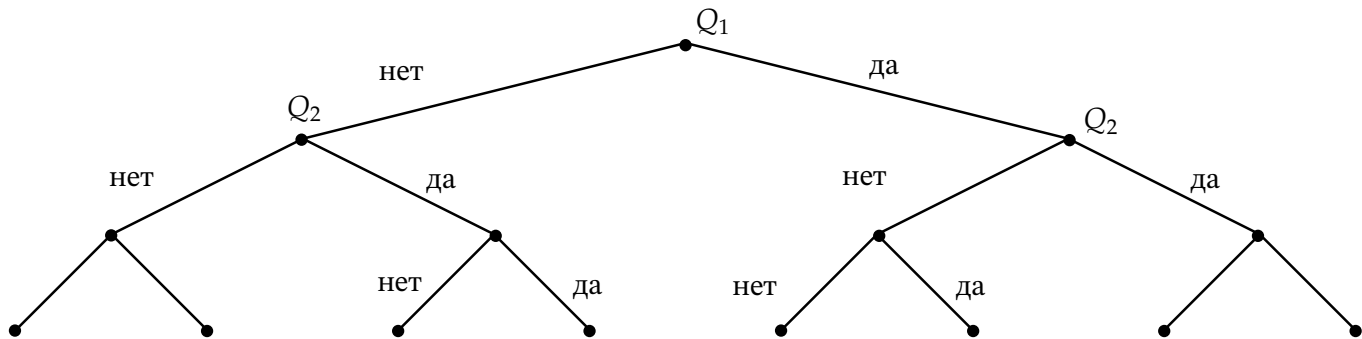


Figure 1.3: Граф вопросов

Теперь подумаем с точки зрения теории информации. Пусть h — число запросов, Q_i — ответ на i -ый запрос (один бит) по некоторому протоколу,

$$B := Q_1 \times \dots \times Q_h,$$

Мы хотим минимизировать h .

Рассмотрим $[n] \times B$ — все возможные пары по всем возможным значениям искомого числа и B . Нас интересует множество $A \subseteq [n] \times B$, соответствующее некоторым корректным запросам:

$$A = \{(m, b) \mid b = (q_1, \dots, q_h), m \text{ — согласовано с ответом}\}.$$

1. $\chi_{[n]|B}(A) = 0$. Ответы на запросы должны однозначно определять число m . Это свойство говорит о корректности протокола, то есть нам ничего не нужно, чтобы, зная ответы, получить m .
2. $\chi(A) \geq \log n$, так как уже первая проекция A имеет n элементов.
С другой стороны, $\chi(A) \leq \chi(A_B) + \chi_{[n]|B}(A) = \chi_B(A) \leq \chi(B) \leq \sum_{i=1}^h \chi(Q_i)$.
А так как $\chi(Q_i) = 1$, $h \geq \log n$.

Асимметричный вариант Пусть теперь за ответ «да» мы платим 1, а за «нет» 2, вариант игры адаптивный. И мы хотим минимизировать не число запросов, а стоимость в худшем случае.

Будем делить запросом множество «пополам» с точки зрения стоимости.

Пусть Q_i — ответ на вопрос «верно ли, что загаданное число m лежит в множестве T_i ?

Пусть A_i — множество элементов, в котором может лежать m после первых $i - 1$ вопросов. В начале это все $[n]$, в конце — одно число.

$$A_i = \{a \in [n] \mid a \text{ согласовано с } Q_1, \dots, Q_{i-1}\}.$$

Стратегия минимальной цены бита информации: берем такое $T_i \subseteq A_i$, что

$$2(\chi(A_i) - \underbrace{\chi(T_i)}_{A_{i+1}}) = \chi(A_i) - \underbrace{\chi(A_i \setminus T_i)}_{A_{i+1}}.$$

Где разность $\chi(A_i) - \chi(A_{i+1})$ обозначает количество информации, которое мы получаем задав i -ый вопрос. То есть слева получили ответ «нет» ($m \in T_i$), а справа «да» ($m \in A_i \setminus T_i$).

Распишем по определению:

$$\begin{aligned} 2(\log|A_i| - \log|T_i|) &= \log|A_i| - \log|A_i \setminus T_i| \\ \log|A_i| &= 2\log|T_i| - \log|A_i \setminus T_i| \\ |A_i| &= \frac{|T_i|^2}{|A_i \setminus T_i|} \end{aligned}$$

Обозначим $|A_i| = k$, $|T_i| = t$ и решим уравнение:

$$\begin{aligned} k &= \frac{t^2}{k-t} \iff t^2 = k(k-t) = k^2 - kt \iff \\ t^2 + kt - k^2 &= 0 \iff t = \frac{-k \pm \sqrt{k^2 + 4k^2}}{2} = k \cdot \frac{-1 + \sqrt{5}}{2} = k \cdot \varphi \end{aligned}$$

То есть нужно выбирать T_i , чтобы $|T_i| = \varphi \cdot |A_i|$.

Тогда «средняя цена» бита будет равна $2(\chi(A_i) - \chi(T_i)) = 2\log \frac{1}{\varphi}$

Докажем оптимальность. Пусть второй игрок меняет число, чтобы мы заплатили как можно больше, причем он знает нашу стратегию.

Если в нашем неравенстве знак \geq , он будет направлять на по «нет», а при \leq «да», за счет чего каждый бит он будет отдавать по цене большей, чем, если бы мы действовали в точности по стратегии.

Следовательно, любая другая стратегия будет требовать большего вклада.

Упражнение (Задача про взвешивания монеток). Есть n монеток и рычажные весы. Хотим найти фальшивую (она одна).

1. Пусть $n = 30$ и мы знаем тяжелее фальшивка или легче. Теперь запрос приносит $\log 3$ информации, так как три ответа.

$$\log 30 \leq \sum_{i=1}^h \chi(a_i) \leq h \log 3.$$

2. $n = 15$, но мы не знаем относительный вес фальшивой монеты. В прошлом неравенстве можно заменить 30 на 29. Если в какой-то момент у нас было неравенство, можем в конце узнать не только номер, но и относительный вес, поэтому у нас 29 исходов.
3. Вопрос: можно ли при $n = 14$? Нет.

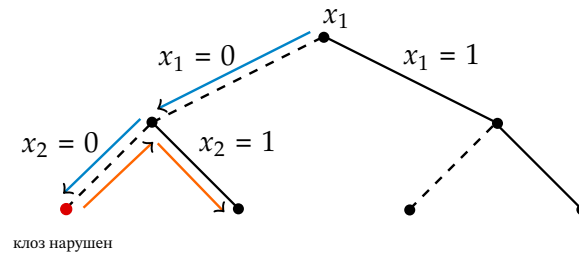
То есть информация по Хартли нам помогла понять два раза, что 3 взвешиваний не хватит. Но её не хватает для того, чтобы решить 3 задачу.

1.3 Задача выполнимости

Вход: $\Phi = \bigwedge_{i=1}^n C_i$ — формула в КНФ.

Будем подставлять значения во все переменные по очереди, тем самым перемещаться по двоичному дереву.

Подставим $x_i = 0$. Если пока клозы не нарушены, подставляем далее $x_{i+1} = 0$ и проверяем клозы аналогично. Если же один из клозов нарушился, вернемся на шаг назад и подставим $x_i = 1$.



Это достаточно эффективный алгоритм, причем мы не ограничиваем выбор последовательности подстановок, порядок 0 и 1.

Таким образом, если есть выполняющий набор, и мы на первом шаге подберем верное значение первой переменной, на второй для второй и так далее, то получим оптимальное решение. Однако не всегда такой алгоритм работает быстро. Пример: задача про рассадку голубей.

1.4 Рассадка голубей

У нас есть $n + 1$ голубь и n клеток, хотим показать, что нельзя рассадить в клетку по одному голубю.

Введем для каждой пары (голубь, клетка) переменную x_{ij} , которая равна 1, если i -ый голубь сидит в j -ой клетке, и $x_{ij} = 0$ иначе.

Тогда, чтобы рассадка была удачной, должны выполняться следующие условия:

- для всех $i \in [n + 1]$ верно $\prod_{j=1}^n (1 - x_{ij}) = 0$. То есть для каждого голубя нашлась клетка.
- для всех $i, i' \in [n + 1]$ и $j \in [n]$, где $i \neq i'$ верно $x_{ij} \cdot x_{i'j} = 0$. То есть никакие два голубя не сидят в одной клетке.

Пусть один игрок загадал расстановку голубей, а второй хочет найти дизъюнкт, для которого нарушается эта расстановка. Игра с монетками позволяет показать, что предыдущий алгоритм плохо работает на этой модели.

Chapter 2

Информация по Шеннону

2.1 Определения и свойства

На прошлой лекции поняли, что не всегда можем отличить некоторые множества.

Попробуем исправить данную ситуацию. Хотим понять состояния в Y , зная информацию об X . В среднем нам нужно сильно меньше информации, чем в крайнем случае.

Введем новую меру информации $\mu(\alpha)$, где α — распределение (множество и вероятности каждого элемента). Причем хотим, чтобы основные свойства были согласованы: ¹

1. $\mu(U_n) = \log n$;
2. $\mu(\alpha) \geq 0$;
3. $\mu(\alpha, \beta) = \mu(\alpha) + \mu(\beta)$, если α и β независимы.²

Если действовать как настоящие математики, можно переписать эти свойства в более общие:

1. *монотонность*: $\mu(U_M) \geq \mu(U_{M'})$, если $|M| \geq |M'|$;
2. *аддитивность*: $\mu(\alpha, \beta) = \mu(\alpha) + \mu(\beta)$, если α и β независимы;
3. *непрерывность*: $\mu(B_p)$ непрерывно по $p \in [0, 1]$, где B_p — распределение Бернулли для p .
4. *согласованность с условной вероятностью*:

$$\mu(B_p, \alpha) = \mu(B_p) + \Pr[B_p = 0] \cdot \mu(\alpha \mid B_p = 0) + \Pr[B_p = 1] \cdot \mu(\alpha \mid B_p = 1).$$

Этим аксиомам удовлетворяет примерно одна функция $\mu(X) := \sum p_i \log \frac{1}{p_i}$ с точностью до домножения на константу.

2.1.1 Энтропия

Определение 80: Энтропия

Для случайной величины α с вероятностями событий (p_1, p_2, \dots, p_n) меру

$$H(\alpha) = \sum_{i=1}^{|\text{supp}(\alpha)|} p_i \log \frac{1}{p_i}$$

будем называть **энтропией** и обозначать H .^a

¹ $\mu(x, y) = \mu((x, y))$

² (α, β) — распределение на парах.

Энтропия обозначает среднее по распределению α необходимое количество информации для записи элемента.

^a $\text{supp } \alpha$ — все возможные события, то есть имеющие ненулевую вероятность

Замечание. Энтропия равномерного распределения U_n равна $\log n$, так как вероятность каждого события $\frac{1}{n}$.

Замечание. Далее $H(p)$ обозначает энтропию для распределения нечестной монетки.

$$H(B_p) = H(p) = p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}.$$

Теорема 2.1.1. $H(\alpha) \leq \log |\text{supp}(\alpha)|$

□ Применим неравенство Йенсена

$$\sum_{i=1}^{|\text{supp}(\alpha)|} p_i \log \frac{1}{p_i} \leq \log \left(\sum_i p_i \frac{1}{p_i} \right) = \log |\text{supp}(\alpha)|$$

■

Теорема 2.1.2. $H(\alpha, \beta) \leq H(\alpha) + H(\beta)$

□

$$\begin{aligned} H(\alpha, \beta) &= \sum_{i,j} p_{i,j} \log \frac{1}{p_{i,j}} \\ H(\alpha) + H(\beta) &= \sum_i p_i \log \frac{1}{p_i} + \sum_j p_j \log \frac{1}{p_j} \end{aligned}$$

Заметим, что $p_i = \sum_j p_{i,j}$ и $p_j = \sum_i p_{i,j}$.

$$H(\alpha, \beta) - H(\alpha) - H(\beta) = \sum_{i,j} p_{i,j} \log \frac{1}{p_{i,j}} - \sum_i p_i \log \frac{1}{p_i} - \sum_j p_j \log \frac{1}{p_j} = \sum_{i,j} p_{i,j} \log \frac{p_i p_j}{p_{i,j}}.$$

Если α и β независимы, то все логарифмы обнуляются. Иначе по неравенству Йенсена

$$\sum_{i,j} p_{i,j} \log \frac{p_i p_j}{p_{i,j}} \leq \log \left(\sum_{i,j} p_i p_j \right) = 0.$$

■

2.1.2 Условная энтропия

Определение 81: Условная энтропия

Энтропией α при $\beta = b$ будем называть энтропию распределения α при условии, что $\beta = b$, то есть:

$$H(\alpha \mid \beta = b) := \sum_i \Pr[\alpha = i \mid \beta = b] \cdot \log \frac{1}{\Pr[\alpha = i \mid \beta = b]}.$$

Тогда **энтропией α при условии β** назовем среднее значение по β энтропии α при $\beta = b$:

$$H(\alpha \mid \beta) := \mathbb{E}_{\beta \rightarrow b} H(\alpha \mid \beta = b) = \sum_b H(\alpha \mid \beta = b) \cdot \Pr[\beta = b].$$

Свойства.

1. $H(\alpha \mid \beta) \geq 0$

2. $H(\alpha \mid \beta) = 0 \iff \alpha$ однозначно определяется β

□

$$H(\alpha \mid \beta) = 0 \iff \forall b H(\alpha \mid \beta = b) = 0 \iff \forall b : \Pr[\alpha = i \mid \beta = b] = 1$$

3. $\forall f: H(\alpha \mid \beta) \geq H(f(\alpha) \mid \beta)$

□

Заметим, что достаточно показать неравенство $H(\alpha \mid \beta = b) \geq H(f(\alpha) \mid \beta = b)$, поэтому будем считать, что все дальнейшие рассуждения проводятся при условии $\beta = b$.

Пусть α принимает значения a_1, \dots, a_n с вероятностями p_1, \dots, p_n соответственно, а $f(\alpha)$ принимает значения f_1, \dots, f_k так, что для $\forall j f_i := f(a_{i_j})$ (разбили a_i -ые на k частей). Обозначим $P_i := \sum_j p_{i_j}$ - вероятность f_i .

$$\begin{aligned} H(\alpha \mid \beta = b) &= \sum_i \Pr[\alpha = i \mid \beta = b] \cdot \log \frac{1}{\Pr[\alpha = i \mid \beta = b]} && \text{(по определению } H) \\ &= \sum_i p_i \cdot \log \frac{1}{p_i} && \text{(по определению } p_i) \\ &= \sum_i \left(\sum_j p_{i_j} \cdot \log \frac{1}{p_{i_j}} \right) && \text{(перегруппировали слагаемые)} \\ &\geq \sum_i \left(\log \frac{1}{P_i} \cdot \sum_j p_{i_j} \right) && \text{(оценили } P_i \geq p_{i_j}) \\ &= \sum_i P_i \cdot \log \frac{1}{P_i} && \text{(свернули сумму в } P_i) \\ &= H(f(\alpha) \mid \beta = b) \end{aligned}$$

4. $H(\alpha, \beta) = H(\alpha) + H(\beta \mid \alpha) = H(\beta) + H(\alpha \mid \beta)$

□

$$\begin{aligned} H(\beta) + H(\alpha \mid \beta) &= \sum_j p_j \log \frac{1}{p_j} + \sum_j H(\alpha \mid \beta = j) \cdot \Pr[\beta = j] = \\ &= \sum_j p_j \log \frac{1}{p_j} + \sum_j \left(\sum_i \Pr[\alpha = i \mid \beta = j] \cdot \log \frac{1}{\Pr[\alpha = i \mid \beta = j]} \right) \cdot \Pr[\beta = j] = \\ &= \sum_j p_j \log \frac{1}{p_j} + \sum_j \left(\sum_i \Pr[\alpha = i, \beta = j] \cdot \log \frac{\Pr[\beta = j]}{\Pr[\alpha = i, \beta = j]} \right) = \\ &= \sum_j \sum_i p_{i,j} \log \frac{1}{p_j} + \sum_j \left(\sum_i p_{i,j} \cdot \log \frac{p_j}{p_{i,j}} \right) = \sum_{i,j} p_{i,j} \log \frac{1}{p_{i,j}} = H(\alpha, \beta) \end{aligned}$$

подсказка: $\Pr[A, B] = \Pr[A \mid B] \cdot \Pr[B]$

5. $H(\alpha, \beta) \geq H(\alpha)$

□

Очевидно из предыдущего свойства.

6. $H(\alpha) \geq H(\alpha | \beta)$



$$\begin{aligned}
 H(\alpha | \beta) - H(\alpha) &= \sum_j \sum_i \left(\Pr[\alpha = i | \beta = j] \log \frac{1}{\Pr[\alpha = i | \beta = j]} \right) \cdot \Pr[\beta = j] - \sum_i p_i \log \frac{1}{p_i} \\
 &= \sum_{i,j} p_{i,j} \log \frac{p_j}{p_{i,j}} - \sum_{i,j} p_{i,j} \log \frac{1}{p_i} \\
 &= \sum_{i,j} p_{i,j} \log \frac{p_i p_j}{p_{i,j}} \\
 &\leq \log \sum p_i p_j = \log 1 = 0
 \end{aligned}$$

(по неравенству Йенсена)



7. Формула условной энтропии через отдельные вероятности



$$\begin{aligned}
 H(\alpha | \beta) &= \sum_j H(\alpha | \beta = b_j) \cdot \Pr[\beta = b_j] \\
 &= \sum_j \left(\Pr[\beta = b_j] \cdot \sum_i \Pr[\alpha = a_i | \beta = b_j] \cdot \log \frac{1}{\Pr[\alpha = a_i | \beta = b_j]} \right) \\
 &= \sum_{i,j} p_{i,j} \cdot \log \frac{p_j}{p_{i,j}}
 \end{aligned}$$



8. $H(\alpha | \beta) \geq H(\alpha | \beta, \gamma)$, причем равенство достигается, если $(\gamma | \beta)$ не зависимо с $(\alpha | \beta)$.



Обозначим $p_{ijk} := \Pr[\alpha = i, \beta = j, \gamma = k]$, аналогично p_{ij}, p_{jk} .
Распишем левую и правую части по определению:

$$\begin{aligned}
 H(\alpha | \beta) &= \sum_{ij} p_{ij} \log \frac{p_j}{p_{ij}} \\
 H(\alpha | \beta, \gamma) &= \sum_{ij} \left(\sum_k p_{ijk} \log \frac{p_{jk}}{p_{ijk}} \right)
 \end{aligned}$$

Достаточно показать неравенство для фиксированных i, j :

$$\begin{aligned}
 \sum_k p_{ijk} \log \frac{p_{jk}}{p_{ijk}} &= p_{ij} \sum_k \frac{p_{ijk}}{p_{ij}} \log \frac{p_{jk}}{p_{ijk}} && \text{(чтобы сумма коэффициентов была 1)} \\
 &\leq p_{ij} \log \left(\sum_k \frac{p_{jk}}{p_{ij}} \right) && \text{(по неравенству Йенсена)} \\
 &= p_{ij} \log \left(\frac{\sum_k p_{jk}}{p_{ij}} \right) \\
 &= p_{ij} \log \frac{p_j}{p_{ij}}
 \end{aligned}$$

Для независимых $(\alpha | \beta)$ и $(\gamma | \beta)$, то есть:

$$\begin{aligned}
 \Pr[\alpha, \gamma | \beta] &= \Pr[\alpha | \beta] \cdot \Pr[\gamma | \beta] \implies \\
 \frac{p_{ijk}}{p_j} &= \frac{p_{ij}}{p_j} \cdot \frac{p_{jk}}{p_j} \implies \frac{p_{ijk}}{p_{jk}} = \frac{p_{ij}}{p_j} \implies \\
 \Pr[\alpha = i | \beta = j, \gamma = k] &= \Pr[\alpha = i | \beta = j]
 \end{aligned}$$

Теперь можем расписать энтропию:

$$\begin{aligned}
 H(\alpha \mid \beta, \gamma) &= \sum_{jk} p_{jk} H(\alpha \mid \beta = j, \gamma = k) \\
 &= \sum_{jk} p_{jk} \left(\sum_i \Pr[\alpha = i \mid \beta = j, \gamma = k] \log \frac{1}{\Pr[\alpha = i \mid \beta = j, \gamma = k]} \right) \\
 &= \sum_{jk} p_{jk} \left(\sum_i \Pr[\alpha = i \mid \beta = j] \log \frac{1}{\Pr[\alpha = i \mid \beta = j]} \right) \quad (\text{по независимости}) \\
 &= \sum_j \left(\sum_k p_{jk} \right) \cdot \left(\sum_i \Pr[\alpha = i \mid \beta = j] \log \frac{1}{\Pr[\alpha = i \mid \beta = j]} \right) \\
 &= \sum_j p_j H(\alpha \mid \beta = j) \\
 &= H(\alpha \mid \beta)
 \end{aligned}$$

$$9. 2H(\alpha, \beta, \gamma) \leq H(\alpha, \beta) + H(\alpha, \gamma) + H(\beta, \gamma).$$

□

Заметим, что $H(\alpha, \beta, \gamma) = H(\alpha) + H(\beta, \gamma \mid \alpha) = H(\alpha) + H(\beta \mid \alpha) + H(\gamma \mid \alpha, \beta)$.

Теперь распишем аналогичное для правой части и воспользуемся $H(\alpha) \geq H(\alpha \mid \beta)$:

$$\begin{aligned}
 H(\alpha, \beta) &= H(\alpha) + H(\beta \mid \alpha) \\
 H(\alpha, \gamma) &= H(\alpha) + H(\gamma \mid \alpha) \geq H(\alpha) + H(\gamma \mid \alpha, \beta) \\
 H(\beta, \gamma) &= H(\beta) + H(\gamma \mid \beta) \geq H(\beta \mid \alpha) + H(\gamma \mid \alpha, \beta)
 \end{aligned}$$

Теперь, если сложить все три формулы, то получим то, что надо.

2.2 Взаимная информация

Определение 82

Взаимная информация между случайными величинами α и β :

$$I(\alpha : \beta) := H(\alpha) - H(\alpha \mid \beta).$$

Взаимная информация в α и β при условии γ :

$$I(\alpha : \beta \mid \gamma) := H(\alpha \mid \gamma) - H(\alpha \mid \beta, \gamma).$$

Свойства.

$$1. I(\alpha : \beta) = I(\beta : \alpha).$$

$$\square I(\alpha : \beta) = H(\alpha) - H(\alpha \mid \beta) = H(\alpha) - H(\alpha, \beta) + H(\beta) = H(\beta) - H(\beta \mid \alpha) = I(\beta : \alpha)$$

$$2. \alpha \text{ и } \beta \text{ независимы тогда и только тогда, когда } I(\alpha : \beta) = 0.$$

$$\square I(\alpha : \beta) = 0 \iff H(\alpha) = H(\alpha \mid \beta) \iff \text{независимы } \alpha \text{ и } \beta.$$

$$3. I(f(\alpha) : \beta) \leq I(\alpha : \beta) \text{ для любой функции } f.$$

□ Сначала распишем левую часть:

$$\begin{aligned}
 I(\alpha : \beta) &= H(\alpha) - H(\alpha | \beta) \\
 &= \sum_{i,j} p_{i,j} \log \frac{1}{p_i} - \sum_{i,j} p_{i,j} \log \frac{p_j}{p_{i,j}} && \text{(расписали энтропию через вероятности)} \\
 &= \sum_{i,j} p_{i,j} \log \frac{p_{i,j}}{p_i p_j} \\
 &= \sum_{k,j} \left(\sum_{i \in f^{-1}(k)} p_{i,j} \log \frac{p_{i,j}}{p_i p_j} \right) && \text{(сгруппировали с помощью прообразов)}
 \end{aligned}$$

Аналогично правую часть:

$$\begin{aligned}
 I(f(\alpha) : \beta) &= \sum_{k,j} \Pr[f(\alpha) = k, \beta = j] \log \frac{\Pr[f(\alpha) = k, \beta = j]}{\Pr[f(\alpha) = k] \Pr[\beta = j]} \\
 &= \sum_{k,j} \left(\sum_{i \in f^{-1}(k)} p_{i,j} \right) \log \frac{\left(\sum_{i \in f^{-1}(k)} p_{i,j} \right)}{\left(\sum_{i \in f^{-1}(k)} p_i \right) p_j} \\
 &= \sum_{k,j} P_{kj} \log \frac{P_{kj}}{P_k p_j}
 \end{aligned}$$

Где $P_k := \Pr[f(\alpha) = k] = \sum_{i \in f^{-1}(k)} p_i$, аналогично $P_{kj} := \Pr[f(\alpha) = k, \beta = j]$.

Теперь для каждой пары k, j покажем неравенство отдельно

$$\begin{aligned}
 - \sum_{i \in f^{-1}(k)} p_{i,j} \log \frac{p_{i,j}}{p_i p_j} &= \sum_{i \in f^{-1}(k)} p_{i,j} \log \frac{p_i p_j}{p_{i,j}} \\
 &= P_{kj} \sum_{i \in f^{-1}(k)} \frac{p_{i,j}}{P_{kj}} \log \frac{p_i p_j}{p_{i,j}} && \text{(поделили и домножили, чтобы сумма коэффициентов была 1)} \\
 &\leq P_{kj} \log \left(\sum_{i \in f^{-1}(k)} \frac{p_i p_j}{P_{kj}} \right) && \text{(по неравенству Йенсена)} \\
 &= P_{kj} \log \frac{P_k p_j}{P_{kj}} = -P_{kj} \log \frac{P_{kj}}{P_k p_j}
 \end{aligned}$$



4. Следующие формулы для взаимной информации эквивалентны

$$\begin{aligned}
 I(\alpha : \beta) &= H(\alpha) - H(\alpha | \beta) \\
 &= H(\beta) - H(\beta | \alpha) \\
 &= H(\alpha) + H(\beta) - H(\alpha, \beta) \\
 &= H(\alpha, \beta) - H(\alpha | \beta) - H(\beta | \alpha)
 \end{aligned}$$

2.3 Применение энтропии

2.3.1 Взвешивания монеток

Попробуем решить задачу с монетками. Мы взвешиваем 14 монеток и хотим найти фальшивую за три взвешивания, причем неизвестен относительный вес. Всего будет $14 \cdot 2$ исходов (одна из 14 монет фальшивая и относительный вес).

Предположим, что есть стратегия, позволяющая решить задачу.

Построим граф, соответствующий взвешиваниям: каждое имеет три исхода, всего три взвешивания. Получается дерево с 27 листьями, где во всех листьях, кроме одного, записана пара: номер фальшивой монетки и ее относительный вес. В единственной ветке, где все взвешивания давали равенство, будет записан только номер фальшивой монетки (теперь количество листьев совпадает с нужным количеством исходов³). Построим равномерное распределение на этих исходах.

При равномерном распределении энтропия $\log 27$.

Если стратегия верная, то

$$\begin{aligned} \log 27 = H(\alpha) &\leq H(\alpha, q_1, q_2, q_3) = \\ &= H(q_1) + H(q_2 | q_1) + H(q_3 | q_1, q_2) + H(\alpha | q_1, q_2, q_3) \leq \\ &\leq H(q_1) + H(q_2) + H(q_3) + 0 \end{aligned} \quad (\text{Chain rule})$$

Так как $H(q_i) \leq \log 3$, для всех i должно быть $H(q_i) = \log 3$.

Чтобы было так, мы должны в каждый ход равновероятно получать все три ответа.

Пусть мы первым ходом взвешиваем кучки из k монет⁴. Вероятность того, что левая кучка будет легче в результате взвешивания $\frac{2k}{27}$ (так как у нас либо в левой кучке фальшивая, и она легче, либо в правой кучке фальшивая, и она тяжелее). Тогда $\frac{2k}{27} = \frac{1}{3}$. k получилось нецелым. Противоречие.

2.3.2 Оценка на биномиальные коэффициенты

$$C = \sum_{i=0}^k \binom{n}{i} \leq 2^{nH(\frac{k}{n})}.$$

Будем выбирать множество размера не больше k равновероятно, а затем проверять, попало ли i в наше множество. Пусть X_i — индикатор того, что i выбрали.

$$\begin{aligned} \log C = H(X) &= H(X_1, \dots, X_n) = \\ &= \sum H(X_i | X_{<i}) \leq \\ &\leq \sum H(X_i) = nH(X_1) \leq \\ &\leq nH\left(\frac{k}{n}\right) \end{aligned} \quad (\text{Chain rule})$$

(считаем, что $k \leq \frac{n}{2}$)

Пояснение к последнему переходу. $H(\frac{k}{n}) = \frac{k}{n} \log \frac{n}{k} + (1 - \frac{k}{n}) \log \frac{1}{1 - \frac{k}{n}}$. С другой стороны $H(X_1) = p \log \frac{1}{p} + (1 - p) \log \frac{1}{1 - p}$, где p — вероятность того, что первый элемент попал в множество. Заметим, что функция $f(x) = x \log \frac{1}{x} + (1 - x) \log \frac{1}{1 - x}$ возрастает на $(0, \frac{1}{2})$. То есть нужно лишь показать, что $p \leq \frac{k}{n}$. Для этого посчитаем матожидание числа элементов, которые вошли в выбранное множество.

$$k \geq \mathbb{E}(X) = \mathbb{E}\left(\sum X_i\right) = \sum \mathbb{E}(X_i) = \sum p_i$$

Но каждый элемент войдет в множество равновероятно, тогда $p \leq \frac{k}{n}$.

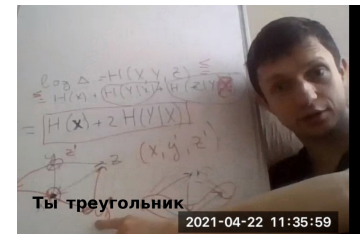
³Далее будем считать, что исходов 27

⁴Очевидно, что если взвешивать кучки разного размера, информацию извлечь не получится даже по Хартли, так как мы не знаем ничего про относительный вес

2.3.3 Подсчет углов в графе

Рассмотрим **ориентированный** граф без кратных ребер и петель.
Назовем *треугольником* упорядоченную тройку (x, y, z) , если это цикл из трех вершин. *Углом* назовем тройку (x, y, z) , если есть ребра xu и xz , при этом y может совпадать z .

Чего в графе больше: углов или треугольников?



Замечание. Каждое ребро тоже угол, например, (x, y, y) .

Замечание. Треугольником мы называем именно упорядоченную тройку вершин, которые соединены ребрами между собой. Поэтому просто треугольник даёт вклад равный 3 «упорядоченным» треугольникам.

Теорема 2.3.1. Число углов в графе всегда больше или равно числу треугольников.

□ Пусть случайная величина $\alpha = (X, Y, Z)$ равна случайному треугольнику, где X, Y, Z — с.в., соответствующие первой, второй и третьей вершине треугольника в равномерном распределении на треугольниках.

Так как распределение количества треугольников равномерно,

$$\begin{aligned} \log(\#\Delta) &= H(X, Y, Z) = \\ &= H(X) + H(Y | X) + H(Z | Y, X) \leq && \text{(Chain rule)} \\ &\leq H(X) + H(Y | X) + H(Z | Y) = && \text{(циклический сдвиг в треугольнике)} \\ &= H(X) + 2H(Y | X) \end{aligned}$$

Найдем какое-то распределение на углах, энтропия которого хотя бы $H(X) + 2H(Y | X)$, тогда эта сумма будет не более $\log(\#\Delta)$.

Пусть мы выбрали случайный треугольник (x, y, z) . Оставим x и выберем для него случайный треугольник с x и возьмем из него следующую за x вершину y' . Повторяем эту операцию еще раз для x и находим z' . Тогда (x, y', z') — угол.

$$\begin{aligned} H(x, y', z') &= H(x) + H(y' | x) + H(z' | x, y') = && \text{(Так как } y' \text{ и } z' \text{ независимы при выбранном } x) \\ &= H(x) + H(y' | x) + H(z' | x) = && \text{(Выбор аналогичный)} \\ &= H(x) + 2H(y' | x) \end{aligned}$$

$H(x)$ здесь совпадает с $H(x)$ выше, так как мы выбираем треугольник и вершину аналогично.

y' выбирается при фиксированном x также, как и выше (выбрали случайный треугольник и в нем вершиной после x будет y').

Таким образом, мы нашли какое-то распределение на углах с такой же энтропией, следовательно, углов не меньше, чем треугольников. ■

Chapter 3

Теория кодирования

Определение 83: Код

Будем называть **кодом** функцию $C: \Sigma = \{a_1, \dots, a_n\} \rightarrow \{0, 1\}^*$, сопоставляющую буквам некоторого алфавита кодовые слова.

Если любое сообщение, полученное применением кода C , однозначно декодируется, то такой код будем называть **однозначно декодируемым**.

3.1 Префиксные коды

Определение 84

Код называется **префиксным**, если никакое кодовое слово не является префиксом другого кодового слова.

Замечание. Очевидно, из этого следует однозначная декодируемость.

Теорема 3.1.1. Пусть набор целых чисел l_1, \dots, l_n удовлетворяет неравенству

$$\sum_{i=1}^n 2^{-l_i} \leq 1.$$

Тогда существует префиксный код с кодовыми словами c_1, \dots, c_n , где $|c_i| \leq l_i$.

□ Индукция по n .

- База: $n = 1$. Очевидно.
- Переход: $n \rightarrow n + 1$. Пусть нам даны числа l_1, \dots, l_{n+1} .
 - Если среди них есть два одинаковых, $l_1 = l_2$, заменим их на одно $l = l_1 - 1$, чтобы сохранилась сумма обратных степеней двойки. Теперь можем применить предположение для n и найти префиксный код со словами c, c_3, \dots, c_{n+1} . Заменим слово c на $c0$ и $c1$, которые будут соответствовать l_1 и l_2 .
 - Если все различны, то можем выбрать слова длины l_1, l_2, \dots, l_n из слов вида $0, 10, 110, \dots$. Этот код будет префиксным, так как у всех разное число единиц перед нулем.



Теорема 3.1.2 (Неравенство Крафта-Макмиллана). Для любого однозначно декодируемого кода с n кодовыми словами c_1, \dots, c_n выполнено неравенство

$$\sum_{i=1}^n 2^{-|c_i|} \leq 1.$$

□ Построим многочлен для всех слов длины L .

Для этого воспользуемся следующей идеей.

Строка длины L переходит в конкатенацию кодовых слов для каждого символа исходной строки. Давайте в кодовых словах заменим 0 на x , а 1 на y .

Тогда всем кодовым словам соответствуют некоторые мономы $p_i(x, y)$ (например, $c_i = 010$ соответствует моному xux).

Считаем, что x и y не коммутируют, чтобы удобнее было различать слова xux , соответствующие 010 и xxu , соответствующие 001.

Будем говорить, что слову c_i соответствует $p_i(x, y)$. Тогда строка длины L есть произведение L таких мономов. Тогда коды всех слов длины L можно представить в виде многочлена $P(x, y)$, в котором операция сложения будет разделять разные коды:

$$P(x, y) = \left(\sum_i p_i(x, y) \right)^L = \sum_{j=L}^{L \cdot \max |c_i|} M_j(x, y).$$

В $M_j(x, y)$ сгруппированы в сумму все мономы степени j , получающиеся при раскрытии скобок (то есть все различные слова длины j , получаемые из кодирования слов длины L).

Так как код однозначно декодируемый, каждое слово является образом не более чем одной исходной строки. Следовательно, в $M_j(x, y)$ не более 2^j мономов.

Посчитаем $P(\frac{1}{2}, \frac{1}{2})$:

$$P(\frac{1}{2}, \frac{1}{2}) = \sum_{j=L}^{L \cdot \max |c_i|} M_j(\frac{1}{2}, \frac{1}{2}) \leq \sum_{j=L}^{L \cdot \max |c_i|} 2^j \cdot 2^{-j} = \mathcal{O}(L).$$

Посчитаем еще раз по второму представлению

$$P(\frac{1}{2}, \frac{1}{2}) = \left(\sum_i 2^{-|c_i|} \right)^L.$$

Если сумма в скобках больше 1, получаем экспоненциальную оценку снизу.

Следовательно, для больших L она обгонит линейную. Противоречие. ■

Теорема 3.1.3. Любой однозначно декодируемый код можно переделать в префиксный с сохранением длин кодовых слов.

□ Пусть c_1, \dots, c_n — кодовые слова.

По теореме 3.1.2 (неравенство Крафта-Макмиллана) для исходного декодируемого кода выполнено неравенство $\sum 2^{-|c_i|} \leq 1$.

Тогда по еще одной теореме 3.1.1 существует префиксный код с кодовыми словами такой же длины (если какие-то слова стали меньше, можно просто дополнить чем-то в конце, на префиксы это не повлияет). ■

Теорема 3.1.4 (Шеннон). Для любого распределения p^a и однозначно декодируемого кода выполнено неравенство:

$$\sum_i p_i |c_i| \geq H(p).$$

^aТо есть каждому символу из конечного алфавита сопоставлена его встречаемость

□

$$\begin{aligned} H(p) - \sum_i p_i |c_i| &= \sum_i p_i \log \frac{2^{-|c_i|}}{p_i} \\ &\leq \log \sum_i p_i \cdot \frac{2^{-|c_i|}}{p_i} && \text{(Неравенство Йенсена)} \\ &\leq 0 && \text{(Неравенство Крафта-Макмиллана)} \end{aligned}$$

■

Теорема 3.1.5 (Шеннон). Для любого распределения p существует такой префиксный код, что^a

$$\sum_i p_i |c_i| \leq H(p) + 1.$$

^aЕдиница обязательно возникает, так как мы приводим непрерывную энтропию к дискретной величине

□ Угадаем длины кодов, чтобы выполнялось неравенство Крафта-Макмиллана. Пусть $|c_i| = \lceil \log \frac{1}{p_i} \rceil$, тогда неравенство из условия выполнено, так как $p_i |c_i| \leq p_i (\log \frac{1}{p_i} + 1) = p_i \log \frac{1}{p_i} + p_i$ и $\sum p_i = 1$. Кроме этого:

$$\sum_i 2^{-|c_i|} = \sum_i 2^{-\lceil \log \frac{1}{p_i} \rceil} \leq \sum_i 2^{-\log \frac{1}{p_i}} = \sum_i p_i = 1.$$

Теперь по [теореме 3.1.1](#) такой код действительно существует и удовлетворяет условию теоремы. ■

3.2 Примеры эффективных кодов

3.2.1 Код Шэннона-Фано

Отсортируем вероятности по убыванию $p_1 \geq p_2 \geq \dots \geq p_n$. Затем «уложим» их в отрезок $[0, 1]$, получая при этом такие точки:

$$0 \leq p_1 < p_1 + p_2 < \dots < p_1 + p_2 + \dots + p_n \leq 1.$$

Разделим отрезок $[0, 1]$ пополам и скажем, что слева кодовые слова начинаются с 0, справа с 1, а центральный p_i будет начинаться с нуля, если это p_1 , с единицы (никогда не выполняется), если p_n , и, наконец, иначе выбираем любое значение.

Далее рекурсивно продолжаем процесс на группе нулей и на группе единиц.

Когда остался один кусок, останавливаемся.

Теорема 3.2.1.

$$\sum_{i=0}^n p_i \cdot |c_i| \leq H(p) + \mathcal{O}(1), \quad n \rightarrow \infty, \quad \mathcal{O}(1) \approx 3 \text{ или } 5.$$

Без доказательства. Дано как «упражнение со звездочкой».

3.2.2 Код Хаффмана

Опять отсортируем по убыванию $p_1 \geq p_2 \geq \dots \geq p_n$. Возьмем p_{n-1} и p_n . Заменяем их на один символ с вероятностью $p_{n-1} + p_n$, теперь продолжаем по индукции строить код для $n - 1$ символа.

Если объединенному символу соответствовал код \bar{c} , то для p_{n-1} задаем код $\bar{c}0$, а для p_n код $\bar{c}1$

Теорема 3.2.2. Для кода Хаффмана выполнено неравенство:

$$\sum_{i=1}^n p_i |c_i| \leq H(p) + 1,$$

причем для любого другого однозначно декодируемого кода c'_i верно:

$$\sum_{i=1}^n p_i |c_i| \leq \sum_{i=1}^n p_i |c'_i|.$$

□ Достаточно доказать второе, а потом сравнить с кодом, который нам дает [теорема Шеннона 3.1.5](#), и получить нужное неравенство.

Рассмотрим набор c'_1, \dots, c'_n . Будем считать, что этот код префиксный, так как мы научились любой декодируемый код переделывать в префиксный с той же длиной кодовых слов.

- **Шаг 1.** Возьмем два наименее вероятных кодовых слова c'_{n-1} и c'_n .

Заметим, что можно поменять их с символами максимальной длины c'_i и c'_j , при этом средняя длина кода не увеличится.

Значит, перестроили так, что код не ухудшился и c'_n, c'_{n-1} соответствуют символам с самой маленькой вероятностью и самой большой длиной кодовых слов.

- **Шаг 2.** Изучим коды c'_{n-1} и c'_n . Пусть они не имеют вид $\bar{v}0$ и $\bar{v}1$. И пусть $|c'_{n-1}| \leq |c'_n|$.

Посмотрим на c'_{n-1} : не умаляя общности он будет заканчиваться на 0 ($c'_{n-1} = \bar{s}0$).

Заменяем c'_n на $\bar{s}1$. Что при этом могло сломаться?

Так как наш код префиксный, нам нужно проверить, что он префиксным и остался.

Заметим, что так как $c'_{n-1} = \bar{s}0$, префиксов s нет среди кодовых слов.

Единственная проблема тогда: среди кодовых слов может быть само $\bar{s}1$.

Тогда поступим следующим образом. Заменяем c'_n на слово длины $|s| + 1$, а затем поменяем его местами с кодовым словом, равным $\bar{s}1$.

Почему мы можем найти новое слово подходящей длины?

Например, обрежем c'_n до длины $|s| + 1$ и обозначим результат как \tilde{c} . Получили то, что требовалось:

- У \tilde{c} нет префиксов среди остальных кодовых слов, так как их не было у c'_n .
- \tilde{c} само не является префиксом другого слова. Пусть не так и $\exists d$ такое, что \tilde{c} его префикс. Тогда возможны 2 варианта:
 1. $|d| \geq |s| + 2 = \tilde{c} + 1 \geq |c'_{n-1}| + 1$, что невозможно исходя из максимальной длин c'_{n-1} и c'_n ;
 2. $d = \tilde{c}$, что тоже невозможно, так как иначе d - префикс c'_n .

В итоге перестроили так, что средняя длина кода не увеличилась и $c_{n-1} = \bar{v}0$, а $c_n = \bar{v}1$.

- **Шаг 3.** Заменяем c_{n-1} и c_n на один символ с кодовым словом v . И применим предположение, что код Хаффмана оптимален для алфавита из $n - 1$ символа.

Тогда, заменив v обратно на c_{n-1} и c_n получим, что код Хаффмана оптимален и для n .

3.2.3 Арифметическое кодирование

Уложим вероятности p_i (частоты) аналогично на отрезок, при этом не обязательно в порядке убывания.

Назовем **стандартным** полуинтервал с таким бинарным представлением: $[\overline{0.v_10}, \overline{0.v_11})$.

Найдем максимальный по длине стандартный интервал в отрезке $[p_1 + \dots + p_{i-1}, p_1 + \dots + p_i]$. Пусть это $(0.v_i0, 0.v_i1)$.

Сопоставим i -ой букве код v_i0 .

Заметим, что такой код будет префиксным, так как отрезки не пересекаются, а, чтобы v_i было префиксом v_j , интервал $(0.v_i0, 0.v_i1)$ должен быть вложен в $(0.v_j0, 0.v_j1)$.

Лемма 21. В отрезке $[a, b]$ длина наибольшего стандартного интервала не меньше $\frac{b-a}{8}$.

□ Пусть $2^{-k-1} < \frac{b-a}{8} < 2^{-k}$. Рассмотрим все стандартные интервалы длины 2^{-k} . Заметим, что соседние интервалы находятся на расстоянии 2^{-k} (от конца левого до начала правого).

Предположим, что ни один стандартный интервал длины 2^{-k} не попал полностью в $[a, b]$. Тогда длина $[a, b]$ меньше суммы длин двух стандартных и расстояния между ними, то есть

$$2^{-k} \cdot 2 + 2^{-k} < 2^{-k+2} \implies 2^{-k+2} > b - a \implies 2^{-k-1} > \frac{b-a}{8}$$

Противоречие. Следовательно, какой-то отрезок попал внутрь $[a, b]$. ■

Теорема 3.2.3. Для арифметического кода выполнено неравенство

$$\sum_i p_i |v_i| \leq H(p) + 2.$$

□ Из **леммы 21** следует, что если $|v_i| = k$, то

$$0.v_i1 - 0.v_i0 = 2^{-k-1} \geq \frac{p_i}{8}.$$

Поэтому $k + 1 \leq \log \frac{8}{p_i}$, следовательно, $|v_i| = k \leq \log \frac{1}{p_i} + 2$ и

$$\sum_i p_i |v_i| \leq H(p) + 2(p_1 + \dots + p_n) \leq H(p) + 2.$$

3.3 Кодирование с ошибками

Пусть есть алфавит Σ размером k , «кодер» $E: [k]^n \rightarrow \{0, 1\}^{L_n}$ и «декодер» $D: \{0, 1\}^{L_n} \rightarrow [k]^n$.

Пусть есть распределение на буквах p_1, p_2, \dots, p_k .¹

Откажемся от условия однозначного декодирования, но будем требовать

$$\varepsilon_n \xrightarrow{n \rightarrow \infty} 0^2$$

где $|x| = n, \varepsilon_n := \Pr[D(E(x)) \neq x]$

¹считаем, что слово состоит из независимых букв

²Если сделать равенство нулю, то особого сжатия не будет

Теорема 3.3.1 (Шеннон).

1. Если $L_n > \lceil hn \rceil$, где $h > H(p)$, то кодирование есть, то есть существуют такие функции E и D , что $\varepsilon_n \rightarrow 0$.
2. Если $L_n < \lceil hn \rceil$, где $h < H(p)$, то $\varepsilon_n \rightarrow 1$ для любых E и D .

□ Зафиксируем $\delta = n^{-0.02}$.

Определение 85

Будем называть слово w δ -**типичным**, если

$$\forall i \left| \frac{n_i}{n} - p_i \right| \leq \delta, \quad n_i = \text{\#вхождений буквы } i.$$

- Докажем, что можем закодировать такие типичные слова.
 - Пусть X_{ij} — характеристическая функция того, что в слове на позиции j находится буква i . Для случайной величины $X_i := \sum_j X_{ij}$ применим неравенство Чебышева для вероятности того, что условие типичности не выполняется для конкретной буквы:

$$\Pr[|X_i - np_i| \geq \delta n] \leq \frac{\mathbb{D}[X_i]}{(\delta n)^2} = \frac{np_i(1-p_i)}{(\delta n)^2} = \mathcal{O}\left(\frac{1}{\delta^2 n}\right),$$

так как $\mathbb{E}X_i = np_i$.

Мы расписали вероятность, что какой-то символ нарушает условие δ -типичности.

$$\begin{aligned} \Pr[\text{слово не } \delta\text{-типично}] &= \Pr[\exists i \text{ такой, что на нём нарушается неравенство}] \\ &= \Pr\left[\exists i : \left| \frac{X_i}{n} - p_i \right| > \delta\right] \\ &\leq \sum_i \Pr\left[\left| \frac{X_i}{n} - p_i \right| > \delta\right] \\ &= \mathcal{O}\left(\frac{k}{\delta^2 n}\right) = \mathcal{O}\left(\frac{1}{\delta^2 n}\right) \end{aligned}$$

То есть, раз букв константное количество, вероятность нетипичности все равно останется очень маленькой и будет стремиться к нулю.

- Теперь докажем, что типичных слов не очень много. Число слов, где буквы встречаются в количествах n_1, \dots, n_k равно

$$N_{n_1, \dots, n_k} = \frac{n!}{n_1! \cdot \dots \cdot n_k!}.$$

Обозначим $\delta_i := \frac{n_i}{n} - p_i$.

$$\begin{aligned} \log N_{n_1, \dots, n_k} &= \quad \quad \quad (\text{так как } n! = \mathbf{poly}(n) \left(\frac{n}{e}\right)^n) \\ &= \log \left(\left(\frac{n}{n_1}\right)^{n_1} \cdot \left(\frac{n}{n_2}\right)^{n_2} \cdot \dots \cdot \left(\frac{n}{n_k}\right)^{n_k} \right) + \mathcal{O}(\log n) = \\ &= \sum_i n_i \log \frac{n}{n_i} + \mathcal{O}(\log n) = \\ &= n \sum_i (p_i + \delta_i) \cdot \log \frac{1}{p_i + \delta_i} + \mathcal{O}(\log n) \quad (n_i \text{ по определению}) \end{aligned}$$

- Теперь оценим число типичных слов.

Для этого можно посчитать для каждого распределения букв, сколько слов с таким распределением будут типичными.

Число способов разбить n на k слагаемых грубо оценивается сверху как n^k ³. А число слов в соответствующем разбиении можно оценить максимумом по всем δ_i , таким что $|\delta_i| \leq \delta$ (так как слово δ -типичное).

$$\begin{aligned} \log(\#\delta\text{-типичных слов}) &\leq \log\left(n^k \cdot \max_{\delta_i} N_{n_1, \dots, n_k}\right) \leq \\ &\leq n \cdot \max_{\delta_i} \sum_i (p_i + \delta_i) \cdot \log \frac{1}{p_i + \delta_i} + \mathcal{O}(\log n) \leq \\ &\leq n \cdot \left(\sum_i p_i \log \frac{1}{p_i} + \delta \cdot \log \frac{1}{p_i} \right) + \mathcal{O}(\log n) = \\ &= nH(p) + \mathcal{O}(\delta \cdot n) \end{aligned}$$

Если теперь «кодер» может отобразить инъективно все типичные слова в набор битовых слов длины hn , при этом ошибаться он будет на нетипичных, количество которых стремится к нулю.

$$\frac{\log(\#\delta\text{-типичных слов})}{L_n} = \frac{nH(p) + \mathcal{O}(\delta n)}{L_n} \leq \frac{nH(p) + \mathcal{O}(n^{0.98})}{hn} = \frac{H(p)}{h} + \mathcal{O}(n^{-0.02})$$

Так как $H(p) < h$ и n растет быстрее $\mathcal{O}(\delta n) = \mathcal{O}(n^{0.98})$, то с какого то момента выражение меньше единицы

- Во второй части докажем, что вероятность ошибки декодера на δ -типичных словах равна 1. Понятно, что этого будет достаточно.

Покажем, что вероятность того, что нам на вход придет конкретное δ -типичное слово очень мала.

Пусть $L_n \leq hn$. Посмотрим на любое распределение кодовых слов. Покажем, что вероятность по нашему определению для δ -типичных слов больше, чем $\frac{1}{2^{L_n}}$.

$$\begin{aligned} \Pr[w = x] &= p_1^{n_1} \cdot p_2^{n_2} \cdot \dots \cdot p_k^{n_k} = & (x - \text{конкретное слово}) \\ &= 2^{\log \prod_i p_i^{n_i}} = \\ &= 2^{-\sum_i n_i \log \frac{1}{p_i}} = \\ &= 2^{-n \sum_i (p_i + \delta_i) \log \frac{1}{p_i}} = \\ &= 2^{-H(p)n + \mathcal{O}(\delta n)} \end{aligned}$$

С какой вероятностью декодер ответит правильно?

$$\begin{aligned} \Pr[D(E(w)) = w] &= \sum_x \Pr[D(x) = w, E(w) = x] \\ &\leq \sum_x \Pr[D(x) = w] \leq & (D(x) — \text{какое-то фиксированное слово}) \\ &\leq 2^{L_n} \cdot \max_{t - \text{слово}} \Pr[w = t] \leq \\ &\leq 2^{L_n - H(p)n + \mathcal{O}(\delta n)} \rightarrow 0 \end{aligned}$$

Предельный переход верен, потому что $L_n - H(p)n + \mathcal{O}(\delta n) \leq (h - H(p))n + \mathcal{O}(\delta n) \rightarrow -\infty$, так как $h - H(p) < 0$ константа.



³Выбираем распределение n букв по k возможным значениям.

Chapter 4

Коммуникационная сложность

Лекция 5

29 April

Пусть у нас есть два игрока: Алиса и Боб. Они могут отправлять друг другу сообщения и хотят посчитать функцию (или отношение) $f: X \times Y \rightarrow Z$. Будем говорить, что они *решают коммуникационную задачу* для функции f , если:

1. множества X, Y, Z и функция f известны обоим игрокам,
2. Алиса знает некоторое $x \in X$,
3. Боб знает некоторое $y \in Y$,
4. Алиса и Боб стремятся вычислить $f(x, y)$.

4.1 Детерминированная модель

Определение 86: Коммуникационный протокол

Коммуникационный протокол для функции $f: X \times Y \rightarrow Z$ — корневое двоичное дерево, которое описывает совместное вычисление Алисой и Бобом функции f . В этом дереве:

- каждая внутренняя вершина v помечена меткой a или b , означающей очередь хода Алисы или Боба соответственно;
- для каждой вершины v , помеченной a , определена функция $g_v: X \rightarrow \{0, 1\}$, которая задает бит, который Алиса отправит Бобу, если вычисление будет находиться в v ; аналогично для каждой вершины v с пометкой b определена функция $h_v: Y \rightarrow \{0, 1\}$ для сообщений Боба;
- у каждой внутренней вершины есть два потомка, ребро к первому помечено 0, ко второму — 1;
- каждый лист помечен значением из множества Z .

Каждая пара входов (x, y) определяют путь от корня до листа в этом дереве. Будем говорить, что коммуникационный протокол **вычисляет** функцию f , если для всех пар $(x, y) \in X \times Y$ этот путь заканчивается в листе с пометкой $f(x, y)$.

Коммуникационной сложностью функции f называется наименьшая глубина протокола, вычисляющего f . Обозначается $D(f)$ или $D^{cc}(f)$.

Каждой функции можем сопоставить матрицу $X \times Y$, где в клетке (x_i, y_j) стоит значение $f(x_i, y_j)$.

4.1.1 Нижние оценки для детерминированного случая

Пусть наша функция $f: X \times Y \rightarrow Z$. Запишем для нее коммуникационную матрицу M размера $|X| \times |Y|$, где $M_{x,y} = f(x, y)$.

Рассмотрим такое подмножество R_v множества $X \times Y$, что $(x, y) \in R_v \iff$ протокол приходит в v .

Лемма 22. $R_v = X_v \times Y_v$ — комбинаторный прямоугольник.

Рассмотрим **два** доказательства:

□ Пусть (x, y) и (x', y') принадлежат R_v . Тогда (x, y') и (x', y) тоже принадлежат R_v , так как $a(x) = a(x')$ и $b(y) = b(y')$, где a и b — ответы Алисы и Боба.

А из этого следует, что это комбинаторный прямоугольник. ■

□ Посмотрим на множество $X \times Y$ как на таблицу.

Пусть Алиса перешла по какому-то ребру. Вся таблица разделилась на две части по горизонтали: какие-то строки соответствуют $x \in X$, для которых Алиса отправляет 0, а какие-то для 1.

Если потом ход делает Боб, то он делит все текущие прямоугольники на две части по вертикали.

И так далее.

В прямоугольнике для листа u всех элементов одинаковый ответ. То есть исходную матрицу можно разбить на комбинаторные прямоугольники, причем они естественно не пересекаются. ■

Давайте продолжим смотреть на матрицу. Зафиксируем какой-то $O_1 \in Z$. Тогда есть входы (x, y) такие, что протокол ведет их к O_1 . По предыдущей лемме множество всех таких входных данных образует комбинаторный прямоугольник. Тогда, если $Z = \{O_1, \dots, O_{|Z|}\}$ и для каждого O_i смотрим на соответствующий комбинаторный прямоугольник мы получаем разбиение исходной матрицы.

Пусть $Z = \{0, 1\}$. Рассмотрим величины $\chi_0(f)$ и $\chi_1(f)$, первая равна минимальному числу непересекающихся комбинаторных прямоугольников, которыми можно покрыть все нули в таблице, вторая — все единицы.

Тогда листьев в двоичном дереве протокола будет хотя бы $\chi_0(f) + \chi_1(f)$. Следовательно, $D(f) \geq \log(\chi_0(f) + \chi_1(f))$.

Но эта оценка не всегда точна. Можно рассмотреть следующее разбиение на картинке 4.1.

Здесь $\chi_0(f) + \chi_1(f) = 4 + 1 = 5$. Заметим, что для такого разбиения не существует дерева протокола.

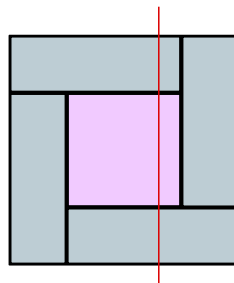


Figure 4.1: Пример неточной оценки

Посмотрим на первое действие игроков. Прямоугольник должен разделиться на две части, но любой разрез вдоль сторон разрежет один из внутренних прямоугольников.

Поэтому любой протокол не будет соответствовать этому разбиению. А тогда листьев будет больше пяти.

Теорема 4.1.1 (G,PW 16). Пусть χ — минимальное число прямоугольников в разбиении. Существует f для которой

$$D(f) \geq \log^{2-\varepsilon} \chi(M_f).$$

Без доказательства.

4.2 Методы оценки коммуникационной сложности

4.2.1 Метод ранга

Пусть M_f — таблица некоторой функции f со значениями $\{0, 1\}$.

Обозначим за x_1, \dots, x_n листья дерева коммуникационного протокола для функции f , а R_{x_1}, \dots, R_{x_n} соответствующие им прямоугольники в M_f . $\text{rk}(M_f)$ — ранг матрицы M_f .

$$\text{rk}(M_f) \leq \sum_{i=1}^n \text{rk} R_{x_i} \leq \sum_{i=1}^n 1 = \# \text{ одноцветных прямоугольников в разбиении.}$$

Тогда коммуникационная сложность не меньше $\log \text{rk}(M_f)$.

Этот метод называется **методом ранга**.

Лемма 23. Для любой функции $f: \{0, 1\}^n \times \{0, 1\}^m \rightarrow \{0, 1\}^k$ коммуникационная сложность не больше $\min\{n + k, n + m, m + k\}$.

□ Пусть Алиса и Боб хотят вычислить значение $f(x, y)$, причем x знает только Алиса, а y — только Боб.

- Для $n + m$ они просто отправляют друг другу свои числа и локально вычисляют f .
- Для $n + k$ и $m + k$ достаточно просто отправить свое число второму, он посчитает значение функции и отправит обратно.



Оценка для EQ Определим функцию $\text{EQ}(x, y) = (x = y): \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$.

Для функции EQ матрица M_{EQ} будет диагональной $2^n \times 2^n$.

Поэтому ее ранг равен 2^n , а тогда коммуникационная сложность $D(\text{EQ}) \geq \log 2^n + 1 = n + 1$.¹

С другой стороны, она не больше $n + 1$, так как за $n + 1$ вопрос можно просто сравнить все биты (n) и отослать результат обратно (1).

Оценка для GT Определим функцию $\text{GT}(x, y) = (x \geq y): \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$.

Докажем, что ее коммуникационная сложность $D(\text{GT}) = n + 1$.

Матрица будет треугольной, поэтому $\text{rk} M_{\text{GT}} = 2^n$. А тогда $D(\text{GT}) \geq \log 2^n + 1 = n + 1$.

По [лемме 23](#) $D(\text{GT}) \leq n + 1$, поэтому она в точности $n + 1$.

4.2.2 Fooling Set

Рассмотрим коммуникационную матрицу некоторой функции f . Пусть мы хотим выбрать некоторое множество клеток

$$S = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\},$$

такое что каждая пара точек не лежит в одном прямоугольнике.

Чтобы все были в разных прямоугольниках, для всех $i \neq j \in [n]$ либо клетка (x_i, y_j) , либо (x_j, y_i) была другого цвета².

Из условия на разные прямоугольники следует, что в дереве не менее n листьев, а поэтому высота хотя бы $\log n$.

¹Дополнительная единица возникает из-за того, что нам нужен еще хотя бы один лист для нуля.

²Цвет = 0 или 1

Оценка для EQ Применим этот метод к EQ. Берем в качестве точек главную диагональ из единиц. Никакие из них не находятся в одном одноцветном прямоугольнике.

Следовательно, высота дерева не меньше n .

Плюс, так как нужен хотя бы один лист для нуля, n не хватит, следовательно, $D(EQ) \geq n + 1$.

Теорема 4.2.1. Если существует **Fooling set** размера s , то $\text{rk}_R s \geq \sqrt{s} - o(1)$. Без доказательства.

4.3 Балансировка протоколов

Теорема 4.3.1. Пусть для функции f существует коммуникационный протокол P с l листьями. Тогда $D(f) \leq 3 \log l$.^a

^aВроде на лекциях этого не было, зато есть в билетах



Лемма 24. В двоичном дереве с l листьями можно найти внутреннюю вершину u , которая разбивает граф на три части, каждая из которых содержит не более $\frac{l}{2}$ листьев.

Обозначим через l_u поддерево с корнем во внутренней вершине u .

Найдем самую низкую вершину u , для которой $|l_u| > \frac{l}{2}$. Пусть ее дети v и w , причем $l_v \leq \frac{l}{2}$ и $l_w \leq \frac{l}{2}$ (если бы было не так, то могли бы взять ребенка у которого нарушается неравенство).

При этом количество листьев не из поддерева u будет меньше $l - l_u < \frac{l}{2}$.

Следовательно, u подходит.

Замечание. Такая вершина называется *центроид*.

Интуиция: Хотим перебалансировать дерево так, чтобы прыгать только по центроидам, в следствие чего глубина нового протокола уменьшится (центроидная декомпозиция).

Опишем протокол P_0 для вычисления f .

- Зафиксируем вершину u , которая делит исходный протокол на три части, в каждой из которых не более $\frac{l}{2}$ листьев.
- Пусть вершина u соответствует ходу Алисы. Сначала проверим, что и Алиса, и Боб доходят до этой вершины в исходном протоколе P : отправляем по одному биту (сначала Алиса, потом Боб обратно отправляет результат).
 - Если и Боб, и Алиса могут добраться (по [лемме 22](#)), то Алиса говорит Бобу, в какое поддерево она бы пошла с ее информацией. Тем самым мы опускаемся в одно из поддеревьев, листьев в котором не больше $\frac{l}{2}$.
 - Если нет, то нам точно не подходят оба поддерева u (в них нет листа ответа для (x, y)), а поэтому нужно подняться в третье поддерево (к родителю u) — для этого Алиса может ничего не отправлять, так как Боб уже знает, что спуска вниз не будет.

Таким образом, обменявшись \leq тремя битами, Алиса и Боб уменьшают количество листьев вдвое. Теперь мы можем не рассматривать недостижимые вершины и составить рекурсивно новый протокол P_1 , где листьев $l_1 \leq \frac{l}{2}$.

В итоге на один «шаг» нужно не более трех битов, при этом количество листьев уменьшается в два раза, следовательно, сложность такого протокола будет не более $3 \log l$.

4.4 Сложная функция. Теорема Шеннона

Определение 87

Формульная (схемная) сложность $L(f)$ формулы f — минимальное возможное число вершин двоичного дерева, вычисляющего f .

Теорема 4.4.1 (Шеннон). Существует $f: \{0, 1\}^n \rightarrow \{0, 1\}$, такая что

$$L(f) \geq \Omega\left(\frac{2^n}{n}\right).$$

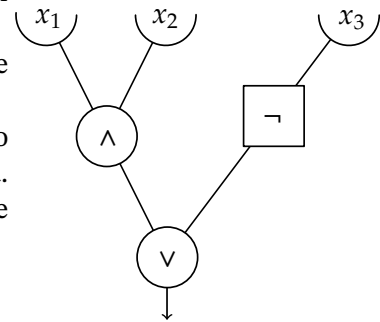
□

Всего функций такого вида 2^{2^n} , так как можно задать таблицей истинности.

Посчитаем число схем. Каждую схему можно представить в виде ациклического графа и того, что записано в его узлах.

Пусть каждая вершина (считаем, что размер схемы S , поэтому их столько же) выбирает себе двух предков. Для этого она должна хранить их номера. Так же в каждую вершину нужно записать операцию (\wedge или \vee), а на ребре можно ставить отрицание. Чтобы это закодировать хватит

$$\underbrace{S \cdot 2(\log S + \log n)}_{\text{у вершины 2 предка: либо другие вершины, либо вход}} + \underbrace{3S}_{\text{операция плюс отрицания}}$$



Итого схем может быть: $2^{S \cdot 2(\log S + \log n) + 3S}$.

Чтобы мы могли закодировать все функции в виде схем тогда и только тогда, когда схем не меньше количества функций:

$$2^{S \cdot 2(\log S + \log n) + 3S} \geq 2^{2^n}$$

$$S \cdot (2 \log S + 2 \log n + 3) \geq 2^n$$

Пусть для всех схем $S < \frac{1}{4} \cdot \frac{2^n}{n}$, тогда

$$S \cdot (2 \log S + 2 \log n + 3) < \frac{2^n}{4n} (2n - 4 + 3) = 2^n \left(\frac{1}{2} - \frac{1}{4n} \right) \leq 2^n \cdot \frac{1}{2} < 2^n$$

Противоречие. Значит есть схема со сложностью $\geq \frac{1}{4} \cdot \frac{2^n}{n}$.

■

Теорема 4.4.2 (Существование сложной монотонной функции). Существует монотонная $f: \{0, 1\}^n \rightarrow \{0, 1\}$, такая что

$$L(f) \geq 2^n - o(2^n).$$

□ Оценим снизу количество монотонных функций. Рассмотрим такие функции g , что $g(u) = 0$, если в $u < \lfloor \frac{n}{2} \rfloor$ единиц, и $g(u) = 1$, если в $u > \lfloor \frac{n}{2} \rfloor$ единиц. Если же единиц ровно $\lfloor \frac{n}{2} \rfloor$, то функция может быть определена как угодно. Заметим, что всего функций такого вида будет $2^{\lfloor \frac{n}{2} \rfloor}$, так как на входах, где $\lfloor \frac{n}{2} \rfloor$ единиц, ответ может быть любым из 0 или 1, входов с $\lfloor \frac{n}{2} \rfloor$ единиц как раз $\binom{n}{\lfloor \frac{n}{2} \rfloor}$. Можем оценить $\binom{n}{\lfloor \frac{n}{2} \rfloor} \leq 2^n$, например, через бином Ньютона: $(1 + 1)^n = \sum_{k=0}^n \binom{n}{k} \geq \binom{n}{\lfloor \frac{n}{2} \rfloor}$.

В предыдущей теореме 4.4.1 показали, что всего схем сложности S может быть $2^{S \cdot 2(\log S + \log n) + 3S}$. Тогда для схем, вычисляющих монотонные функции, имеем неравенство:

$$2^{S \cdot 2(\log S + \log n) + 3S} \geq 2^{\left(\lfloor \frac{n}{2} \rfloor\right)}$$

$$S \cdot (2 \log S + 2 \log n + 3) \geq 2^n \geq \left(\frac{n}{2}\right)$$

$$\log S \geq n - \log(2 \log S + 2 \log n + 3) = n - o(n)$$

■

Открытый вопрос: Можно ли предъявить $f \in \mathbf{NP}$, что $L(f) \geq 10n$

4.5 Теорема Карчмера-Вигдерсона

Пусть нам дана $f: \{0, 1\}^n \rightarrow \{0, 1\}$.

Алиса получает число $x \in f^{-1}(1)$, а Боб $y \in f^{-1}(0)$. Их цель найти любой бит, в котором x и y отличаются.

Теорема 4.5.1 (Karchmer-Wigderson, 1990). $L(f)$ — размер минимальной формулы для f , тогда и только тогда $L(f)$ — размер минимального протокола для KW_f

□

1 \Rightarrow 2 Нарисуем дерево вверх корнем. Также спустим все отрицания к листьям. Пусть в корне считается функция $f = g \vee h$, где g и h — соседи f .

Тогда $f(x) = 1, f(y) = 0$. Тогда $g(y) = h(y) = 0$, а для Алисы хотя бы одно из двух значений единица. Тогда Алиса посылает информацию, где именно единица, то есть куда нам нужно спуститься (потому что в этом отрезке битов точно есть отличие). Далее игра продолжается по тем же правилам рекурсивно. Стоит заметить, что если в вершине конъюнкция, то Боб пошлет информацию, где происходит обнуление. Так за глубину формулы мы нашли решение для KW_f той же глубины. Т.е. оптимальный размер протокола меньше или равен $L(f)$.

2 \Rightarrow 1 Пусть у нас есть некоторый протокол для игры. Это некоторое дерево. Обозначим за $R_v := X_v \times Y_v$ — прямоугольник входов для вершины v , из которых мы получаем v .

Мы хотим построить $f: \{0, 1\}^n \rightarrow \{0, 1\}$.

Будем подниматься снизу и строить формулу по протоколу. Обозначим за f_v построенную формулу в вершине v . Хотим получить следующие свойства для формулы: $f_v(X_v) = 1$ и $f_v(Y_v) = 0$.

Если они выполняются, то $\forall x \in X_v: f(x) = 1$ и $\forall y \in Y_v: f(y) = 0$.

В корне $f = f_r$.

- Если мы в листе l . Здесь написан некоторый ответ i . То есть $\forall x \in X_l \forall y \in Y_l: x_i \neq y_i$.

Тогда либо $x_i = 0 \wedge y_i = 1$, либо наоборот. (Причём одинаково для всех пар x_i, y_i)

В качестве $f_l(z)$ можем в первом случае взять $\neg z_i$, во втором z_i .

- Теперь мы находимся в вершине v с потомками a и b .

Если ходит Алиса, то прямоугольник R_v разрезается на R_a и R_b горизонтально (если Боб, то наоборот, вертикально).

У нас уже есть две функции f_a и f_b , построенные по предположению индукции. $\forall y \in Y_v: f_a(y) = f_b(y) = 0$, так как $Y_v = Y_a = Y_b$. А так как $X_v \subseteq X_a \cup X_b$, $\forall x \in X_v$ либо $f_a(x) = 1$, либо $f_b(x) = 1$. Поэтому нам подходит $f_v := f_a \vee f_b$.

Если же ходит Боб нужно будет сделать конъюнкцию.

■

4.6 Вероятностная модель

Теперь Алиса и Боб могут подбрасывать монетки. Либо эти монетки (оракулы) *публичны* (оба видят значения), либо *приватны* (тогда никто не видит, кроме пользователя).

Каждый шаг будет зависеть от входа x и случайной строки $r \in \{0, 1\}^l$.

Так как Алиса или Боб в случае публичного оракула, могут закрыть глаза на сообщения другого, публичный протокол не меньше приватного.

Вероятность ошибки на входе (x, y) — доля случайных слов r , при которых протокол выдает неправильный ответ.

Скажем, что **протокол отработал корректно**, если³

$$\forall x, y: \Pr[\pi(x, y) = f(x, y)] \geq \frac{2}{3}, \quad \pi(x, y) \text{ — результат работы.}$$

Определение 88

Вероятностный протокол ε -**вычисляет** f , если для любой пары x, y с вероятностью не менее $1 - \varepsilon$ результат протокола равен $f(x, y)$ (с точки зрения обоих игроков).

Через $R_\varepsilon(f)$ обозначается минимальная высота вероятностного протокола ε -вычисляющего f .

Через R_ε^{pub} — минимальная высота в случае публичного протокола.

4.6.1 Эффективные протоколы для EQ и GT

Лемма 25. $R_\varepsilon^{pub}(\text{EQ}) = \mathcal{O}(\log \varepsilon^{-1})$

□ Рассмотрим следующий протокол. На входах $x, y \in \{0, 1\}^n$ Алиса и Боб выбирают случайную строку $r \in \{0, 1\}^n$, затем Алиса считает скалярное произведение $\langle x, r \rangle$ над \mathbb{F}_2 и передает результат (один бит) Бобу, который считает свое скалярное произведение $\langle y, r \rangle$. Если биты равны, отвечает 1, иначе 0.

- Если $x = y$, то ответ точно будет одинаковый и протокол работает правильно.
- Если $x \neq y$, протокол ошибается (то есть выдает единицу, хотя должен нуль), если $\langle x - y, r \rangle = 0$. Это линейное уравнение на r . Соответственно, его решения — некоторое линейное пространство размерности $n - 1$ над \mathbb{F}_2 .

Тогда решений будет ровно 2^{n-1} . А это половина от всех возможных r . Следовательно, вероятность ошибки $\frac{1}{2}$.

Теперь просто повторяем этот протокол k раз с независимыми битами r , причем будем выдавать 1, если получили все k единиц. За счет этого теперь вероятность ошибки станет 2^{-k} .

Если мы хотим добиться ошибки ε , достаточно повторить протокол $\log \varepsilon^{-1}$ раз. ■

Лемма 26. $R_\varepsilon^{pub}(\text{GT}) = \mathcal{O}(\log n \cdot \log \frac{\log n}{\varepsilon})$

□ Чтобы сравнить $x, y \in \{0, 1\}^n$, Алисе и Бобу нужно найти первый (самый старший) бит, в котором их входы отличаются, и обменяться этими битами.

Будем искать этот бит бинарным поиском. Каждый раз мы делим текущий блок на две части и проводим тест на равенство левых частей (который мы умеем делать по [лемме 25](#)). Если они оказались равны, то выбираем правые, иначе левые, и запускаемся рекурсивно.

³Вместо $\frac{2}{3}$ может быть любое $\varepsilon \in (\frac{1}{2}, 1)$

После каждого деления блок уменьшается не меньше, чем в два раза, следовательно, тест на равенство нужно провести $\mathcal{O}(\log n)$ раз.

Чтобы общая вероятность была меньше ε , вероятность ошибки на одной проверке должна быть меньше $\delta = \frac{\varepsilon}{\log n}$.

В итоге получается $R_\varepsilon^{pub}(\text{GT}) = \mathcal{O}(\log n \cdot \log \frac{\log n}{\varepsilon})$. ■

4.7 Теория информации в коммуникационной сложности

4.7.1 Подсчет функции индексов

Лекция 6
6 May

Определение 89

Определим **функцию индексирования** следующим образом:

$$\text{Ind}: [n] \times \{0, 1\}^n \rightarrow \{0, 1\}, \quad \text{Ind}(x, y) = y_x.$$

Алиса и Боб хотят посчитать эту величину, причем x у Алисы, а y у Боба.

Пусть сообщения идут только от Боба до Алисы. Несложно понять, что Бобу придется послать всю информацию.

Теперь предположим, что они хотят, чтобы Алиса посчитала Ind верно с вероятностью $\frac{1}{2} + \delta$, если x и y выбираются равномерно. Очевидно, для $\delta = 0$, просто ничего не нужно пересылать. А вот для других положительных значений все испортится.

.....
Пусть $M(y)$ — случайная величина, это сообщение, которое передает Боб Алисе в зависимости от своего входа. Легко получить

$$D(\pi) \geq \log |L| \geq H(M) \geq I(M : y)$$

Где L — множество различных сообщений, которые может отправить Боб (то есть множество различных листьев в протоколе), $D(\pi)$ — глубина протокола (то есть количество бит, которое нужно отправить).

$$\begin{aligned} I(M : y) &= \sum_i I(M : y_i \mid y_{<i}) = && \text{(Chain rule)} \\ &= \sum_i H(y_i \mid y_{<i}) - H(y_i \mid M, y_{<i}) = && (y_i \text{ независимы}) \\ &= \sum_i H(y_i) - H(y_i \mid M, y_{<i}) \geq && \text{(Выкинули часть условий)} \\ &\geq \sum_i H(y_i) - H(y_i \mid M) = \\ &= \sum_i I(M : y_i) \end{aligned}$$

Покажем, что полученная сумма большая.

Зафиксируем i и распишем по определению взаимной информации:

$$\begin{aligned}
 \sum_i I(M : y_i) &= \sum_i H(y_i) - H(y_i | M) = & (H(y_i) = 1) \\
 &= \sum_i 1 - \mathbb{E}_m (H(y_i | M = m)) = \\
 &\quad (x \text{ не зависит от } (M, y), \text{ значит } (y | M) \text{ не зависит от } (x | M)) \\
 &= \sum_i 1 - \mathbb{E}_m (H(y_i | M = m, x = i)) = \\
 &= \sum_i 1 - \mathbb{E}_m (H(r_m^i)) \geq & (\text{Неравенство Йенсена для выпуклой вверх } H(q)) \\
 &\geq \sum_i 1 - H(\mathbb{E}_m(r_m^i)) = \\
 &= n - n \sum_i \frac{1}{n} H(\mathbb{E}_m(r_m^i)) \geq & (\text{Опять неравенство Йенсена для } H(q)) \\
 &\geq n \cdot \left(1 - H\left(\sum_i \frac{1}{n} \mathbb{E}_m(r_m^i)\right) \right) = n \cdot \left(1 - H(\mathbb{E}_i(\mathbb{E}_m(r_m^i))) \right) \\
 &\geq n \cdot \left(1 - H(\mathbb{E}_{m,i}(r_m^i)) \right) \geq & (\text{Монотонность энтропии на } (0, \frac{1}{2})) \\
 &\geq n \cdot \left(1 - H\left(\frac{1}{2} - \delta\right) \right) = & (\text{Ряд Тейлора для энтропии}) \\
 &= \Omega(\delta^2 n)
 \end{aligned}$$

Здесь r_m^i — характеристическая функция ошибки $M = m, x = i$.

Пояснение к переходу к r_m^i : Так как у Алисы алгоритм детерминированный, то при фиксированных x, M она решает, что y_x равно конкретному значению (0 или 1) при этом, если $\Pr[y_i = 1 | M = m, x = i] = p$, то рассмотрим 2 случая:

1. Алиса решает, что $y_x = 1$, то она ошибется с вероятностью $1 - p$
2. Алиса решает, что $y_x = 0$, то она ошибется с вероятностью p

То есть в обоих случаях распределения совпадают, а значит энтропия одинакова.

Чтобы алгоритм был корректен, $\mathbb{E}_{i,m}(r_m^i) \leq \frac{1}{2} - \delta$.

Теперь $D(\pi) \geq \Omega(\delta^2 n)$. То есть нам нужно передать хотя бы $\Omega(\delta^2 n)$ бит. С другой стороны можем построить простой протокол с $2\delta n$ битами. Для этого Боб передаст первые $2\delta n$ бит, а затем Алиса, если нужный ей бит есть, выдаст его, иначе выдаст 0. Вероятность ошибки в таком случае ровно такая, как мы хотели: $\frac{1}{2}(1 - 2\delta)$. Эта верхняя оценка больше нашей нижней. На самом деле она улучшается.

Определение 90

Пусть μ — вероятностная мера на $X \times Y$.

$IC_\mu^{ext}(\pi) := I(\pi(X, Y) : (X, Y))$ — внешнее информационное разглашение.

$IC_\mu^{int}(\pi) := I(\pi(X, Y) : X | Y) + I(\pi(X, Y) : Y | X)$ — внутреннее информационное разглашение.

Интуиция:

1. Внешнее информационное разглашение — сколько информации получит внешний наблюдатель об (X, Y) , зная значение протокола $\pi(X, Y)$.
2. Внутреннее информационное разглашение — сколько информации получают внутренние наблюдатели о входах друг друга, посмотрев значения протокола.

Теорема 4.7.1. $D(\pi) \geq IC_{\mu}^{ext}(\pi) \geq IC_{\mu}^{int}(\pi)$

□

- Первое неравенство очевидно, так как

$$I(\pi(X, Y) : (X, Y)) \leq H(\pi(X, Y)) \leq \log |L| \leq D(\pi(X, Y))$$

L — листья протокола.

- Докажем вторую часть теоремы. Для этого покажем $I(\pi : x, y) \geq I(\pi : x | y) + I(\pi : y | x)$.

Если оставить только одно слагаемое, слева останется $H(\pi) - H(\pi | x, y)$, а справа $H(\pi | y) - H(\pi | x, y)$, которое точно не больше.

Пусть π_i — i -ый бит π , то есть то, что Алиса и Боб отправили за i -ый раунд.

$$I(\pi : x, y) = \sum_i I(\pi_i : x, y | \pi_{<i}) \quad (\text{Chain rule})$$

Аналогично попробуем нарезать слагаемые правой части и оценим каждую часть по отдельности:

$$I(\pi : x | y) = \sum_i I(\pi_i : x | y, \pi_{<i})$$

$$I(\pi : y | x) = \sum_i I(\pi_i : y | x, \pi_{<i})$$

Вход Боба первое слагаемое «равно нулю», так как π_i определяется y -ом. Аналогично второе «равно нулю», когда ходит Алиса. Поэтому каждый раз неравенство сохраняется. На самом деле есть ошибка в рассуждениях, т.к. $\pi_{<i}$ с.в., то при разных значениях $\pi_{<i}$ ходить может либо Боб, либо Алиса, а не всегда кто-то один при всех значениях, как мы пытались обосновать.

Чтобы исправить скрытую фундаментальную ошибку распишем через матожидания

$$I(\pi : x, y) = \sum_i \mathbb{E}_m I(\pi_i : x, y | \pi_{<i} = m)$$

$$I(\pi : x | y) = \sum_i \mathbb{E}_m I(\pi_i : x | y, \pi_{<i} = m)$$

$$I(\pi : y | x) = \sum_i \mathbb{E}_m I(\pi_i : y | x, \pi_{<i} = m)$$

Это уже корректное утверждение, так как для каждого конкретного m одно из слагаемых действительно обнуляется, значит для матожиданий неравенство есть, ну и значит для суммы тоже.

■

Теорема 4.7.2 (Храпченко). $L(XOR) \geq \Omega(n^2)$

□ Покажем, что для любого протокола π задачи KW_{\oplus_n} существует такое распределение μ , что $IC_{\mu}^{int}(\pi) \geq 2 \log n$. Отсюда будет следовать, что (пользуясь выводами в [теореме 4.7.1](#)) $D(\pi) \geq \log m \geq \log |L| \geq 2 \log n$, где m — кол-во вершин, $|L|$ — кол-во листьев, а значит и $L(\oplus_n) \geq n^2$. Пусть распределение μ будет равномерным на всех парах вида $(x, x \oplus e_i)$, где $\oplus_n(x) = 1$, то есть в x нечётное число единиц, а строка e_i

имеет единицу в позиции i и нули во всех остальных. Таким образом, пары входов из распределения μ всегда будут отличаться только в одном бите. По определению,

$$\text{IC}_{\mu}^{\text{int}}(\pi) = I(\pi : X | Y) + I(\pi : Y | X).$$

Рассмотрим одно из слагаемых $I(\pi : Y | X)$. Имеем:

$$I(\pi : Y | X) = I(\pi : X \oplus e_i | X) = H(X \oplus e_i | X) - \underbrace{H(X \oplus e_i | X, \pi)}_{=0} = H(e_i) = \log n$$

Аналогичное равенство верно и для $I(\pi : X | Y)$. Таким образом, $\text{IC}_{\mu}^{\text{int}}(\pi) \geq 2 \log n$, что и требовалось. Верхнюю оценку на $D(\pi)$ нетрудно доказать, явно построив функцию. ■

Chapter 5

Колмогоровская сложность

Лекция 7
20 May

5.1 Определения

Пусть F — вычислимая декодирующая функция. Определим $K_F(x) := \min\{|p| \mid F(p) = x\}$. Это «критерий сжимаемости» функции F .

Будем считать, что $F \leq G$ (F не хуже G), если $\forall x K_F(x) \leq K_G(x) + c_{FG}$.

Назовем способ описания (функцию) оптимальным, если она не хуже всех остальных.

Теорема 5.1.1. Существует оптимальный способ описания.

Определение 91

Колмогоровская сложность для x — $K(x) := K_U(x)$, где U — оптимальный способ описания.

Свойства.

1. $K(x) \leq |x| + c$, так как можно взять $K_{id}(x) = |x|$
2. $K(XX) \leq |x| + c$, можно взять описание $F(p) = pp$
3. Пусть Gx не более p единиц, тогда $K(x) \leq H(p) \cdot |x| + c$.

Можем взять $F(p) = p$ -ое слово с не более p единицами.

$$\sum_{i=0}^n \binom{i}{n} \leq 2^{H(p)n}.$$

Теорема 5.1.2. Пусть M — всюду вычислимая функция. Если $M(x) \leq K(x)$ и $\forall c \exists x: M(x) \geq c$, то M не вычислима.

□ Зафиксируем c . Найдем x_c — первое слово, где $M(x_c) \geq c$. Так как M вычислима всюду, определено $F(c)$. Тогда из $F(c) = x_c$ следует, что $K(x) \leq \log c + c_0$ по определению. ■

Следствие 22. У почти всех слов колмогоровская сложность равна $n - \text{const}$.

5.2 Применение

Мы лишаемся вычислимости, поэтому мы лишаемся практических применений, как с энтропией. Но есть математическое применение, так как это оптимальный алгоритм.

Можно доказать, что одноленточная машина Тьюринга, копирующая вход будет работать $\Omega(|x|^2)$.

5.3 Условная сложность

$$K(x \mid y) = K_U(x \mid y).$$

Это способ описания, который может еще использовать y : $K_F(x \mid y) := \min\{|p| \mid F(p, y) = x\}$. Аналогично U — оптимальный способ описания¹.

Замечание. $K(x) = K(x \mid \emptyset) + \text{const}$

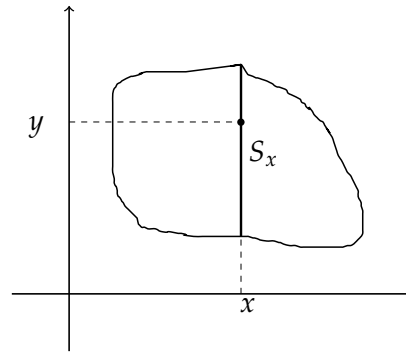
$$K(x, y) := K(\langle x, y \rangle) \quad \langle \cdot, \cdot \rangle \text{ — какая-то кодировка пары.}$$

Свойства.

1. $K(x \mid y) \leq K(x) + \mathcal{O}(1)$
2. $K(x, y) \leq K(x) + K(y \mid x)$

Теорема 5.3.1 (Колмогоров, Левин). $K(x, y) = K(x) + K(y \mid x)$

□ В одну сторону по свойству. Пусть $n = K(x, y)$ Пусть $S := \{(a, b) \mid K(a, b) \leq n\}$,



$$K(y \mid x) \leq \underbrace{\log |S_x|}_m + \mathcal{O}(\log n).$$

Рассмотрим все x , для которых $\log |S_x| \geq m$. Таких не более 2^{n-m+} , так как мы знаем, что $K(x, y) = n$, то внутри множества размером 2^n , есть элемент с большой сложностью, теперь множество S по размеру не более 2^{n+c} , но так как в одном сечении не более 2^m , получаем 2^{n-m+c} .

Тогда $K(x) \leq n - m$. Если мы подставим в неравенство выше, то получим то, что хотели. ■ Конец.

¹Упражнение — аналогично доказать существования