Дополнительные главы дискретной математики и кибернетики

Презентации к лекциям по частям I, II: конечные автоматы, машины Тьюринга, рекурсивные функции и сложностные классы

Савицкий Игорь Владимирович

факультет ВМК МГУ

осень 2023

Лекция 1
Конечные автоматы-распознаватели. Правоинвариантные отношения эквивалентности. Теоретико-множественные операции над автоматными множествами.

Операции над словами

Определение

- Алфавит A — это непустое множество символов.
- A^* — это множество слов (конечной длины) в алфавите A, включая пустое слово Λ.
- Длина $|w|$ слова $w \in A^*$ — это количество символов в слове w. Длина пустого слова Λ есть нуль.

Определение

- Конкатенация слов $u = a_{i_1} \ldots a_{i_k} \in A^*$ и $v = b_{j_1} \ldots b_{j_l} \in A^*$ — это слово
$$u * v = uv = a_{i_1} \ldots a_{i_k} b_{j_1} \ldots b_{j_l} \in A^*.$$
При этом для любого $w \in A^*$ определяем $\Lambda w = w \Lambda = w$.
- Возведение в степень: $a^n = \underbrace{a * \ldots * a}_{n \text{ раз}}$ при $n \in \mathbb{N}$; $a^0 = \Lambda$.

Конечные автоматы

- Конечный автомат-распознаватель — это абстрактное вычислительное устройство, предназначенное для распознавания множества слов.

Определение

Конечный автомат (распознаватель) — это $\mathcal{A} = (A, Q, f, q_1, F)$, где

- $A \neq \varnothing$ — входной алфавит (часто задан заранее и не является частью автомата),
- $Q \neq \varnothing$ — множество состояний,
- $f : A \times Q \to Q$ — функция переходов,
- $q_1 \in Q$ — начальное состояние,
- $F \subseteq Q$ — множество заключительных состояний.

Конечные автоматы

Работа автомата

- На вход автомату подаётся слово $x \in A^*$. Через $x(t)$ обозначаем t-й символ входного слова.
- Автомат работает в дискретном времени: $t = 1, 2, \ldots$ На каждом такте t автомату подаётся очередной символ $x(t)$.
- На каждом такте t автомат меняет своё состояние $q(t)$ согласно канонически уравнениям:

$$\begin{cases} q(t) = f(x(t), q(t-1)), \\ q(0) = q_1. \end{cases}$$

- После обработки всего слова x автомат останавливается в состоянии $q(|x|)$. Если это состояние принадлежит F, то автомат допускает слово x. Иначе он отвергает это слово.

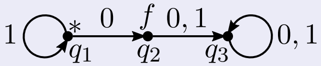
Конечные автоматы

Диаграмма Мура

- Пусть A — входной алфавит автомата, $k = |A|$.
- Каждому состоянию автомата соответствует вершина графа.
- Из каждой вершины исходит k дуг, помеченных символами алфавита A. Они показывают, куда переходит автомат из каждого состояния под действием каждого символа.
- Начальное состояние помечено $*$. Заключительные состояния помечены f.

Пример

- $A = \{0, 1\},\ Q = \{q_1, q_2, q_3\},\ F = \{q_2\}$, диаграмма Мура автомата:



- Автомат допускает слова вида $1^n0,\ n \geqslant 0$.

Конечные автоматы

Конечно-автоматные множества

- Пусть $\mathcal{A}$ — автомат с входным алфавитом A. Тогда $D(\mathcal{A})$ — это множество всех слов из A^*, которые допускает автомат $\mathcal{A}$.
- Множества вида $D(\mathcal{A})$ (где $\mathcal{A}$ — конечный автомат), называются конечно-автоматными.

Правоинвариантные отношения эквивалентности

Определение

Отношение $\sim \subseteq A^* \times A^*$ — отношение эквивалентности, если

- $\forall\, a \in A^* \qquad a \sim a,$
- $\forall\, a, b \in A^* \qquad a \sim b \equiv b \sim a,$
- $\forall\, a, b, c \in A^* \quad (a \sim b)\ \&\ (b \sim c) \to a \sim c.$

Определение

- Множество A^* разбивается отношением $\sim$ на классы эквивалентности: максимальные множества попарно эквивалентных элементов.
- Индекс отношения эквивалентности — это число классов эквивалентности.

Правоинвариантные отношения эквивалентности

Отношение эквивалентности, связанное с автоматом

- Пусть $\mathcal{A} = (A, Q, f, q_1, F)$ — автомат, $Q = \{q_1, \ldots, q_r\}$.
- Тогда $A^* = X_1 \cup \ldots \cup X_r$, где X_i — это множество слов, которые переводят автомат $\mathcal{A}$ из состояния q_1 в состояние q_i. Ясно, что множества X_i попарно не пересекаются. В частности, $\Lambda \in X_1$.
- Пустые множества X_i исключаем из набора.
- По разбиению A^* на X_i обычным образом введём на A^* отношение эквивалентности $\sim$: $a \sim b \iff (\exists i)(a, b \in X_i)$.
- Это отношение называем отношением эквивалентности автомата $\mathcal{A}$ и обозначаем $\sim_{\mathcal{A}}$. Оно обладает следующими свойствами:
  1. Отношение $\sim_{\mathcal{A}}$ имеет конечный индекс (конечное число классов эквивалентности).
  2. Отношение $\sim_{\mathcal{A}}$ правоинвариантно: если $a \sim_{\mathcal{A}} b$ и $c \in A^*$, то $ac \sim_{\mathcal{A}} bc$.

Правоинвариантные отношения эквивалентности

Определение

- Отношение эквивалентности $\sim$ имеет конечный индекс, если число его классов эквивалентности конечно.
- Отношение эквивалентности $\sim \subseteq A^* \times A^*$ является правоинвариантным, если

$$\forall\, a, b, c \in A^* \quad (a \sim b) \to (ac \sim bc).$$

Правоинвариантные отношения эквивалентности

Построение автомата по правоинвариантной эквивалентности

- Пусть на A^* задано правоинвариантное отношение эквивалентности $\sim$, которое разбивает A^* на конечное число классов эквивалентности $K_1, \ldots, K_r$, причём $\Lambda \in K_1$.
- Определим автомат $\mathcal{A} = (A, \{K_1, \ldots, K_r\}, h, K_1, F)$. Множество F определяется произвольно.
- Определим функцию переходов h:
  Для каждых класса K_i и $a_j \in A$ выбираем любое $a \in K_i$. Тогда $aa_j \in K_l$ для некоторого l. Задаём $h(a_j, K_i) = K_l$.
- За счёт правоинвариантности отношения класс K_l не зависит от выбора a: если $a, b \in K_i$, то aa_j и ba_j принадлежат одному и тому же классу K_l. Поэтому функция переходов задана корректно.
- Каждый класс K_i совпадает со множеством слов, которые переводят автомат $\mathcal{A}$ из состояния K_1 в состояние K_i.

Правоинвариантные отношения эквивалентности

- Отношение эквивалентности $\sim_\mathcal{A}$ автомата $\mathcal{A}$, построенного по правоинвариантному отношению эквивалентности $\sim$ конечного индекса, совпадает с отношением $\sim$.
- Результаты построений сформулируем в виде теоремы.

Теорема 1

- *Отношение эквивалентности $\sim_\mathcal{A}$ любого автомата $\mathcal{A}$ является правоинвариантным и имеет конечный индекс.*
- *Для каждого правоинвариантного отношения эквивалентности $\sim$ конечного индекса можно построить конечный автомат $\mathcal{A}$, отношение эквивалентности $\sim_\mathcal{A}$ которого совпадает с $\sim$.*

Правоинвариантные отношения эквивалентности

Теорема 2

- *Всякое непустое конечно-автоматное множество есть объединение некоторого числа классов подходящего правоинвариантного отношения эквивалентности конечного индекса.*
- *Обратно, объединение любого числа классов произвольного правоинвариантного отношения эквивалентности конечного индекса является конечно-автоматным множеством.*

Правоинвариантные отношения эквивалентности

Доказательство (автоматность $\Rightarrow$ классы эквивалентности)

- Пусть имеется непустое конечно-автоматное множество $D(\mathcal{A})$. Автомат $\mathcal{A}$ всегда можно выбрать так, чтобы в нём не было недостижимых из q_1 состояний.
- Пусть $\mathcal{A} = (A, Q, f, q_1, F)$ — автомат, $Q = \{q_1, \ldots, q_r\}$, а $F = \{q_{i_1}, \ldots, q_{i_s}\}$.
- $A^* = X_1 \cup \ldots \cup X_r$, где X_i — это множество слов, которые переводят автомат $\mathcal{A}$ из состояния q_1 в состояние q_i.
- $X_1, \ldots, X_r$ — классы эквивалентности отношения эквивалентности $\sim_{\mathcal{A}}$ автомата $\mathcal{A}$. По теореме 1 отношение $\sim_{\mathcal{A}}$ правоинвариантно.
- Ясно, что $D(\mathcal{A}) = X_{i_1} \cup \ldots \cup X_{i_s}$. То есть конечно-автоматное множество является объединением некоторых классов эквивалентности некоторого правоинвариантного отношения эквивалентности конечного индекса.

Правоинвариантные отношения эквивалентности

Доказательство (классы эквивалентности $\Rightarrow$ автоматность)

- Пусть имеется правоинварантное отношение эквивалентности $\sim$ конечного индекса на A^* с классами эквивалентности $K_1, \ldots, K_r$ и $X = K_{i_1} \cup \ldots \cup K_{i_s}$ — объединение некоторых классов эквивалентности.
- Тогда по теореме 1 мы можем построить конечный автомат $\mathcal{A}$, отношение эквивалентности $\sim_{\mathcal{A}}$ которого совпадает с $\sim$.
- Автомат будет иметь состояния $K_1, \ldots, K_r$, причём K_i совпадает со множеством слов, которые переводят автомат $\mathcal{A}$ из состояния K_1 в состояние K_i.
- Тогда выберем множество заключительных состояний $F = \{K_{i_1}, \ldots, K_{i_s}\}$. Получится, что $D(\mathcal{A}) = K_{i_1} \cup \ldots \cup K_{i_s}$, то есть множество X конечно-автоматно.

$\square$

Правоинвариантные отношения эквивалентности

- Правоинвариантные отношения эквивалентности можно использовать для доказательства того, что множество не является конечно-автоматным.

Пример

- Докажем, что $X = \{a_1^n a_2^n \mid n \in \mathbb{N}\}$, где $a_1, a_2 \in A$, не является конечно-автоматным. От противного.
- Пусть X — конечно-автоматное множество. Тогда оно является объединением некоторых классов эквивалентности правоинвариантного отношения $\sim$ конечного индекса.
- Выберем такие $i \neq j$, что $a_1^i \sim a_1^j$. Это возможно, так как классов эквивалентности конечное число.
- Тогда $a_1^i a_2^i \sim a_1^j a_2^i$. Но это невозможно, т. к. $a_1^i a_2^i \in X,\ a_1^j a_2^i \notin X$.
- Значит X не конечно-автоматно.

Операции над автоматными множествами

Операция дополнения $\overline{X}$

- Дополнение: $\overline{X} = A^* \setminus X$.
- Пусть X — конечно-автоматное множество.
  $X = D(\mathcal{A}),\ \mathcal{A} = (A, Q, f, q_1, F)$.
- Тогда $\overline{X} = D(\mathcal{A}')$, где $\mathcal{A}' = (A,\ Q,\ f,\ q_1,\ Q \setminus F)$.
- Поэтому $\overline{X}$ — конечно-автоматное множество.
- Операция дополнения сохраняет конечную автоматность множеств.

Операции над автоматными множествами

Операция пересечения $X \cap Y$

- Пусть $X, Y \subseteq A^*$ — конечно-автоматны. Тогда существуют два правоинвариантных отношения эквивалентности конечного индекса $\sim_1, \sim_2$ такие, что
  $K_1, \ldots, K_u$ — классы эквивалентности $\sim_1$ и $X = K_{i_1} \cup \ldots \cup K_{i_s}$,
  а $L_1, \ldots, L_v$ — классы эквивалентности $\sim_2$ и $Y = L_{i_1} \cup \ldots \cup L_{i_t}$.
- Введём отношение эквивалентности $\sim_3$ с классами эквивалентности $M_1, \ldots, M_p$ — всеми непустыми пересечениями вида $K_i \cap L_j$. Оно правоинвариантно.
- Тогда $X \cap Y$ — объединение всех непустых пересечений вида $K_{i_m} \cap L_{i_n}$, то есть некоторых классов $\sim_3$.
- Поэтому $X \cap Y$ конечно-автоматно. Операция пересечения сохраняет конечную автоматность множеств.

Операции над автоматными множествами

Иллюстрация пересечений классов

	K_1	K_2	K_3
L_1	$K_1 \cap L_1$	$K_2 \cap L_1$	$K_3 \cap L_1$
L_2	$K_1 \cap L_2$	$K_2 \cap L_2$	$K_3 \cap L_1$
L_3	$K_1 \cap L_3$	$K_2 \cap L_3$	$K_3 \cap L_1$

- $X \cup Y = \overline{\overline{X} \cap \overline{Y}}$. Поэтому операция объединения тоже сохраняет конечную автоматность множеств.

Операции над автоматными множествами

- Сформулируем полученные результаты в виде теоремы.

Теорема 3

Класс всех конечно-автоматных множеств замкнут относительно теоретико-множественных операций дополнения, объединения и пересечения.

- Другие теоретико-множественные операции выражаются с помощью операций объединения, пересечения и дополнения и тоже сохраняют конечную автоматность множеств.
- Например, $X \setminus Y = X \cap \overline{Y}$.

Лекция 2
Недетерминированные конечные автоматы. Операции произведения и итерации автоматных множеств. Регулярные выражения и регулярные множества.

Недетерминированные конечные автоматы

- В отличие от обычного конечного автомата, недетерминированный конечный автомат из одного и того же состояния под действием одной и той же буквы может переходить в разные состояния.

Определение

Недетерминированный конечный автомат — это (A, Q, f, q_1, F), где

- $A \neq \varnothing$ — входной алфавит,
- $Q \neq \varnothing$ — множество состояний,
- $f : A \times Q \to 2^Q \setminus \{\varnothing\}$ — функция переходов (по символу и состоянию выбирается подмножество состояний),
- $q_1 \in Q$ — начальное состояние,
- $F \subseteq Q$ — множество заключительных состояний.

Недетерминированные конечные автоматы

Работа недетерминированного автомата

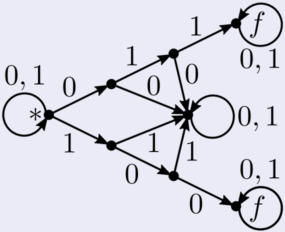
- На вход автомату подаётся слово $x \in A^*$. На каждом такте t автомату подаётся очередной символ $x(t)$.
- На каждом такте t автомат меняет своё состояние $q(t)$ согласно следующим условиям:

$$\begin{cases} q(t) \in f(x(t),\, q(t-1)), \\ q(0) = q_1. \end{cases}$$

- После обработки слова x автомат останавливается в состоянии $q(|x|)$. Автомат может обработать одно и то же слово разными способами в зависимости от выбора $q(t)$ на каждом шаге.
- Если хотя бы один способ обработки слова x приводит к состоянию из F, то автомат допускает слово x. Иначе он отвергает это слово.

Недетерминированные конечные автоматы

Пример недетерминированного автомата



На диаграмме Мура из одного состояния может исходить несколько стрелок с одним и тем же символом.

- Обычный автомат является частным случаем недетерминированного.

Недетерминированные конечные автоматы

Теорема 4

Класс множеств, допускаемых недетерминированными конечными автоматами, совпадает с классом конечно-автоматных множеств.

Доказательство

- Если множество конечно-автоматно, то оно допускается недетерминированным конечным автоматом, так как обычный автомат является частным случаем недетерминированного.
- Пусть $\mathcal{A} = (A, Q, f, q_1, F)$ — недетерминированный автомат, $X = D(\mathcal{A})$. Обозначим $r = |Q|$.
- Построим конечный автомат $\mathcal{A}'$, который допускает множество X. Выберем $\mathcal{A}' = (A,\ 2^Q \setminus \{\varnothing\},\ h,\ \{q_1\},\ F')$.
- F' — это множество всех подмножеств Q, которые пересекаются с F.

Недетерминированные конечные автоматы

Доказательство (продолжение)

- Задаём h: $h(a_i, \{q_{j_1}, \ldots, q_{j_s}\}) = f(a_i, q_{j_1}) \cup \ldots \cup f(a_i, q_{j_s})$.
- Моделирование автоматом $\mathcal{A}'$ работы $\mathcal{A}$:
  1. В начальный момент $\mathcal{A}'$ находится в состоянии $\{q_1\}$.
  2. Во второй момент времени $\mathcal{A}'$ находится в состоянии $f(x(1), q_1)$.
  3. В каждый момент времени $\mathcal{A}'$ находится в состоянии U, которое состоит из всех состояний q_i, в которые $\mathcal{A}$ мог бы прийти к этому моменту времени.
  4. В конце работы автомат $\mathcal{A}'$ попадает в некоторое состояние V. Если V пересекается с F, то хотя бы в одном способе обработке слова $\mathcal{A}$ попадает в состояние из F, и входное слово входит в $X = D(\mathcal{A})$. В противном случае входное слово не входит в X.
- Таким образом, построен конечный автомат $\mathcal{A}'$ такой, что $X = D(\mathcal{A}')$. Значит, X конечно-автоматно.

$\square$

Операция произведения множеств

Определение

- Пусть $X, Y \subseteq A^*$. Произведение X и Y есть

$$X \cdot Y = \{xy \mid x \in X,\ y \in Y\}, \quad X \cdot \varnothing = \varnothing \cdot X = \varnothing.$$

Теорема 5

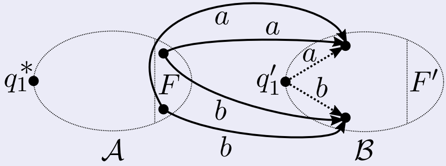
Класс конечно-автоматных множеств замкнут относительно операции произведения.

Доказательство

- Пусть $\mathcal{A} = (A, Q, f, q_1, F), \mathcal{B} = (A, Q', f', q_1', F')$ — конечные автоматы, $X = D(\mathcal{A})$, $Y = D(\mathcal{B})$, $Q \cap Q' = \varnothing$.
- Строим недетерминированный конечный автомат $\mathcal{C} = (A,\ Q \cup Q',\ h,\ q_1,\ \tilde{F})$, допускающий множество $X \cdot Y$.

Операция произведения множеств

Доказательство (продолжение)



- $h(a_i, q) = \begin{cases} \{f(a_i, q)\}, & q \in Q \setminus F, \\ \{f(a_i, q),\ f'(a_i, q_1')\}, & q \in F, \\ \{f'(a_i, q)\}, & q \in Q'. \end{cases}$
- Если $q_1' \notin F'$, то $\tilde{F} = F'$. Если $q_1' \in F'$, то $\tilde{F} = F \cup F'$.
- Автомат на состояниях Q распознаёт слово из X, а на состояниях Q' — слово из Y. Поскольку переход из Q в Q' обязателен и однократен, итоговый автомат распознаёт слова из XY. Если $\Lambda \in Y$, то $X \subseteq X \cdot Y$, поэтому $F \subseteq \tilde{F}$.

$\square$

Операция итерации множества

Определение

- Пусть $X \subseteq A^*$. $X^n = \underbrace{X \cdot X \cdot \ldots \cdot X}_{n}$, $X^0 = \{\Lambda\}$.
- Пусть $X \subseteq A^*$. Итерация X есть

$$X^* = X^0 \cup X^1 \cup X^2 \cup \ldots, \quad \varnothing^* = \varnothing.$$

Особенности итерации

- $\varnothing^* = \varnothing$, $\{\Lambda\}^* = \{\Lambda\}$.
- Если $a \neq \Lambda$, $a \in X$, то $\Lambda, a, a^2, \ldots \in X^*$ и X^* — бесконечное множество.

Операция итерации множества

Теорема 6

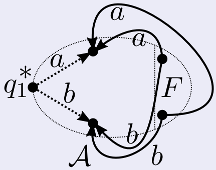
Класс конечно-автоматных множеств замкнут относительно операции итерации.

Доказательство

- Если $X = \varnothing$, то утверждение теоремы очевидно. Далее считаем $X \neq \varnothing$.
- Пусть $\mathcal{A} = (A, Q, f, q_1, F)$ — конечный автомат, $X = D(\mathcal{A})$.
- Строим недетерминированный конечный автомат $\mathcal{C} = (A,\ Q,\ h,\ q_1,\ F)$, допускающий множество $X^1 \cup X^2 \cup \dots$
- $h(a_i, q) = \begin{cases} \{f(a_i, q)\}, & q \in Q \setminus F, \\ \{f(a_i, q),\ f(a_i, q_1)\}, & q \in F. \end{cases}$

Операция итерации множества

Доказательство (продолжение)



- Автомат на состояниях Q распознаёт слово из X. Когда слово распознано, он может продолжить распознавать слово из X или начать новую итерацию и распознавать слово из X сначала.
- Если автомат $\mathcal{C}$ не использует новые «обратные» переходы, то он допускает слова из X. Если он использует их один раз, то допускает слова из X^2 и т. д. Поэтому $\mathcal{C}$ допускает $X^1 \cup X^2 \cup \dots$
- Очевидно, $\{\Lambda\}$ конечно-автоматно. Тогда X^* конечно-автоматно как объединение $X^1 \cup X^2 \cup \dots$ и $\{\Lambda\}$.

□

Промежуточные итоги

- Класс конечно-автоматных множеств можно охарактеризовать в терминах правоинвариантных отношений эквивалентности.
- Класс конечно-автоматных множеств замкнут относительно операций $\overline{}$, $\cup$, $\cap$, $\cdot$, $*$.

Регулярные выражения и множества

Определение

Пусть $A = \{a_1, \ldots, a_m\}$ — конечный алфавит.

- $\varnothing$, $\{\Lambda\}$, $\{a_i\}$, $i = \overline{1, m}$ — регулярные множества, обозначаемые регулярными выражениями $\varnothing$, Λ, a_i, $i = \overline{1, m}$ соответственно.
- Если X, Y — регулярные множества, обозначаемые регулярными выражениями α, β, то $X \cup Y$, $X \cdot Y$, X^* — регулярные множества, обозначаемые регулярными выражениями $(\alpha \cup \beta)$, $(\alpha \cdot \beta)$, $(\alpha)^*$.

Схема задания регулярных выражений и множеств

Регулярное выражение	Регулярное множество
$\varnothing$	пустое множество
Λ	$\{\Lambda\}$
a_i, $i = \overline{1, m}$	$\{a_i\}$, $i = \overline{1, m}$
α, β — регулярные выражения	X, Y — регулярные множества
α^*, $\alpha \cdot \beta$, $\alpha \cup \beta$	X^*, $X \cdot Y$, $X \cup Y$

Регулярные выражения и множества

Запись регулярных выражений

- Скобки можно опускать с учётом приоритета операций: $*$, $\cdot$, $\cup$ (перечислены в порядке убывания приоритета).
- Знак $\cdot$ можно опускать.

- Регулярное выражение является формулой, то есть строкой из символов, записанных по определённым правилам. Регулярное множество является подмножеством A^*.
- Регулярное выражение можно рассматривать как «шаблон», показывающий устройство слов в регулярном множестве.
- Например, выражение $1^*(010 \cup 0110)1^*$ задаёт слова, в которых сначала присутствует некоторое (возможно, нулевое) количество единиц, далее следует подслово 010 или 0110, после чего снова следует некоторое количество единиц.

Лекция 3
Теорема Клини. Детерминированные функции. Конечные автоматы-преобразователи.

Теорема Клини

Теорема 7 (Клини)

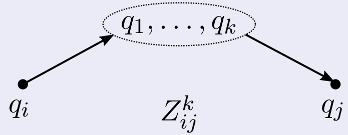
Класс конечно-автоматных множеств совпадает с классом регулярных множеств.

Доказательство

- $\supseteq$. Множества $\varnothing$, $\{\Lambda\}$, $\{a_i\}$, $i = \overline{1, m}$ конечно-автоматны. Ранее было доказано, что операции $\cup, \cdot, *$ сохраняют конечную автоматность множеств. Поэтому все регулярные множества конечно-автоматны.
- $\subseteq$. Пусть $\mathcal{A} = (A, Q, f, q_1, F)$ — произвольный конечный автомат. Будем доказывать, что множество $D(\mathcal{A})$ регулярно.
- Пусть $Q = \{q_1, \ldots, q_r\}$, $F = \{q_{j_1}, \ldots, q_{j_s}\}$.
Тогда $D(\mathcal{A}) = X_1 \cup \ldots \cup X_s$, где $X_l = D((A,\, Q,\, f,\, q_1,\, \{q_{j_l}\}))$.
- Достаточно доказать регулярность каждого множества X_l: тогда $D(\mathcal{A})$ будет регулярным как объединение регулярных множеств.

Теорема Клини

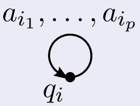
Доказательство (продолжение)



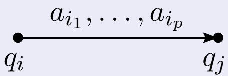
- Пусть $i, j = \overline{1, r}$, $k = \overline{0, r}$. Обозначим Z_{ij}^k множество слов, по которым автомат $\mathcal{A}$ переходит из q_i в q_j используя качестве промежуточных состояний только элементы $\{q_1, \ldots, q_k\}$.
- Если $k = 0$, то допускается только переход из q_i в q_j напрямую, без использования промежуточных состояний.
- Заметим, что $X_l = Z_{1j_l}^r$. Докажем, что все множества Z_{ij}^k регулярны, с помощью индукции по k.

Теорема Клини

Доказательство (продолжение)



Z^0_{ii}

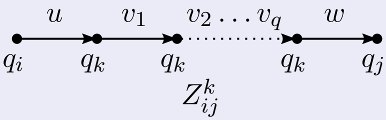


Z^0_{ij}

- Базис индукции: $k = 0$
  1. $i = j$. Если нет переходов из q_i в q_i, то $Z^0_{ii} = \{\Lambda\}$.
     Если есть переходы из q_i в q_i по символам $a_{i_1}, \dots, a_{i_p}$, то $Z^0_{ii} = \{\Lambda, a_{i_1}, \dots, a_{i_p}\}$. В обоих случаях множество регулярно.
  2. $i \neq j$. Если нет переходов из q_i в q_j, то $Z^0_{ij} = \varnothing$.
     Если есть переходы из q_i в q_j по символам $a_{i_1}, \dots, a_{i_p}$, то $Z^0_{ij} = \{a_{i_1}, \dots, a_{i_p}\}$. В обоих случаях множество регулярно.
- Предположим, что все множества Z^{k-1}_{ij} регулярны.
  Шаг индукции: докажем регулярность Z^k_{ij}.

Теорема Клини

Доказательство (продолжение)



- Пусть $a \in Z_{ij}^k \setminus Z_{ij}^{k-1}$. Тогда $a = uv_1 \dots v_q w$, где $q \geqslant 0$ и $u \in Z_{ik}^{k-1}$, $v_1, \dots, v_q \in Z_{kk}^{k-1}$, $w \in Z_{kj}^{k-1}$.
- Тогда $Z_{ij}^k = Z_{ij}^{k-1} \cup Z_{ik}^{k-1}(Z_{kk}^{k-1})^* Z_{kj}^{k-1}$.
- Поскольку множества Z_{ij}^{k-1}, Z_{ik}^{k-1}, Z_{kk}^{k-1}, Z_{kj}^{k-1} регулярны, множество Z_{ij}^k тоже регулярно.
- Получаем, что $X_l = Z_{1j_l}^r$ тоже регулярно, а значит и $D(\mathcal{A})$ регулярно. Таким образом, любое конечно-автоматное множество является регулярным.

$\square$

Регулярные выражения

Практическое использование

- Во многих текстовых редакторах и файловых менеджерах есть опция поиска/фильтра по регулярным выражениям.
- Эти регулярные выражения основаны на регулярных выражениях Клини, но в них добавлены дополнительные операции для сокращения записи.
- Например, `<[^<>]*>` ищет пару угловых скобок с произвольным текстом (не содержащим других угловых скобок) внутри.
- Существует стандартный язык регулярных выражений, который несложно изучить. Он описан, например, в документации языка Python [5] или на Википедии [6].
- Обработчики регулярных выражений иногда поддерживают возможности, выходящие за рамки возможностей регулярных выражений Клини. Но наиболее эффективно реализуемые возможности используют регулярные выражения Клини.

Регулярные выражения

Реализация в программах

- По любому регулярному выражению можно построить конечный автомат, который распознаёт слова, соответствующие данному регулярному выражению.
- Конечный автомат работает очень быстро: он проходит по символам текста только один раз, и для каждого символа совершает простую операцию изменения состояния.
- Память автомата конечна: она зависит только от регулярного выражения, но не от текста, по которому идёт поиск. Поэтому автомат может работать с очень большими текстами.
- Теорема Клини гарантирует, что всё, что может быть найдено быстрым поиском с помощью автомата, можно задать регулярными выражениями.

Детерминированные функции

Бесконечные последовательности

Пусть A — непустое множество.

- A^∞ — это множество счётно-бесконечных последовательностей вида $a_{i_1}a_{i_2}\ldots$, где $a_{i_n} \in A$ при $n \in \mathbb{N}$.
- Пусть $a = a_{i_1}a_{i_2}\ldots \in A^\infty$. Обозначим $a(t) = a_{i_t}$ при $t \in \mathbb{N}$.
- Для введения индексации с нуля пишем $a = a(0)a(1)\ldots \in A^\infty$.
- Конкатенация слова $u = u_1 \ldots u_k \in A^*$ и последовательности $a = a(1)a(2)\ldots \in A^\infty$ — это последовательность

  $$u * a = ua = u_1 \ldots u_k a(1)a(2)\ldots \in A^\infty.$$

  При этом для любого $a \in A^\infty$ определяем $\Lambda a = a$.
- Бесконечное повторение слова $u = u_1 \ldots u_k \in A^*$, $u \neq \Lambda$ есть

  $$u^\omega = u_1 \ldots u_k u_1 \ldots u_k \ldots \in A^\infty.$$

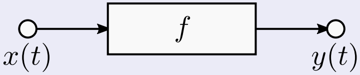
Детерминированные функции

Основные обозначения

- $E_2 = \{0, 1, \}, \quad E_2^n = \underbrace{E_2 \times \ldots \times E_2}_{n}$
- Мы будем рассматривать алфавит $A = E_2$ и бесконечные слова $a(1)a(2)\ldots \in E_2^\infty$, где $a(t) \in E_2$.
- Если $x = (x_1, \ldots, x_n) \in (E_2^\infty)^n$, то $x(t) = (x_1(t), \ldots, x_n(t)) \in E_2^n$.
- Будем рассматривать функции $y = f(x_1, \ldots, x_n)\colon (E_2^\infty)^n \to E_2^\infty$.
- P_2^∞ — множество всех функций $f\colon (E_2^\infty)^n \to E_2^\infty$ при $n \geqslant 1$.

Детерминированные функции

Содержательное понимание детерминированности



$t = 1, 2, \ldots$ — время

- Можно считать, что функция $y = f(x)$ над бесконечными словами действует не сразу, а растянуто во времени: в каждый момент $t \geqslant 1$ функция получает на вход символ $x(t)$ и выдаёт символ $y(t)$.
- Детерминированная функция «не может заглядывать в будущее»: её выход в момент t зависит только от входов $x(1), \ldots, x(t)$, которые были получены ранее, и не зависит от будущих входов $x(t+1), x(t+2), \ldots$

Детерминированные функции

Определение

- Функция $y = f(x_1, \ldots, x_n)\colon (E_2^\infty)^n \to E_2^\infty$ является детерминированной, если для каждого $t \geqslant 1$ существует такая булева функция $\varphi_t(x_1^1, \ldots, x_n^1, \ldots, x_1^t, \ldots, x_n^t)$, что

$$y(t) = \varphi_t(x_1(1), \ldots, x_n(1), \ldots, x_1(t), \ldots, x_n(t)).$$

- $P_{\text{д},2}$ — множество всех детерминированных функций на E_2^∞ (т. е. из P_2^∞).

Детерминированные функции

Примеры

Рассматриваем функции $f\colon E_2^\infty \to E_2^\infty, \quad f(x) = y$.

- f **детерминированная**:
  $$y(t) = \begin{cases} 0, & t = 1, \\ x(t-1) \oplus x(t). & t = \overline{2, \infty}. \end{cases}$$
- f **детерминированная**:
  $$y(t) = \begin{cases} 1, & \text{слово } x(1)\dots x(t) \text{ симметрично}, \\ 0 & \text{в ином случае}, \end{cases} \qquad t = \overline{1, \infty}.$$
- f **не детерминированная**:
  $y(t) = x(t+1)$.
- f **не детерминированная**:
  $$f(x) = \begin{cases} 0^\infty, & x = 0^\infty, \\ 1^\infty & \text{в ином случае}. \end{cases}$$

Конечные автоматы-преобразователи

Определение

Конечный автомат (преобразователь) — это $\mathcal{A} = (A, B, Q, F, G, q_1)$, где

- $A \neq \varnothing$ — входной алфавит,
- $B \neq \varnothing$ — выходной алфавит,
- $Q \neq \varnothing$ — множество состояний,
- $F\colon A \times Q \to B$ — функция выходов,
- $G\colon A \times Q \to Q$ — функция переходов,
- $q_1 \in Q$ — начальное состояние.

- В качестве алфавитов A, B мы будем рассматривать множества E_2 или E_2^n.

Конечные автоматы-преобразователи

Работа автомата

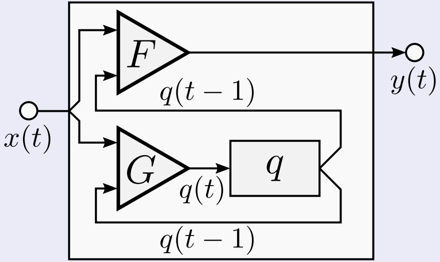
- На вход автомату подаётся бесконечное слово $x \in A^\infty$. На выходе получается бесконечное слово $y \in B^\infty$.
- Автомат работает в дискретном времени: $t = 1, 2, \ldots$ На каждом такте t автомату подаётся очередной символ $x(t)$.
- На каждом такте t автомат меняет своё состояние $q(t)$ и выдаёт символ выхода $y(t)$ согласно канонических уравнениям:

$$\begin{cases} y(t) = F(x(t), q(t-1)), \\ q(t) = G(x(t), q(t-1)), \\ q(0) = q_1. \end{cases}$$

- Автомат $\mathcal{A}$ реализует функцию $\varphi\colon A^\infty \to B^\infty$: $\varphi(x) = y$.

Конечные автоматы-преобразователи

Схема работы автомата



$t = 1, 2, \ldots$ — время

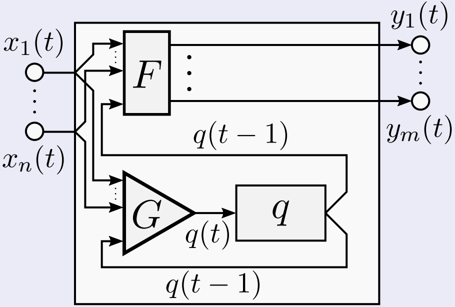
В начальный момент $q = q_1$

$$\begin{cases} y(t) = F(x(t),\, q(t-1)), \\ q(t) = G(x(t),\, q(t-1)), \\ q(0) = q_1. \end{cases}$$

Конечные автоматы-преобразователи

- Если $A = E_2^n$, то у автомата несколько входов $x_1, \dots, x_n \in E_2^\infty$.
- Если $B = E_2^m$, то у автомата несколько входов $y_1, \dots, y_m \in E_2^\infty$.

Автомат с несколькими входами и выходами



$t = 1, 2, \dots$ — время

В начальный момент $q = q_1$

$$x(t) = (x_1(t), \dots, x_n(t))$$
$$y(t) = (y_1(t), \dots, y_m(t))$$
$$\begin{cases} y(t) = F(x(t),\, q(t-1)), \\ q(t) = G(x(t),\, q(t-1)), \\ q(0) = q_1. \end{cases}$$

Конечные автоматы-преобразователи

Конечно-автоматные функции

- Функция $f\colon (E_2^\infty)^n \to E_2^\infty$ называется конечно-автоматной (ограниченно-детерминированной), если она реализуется некоторым автоматом с входным алфавитом E_2^n и выходным алфавитом E_2.
- $P_{\text{ка},2}$ — множество всех конечно-автоматных функций $f\colon (E_2^\infty)^n \to E_2^\infty,\ n \in \mathbb{N}$.
- Автомат с несколькими выходами реализует одновременно несколько функций из $P_{\text{ка},2}$, используя одни и те же состояния.
- Любая конечно-автоматная функция является детерминированной.

Конечные автоматы-преобразователи

Моделирование реальных систем

- Машина Тьюринга — это модель алгоритма: процесса, который по входным данным за конечное число шагов выдаёт результат.
- Автомат-преобразователь — это модель системы, которая работает неопределённо долгое время, в каждый момент получает определённый входные сигналы и выдаёт некоторые результаты.
- Процессор компьютера является автоматом-преобразователем:
  - Состояния (конечная память) — регистры.
  - Входные сигналы — данные из оперативной памяти и с внешних устройств (клавиатуры, мыши).
  - Выходные сигналы — данные для записи в оперативную память, позиция чтения/записи в оперативной памяти, вывод на внешние устройства (дисплей).
- Автомат — вычислительно слабое устройство, так как имеет лишь конечную память. Компьютер является универсальным за счёт наличия (условно) бесконечной оперативной памяти.

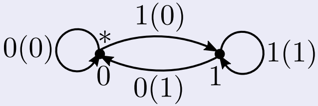
Конечно-автоматные функции

Диаграмма Мура

- Диаграмма Мура автомата-преобразователя строится аналогично диаграмме Мура автомата-распознавателя.
- На каждой дуге, помимо входа, подписывается (в скобках) выход $y(t)$. Заключительных состояний нет.

Пример

- $A = B = E_2$, $Q = \{q_1, q_2\}$, диаграмма Мура автомата:



- Реализуемая автоматом функция называется единичной задержкой. $\mathfrak{z}\colon E_2^\infty \to E_2^\infty$, $\mathfrak{z}(x) = 0x$.

Конечно-автоматные функции

Истинностные функции

- Пусть $\varphi\colon E_2^n \to E_2$ — булева функция. Ей соответствует истинностная функция $f_\varphi\colon (E_2^\infty)^n \to E_2^\infty$ такая, что

$$f_\varphi(x_1, \ldots, x_n) = \varphi(x_1(1), \ldots, x_n(1))\varphi(x_1(2), \ldots, x_n(2)) \ldots$$

- Иными словами, если $y = f_\varphi(x_1, \ldots, x_n)$, то $y(t) = \varphi(x_1(t), \ldots, x_n(t))$ при всех $t \geqslant 1$.
- Истинностная функция является конечно-автоматной и задаётся каноническими уравнениями:

$$\begin{cases} y(t) = \varphi(x_1(t), \ldots, x_n(t)), \\ q(t) = q_1, \\ q(0) = q_1. \end{cases}$$

Конечно-автоматные функции

Канонические уравнения в скалярной форме

- Пусть $|Q| = r$. Выбираем наименьшее l такое, что $2^l \geqslant r$.
- Кодируем состояния из Q векторами из E_2^l. Код состояния $q(t)$ обозначим $(q_1(t), \ldots, q_l(t)) \in E_2^l$. Код q_1 есть $(0, \ldots, 0)$.
- Тогда канонические уравнения можно переписать в скалярной форме:

$$\begin{cases} y_1(t) = f_1(x_1(t), \ldots, x_n(t),\ q_1(t-1), \ldots, q_l(t-1)), \\ \ldots \\ y_m(t) = f_m(x_1(t), \ldots, x_n(t),\ q_1(t-1), \ldots, q_l(t-1)), \\ q_1(t) = g_1(x_1(t), \ldots, x_n(t),\ q_1(t-1), \ldots, q_l(t-1)), \\ \ldots \\ q_l(t) = g_l(x_1(t), \ldots, x_n(t),\ q_1(t-1), \ldots, q_l(t-1)), \\ q_1(0) = \ldots = q_l(0) = 0. \end{cases}$$

Конечно-автоматные функции

Канонические уравнения в скалярной форме (продолжение)

- В полученных канонических уравнениях функции $f_1, \ldots, f_m, g_1, \ldots, g_l$ являются булевыми функциями.
- Эти функции определяются по исходным функциям F, G и по кодированию состояний.
- Если $r < 2^l$, то на части наборов функции f_i, g_i окажутся не определены. Мы доопределяем их произвольным образом.

Конечно-автоматные функции

Канонические уравнения в скалярной форме (пример 1)

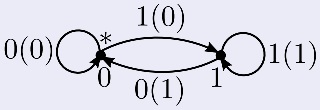
- $y = f_{\&}(x_1, x_2)$ — истинностная функция на E_2^{∞}, $y(t) = x_1(t)x_2(t)$.
- Её можно задать следующими каноническими уравнениями в скалярной форме:

$$\begin{cases} y(t) = x_1(t)x_2(t), \\ q(t) = 0, \\ q(0) = 0. \end{cases}$$

Конечно-автоматные функции

Канонические уравнения в скалярной форме (пример 2)

- $y = \mathfrak{z}(x)$ — единичная задержка.
- Диаграмма Мура:



- Канонические уравнения в скалярной форме:

$$\begin{cases} y(t) = q(t-1), \\ q(t) = x(t), \\ q(0) = 0. \end{cases}$$

Лекция 4
Операции суперпозиции и введения обратной связи. Полные системы конечно-автоматных функций. Машина Тьюринга.

Суперпозиция конечно-автоматных функций

Операция суперпозиции

- Операция суперпозиции включает в себя
  1. Подстановку функции вместо переменной: $f(g(x_1, \ldots, x_n), y_2, \ldots, y_m)$.
  2. Перестановку и отождествление переменных.
  3. Добавление и удаление фиктивных переменных.
- Операцию суперпозиции можно определить с помощью формул, как для булевых функций.
- Обычно рассматривается регулярная суперпозиция:

$$h(\bar{x}) = f(g_1(\bar{x}), \ldots, g_m(\bar{x})),$$

  где $\bar{x} = (x_1, \ldots, x_n)$.
- Если некоторое утверждение доказано для регулярной суперпозиции, обычно оно легко переносится и на общий случай.

Суперпозиция конечно-автоматных функций

Теорема 8

Класс $P_{\text{ка},2}$ замкнут относительно операции суперпозиции.

Доказательство

$$f(\bar{x}) = f_0(f_1(\bar{x}), \ldots, f_m(\bar{x}))$$

- Пусть все функции $f_0, f_1, \ldots, f_m$ конечно-автоматны:

$$f_0\colon \begin{cases} y(t) = F_0(y_1(t), \ldots, y_m(t),\, q_0(t-1)), \\ q_0(t) = G_0(y_1(t), \ldots, y_m(t),\, q_0(t-1)), \\ q_0(0) = q_0'; \end{cases}$$

$$f_i\colon \begin{cases} y(t) = F_i(\bar{x}(t),\, q_i(t-1)), \\ q_i(t) = G_i(\bar{x}(t),\, q_i(t-1)), \qquad i = \overline{1,m}. \\ q_i(0) = q_i', \end{cases}$$

Суперпозиция конечно-автоматных функций

Доказательство (продолжение)

- Составим канонические уравнения для суперпозиции:

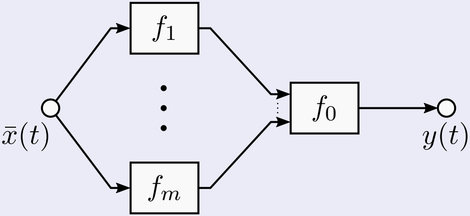
$$\begin{cases} y(t) = F_0(F_1(\bar{x}(t), q_1(t-1)), \ldots, F_m(\bar{x}(t), q_m(t-1)),\ q_0(t-1)), \\ q_0(t) = G_0(F_1(\bar{x}(t), q_1(t-1)), \ldots, F_m(\bar{x}(t), q_m(t-1)),\ q_0(t-1)), \\ q_1(t) = G_1(\bar{x}(t), q_1(t-1)), \\ \ldots \\ q_m(t) = G_m(\bar{x}(t), q_m(t-1)), \\ q_i(0) = q_i', \quad i = \overline{1, m}. \end{cases}$$

- Если обозначить $q(t) = (q_0(t), \ldots, q_m(t))$, то эти уравнения можно переписать в стандартной форме.
- Получаем, что функция f конечно-автоматна.

□

Суперпозиция конечно-автоматных функций

Иллюстрация



$t = 1, 2, \ldots$ — время

- Если автомат, реализующий f_i, имел r_i состояний $(i = \overline{0, m})$, то автомат для суперпозиции будет иметь $r_0 \cdot r_1 \cdot \ldots \cdot r_m$ состояний.
- Некоторые из этих состояний могут оказаться недостижимыми или эквивалентными, но бывают примеры функций, для суперпозиции которых число состояний нельзя уменьшить.

Операция введения обратной связи

Определение

Детерминированная функция $y = f(x_1, \ldots, x_n)$ зависит с запаздыванием от x_i, если $y(t)$ не зависит от $x_i(t)$ при любом $t \geqslant 1$.

- При зависимости с запаздыванием $y(t)$ может зависеть от $x_i(1), \ldots, x_i(t-1)$, а также от $x_j(1), \ldots, x_j(t)$ при $j \neq i$.

Пример

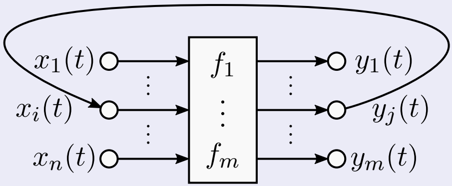
- Единичная задержка $\mathfrak{z}\colon E_2^\infty \to E_2^\infty$, $\mathfrak{z}(x) = 0x$.

$$\begin{cases} y(t) = q(t-1), \\ q(t) = x(t), \\ q(0) = 0. \end{cases}$$

- При любом $t \geqslant 2$ верно $(\mathfrak{z}(x))(t) = x(t-1)$. Она зависит с запаздыванием от x.

Операция введения обратной связи

Иллюстрация



- $y_1, \ldots, y_m$ — выходы, на которых реализуются детерминированные функции $f_1, \ldots, f_m$.
- y_j (т. е. f_j) зависит с запаздыванием от x_i.
- На рисунке изображено введение обратной связи по переменным x_i, y_j.
- У получившейся конструкции вход x_i и выход y_j пропадают. Теперь она реализует $m-1$ функцию от $n-1$ переменных.

Операция введения обратной связи

Работа набора функций, полученного в результате обратной связи

- $y_j(t)$ не зависит от $x_i(t)$. Мы хотим выразить все $y_k(t)$, $k \neq j$ через $x_k(1), \dots x_k(t)$, $k \neq i$ при всех t.
- В начале $y_j(1) = \varphi_1^j(x_1(1), \dots, x_{i-1}(1), x_{i+1}(1), \dots, x_n(1))$.
- Для получения $y_k(1)$, $k \neq j$ подставляем в их выражение через $x_k(1)$, $k = \overline{1,n}$ вместо $x_i(1)$ выражение для $y_j(1)$.
- Пусть для момента времени $t-1$ получены $y_k(1), \dots, y_k(t-1)$ при всех $k = \overline{1,m}$ (зависят от $x_k(1), \dots, x_k(t-1)$, $k \neq i$).
- Тогда $y_j(t)$ определяется через $x_k(1), \dots, x_k(t)$ при $k \neq i$ и через $x_i(1) = y_j(1), \dots, x_i(t-1) = y_j(t-1)$.
- Остальные $y_k(t)$ определяются через $x_k(1), \dots, x_k(t)$, $k = \overline{1,n}$, где вместо $x_i(1), \dots, x_i(t)$ подставлены выражения $y_j(1), \dots, y_j(t)$.
- Таким образом, в полученном наборе все функции детерминированные.

Операция введения обратной связи

Теорема 9

Класс $P_{\text{ка},2}$ замкнут относительно операции введения обратной связи.

Доказательство

- Имеем набор функций из $P_{\text{ка},2}$ с каноническими уравнениями:

$$\begin{cases} y_1(t) = F_1(x_1(t), \ldots, x_n(t),\ q(t-1)), \\ \ldots \\ y_m(t) = F_m(x_1(t), \ldots, x_n(t),\ q(t-1)), \\ q(t) = G(x_1(t), \ldots, x_n(t),\ q(t-1)), \\ q(0) = q_0. \end{cases}$$

- Пусть выход y_j зависит от входа x_i с запаздыванием:
$y_j(t) = F_j(x_1(t), \ldots, x_{i-1}(t), x_{i+1}(t) \ldots, x_n(t),\ q(t-1))$.

Операция введения обратной связи

Доказательство (продолжение)

- Применим операцию обратной связи по переменным x_i, y_j: подставим выражение для $y_j(t)$ вместо $x_i(t)$
- Функции из полученного набора конечно-автоматны, их канонические уравнения:

$$
\begin{cases}
y_k(t) = F_k(x_1(t), \dots, x_{i-1}(t), \\
\qquad\qquad F_j(x_1(t), \dots, x_{i-1}(t), x_{i+1}(t) \dots, x_n(t),\ q(t-1)), \\
\qquad\qquad x_{i+1}(t) \dots, x_n(t),\ q(t-1)), \quad k \neq i,\ k = \overline{1, m}, \\
q(t) = G(x_1(t), \dots, x_{i-1}(t), \\
\qquad\qquad F_j(x_1(t), \dots, x_{i-1}(t), x_{i+1}(t) \dots, x_n(t),\ q(t-1)), \\
\qquad\qquad x_{i+1}(t) \dots, x_n(t),\ q(t-1)), \\
q(0) = q_0.
\end{cases}
$$

$\square$

Полная система конечно-автоматных функций

Истинностные функции

- Пусть $\varphi\colon E_2^n \to E_2$ — булева функция. Ей соответствует истинностная функция $f_\varphi\colon (E_2^\infty)^n \to E_2^\infty$ такая, что если $y = f_\varphi(x_1, \ldots, x_n)$, то $y(t) = \varphi(x_1(t), \ldots, x_n(t))$ при всех $t \geqslant 1$.

Задержка

- Единичная задержка $\mathfrak{z}\colon E_2^\infty \to E_2^\infty$, $\mathfrak{z}(x) = 0x$ реализуется автоматом с 2 состояниями ($Q = E_2$):

$$\begin{cases} y(t) = q(t-1), \\ q(t) = x(t), \\ q(0) = 0. \end{cases}$$

Полная система конечно-автоматных функций

Исходные функции

- Полная в P_2:
$$\{\&, \vee, \neg\}.$$
- Рассмотрим систему конечно-автоматных функций, состоящую из истинностных функций и единичной задержки:
$$\{f_{\&},\ f_{\vee},\ f_{\neg},\ \mathfrak{z}\}.$$

Полная система конечно-автоматных функций

Теорема 10

Система $\{f_{\&},\ f_{\vee},\ f_{\neg},\ \mathfrak{z}\}$ *полна в классе* $P_{\text{ка},2}$ *относительно операций суперпозиции и введения обратной связи.*

Доказательство

- Пусть функция входит в класс $P_{\text{ка},2}$. Тогда она реализуется системой канонических уравнений:

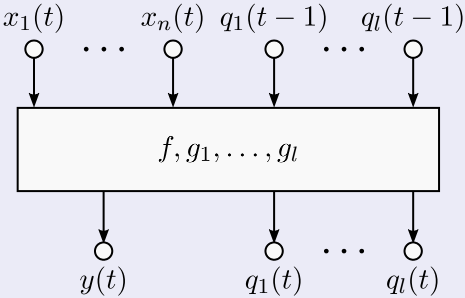
$$\begin{cases} y(t) = f(x_1(t),\ \ldots,\ x_n(t),\ q_1(t-1),\ \ldots,\ q_l(t-1)), \\ q_1(t) = g_1(x_1(t),\ \ldots,\ x_n(t),\ q_1(t-1),\ \ldots,\ q_l(t-1)), \\ \ldots \\ q_l(t) = g_l(x_1(t),\ \ldots,\ x_n(t),\ q_1(t-1),\ \ldots,\ q_l(t-1)), \\ q_1(0) = \ldots = q_l(0) = 0. \end{cases}$$

- Здесь $f, g_1, \ldots, g_l \in P_2$.

Полная система конечно-автоматных функций

Доказательство (продолжение)

- Моделируем работу функций $f, g_1, \dots, g_l$ с помощью суперпозиции истинностных функций:

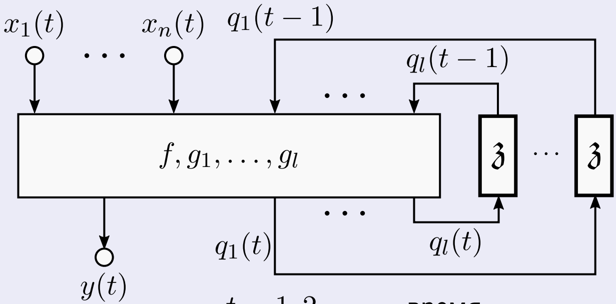


$t = 1, 2, \dots$ — время

Полная система конечно-автоматных функций

Доказательство (продолжение)

- Соединяем выходы $q_i(t)$ со входами $q_i(t-1)$ через задержки. Используем для этого операцию обратной связи. Зависимость с запаздыванием обеспечивается задержками.



$t = 1, 2, \ldots$ — время

- Таким образом, построена нужная функция.

$\square$

Базис из одной конечно-автоматной функции

Шефферовы функции

- Штрих Шеффера: $x \mid y = \overline{xy} = \overline{x} \vee \overline{y}$.
- $[x \mid y] = P_2$, так как $\neg x = x \mid x,\ xy = \overline{x \mid y},\ x \vee y = \overline{x} \mid \overline{y}$.

Теорема 11

В классе $P_{\text{ка},2}$ существует полная система, состоящая из одной функции.

Доказательство

- Рассмотрим функцию $F(x_1, x_2, x_3, x_4) = ((x_1 \oplus x_3)x_4 \oplus x_3) \mid x_2$.
- $F(x_1, x_2, x_1, x_4) = x_1 \mid x_2$.
- $\overline{F}(1, 1, 0, x_4) = x_4$.

Базис из одной конечно-автоматной функции

Доказательство (продолжение)

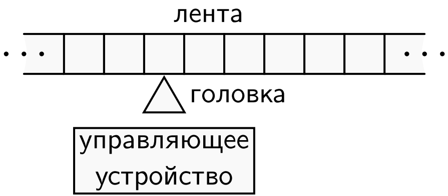
- Теперь рассмотрим функцию $f(x_1, x_2, x_3, x_4) = f_F(x_1,\ x_2,\ x_3,\ \mathfrak{z}(x_4))$ и докажем, что $[f] = P_{\text{ка},2}$.
- $f_|(x_1, x_2) = f(x_1, x_2, x_1, x_4)$. С помощью $f_|$ получаем $f_\neg(x),\ f_0(x),\ f_1(x)$.
- $\mathfrak{z}(x) = f_\neg(f(f_1(x),\ f_1(x),\ f_0(x),\ x))$.
- Получили систему из истинностных функций, соответствующих полной в P_2 системе, и функцию задержки. Значит, система $\{f\}$ полна в $P_{\text{ка},2}$.

$\square$

Машина Тьюринга

- В середине 1930-х годов математикам удалось формализовать понятие алгоритма.
- Разными математиками практически в одно время было предложено несколько формализаций, основанных на разных идеях.
- Одна из них (предложенная независимо Тьюрингом и Постом) основана на представлении алгоритма как программы для абстрактного вычислительного устройства определённого вида.
- Эта формализация и по сей день является одной из самых используемых и пригодных для анализа алгоритмов.

Машина Тьюринга



- Машина Тьюринга состоит из бесконечной в обе стороны ленты, разделённой на ячейки, считывающе-записывающей головки и управляющего устройства.
- Ячейки ленты содержат символы — это бесконечная память машины.
- Головка в каждый момент обозревает какую-то ячейку и может двигаться по ленте влево и вправо.
- Управляющее устройство содержит программу, которая управляет поведением головки.

Машина Тьюринга

Определение

Машина Тьюринга $\mathcal{M}$ — это набор (A, Q, f, q_1, q_0), где

- $A = \{a_0, \ldots, a_k\}$, $k \geqslant 1$ — рабочий алфавит. a_0 — пустой символ.
- $Q \neq \varnothing$ — множество состояний.
- $q_1 \in Q$ — начальное состояние.
- $q_0 \in Q$, $q_0 \neq q_1$ — заключительное состояние.
- $f\colon A \times Q \to A \times \{L, R, S\} \times Q$ — программа машины.

- Программу машины можно считать набором команд вида $a_i q_j \to a_r D q_s$, $j \neq 0$. В программе имеется ровно одна команда с каждой допустимой левой частью.

Машина Тьюринга

Работа машины

- В каждой ячейке ленты записан символ алфавита A. Ячейки с символом a_0 считаем пустыми.
- В каждый момент времени головка машины обозревает некоторую ячейку ленты и машина находится в одном из состояний Q.
- В начальный момент времени машина находится в состоянии q_1.
- В каждый момент времени машина считывает символ a из обозреваемой головкой ячейки. По этому символу и текущему состоянию q_i машина с помощью своей программы получает набор $f(a, q_i) = (b, D, q_j) \in A \times \{R, S, L\} \times Q$.
- После этого машина записывает в текущую ячейку символ b, передвигает головку на другую ячейку ленты (L — на 1 влево, R — на 1 вправо, S — не передвигает) и переходит в состояние q_j.
- Машина останавливается при переходе в состояние q_0. Если этого не происходит, машина работает бесконечно.

Машина Тьюринга

Запись программ машин Тьюринга

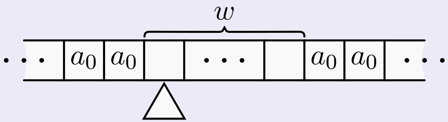
- Записываем программы машин Тьюринга в виде таблиц:

	1^*	2	3	4
0	$0R1$	$0L3$		$0Rq_0$
1	$1R2$	$1R2$	$0L4$	$1L1$

- Состояния (кроме q_0) обозначаем цифрами или другим удобным образом. Начальное состояние (если оно не обозначено q_1) отмечаем звёздочкой.
- Ячейки таблицы, которые не могут выполниться, оставляем пустыми. Для определённости считаем, что указанная там команда не меняет символ в ячейке, не двигает головку и переходит в q_0.

Машина Тьюринга

Выполнение преобразований в общем случае



- Считаем, что в начальный момент на ленте находится слово w в алфавите $A \setminus \{a_0\}$, а все остальные символы пусты. Головка машины обозревает самый левый непустой символ.
- Машина работает в дискретном времени согласно программе.
- Если машина остановилась, то результат её работы — это участок ленты от самого левого непустого символа до самого правого. В противном случае результат не определён.

Лекция 5
Вычислимые функции. Композиция и итерация машин Тьюринга. Вычислимость простейших функций.

Вычислимые функции

Базовые понятия

- $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$ — множество натуральных чисел с добавлением нуля.
- $f\colon \mathbb{N}_0^n \to \mathbb{N}_0$ — функции натурального аргумента.
- Мы расширяем понятие функции до частичной функции: частичная функция может быть определена не на всех элементах базового множества.

Примеры

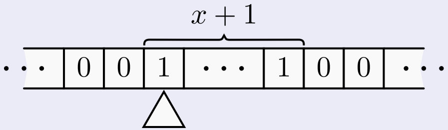
- $x + y$ — всюду определённая функция.
- Функция $x - y$ определена при $x \geqslant y$ и не определена при $x < y$.
- Усечённая разность $x \mathbin{\dot{-}} y = \begin{cases} x - y, & x \geqslant y, \\ 0, & x < y. \end{cases}$ всюду определена.
- $x/2$ определена только при чётных x, а $\lfloor x/2 \rfloor$ всюду определена.

Вычислимые функции

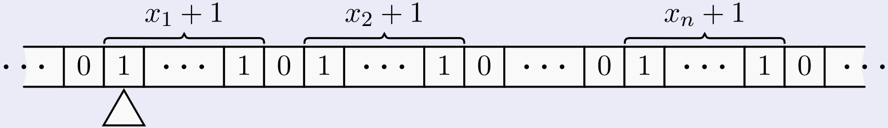
- Будем использовать машины Тьюринга с алфавитом $A = \{0, 1, a_2, \ldots, a_k\}$. При этом считаем $a_0 = 0$ — пустой символ.
- Для записи входных значений на ленте машины Тьюринга будем использовать основной код.

Основной код

- Кодируем число $x \in \mathbb{N}_0$ в виде 1^{x+1}.



- Кодируем набор $(x_1, \ldots, x_k)$ из $\mathbb{N}_0^n$ в виде $1^{x_1+1}01^{x_2+1}0\ldots01^{x_n+1}$.



Вычислимые функции

Определение

Машина Тьюринга $\mathcal{M}$ **вычисляет** частичную функцию $f(\bar{x})$, если, начиная работу на первой единице основного кода набора $\bar{x}$ (остальные символы ленты — нули) в состоянии q_1, машина:

1. Если $f(\bar{x})$ определено, то $\mathcal{M}$ через конечное число тактов останавливается, и в этот момент на ленте представлено значение $f(\bar{x})$ в основном коде (остальные символы ленты — нули, головка может находиться где угодно).
2. Если $f(\bar{x})$ не определено, то $\mathcal{M}$ либо не останавливается, либо останавливается, но на ленте не оказывается основной код числа из $\mathbb{N}_0$ (либо все символы ленты — нули, либо на ленте несколько массивов из единиц).

Вычислимые функции

Определение

Машина Тьюринга $\mathcal{M}$ правильно вычисляет частичную функцию $f(\bar{x})$, если, начиная работу на первой единице основного кода набора $\bar{x}$ (остальные символы ленты — нули) в состоянии q_1, машина:

1. Если $f(\bar{x})$ определено, то $\mathcal{M}$ через конечное число тактов останавливается, и в этот момент на ленте представлено значение $f(\bar{x})$ в основном коде (остальные символы ленты — нули), причём головка машина находится на первом символе этого основного кода.
2. Если $f(\bar{x})$ не определено, то $\mathcal{M}$ не останавливается.

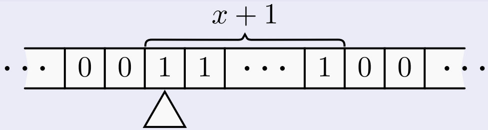
- Если машина вычисляет некоторую функцию, то можно изменить её так, чтобы она правильно вычисляла эту функцию.
- В дальнейшем мы будем не будем ссылаться на простое вычисление функций, а будем использовать только правильные вычисления.

Вычислимые функции

Определение

Частичная функция называется **вычислимой** (на машинах Тьюринга), если существует машина Тьюринга, правильно вычисляющая эту функцию.

Примеры



	1^*
0	$1Sq_0$
1	$1L1$

$x+1$

	1^*
0	$1Sq_0$
1	$0R1$

0

	1^*	2
0		$1Sq_0$
1	$0R2$	$1Sq_0$

$x \dot{-} 1$

Композиция машин Тьюринга

Безусловная композиция

- Имеем две машины Тьюринга $\mathcal{M}_1 = (A, Q_1, f_1, q_1', q_0')$ и $\mathcal{M}_2 = (A, Q_2, f_2, q_1'', q_0'')$. Хотим построить машину $\mathcal{M}$, которая сначала работает как машина $\mathcal{M}_1$, а когда $\mathcal{M}_1$ останавливается, на полученном содержимом ленты запускается работа $\mathcal{M}_2$, и результат её работы будет результатом машины $\mathcal{M}$.

$$\mathcal{M}_1, \mathcal{M}_2 \to \mathcal{M}$$

- Считаем, что $Q_1 \cap Q_2 = \varnothing$. В качестве множества состояний $\mathcal{M}$ выбираем $Q_1 \cup Q_2$. Начальное состояние q_1', заключительное — q_0''.
- В программе $\mathcal{M}_1$ заменяем переходы к q_0' на переходы к q_1''.
- Объединяем программы $\mathcal{M}_1$ и $\mathcal{M}_2$, получаем программу $\mathcal{M}$.

Композиция машин Тьюринга

Иллюстрация

- Имеем

$$\begin{array}{c||c|c} & q_1' & \dots \\ \hline 0 & \dots & \dots \\ 1 & \dots & \dots \end{array} \quad q_0' \qquad\qquad \begin{array}{c||c|c} & q_1'' & \dots \\ \hline 0 & \dots & \dots \\ 1 & \dots & \dots \end{array} \quad q_0''$$

$$\mathcal{M}_1 \qquad\qquad\qquad \mathcal{M}_2$$

- В $\mathcal{M}_1$ заменяем ячейки: $\boxed{a_r D q_0'} \to \boxed{a_r D q_1''}$.
- Объединяем программы $\mathcal{M}_1$ и $\mathcal{M}_2$. Начальное состояние q_1', заключительное q_0''.

Композиция машин Тьюринга

Условная композиция

- Имеем две машины Тьюринга $\mathcal{M}_1 = (A, Q_1, f_1, q_1', q_0')$ и $\mathcal{M}_2 = (A, Q_2, f_2, q_1'', q_0'')$.
- Хотим построить машину $\mathcal{M}$, которая в некоторых случаях работает как машина $\mathcal{M}_1$, а в некоторых — как безусловная композиция $\mathcal{M}_1$ и $\mathcal{M}_2$.
- Строим аналогично безусловной композиции, но заменяем q_0' на q_1'' не во всех ячейках $\mathcal{M}_1$, а только в тех, где нужно запустить работу машины $\mathcal{M}_2$. В остальных ячейках заменяем q_0' на заключительное состояние q_0''.

Композиция машин Тьюринга

Условная композиция трёх машин

- Имеем три машины Тьюринга $\mathcal{M}_1 = (A, Q_1, f_1, q_1', q_0')$, $\mathcal{M}_2 = (A, Q_2, f_2, q_1'', q_0'')$, $\mathcal{M}_3 = (A, Q_3, f_3, q_1''', q_0''')$.
- Хотим построить машину $\mathcal{M}$, которая в некоторых случаях работает как безусловная композиция машин $\mathcal{M}_1$ и $\mathcal{M}_2$, а в некоторых — как безусловная композиция $\mathcal{M}_1$ и $\mathcal{M}_3$.

$$\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3 \to \mathcal{M}$$

- Строим аналогично безусловной композиции, но объединяем программы всех машин и заменяем q_0' на q_1'' или q_1''' в зависимости от того, какую машину нужно запустить при попадании управления в эту ячейку программы.
- Заключительным выбираем состояние q_0'', и у машины $\mathcal{M}_3$ в программе заменяем состояние q_0''' на q_0''.

Композиция машин Тьюринга

Пример

- $\mathrm{rm}(x, y)$ — остаток от деления x на y (0 при $y = 0$).
- $\mathrm{rm}(x,2) = \begin{cases} 0, & x \text{ чётно}, \\ 1, & x \text{ нечётно}. \end{cases}$
- $\mathcal{M}_1$:

	11^*	12
0	$0S{\color{red}q_0'}$	$0S\mathbf{q_0'}$
1	$0R12$	$0R11$

$\mathcal{M}_2$:

	21^*
0	$1Sq_0''$
1	

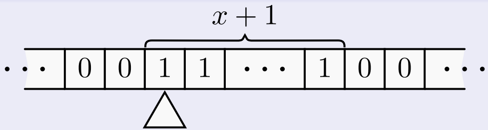
$\mathcal{M}_3$:

	31^*	32
0	$1R32$	$1Lq_0'''$
1		

- Условная композиция этих машин вычисляет функцию $\mathrm{rm}(x, 2)$: выделенное жирным состояние нужно заменить на 21, а выделенное красным — на 31. Состояния q_0'', q_0''' объединяем в общее заключительное состояние.

Композиция машин Тьюринга

Пример (продолжение)



	11^*	12	21	31	32
0	$0S31$	$0S\mathbf{21}$	$1Sq_0$	$1R32$	$1Lq_0$
1	$0R12$	$0R11$			

$\mathrm{rm}(x, 2)$

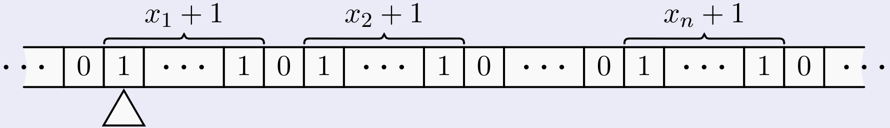
Итерация машин Тьюринга

Итерация

- Имеем машину Тьюринга $\mathcal{M} = (A, Q, f, q_1, q_0)$. Хотим построить машину $\mathcal{M}'$, которая выполняет работу машины $\mathcal{M}$ несколько раз (каждый раз применяя её к результату работы предыдущей машины). Цикл завершается при выполнении некоторого условия.
- В программе машины выделяем клетки с переходами в заключительное состояние, после которых нужно запускать машину $\mathcal{M}$ снова. Заменяем в них состояние q_0 на q_1.
- $\boxed{a_r D q_0} \to \boxed{a_r D q_1}$

Вычислимость простейших функций

Селекторные функции



- $I_m^n(x_1, \ldots, x_n) = x_m,\ m \in \{1, \ldots, m\},\ n \in \mathbb{N}$ — селекторные функции.

	1^*	2	$\ldots$	$m-1$	m
0	$0R2$	$0R3$	$\ldots$	$0Rm$	$0R(m+1)$
1	$0R1$	$0R2$	$\ldots$	$0R(m-1)$	$1Rm$

	$m+1$	$\ldots$	n	$n+1$	$n+2$
0	$0R(m+2)$	$\ldots$	$0R(n+1)$	$0L(n+1)$	$0Rq_0$
1	$0R(m+1)$	$\ldots$	$0Rn$	$1L(n+2)$	$1L(n+2)$

Вычислимость простейших функций

Селекторные функции

- В примерах были построены программы для правильного вычисления следующих простейших функций:
  1. Константа 0;
  2. Функция $x + 1$;
  3. Селекторные функции $I_m^n(x_1, \ldots, x_n) = x_m,\ m \in \{1, \ldots, m\},\ n \in \mathbb{N}$.
- Кроме того, были построены программы для правильного вычисления функций $x \dot{-} 1$ и $\mathrm{rm}(x, 2)$.
- Таким образом, все перечисленные функции являются вычислимыми.

Лекция 6
Моделирование машин Тьюринга. Механизм дорожек.
Универсальные функции.

Моделирование машин Тьюринга

- Мы рассмотрим моделирование машин Тьюринга, работающих в алфавите $\{0, 1, a_2, \ldots, a_k\}$, машинами Тьюринга, работающими в алфавите $\{0, 1\}$.

Теорема 1

При любом $k \geqslant 2$ классы функций, вычислимых машинами Тьюринга в алфавитах $\{0, 1, a_2, \ldots, a_k\}$ и $\{0, 1\}$, совпадают.

Доказательство

- $\supseteq$. Если функция вычислима на машине Тьюринга с алфавитом $\{0, 1\}$, то она вычислима и на машине Тьюринга с алфавитом $\{0, 1, a_2, \ldots, a_k\}$ (дополнительные символы в вычислении можно не использовать).

Моделирование машин Тьюринга

Доказательство (продолжение)

- $\subseteq$. Выберем такое l, что $2^l \geqslant k+1$. Кодируем все символы $\{0, 1, a_2, \ldots, a_k\}$ наборами из l нулей и единиц.
- 0 кодируем в виде 0^l, а 1 — в виде 1^l. Остальные символы кодируем произвольно.
- Пусть $\mathcal{M}$ — машина Тьюринга, работающая в алфавите $\{0, 1, a_2, \ldots, a_k\}$ и правильно вычисляющая некоторую функцию $f(x_1, \ldots, x_n)$.
- Отметим, в начале вычисления и в конце вычисления на ленте этой машины находятся только нули и единицы. Остальные символы могут появляться только на промежуточных шагах.
- Строим машину $\mathcal{M}'$ в алфавите $\{0, 1\}$, моделирующую машину $\mathcal{M}$ и правильно вычисляющую функцию $f(x_1, \ldots, x_n)$.

Моделирование машин Тьюринга

Доказательство (продолжение)

- Моделирование проходит в 3 этапа (машины $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$):
  1. Все единицы и нули на ленте заменяются своими кодами: вход «растягивается» в l раз.
  2. Моделирующая машина «воспроизводит» на ленте работу машины $\mathcal{M}$, обрабатывая коды символов, вместо самих символов.
  3. Получив результат, заменяем коды единиц на единицы: выход «сжимается» в l раз.
- Сперва рассмотрим этап 2. Пусть $Q = \{q_0, \ldots, q_m\}$ — множество состояний исходной машины $\mathcal{M}$.

Моделирование машин Тьюринга

Доказательство (продолжение)

- Машина $\mathcal{M}_2$ будет иметь 3 группы состояний и команд.
- Первая группа — чтение кода символа.
  - Состояния: $[b_1 \dots b_p, j],\ b_1, \dots, b_p \in \{0, 1\},\ p = \overline{1, l},\ j = \overline{1, m}$.
  - Команды: $bq_j \to bR[b, j]$,
    $b[b_1 \dots b_p, j] \to bR[b_1 \dots b_p b, j],\ p = \overline{1,\ l-2}$,
    $b[b_1 \dots b_{l-1}, j] \to bS[b_1 \dots b_{l-1} b, j] \qquad (b \in \{0, 1\})$.
- Первая группа команд обеспечивает чтение кода очередного символа за l тактов. Этот код, а также номер состояния машины в начале чтения, сохраняются с помощью перехода машины в специальные состояния.

Моделирование машин Тьюринга

Доказательство (продолжение)

- Вторая группа — запись кода нового символа на ленту.
  - Пусть $a_i q_j \to a_r D q_s$ — команда машины $\mathcal{M}$. Пусть $b_1 \dots b_l$ — код a_i, а $c_1 \dots c_l$ — код a_r. Для каждой такой команды машина $\mathcal{M}_2$ будет иметь указанные ниже состояния и команды.
  - Состояния: $[i, j, p]$, $p = \overline{1, l-1}$, $i = \overline{0, k}$, $j = \overline{1, m}$;
    Состояние кодирует текущую выполняемую команду (i, j) и номер p текущего записываемого элемента кода символа.
  - Команды: $b[b_1 \dots b_l, j] \to c_l L[i, j, l-1]$,
    $b[i, j, p] \to c_p L[i, j, p-1]$, $p = \overline{2, l-1}$,
    $b[i, j, 1] \to c_1 S\{D, s\}$ $\qquad (b \in \{0, 1\})$.
- Вторая группа команд обеспечивает запись на ленту кода нового символа в соответствии с командой машины $\mathcal{M}$. Она делает это за l тактов, двигая головку справа налево. В конце она приходит в состояние, в котором «записывается» тип движения D и номер s текущей команды машины $\mathcal{M}$.

Моделирование машин Тьюринга

Доказательство (продолжение)

- Третья группа — движение головки машины $\mathcal{M}$.
  - Состояния: $\{D, s\}$, $D \in \{R, L, S\}$, $s = \overline{0, m}$, $\{L, s, p\}$, $\{R, s, p\}$, $p = \overline{1, l}$, $s = \overline{0, m}$.
  - Команды: $b\{S, s\} \to bSq_s$,

$$\begin{aligned}
&b\{L, s\} \to bL\{L, s, 1\},\\
&b\{L, s, p\} \to bL\{L, s, p+1\},\ p = \overline{1, l-1},\\
&b\{L, s, l\} \to bSq_s\\
&b\{R, s\} \to bR\{R, s, 1\},\\
&b\{R, s, p\} \to bR\{R, s, p+1\},\ p = \overline{1, l-1},\\
&b\{R, s, l\} \to bSq_s \qquad\qquad (b \in \{0, 1\}).
\end{aligned}$$

- Третья группа команд совершает движение головки в нужном направлении. Для этого нужно просто остаться на месте, либо сдвинуться l раз влево или вправо. После этого машина переходит в состояние, в котором заново начнётся чтение символа кода.

Моделирование машин Тьюринга

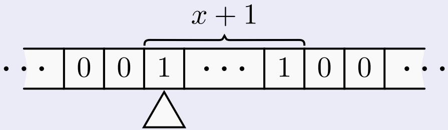
Доказательство (продолжение)

- Таким образом, на втором этапе машина $\mathcal{M}_2$ совершает те же преобразования над кодами символов на ленте, что машина $\mathcal{M}$ совершает над самими символами. Каждая команда машины $\mathcal{M}$ «выполняется» машиной $\mathcal{M}_2$ за $3(l+1)$ тактов.
- Отметим, что машина $\mathcal{M}_2$ также содержит во множестве состояний состояния $\{q_0, \ldots, q_m\}$. В них она попадает в промежутках между итерациями. Начальное состояние q_1, заключительное — q_0.

Моделирование машин Тьюринга

Доказательство (продолжение)

- Теперь рассмотрим этап 1. Выпишем программу машины $\mathcal{M}_1$ для случая функции от одной переменной.

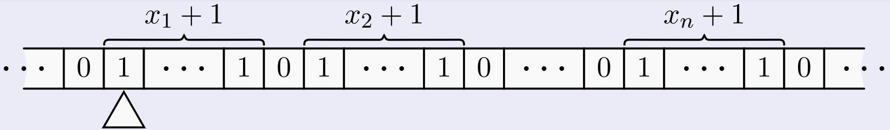


	1^*	2	$[3,1]$	$[3,2]$	$\dots$	$[3,l]$	4	5
0	$0Rq_0$	$0R[3,1]$	$1R[3,2]$	$1R[3,3]$	$\dots$	$1L4$	$0L5$	$0R1$
1	$0R2$	$1R2$	$1R[3,1]$				$1L4$	$1L5$

- Машина стирает единицу в начале, движет головку вправо и записывает там l единиц, после чего возвращает головку в начало. Так продолжается, пока входное слово не кончится.

Моделирование машин Тьюринга

Доказательство (продолжение)



- В общем случае машина действует аналогично, но во время прохода по слову ей нужно «пропускать» нужное количество нулей (разное при обработке разных входных чисел), а также «отлавливать» моменты обработки первого символа очередного слова, чтобы оставлять l разделительных нулей между массивами из единиц.
- Машина $\mathcal{M}_1$ будет получаться как безусловная композиция машин $\mathcal{M}_1^0, \ldots, \mathcal{M}_1^{n-1}$, каждая из которых обрабатывает один вход. Программа машины $\mathcal{M}_1^k$ в общем виде для $k \geqslant 1$ приведена на следующем слайде.

Моделирование машин Тьюринга

Доказательство (продолжение)

	$[1,0]^*$	$[1,1]$	$\dots$	$[1,\ n-k]$	$[2,1,1]$	$[2,1,0]$	$\dots$
0	$0Rq_0$	$0R[1,2]$	$\dots$	$0R[2,1,1]$	$0R[2,1,0]$	$0R[2,1,0]$	$\dots$
1	$0R[1,1]$	$1R[1,1]$	$\dots$	$1R[1,\ n-k]$	$1R[2,1,1]$	$1R[2,2,1]$	$\dots$

	$\dots$	$[2,k,1]$	$[3,2]$	$\dots$	$[3,l]$	$[4,1]$	$[4,2]$	$\dots$	$[4,l]$
0	$\dots$	$0R[3,2]$	$0R[3,3]$	$\dots$	$0R4$	$1R[4,2]$	$1R[4,3]$	$\dots$	$1L5$
1	$\dots$	$1[2,k,1]$		$\dots$		$1R[4,1]$			

	5	6	$[7,k,1]$	$[7,k,0]$	$\dots$	$[7,1,1]$
0	$0L6$	$0L6$	$0L[7,k,0]$	$0L[7,k,0]$	$\dots$	$0L[8,\ n-k]$
1	$1L5$	$1L[7,k,1]$	$1L[7,k,1]$	$1L[7,\ k-1,\ 1]$	$\dots$	$1L[7,1,1]$

	$[8,\ n-k]$	$\dots$	$[8,1]$
0	$0L[8,\ n-k-1]$	$\dots$	$0R[1,0]$
1	$1L[8,\ n-k]$	$\dots$	$1L[8,1]$

Моделирование машин Тьюринга

Доказательство (продолжение)

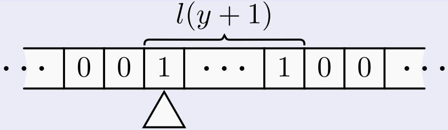
- Программа $\mathcal{M}_1^0$ будет иметь немного другой вид (так как ей нужно «пропустить» только один ноль, а не l). Выпишем её отдельно.

	$[1,0]^*$	$[1,1]$	$\dots$	$[1,n]$	$[4,1]$	$[4,2]$	$\dots$	$[4,l]$
0	$0Rq_0$	$0R[1,2]$	$\dots$	$0R[4,1]$	$1R[4,2]$	$1R[4,3]$	$\dots$	$1L5$
1	$0R[1,1]$	$1R[1,1]$	$\dots$	$1R[1,n]$	$1R[4,1]$			

	5	$[8,\ n]$	$\dots$	$[8,1]$
0	$0L[8,n]$	$0L[8,\ n-1]$	$\dots$	$0R[1,0]$
1	$1L5$	$1L[8,\ n]$	$\dots$	$1L[8,1]$

Моделирование машин Тьюринга

Доказательство (продолжение)



- Машина $\mathcal{M}_3$ строится аналогично машине $\mathcal{M}_1$, только она должна каждый раз стирать l символов, а записывать один. При этом ей всегда требуется обрабатывать только один блок единиц.

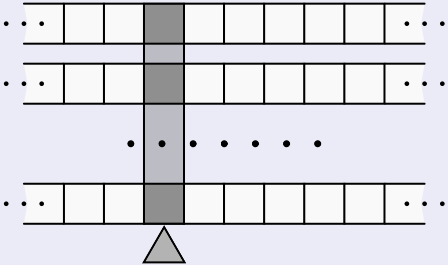
	$[1,1]^*$	$[1,2]$	$\ldots$	$[1,l]$	2	3	4	5
0	$0Rq_0$				$0R3$	$1L4$	$0L5$	$0R[1,1]$
1	$0R[1,2]$	$0R[1,3]$	$\ldots$	$0R2$	$1R2$	$1R3$	$1L4$	$1L5$

- Машина $\mathcal{M}'$ получается путём безусловной композиции машин $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$.

□

Механизм дорожек

Дорожки



- Рассмотрим машину Тьюринга в алфавите $A = \{0, 1, a_2, \dots, a_k\}$ с m дорожками. Эта машина имеет m лент (дорожек), на каждой из которых могут быть записаны символы алфавита A.
- Машина имеет одну головку, которая синхронно перемещается по всем дорожкам и может считывать и записывать символы на всех дорожках одновременно.

Механизм дорожек

Дорожки

- Команды машины имеют вид $a_{i_1} \ldots a_{i_m} q_j \to a_{k_1} \ldots a_{k_m} D q_s$.
- При вычислении функций в начальный момент основной код набора записан на первой дорожке, а остальные дорожки содержат нули.
- Результат вычисления записывается на первую дорожку, при этом все остальные дорожки должны быть «очищены» (на них должны содержаться нули).

Механизм дорожек

Моделирование машины с дорожками

- Моделируем машину $\mathcal{M}$ с m дорожками на обычной машине Тьюринга $\mathcal{M}'$ в алфавите A^m: каждый символ ленты машины $\mathcal{M}'$ кодирует в себе сразу m символов (по одному с каждой дорожки).
- Нулём машины $\mathcal{M}'$ считаем символ $(0, \ldots, 0)$, а единицей — символ $(1, 0, \ldots, 0)$. В начальный и конечный момент содержимое ленты $\mathcal{M}'$ автоматически кодирует содержимое дорожек $\mathcal{M}$.
- Каждая команда $a_{i_1} \ldots a_{i_m} q_j \to a_{k_1} \ldots a_{k_m} D q_s$ моделируется командой $(a_{i_1}, \ldots, a_{i_m}) q_j \to (a_{k_1}, \ldots, a_{k_m}) D q_s$.
- Таким образом, машина $\mathcal{M}'$ будет «воспроизводить» работу $\mathcal{M}$ с помощью одной ленты с «расширенным» алфавитом и будет вычислять ту же функцию.
- С помощью рассматривавшегося ранее моделирования машин Тьюринга теперь можно перейти от машины $\mathcal{M}'$ к машине с одной лентой в алфавите $\{0, 1\}$, вычисляющей ту же функцию.

Универсальные функции

Определение

Пусть $f_0(x), f_1(x), \dots$ — последовательность частичных функций натурального аргумента. Частичная функция $U(n, x)$ — универсальная функция для $\{f_0(x), f_1(x), \dots\}$, если

1. При любом $n_0 \geqslant 0$ функция $U(n_0, x)$ совпадает с одной из функций $f_0(x), f_1(x), \dots$
2. Для любого $i \geqslant 0$ найдётся $n' \geqslant 0$ такое, что $f_i(x) = U(n', x)$.

- Для универсальной функции $\{U(0, x), U(1, x), \dots\} = \{f_0(x), f_1(x), \dots\}$.

Определение

Универсальная машина Тьюригна $\mathcal{U}(n, x)$ — это машина Тьюринга, которая правильно вычисляет функцию, универсальную для последовательности всех вычислимых функций от одной переменной.

Лекция 7
Существование универсальной машины Тьюринга. Операции суперпозиции, примитивной рекурсии и минимизации. Классы примитивно рекурсивных и частично рекурсивных функций.

Существование универсальной машины Тьюринга

Теорема 2

Универсальная машина Тьюринга существует.

Доказательство

- Мы опишем лишь идею построения программы универсальной машины Тьюринга. Полное построение программы было бы слишком громоздко.
- Мы хотим пронумеровать все машины Тьюринга в алфавите $\{0, 1\}$ так, чтобы универсальная машина $\mathcal{U}(n, x)$ могла по поданному на вход номеру машины Тьюринга n воспроизвести её работу на числе x.

Существование универсальной машины Тьюринга

Доказательство (продолжение)

- Для этого мы будем формировать номер машины при помощи кодирования её программы.
    1. Команду $a_i q_j \to a_r D q_s$ кодируем словом $2a_i 2d(j) 2a_r 2d(D) 2d(s)$. Здесь $d(j), d(s)$ — это основные коды чисел. $d(L) = 0$, $d(R) = 1$, $d(S) = 01$.
    2. Пусть $w_1, \ldots, w_p$ — коды всех команд машины (порядок произвольный). Тогда $w_1 3 w_2 3 \ldots 3 w_p$ — код программы машины.
    3. Теперь рассматриваем код программы как число в четверичной системе счисления (сопоставление однозначно, так как первый символ всегда не 0). Это число и будет номером машины.
- Теперь нужно построить программу, которая бы расшифровывала номер-код машины Тьюринга и выполняла бы её программу. Для этого мы будем использовать механизм дорожек.

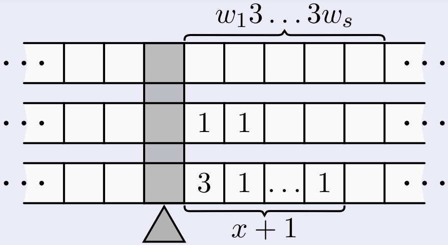
Существование универсальной машины Тьюринга

Доказательство (продолжение)

- Нам нужно построить универсальную машину, которая по номеру n машины Тьюринга $\mathcal{M}_n$ и входу x воспроизводила бы работу $\mathcal{M}_n$ на x и выдавала бы соответствующий результат.
- Будем использовать машину Тьюринга с тремя дорожками в алфавите $\{0, 1, 2, 3\}$:
  1. Первая дорожка содержит номер n в четверичной записи и используется для чтения программы машины $\mathcal{M}_n$.
  2. Вторая дорожка хранит номер текущего состояния машины $\mathcal{M}_n$ в основном коде.
  3. Третья дорожка хранит текущее содержимое ленты машины $\mathcal{M}_n$, а также позицию головки: символ 2 означает, что в ячейке записан 0 и находится головка, а символ 3 — что в ячейке записана 1 и находится головка.

Существование универсальной машины Тьюринга

Доказательство (продолжение)



- Сначала мы переписываем x на третью дорожку, 11 — на вторую дорожку, а на первой дорожке получаем из числа n его запись в четверичной системе счисления (т. е. код программы машины).
- Мы также помечаем позицию головки $\mathcal{M}_n$ в начале слова x.

Существование универсальной машины Тьюринга

Доказательство (продолжение)

- Происходит моделирование работы машины $\mathcal{M}_n$:
  1. Универсальная машина считывает текущий обозреваемый $\mathcal{M}_n$ символ с третьей дорожки (запоминает в состояниях) и ищет на первой дорожке команду, в левой части которой находится этот символ и состояние, записанное на второй дорожке.
  2. Если машина не находит такой команды (или видит нарушения формата кода), то число n не является кодом машины Тьюринга. Тогда машина стирает содержимое всех дорожек, записывает на первую дорожку основной код нуля и останавливается.
  3. Если машина нашла нужную команду, она заменяет номер состояния на второй дорожке, текущий обозреваемый символ на третьей дорожке и передвигает указатель положения головки (символ 2 или 3) на третьей дорожке.
  4. Если машина $\mathcal{M}_n$ перешла в состояние q_0, то универсальная машина переписывает результат её работы на первую дорожку и стирает содержимое остальных дорожек.

Существование универсальной машины Тьюринга

Доказательство (продолжение)

- Если в результате работы $\mathcal{M}_n$ на ленте не ровно один массив единиц, то зацикливаемся. Для этой проверки нужно ещё во время вычисления помечать задействованный участок ленты.
- Теперь можно перейти к машине с одной дорожкой в алфавите $\{0, 1\}$, и мы получим требуемую универсальную машину.
- Отметим, что мы описали лишь общую идею построения универсальной машины Тьюринга. При её реализации может потребоваться ввести дополнительные дорожки для выполнения технических операций универсальной машины.
- Например, при поиске команды в программе нужно возвращаться к содержимому второй дорожки для сравнения номеров состояний. Чтобы запоминать текущую позицию поиска в коде программы, можно использовать дополнительную дорожку.

$\square$

Утверждения об универсальной функции

Утверждение

Для последовательности всех вычислимых всюду определённых функций натурального аргумента от одной переменной не существует вычислимой универсальной функции.

Доказательство

- Пусть $U'(n, x)$ — вычислимая функция, универсальная для вычислимых всюду определённых функций от одной переменной.
- Функция $U'(n, x)$ всюду определена.
- Тогда функция $U'(x, x) + 1$ тоже вычислима и всюду определена. Значит, она имеет некоторый номер n_0: $U'(x, x) + 1 = U'(n_0, x)$.
- Тогда $U'(n_0, n_0) + 1 = U'(n_0, n_0)$, что невозможно, т.к. значение $U'(n_0, n_0)$ определено.

$\square$

Утверждения об универсальной функции

Утверждение

Существует вычислимая частичная функция, которую невозможно доопределить до вычислимой всюду определённой функции.

Доказательство

- Пусть $U(n, x)$ — вычислимая универсальная функция для последовательности вычислимых функций одного аргумента.
- Пусть $V(x)$ всюду определена и есть доопределение $U(x, x) + 1$.
- Если функция $V(x)$ вычислима, то вычисляющая её машина Тьюринга имеет некоторый номер n_1 и верно $V(x) = U(n_1, x)$.
- Тогда $V(n_1) = U(n_1, n_1)$, то есть значение $U(n_1, n_1)$ определено. Но тогда $V(n_1) = U(n_1, n_1) + 1$.
- Противоречие показывает, что функция $V(x)$ не может быть вычислимой.

□

Утверждения об универсальной функции

- Будем считать, что машины Тьюринга нумеруются тем же способом, что и в доказательстве существования универсальной машины Тьюринга.
- Если номер некорректен, то считаем, что он задаёт машину, правильно вычисляющую функцию 0.

Проблема остановки

$$\operatorname{stop}(n, x) = \begin{cases} 1, & \text{машина Тьюринга с номером } n \\ & \text{останавливается на входе } x, \\ 0, & \text{машина Тьюринга с номером } n \\ & \text{не останавливается на входе } x. \end{cases}$$

- Функция $\operatorname{stop}(n, x)$ проверяет, останавливается или зацикливается машина $\mathcal{M}_n$ на входе x. Эта задача называется проблемой остановки.

Утверждения об универсальной функции

Утверждение (неразрешимость проблемы остановки)

Функция $\mathrm{stop}(n, x)$ *невычислима.*

Доказательство

- Пусть функция $\mathrm{stop}(n, x)$ вычислима, а машина для вычисления $U(x, x) + 1$ имеет номер n_0. Тогда рассмотрим функцию

$$V(x) = \begin{cases} U(x, x) + 1, & \text{если } \mathrm{stop}(n_0, x) = 1, \\ 0 & \text{в ином случае.} \end{cases}$$

- Тогда функция $V(x)$ вычислима. Но это невозможно, так как она является доопределением $U(x, x) + 1$.

□

Утверждения об универсальной функции

Содержательный смысл результатов

- Существование универсальной машины Тьюринга на теоретическом уровне обосновывает возможность иметь один компьютер, который за счёт занесения в него разных программ способен выполнять любые алгоритмические задачи.
- При этом невозможно создать устройство или язык программирования, который позволял бы составлять только программы, которые не зацикливаются, но всё ещё позволял бы решать любые (алгоритмически разрешимые) задачи.
- Не существует алгоритмических способов избавиться от зацикливания или даже проверить наличие зацикливаний в произвольной программе.

Операции над частичными функциями

Определение

- Функция $f(x_1, \ldots, x_n)$ получается из функций $g_0(y_1, \ldots, y_m)$, $g_1(x_1, \ldots, x_n)$, $\ldots$, $g_m(x_1, \ldots, x_n)$ с помощью операции суперпозиции, если

$$f(x_1, \ldots, x_n) = g_0(g_1(x_1, \ldots, x_n), \ \ldots, \ g_m(x_1, \ldots, x_n)).$$

- При этом для каждого $\bar{a} \in \mathbb{N}_0^n$ значение $f(\bar{a})$ определено, если определены все значения $g_1(\bar{a}), \ldots, g_m(\bar{a})$, а также значение $g_0(g_1(\bar{a}), \ldots, g_m(\bar{a}))$.
- В противном случае значение $f(\bar{a})$ не определено.

Операции над частичными функциями

Определение

- Функция $f(x_1, \dots, x_n)$ получается из функций $g(x_1, \dots, x_{n-1})$ и $h(x_1, \dots, x_{n+1})$ с помощью операции примитивной рекурсии, если

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

- При этом для каждого $(\bar{a}, a_n) \in \mathbb{N}_0^n$ значение $f(\bar{a}, a_n)$ определено, если определено значение $g(\bar{a})$ и все значения $h(\bar{a}, y, f(\bar{a}, y))$ при $y < a_n$.
- В противном случае значение $f(\bar{a}, a_n)$ не определено.

Операции над частичными функциями

Пример работы рекурсии

- Рассмотрим частный примитивной случай рекурсии — итерацию:

$$\begin{cases} f(0) = a, \\ f(y+1) = h(f(y)). \end{cases}$$

- Здесь $f(1) = h(a),\ f(2) = h(h(a)),\ \ldots,\ f(i) = \underbrace{h(\ldots(h}_{i}(a))\ldots).$

Операции над частичными функциями

Определение

- Функция $f(x_1, \ldots, x_n)$ получается из функции $g(x_1, \ldots, x_n)$ с помощью операции минимизации

$$f(x_1, \ldots, x_n) = (\mu y)(g(x_1, \ldots, x_{n-1}, y) = x_n),$$

  если при любых значениях $x_1, \ldots, x_n$ значение $f(x_1, \ldots, x_n)$ равно минимальному значению y такому, что $g(x_1, \ldots, x_{n-1}, y) = x_n$.
- При этом для каждого $(\bar{a}, a_n) \in \mathbb{N}_0^n$ значение $f(\bar{a}, a_n)$ определено, если существует b такое, что $g(\bar{a}, b) = a_n$, причём все значения $g(\bar{a}, 0), \ldots, g(\bar{a}, b)$ определены.
- В противном случае значение $f(\bar{a}, b)$ не определено.

- Иными словами $f(a_1, \ldots, a_n) = b$, если $g(a_1, \ldots, a_{n-1}, b) = a_n$ и для всех $z < b$ значения $g(a_1, \ldots, a_{n-1}, z)$ определены и отличны от a_n.

Операции над частичными функциями

- Требование того, что $g(\bar{a}, 0), \ldots, g(\bar{a}, b)$ определены, существенно в определении минимизации. Если убрать это требование, то минимизация сможет получать из вычислимых функций невычислимые.

Пример минимизации

- Пусть $g(y) \equiv 1$.
- $f(x) = (\mu y)(1 = x)$.
- Тогда $f(x) = \begin{cases} 0, & x = 1, \\ \text{не определено} & \text{иначе.} \end{cases}$

Некоторые классы функций

Базовые функции

- $I = \{0,\ x+1,\ I_m^n(x_1, \ldots, x_n),\ m = \overline{1,n},\ n \in \mathbb{N}\}$,
  где $I_m^n(x_1, \ldots, x_n) = x_m$.

Определение

Класс примитивно рекурсивных функций $F_{\text{пр}}$ — это замыкание множества I относительно операций суперпозиции и примитивной рекурсии $[I]_{\substack{\text{суперпозиция}\\ \text{прим. рекурсия}}}$.

Определение

Класс частично рекурсивных функций $F_{\text{чр}}$ — это замыкание множества I относительно операций суперпозиции, примитивной рекурсии и минимизации $[I]_{\substack{\text{суперпозиция}\\ \text{прим. рекурсия}\\ \text{минимизация}}}$.

Некоторые классы функций

Простейшие свойства классов

- $F_{\text{пр}}$ содержит только всюду определённые функции. $F_{\text{чр}}$ содержит и частичные функции.
- $F_{\text{пр}} \subsetneq F_{\text{чр}}$.

- Название «частично рекурсивные функции» не вполне корректно с точки зрения русского языка и появилось в результате неудачного перевода. Правильнее было бы говорить «частичные рекурсивные функции». Но название «частично рекурсивные функции» уже стало стандартным и повсеместно используется.

Лекция 8
Вычислимость частично рекурсивных функций. Некоторые примитивно рекурсивные функции.

Некоторые классы функций

Тезис Чёрча

Класс $F_{\text{чр}}$ совпадает с классом эффективно (алгоритмически) вычислимых функций.

- Понятие «эффективно (алгоритмически) вычислимых» не является строгим, поэтому этот тезис невозможно доказать.
- Однако на текущий момент все известные способы конкретизации понятия эффективной вычислимости приводят к классу $F_{\text{чр}}$.
- Далее мы докажем это для одной из конкретизаций: $F_{\text{чр}} = F_{\text{выч}}$, где $F_{\text{выч}}$ — это класс всех вычислимых (на машинах Тьюринга) функций.

Вычислимость частично рекурсивных функций

Теорема 3

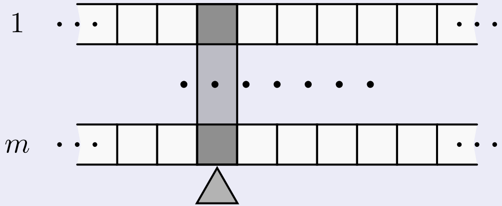
Имеет место включение $F_{\text{чр}} \subseteq F_{\text{выч}}$.

Доказательство

- Ранее были построены машины Тьюринга, правильно вычисляющие функции системы I.
- Для доказательства теоремы осталось доказать замкнутость класса вычислимых функций относительно операций суперпозиции, примитивной рекурсии и минимизации.

Замкнутость $F_{\text{выч}}$ относительно суперпозиции

Доказательство (продолжение)



- Пусть $f(x_1, \ldots, x_n) = g_0(g_1(x_1, \ldots, x_n), \ \ldots, \ g_m(x_1, \ldots, x_n))$, а функции $g_0, \ldots, g_m$ правильно вычисляются машинами Тьюринга $\mathcal{M}_0, \ldots, \mathcal{M}_m$.
- Мы будем строить машину Тьюринга $\mathcal{M}$, правильно вычисляющую функцию f. Машина будет иметь m дорожек, на которых она будет производить вычисление функций $g_1, \ldots, g_m$. Затем она будет записывать результаты на первую дорожку и применять к ним функцию g_0.

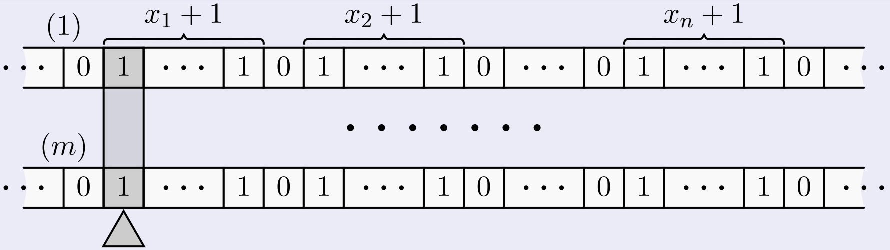
Замкнутость $F_{\text{выч}}$ относительно суперпозиции

Доказательство (продолжение)

- Мы хотим иметь возможность «отслеживать» области ленты, которые были затронуты вычислениями функций $g_1, \ldots, g_m$. Для этого мы вводим в машины $\mathcal{M}_1, \ldots, \mathcal{M}_m$ новый символ ленты 2.
- Машины будут обрабатывать символ 2 так же, как символ 0. При этом они никогда не будут записывать на ленту 0, вместо этого они будут записывать 2.
- В машинах $\mathcal{M}_1, \ldots, \mathcal{M}_m$ команды $a_i q_j \to 0 D q_s$ заменяем на $a_i q_j \to 2 D q_s$. После этого для каждой команды $0 q_j \to a_k D q_s$ добавляем новую команду $2 q_j \to a_k D q_s$.
- Полученные машины обозначим $\mathcal{M}'_1, \ldots, \mathcal{M}'_m$.
- Каждая дорожка машины $\mathcal{M}$ будет содержать символы $0, 1, 2$.
- Далее описываем работу машины $\mathcal{M}$.

Замкнутость $F_{\text{выч}}$ относительно суперпозиции

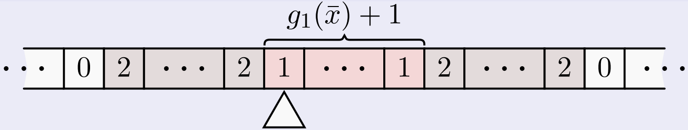
Доказательство (продолжение)



- Вначале машина переносит основной код входного набора на m дорожек и возвращает головку на начало этого основного кода.
- Запускаем машину $\mathcal{M}_1'$ на первой дорожке. Во время своей работы машина $\mathcal{M}_1'$ заменяет нули на всех остальных дорожках на двойки (единицы оставляет без изменения).

Замкнутость $F_{\text{выч}}$ относительно суперпозиции

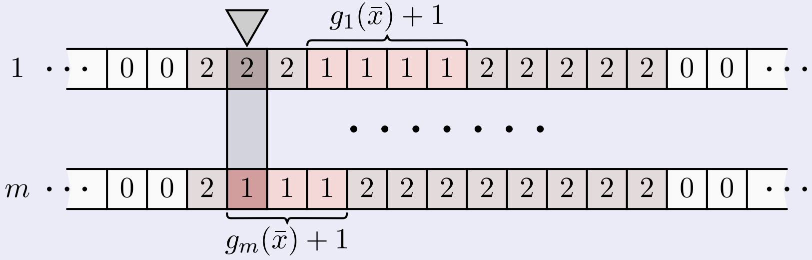
Доказательство (продолжение)



- Если машина $\mathcal{M}'_1$ остановится, то её результатом на первой дорожке будет являться массив из единиц. Он может быть окружён некоторым количеством двоек.
- Возвращаем головку на первый символ входа других дорожек. Для этого на любой из других дорожек двигаемся по единицам и двойкам влево, пока не дойдём до нулей, а затем вправо до первой единицы.
- Далее одну за другой запускаем машины $\mathcal{M}'_2$ на второй дорожке, $\mathcal{M}'_3$ на третьей дорожке, $\ldots$, $\mathcal{M}'_m$ на m-й дорожке. Перед каждым запуском возвращаем головку на первый символ входа.

Замкнутость $F_{\text{выч}}$ относительно суперпозиции

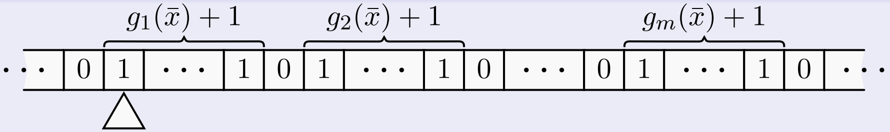
Доказательство (продолжение)



- Если все машины остановятся, то их результаты будут иметь такой же вид, как у $\mathcal{M}_1'$. Блоки из единиц могут находиться в разных местах ленты, но непустые области дорожек выровнены.
- Переносим все результаты вычислений на первую дорожку: как и раньше, найти начало блока единиц можно с помощью двоек.
- Стираем содержимое всех дорожек, кроме первой. Заменяем на первой дорожке двойки на нули.

Замкнутость $F_{\text{выч}}$ относительно суперпозиции

Доказательство (продолжение)



- Мы получили на первой дорожке основной код набора $g_1(\bar{x}), \ldots, g_m(\bar{x})$. Остальные дорожки пусты и больше не будут использоваться.
- Запускаем на первой дорожке машину $\mathcal{M}_0$. Она проведёт вычисление функции g_0 и выдаст требуемый результат. В случае неопределённых значений функционирование машины $\mathcal{M}$ тоже соответствует определению суперпозиции.
- От машины с дорожками в алфавите $\{0, 1, 2\}$ переходим к обычной машине Тьюринга в алфавите $\{0, 1\}$. Замкнутость класса $F_{\text{выч}}$ относительно суперпозиции доказана.

Замкнутость $F_{\text{выч}}$ относительно рекурсии

Доказательство (продолжение)

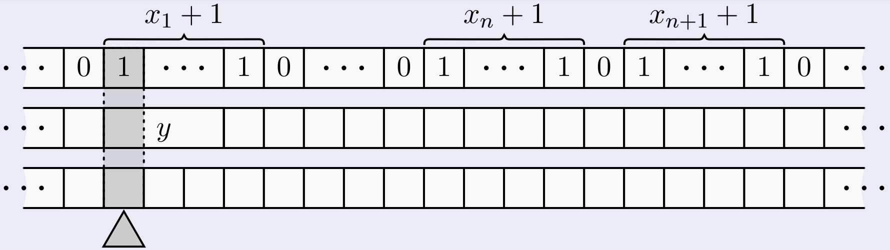
- Пусть $f(x_1, \ldots, x_n, x_{n+1})$ получается из функций $g(x_1, \ldots, x_n)$ и $h(x_1, \ldots, x_n, y, z)$ с помощью примитивной рекурсии:

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

  и функции g и h правильно вычисляются машинами $\mathcal{M}_g$ и $\mathcal{M}_h$.
- Аналогично случаю суперпозиции будем отмечать символом 2 пустые клетки, которые были затронуты работой машин $\mathcal{M}_g$ и $\mathcal{M}_h$. Символы 2 будут использоваться для поиска блоков единиц на дорожках, мы не будем уточнять это далее.
- Будем строить машину $\mathcal{M}$ с тремя дорожками в алфавите $\{0, 1, 2\}$ для вычисления функции f.

Замкнутость $F_{\text{выч}}$ относительно рекурсии

Доказательство (продолжение)



- Первая дорожка постоянно содержит входные значения. Вторая дорожка содержит y — номер текущей итерации рекурсии. Третья дорожка используется для вычислений $\mathcal{M}_g$ и $\mathcal{M}_h$.
- Вначале машина $\mathcal{M}$ переписывает значения $x_1, \ldots, x_n$ с первой дорожки на третью, записывает $y = 0$ на вторую дорожку и запускает машину $\mathcal{M}_g$ на третьей дорожке.

Замкнутость $F_{\text{выч}}$ относительно рекурсии

Доказательство (продолжение)

- Если $y = x_{n+1}$, то машина переписывает результат с третьей дорожки на первую, стирает всё остальное и завершает вычисление.
- Иначе машина формирует на третьей дорожке набор $x_1, \ldots, x_n, y, z$, где z — уже имеющийся на этой дорожке результат прошлого вычисления.
- Запускается машина $\mathcal{M}_h$ на третьей дорожке. После окончания её работы содержимое второй дорожки y увеличивается на 1.
- Если $y = x_{n+1}$, то машина переписывает результат на первую дорожку, стирает всё остальное и завершает вычисление.
- Иначе машина вновь формирует на третьей дорожке набор $x_1, \ldots, x_n, y, z$, где z — уже записанный на ней результат прошлого вычисления, запускает $\mathcal{M}_h$ и продолжает работу циклически.

Замкнутость $F_{\text{выч}}$ относительно рекурсии

Доказательство (продолжение)

- Нетрудно видеть, что полученная машина моделирует работу примитивной рекурсии.
- При этом, если в процессе вычислений встретилось неопределённое значение, то машина никогда не остановится и результат будет неопределён, что соответствует определению примитивной рекурсии.
- От машины с дорожками в алфавите $\{0, 1, 2\}$ переходим к обычной машине Тьюринга в алфавите $\{0, 1\}$. Замкнутость класса $F_{\text{выч}}$ относительно примитивной рекурсии доказана.

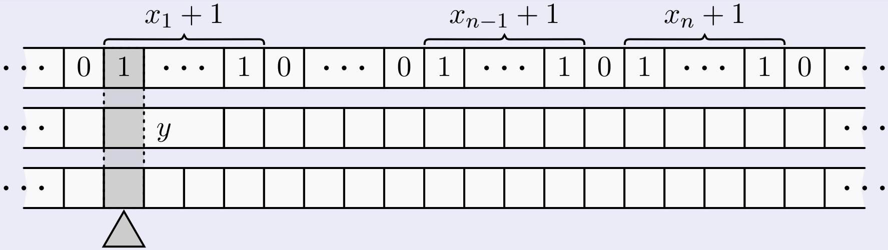
Замкнутость $F_{\text{выч}}$ относительно минимизации

Доказательство (продолжение)

- Пусть $f(x_1, \dots, x_n) = (\mu y)(g(x_1, \dots, x_{n-1}, y) = x_n)$ и функция g вычисляется машиной $\mathcal{M}_g$.
- Аналогично случаю суперпозиции будем отмечать символом 2 пустые клетки, которые были затронуты работой машины $\mathcal{M}_g$. Символы 2 будут использоваться для поиска блоков единиц на дорожках, мы не будем уточнять это далее.
- Будем строить машину $\mathcal{M}$ с тремя дорожками в алфавите $\{0, 1, 2\}$ для вычисления функции f, аналогичную машине для примитивной рекурсии.

Замкнутость $F_{\text{выч}}$ относительно минимизации

Доказательство (продолжение)



- Первая дорожка постоянно содержит входные значения. Вторая дорожка содержит y — номер текущего проверяемого значения. Третья дорожка используется для вычислений $\mathcal{M}_g$.
- Вначале машина $\mathcal{M}$ записывает $y = 0$ на вторую дорожку, переписывает значения $x_1, \ldots, x_{n-1}, y$ на третью дорожку и запускает машину $\mathcal{M}_g$ на третьей дорожке.

Замкнутость $F_{\text{выч}}$ относительно минимизации

Доказательство (продолжение)

- Далее машина сравнивает результат z на третьей дорожке с x_n. Если $z = x_n$, то она переписывает y со второй дорожки на первую, стирает всё остальное и останавливается.
- Иначе машина увеличивает y на 1, формирует на третьей дорожке набор $x_1, \ldots, x_{n-1}, y$ и запускает машину $\mathcal{M}_g$. Далее машина продолжает работу циклически.
- Таким образом, машина находит минимальное y, для которого выполняется $g(x_1, \ldots, x_{n-1}, y) = x_n$. Если в процессе поиска она натыкается на неопределённое значение, то она никогда не остановится, что соответствует определению минимизации.
- От машины с дорожками в алфавите $\{0, 1, 2\}$ переходим к обычной машине Тьюринга в алфавите $\{0, 1\}$. Замкнутость класса $F_{\text{выч}}$ относительно минимизации доказана.

□

Примитивно рекурсивные функции

Класс примитивно рекурсивных функций

- $I = \{0,\ x+1,\ I_m^n(x_1,\ldots,x_n),\ m = \overline{1,n},\ n \in \mathbb{N}\}$,
  где $I_m^n(x_1,\ldots,x_n) = x_m$.
- Суперпозиция:
$$f(x_1,\ldots,x_n) = g_0(g_1(x_1,\ldots,x_n),\ \ldots,\ g_m(x_1,\ldots,x_n)).$$
- Примитивная рекурсия:
$$\begin{cases} f(x_1,\ldots,x_{n-1},\ 0) = g(x_1,\ldots,x_{n-1}), \\ f(x_1,\ldots,x_{n-1},\ y+1) = h(x_1,\ldots,x_{n-1},\ y,\ f(x_1,\ldots,x_{n-1},y)) \end{cases}$$
- Класс примитивно рекурсивных функций $F_{\text{пр}}$ — это замыкание множества I относительно операций суперпозиции и примитивной рекурсии $[I]_{\substack{\text{суперпозиция} \\ \text{прим. рекурсия}}}$.

Примитивно рекурсивные функции

Некоторые простые функции

- $0,\ x+1 \in F_{\text{пр}}$ по определению.
- Константа $d = 0 \underbrace{+1+\ldots+1}_{d} \in F_{\text{пр}}$.
- $\mathrm{sum}(x,y) = x+y$:

$$\begin{cases} \mathrm{sum}(x,\ 0) = x, \\ \mathrm{sum}(x,\ y+1) = \mathrm{sum}(x,y)+1. \end{cases}$$

  Здесь $g(x) = x = I_1^1(x)$, $h(x,y,z) = z+1 = I_3^3(x,y,z)+1$.
- $\mathrm{prod}(x,y) = xy$:

$$\begin{cases} \mathrm{prod}(x,\ 0) = 0, \\ \mathrm{prod}(x,\ y+1) = \mathrm{prod}(x,y)+x. \end{cases}$$

Примитивно рекурсивные функции

Некоторые простые функции

- Усечённая разность: $x \dot{-} y = \begin{cases} x - y, & x \geqslant y, \\ 0, & x < y. \end{cases}$

  1. Сначала докажем, что $x \dot{-} 1 \in F_{\text{пр}}$:

  $$\begin{cases} 0 \dot{-} 1 = 0, \\ (x+1) \dot{-} 1 = x. \end{cases}$$

  2. Теперь можно доказать, что $x \dot{-} y \in F_{\text{пр}}$:

  $$\begin{cases} x \dot{-} 0 = x, \\ x \dot{-} (y+1) = (x \dot{-} y) \dot{-} 1. \end{cases}$$

Примитивно рекурсивные функции

Некоторые простые функции

- $\mathrm{pow}(x,y) = x^y$ (считаем, что $0^0 = 1$):

$$\begin{cases} \mathrm{pow}(x,\ 0) = 1, \\ \mathrm{pow}(x,\ y+1) = \mathrm{pow}(x,y) \cdot x. \end{cases}$$

- $\min(x,y) = x \dot{-} (x \dot{-} y)$.
- $\max(x,y) = (x+y) \dot{-} \min(x,y)$.
- $|x-y| = (x \dot{-} y) + (y \dot{-} x)$.

  Отметим: $|x-y|$ не является суперпозицией функций $|x|$ и $x-y$. Это единая функция, и она всюду определена.
- $\mathrm{sg}\, x = \begin{cases} 0, & x = 0, \\ 1 & x > 0, \end{cases} \qquad \overline{\mathrm{sg}}\, x = \begin{cases} 1, & x = 0, \\ 0 & x > 0, \end{cases}$

  $\overline{\mathrm{sg}}\, x = 1 \dot{-} x,\ \ \mathrm{sg}\, x = \overline{\mathrm{sg}}\, \overline{\mathrm{sg}}\, x \in F_{\text{пр}}$.

Примитивно рекурсивные функции

Замена значений функции в нескольких точках

- Характеристическая функция точки a: $\overline{\mathrm{sg}}\,|x-a| = \begin{cases} 1, & x = a, \\ 0, & x \neq a. \end{cases}$
- Пусть функция $f(x)$ в точках $a_1, \ldots, a_m$ принимает значения $b_1, \ldots, b_m$ соответственно, а в остальных точках она равна 0. Тогда

$$f(x) = b_1\,\overline{\mathrm{sg}}\,|x-a_1| + \ldots + b_m\,\overline{\mathrm{sg}}\,|x-a_m| \in F_{\text{пр}}.$$

- Пусть теперь $g(x)$ — примитивно рекурсивная функция, а $f(x)$ получается из $g(x)$ заменой значений в точках $a_1, \ldots, a_m$ на $b_1, \ldots, b_m$ соответственно. Тогда $f \in F_{\text{пр}}$:

$$f(x) = b_1\,\overline{\mathrm{sg}}\,|x-a_1| + \ldots + b_m\,\overline{\mathrm{sg}}\,|x-a_m| + \\ + g(x)\,\mathrm{sg}\,|x-a_1| \cdot \ldots \cdot \mathrm{sg}\,|x-a_m|.$$

Примитивно рекурсивные функции

Выражение отношений функциями

- Характеристическая функция отношения (предиката) $\rho(\bar{x})$ — это функция, принимающая значения 0 и 1, причём функция принимает значение 1 на тех и только на тех наборах, на которых $\rho(\bar{x})$ истинно.
- Характеристическая функция $x = y$: $\overline{\mathrm{sg}}\,|x - y|$.
- Характеристическая функция $x \neq y$: $\mathrm{sg}\,|x - y|$.
- Характеристическая функция $x < y$: $\mathrm{sg}(y \dot{-} x)$.
- Характеристическая функция $x > y$: $\mathrm{sg}(x \dot{-} y)$.
- Характеристическая функция $x \geqslant y$: $\overline{\mathrm{sg}}(y \dot{-} x)$.
- Характеристическая функция $x \leqslant y$: $\overline{\mathrm{sg}}(x \dot{-} y)$.

Примитивно рекурсивные функции

Утверждение (Разбор случаев по предикатам)

$$f(x_1,\dots,x_n)=\begin{cases} f_1(x_1,\dots,x_n), & \text{если } \rho_1(x_1,\dots,x_n),\\ \dots & \\ f_m(x_1,\dots,x_n), & \text{если } \rho_m(x_1,\dots,x_n),\\ f_{m+1}(x_1,\dots,x_n) & \text{иначе,}\end{cases}$$

где $f_1,\dots,f_{m+1}\in F_{пр}$, $\rho_1,\dots,\rho_m$ — попарно несовместные предикаты, характеристические функции $\chi_1,\dots,\chi_m$ которых примитивно рекурсивны. Тогда функция f примитивно рекурсивна.

Доказательство

$$f(\bar{x})=f_1(\bar{x})\chi_1(\bar{x})+\ldots+f_m(\bar{x})\chi_m(\bar{x})+f_{m+1}(\bar{x})\,\overline{\mathrm{sg}}(\chi_1(\bar{x})+\ldots+\chi_m(\bar{x}))$$

□

Примитивно рекурсивные функции

Ограниченные суммирование и мультиплицирование

- Операция ограниченного суммирования:

$$f(x_1, \dots, x_n) = \sum_{i=0}^{x_n} g(x_1, \dots, x_{n-1}, i).$$

$$\begin{cases} f(x_1, \dots, x_{n-1},\, 0) = g(x_1, \dots, x_{n-1}, 0), \\ f(x_1, \dots, x_{n-1},\, y+1) = f(x_1, \dots, x_{n-1}, y) + g(x_1, \dots, x_{n-1},\, y+1). \end{cases}$$

- Операция ограниченного мультиплицирования:

$$f(x_1, \dots, x_n) = \prod_{i=0}^{x_n} g(x_1, \dots, x_{n-1}, i).$$

$$\begin{cases} f(x_1, \dots, x_{n-1},\, 0) = g(x_1, \dots, x_{n-1}, 0), \\ f(x_1, \dots, x_{n-1},\, y+1) = f(x_1, \dots, x_{n-1}, y) \cdot g(x_1, \dots, x_{n-1},\, y+1). \end{cases}$$

Примитивно рекурсивные функции

Деление с остатком

- Считаем, что $\lfloor x/y \rfloor = 0$ при $y = 0$.
- Чтобы получить значение $\lfloor x/y \rfloor$, нужно получить $i \in \{0, \ldots, x\}$ (оно будет единственным) такое, что $(i = \lfloor x/y \rfloor) \equiv (i \leqslant x/y < i+1) \equiv (iy \leqslant x) \,\&\, ((i+1)y > x)$.
- Это значение i ищем с помощью операции ограниченного суммирования:

$$\lfloor x/y \rfloor = \sum_{i=0}^{x} i \cdot \overline{\mathrm{sg}}(iy \dot{-} x)\, \mathrm{sg}((i+1)y \dot{-} x).$$

- $\mathrm{rm}(x, y)$ — остаток от деления x на y (0 при $y = 0$).
- $\mathrm{rm}(x, y) = (x - y \cdot \lfloor x/y \rfloor)\, \mathrm{sg}\, y$.

Примитивно рекурсивные функции

Извлечение корня

- Аналогично делению можно получить вычисление корня: $(i = \lfloor\sqrt{x}\rfloor) \equiv (i \leqslant \sqrt{x} < i + 1) \equiv (i^2 \leqslant x)\ \&\ ((i + 1)^2 > x)$.
- Поиск этого i с помощью ограниченного суммирования:

$$\lfloor\sqrt{x}\rfloor = \sum_{i=0}^{x} i \cdot \overline{\mathrm{sg}}(i^2 \dot{-} x)\,\mathrm{sg}((i + 1)^2 \dot{-} x).$$

- Схожим образом можно получить функции $\lfloor\sqrt[m]{x}\rfloor$, $\lfloor\log_y x\rfloor$ (при некотором доопределении в нулевых точках) и другие обратные функции.

Лекция 9
Некоторые частично рекурсивные функции. Формула Клини.

Частично рекурсивные функции

Частично рекурсивные функции

- Операция минимизации:

$$f(x_1,\ldots,x_n) = (\mu y)(g(x_1,\ldots,x_{n-1},y) = x_n),$$

- Класс частично рекурсивных функций $F_{\text{чр}}$ — это замыкание множества I относительно операций суперпозиции, примитивной рекурсии и минимизации $[I]_{\substack{\text{суперпозиция}\\ \text{прим. рекурсия}\\ \text{минимизация}}}$.

Нигде не определённая функция

- $g(x) = (\mu y)(1 = x) = \begin{cases} 0, & x = 1, \\ \text{не определено}, & \text{иначе}. \end{cases}$
- $f(x) = (\mu y)(g(y) = x)$ — нигде не определённая функция (она не определена ни в одной точке).

Частично рекурсивные функции

Обратные функции

- $f_1(x) = (\mu y)(y + 1 = x) = x - 1$ (не определена при $x = 0$).
- $f_2(x) = (\mu y)(y^2 = x) = \sqrt{x}$ (не определена, если x не полный квадрат).

Нумерационные функции

- Пусть $\mathrm{c}(x, y)$ — инъективная функция, а $\mathrm{l}(v)$ и $\mathrm{r}(v)$ — такие функции, что $\mathrm{l}(\mathrm{c}(x, y)) = x,\ \mathrm{r}(\mathrm{c}(x, y)) = y$.
  Тогда набор функций $\mathrm{c}(x, y),\ \mathrm{l}(v),\ \mathrm{r}(v)$ называется тройкой нумерационных функций.
- Нумерационные функции позволяют кодировать пары чисел одним числом. Их можно выбирать по разному, мы рассмотрим один конкретный вариант.

Частично рекурсивные функции

Нумерационные функции

- Обозначим

$$\mathrm{c}(x,y)=(x+y)^2+x,\quad \mathrm{l}(v)=v\dot{-}(\lfloor\sqrt{v}\rfloor)^2,\quad \mathrm{r}(v)=\lfloor\sqrt{v}\rfloor\dot{-}l(v).$$

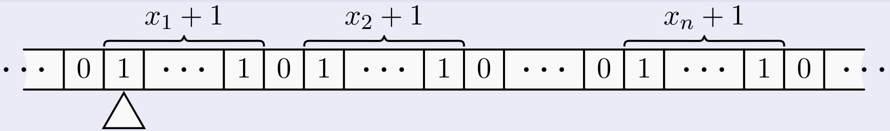
- Легко видеть, что указанные функции составляют тройку нумерационных функций и что эти функции примитивно рекурсивны.
- Для нумерации троек используем функцию $\mathrm{c}^3(x,y,z)=\mathrm{c}(\mathrm{c}(x,y),z)$.
- Обратные функции для c^3:

$$\mathrm{l}_1(v)=\mathrm{l}(\mathrm{l}(v)),\quad \mathrm{l}_2(v)=\mathrm{r}(\mathrm{l}(v)),\quad \mathrm{l}_3(v)=\mathrm{r}(v).$$

- Ясно, что все эти функции примитивно рекурсивны.

Частично рекурсивные функции

Функции для представления основного кода



- Обозначим через $\Theta_n(x_1, \ldots, x_n)$ функцию, которая выдаёт число, двоичным представлением которого является основной код набора $(x_1, \ldots, x_n)$.
- Эти функции примитивно рекурсивны:

$$\begin{cases} \Theta_1(0) = 1, \\ \Theta_1(x+1) = 2\Theta_1(x) + 1, \end{cases}$$

$$\begin{cases} \Theta_{n+1}(\bar{x},\, 0) = 4\Theta_n(\bar{x}) + 1, \\ \Theta_{n+1}(\bar{x},\, x_{n+1}+1) = 2\Theta_{n+1}(\bar{x},\, x_{n+1}) + 1. \end{cases}$$

Формула Клини

Теорема 4 (Формула Клини)

Для любой вычислимой функции $f(x_1, \ldots, x_n)$ *найдутся примитивно рекурсивные функции* $G(x_1, \ldots, x_n, y)$ *и* $H(x_1, \ldots, x_n, y)$ *такие, что*

$$f(x_1, \ldots, x_n) = G(x_1, \ldots, x_n,\ (\mu y)(H(x_1, \ldots, x_n, y) = 0)).$$

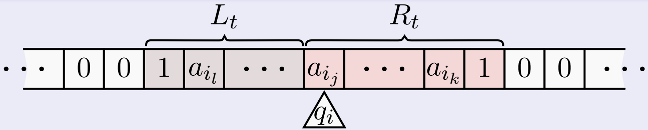
- Для задания любой вычислимой функции достаточно только одного использования операции минимизации.

Доказательство

- Пусть машина $\mathcal{M}$ правильно вычисляет функцию f. Мы будем считать, что в программе машины есть команды для заключительного состояния: $0q_0 \to 0Sq_0$ и $1q_0 \to 1Sq_0$.

Формула Клини

Доказательство (продолжение)



- Предположим, что машина $\mathcal{M}$ начала работать на входе $\bar{x} = (x_1, \ldots, x_n)$. Рассмотрим конфигурацию на ленте в произвольный момент времени t.
- Обозначим через $l(\bar{x}, t)$ число, двоичной записью которой является содержимое ленты левее головки L_t.
- Обозначим через $r(\bar{x}, t)$ число, двоичной записью которой является содержимое ленты справа от головки R_t, включая обозреваемый головкой символ. При этом считаем, что эта запись размещена на ленте справа налево (младшие разряды слева, обозреваемый головкой разряд — самый младший).

Формула Клини

Доказательство (продолжение)

- Обозначим $q(\bar{x}, t)$ номер состояния i в момент времени t при работе на входе $\bar{x}$.
- Легко видеть, что тройка значений $(l(\bar{x}, t),\ r(\bar{x}, t),\ q(\bar{x}, t))$ полностью задаёт конфигурацию машины и её дальнейшее функционирование.
- Будем кодировать всю конфигурацию машины с помощью одного числа:
$$\mathrm{Code}(\bar{x}, t) = \mathrm{c}^3(l(\bar{x}, t),\ r(\bar{x},\ t),\ q(\bar{x}, t)).$$
- Далее мы покажем, что функция Code примитивно рекурсивна. А пока выпишем формулу Клини с использованием этой функции.

Формула Клини

Доказательство (продолжение)

$$\rho\colon (\underbrace{11\ldots1}_{z+1})_2 \to z$$

- Через $\rho(x)$ обозначим функцию, которая удовлетворяет условию $\rho(2^{z+1}-1) = z$ при всех $z \in \mathbb{N}_0$. Эта функция преобразует число, двоичной записью которого является основной код числа z, в само число z. Далее мы покажем, что она примитивно рекурсивна.
- Формула Клини:

$$f(x_1, \ldots, x_n) = \rho(\mathrm{l}_2(\mathrm{Code}(\bar{x},\ (\mu t)(\mathrm{l}_3(\mathrm{Code}(\bar{x}, t)) = 0)))).$$

- Эта формула ищет минимальный момент времени, в котором машина попадает в состояние q_0. Далее она берёт конфигурацию в этот момент времени, извлекает из неё правую часть ленты и выдаёт записанный на ней результат.

Формула Клини

Доказательство (продолжение)

- Сначала покажем примитивную рекурсивность функции ρ:

$$\rho(x) = \sum_{i=0}^{x} i\,\overline{\mathrm{sg}}\,|(2^{i+1} - 1) - x|.$$

- Для завершения доказательства теоремы осталось доказать примитивную рекурсивность функции Code. Будем задавать эту функцию схемой примитивной рекурсии.

$$\begin{cases} \mathrm{Code}(\bar{x},\, 0) = \mathrm{c}^3(0,\, \Theta_n(x_n, \ldots, x_1),\, 1), \\ \mathrm{Code}(\bar{x},\, t+1) = h(\bar{x},\, t,\, \mathrm{Code}(\bar{x}, t)). \end{cases}$$

- У $\Theta_n(x_n, \ldots, x_1)$ аргументы переставлены, так как двоичная запись $r(\bar{x}, t)$ пишется справа налево.
- Задание функции h потребует ряда технических операций.

Формула Клини

Доказательство (продолжение)

$l(\bar{x}, t) = \mathrm{l}_1(\mathrm{Code}(\bar{x}, t)),\ r(\bar{x}, t) = \mathrm{l}_2(\mathrm{Code}(\bar{x}, t)),\ q(\bar{x}, t) = \mathrm{l}_3(\mathrm{Code}(\bar{x}, t))$

- Текущий обозреваемый символ — младший символ правой части: $\nu(\bar{x}, t) = \mathrm{rm}(r(\bar{x}, t), 2)$.
- Пусть $\nu(\bar{x}, t) = a,\ q(\bar{x}, t) = i$ и в программе машины есть команда $aq_i \to b_{a,i} D_{a,i} q_{j_{a,i}}$. Рассмотрим, каким будет значение $\mathrm{Code}(\bar{x},\ t+1)$ в каждом возможном случае.
- Если $D_{a,i} = S$, то

$$
\begin{aligned}
l_{a,i}(\bar{x},\ t+1) &= l(x, t),\\
r_{a,i}(\bar{x},\ t+1) &= r(x, t) \dot{-} \nu(\bar{x}, t) + b_{a,i},\\
q_{a,i}(\bar{x},\ t+1) &= j_{a,i}.
\end{aligned}
$$

Формула Клини

Доказательство (продолжение)

- Если $D_{a,i} = L$, то

$$l_{a,i}(\bar{x},\, t+1) = \lfloor l(\bar{x}, t)/2 \rfloor,$$
$$r_{a,i}(\bar{x},\, t+1) = (r(\bar{x}, t) \dot{-} \nu(\bar{x}, t) + b_{a,i}) \cdot 2 + \mathrm{rm}(l(\bar{x}, t),\, 2),$$
$$q_{a,i}(\bar{x},\, t+1) = j_{a,i}.$$

- Если $D = R$, то

$$l_{a,i}(\bar{x},\, t+1) = 2 \cdot l(\bar{x}, t) + b_{a,i},$$
$$r_{a,i}(\bar{x},\, t+1) = \lfloor r(\bar{x}, t)/2 \rfloor,$$
$$q_{a,i}(\bar{x},\, t+1) = j_{a,i}.$$

Формула Клини

Доказательство (продолжение)

- Пусть Q — множество номеров состояний машины $\mathcal{M}$. Тогда

$$\mathrm{Code}(\bar{x},\, t+1) = \sum_{\substack{a\in\{0,1\}\\ i\in Q}} \overline{\mathrm{sg}}\,|a-\nu(\bar{x},t)|\cdot\overline{\mathrm{sg}}\,|i-q(\bar{x},t)|\times$$
$$\times\,\mathrm{c}^3(l_{a,i}(\bar{x},\, t+1),\, r_{a,i}(\bar{x},\, t+1),\, q_{a,i}(\bar{x},\, t+1)),$$

  где функции $l_{a,i}(\bar{x},\, t+1)$, $r_{a,i}(\bar{x},\, t+1)$, $q_{a,i}(\bar{x},\, t+1)$ для каждой пары a, i определяются индивидуально в зависимости от действия программы машины.
- Отметим, что в задании $\mathrm{Code}(\bar{x},\, t+1)$ не используется операция ограниченного суммирования. С помощью знака суммы сокращена запись обычной конечной суммы $z_1+\ldots+z_k$, которую можно получить суперпозициями функции $x+y$.

Формула Клини

Доказательство (продолжение)

- Итак, мы построили схему примитивной рекурсии для функции $\mathrm{Code}(\bar{x}, t)$. Значит, эта функция примитивно рекурсивна.
- Отметим, что формула Клини корректно работает и тогда, когда машина $\mathcal{M}$ не останавливается. В этом случае результат минимизации будет не определён, а значит не определено и значение $f(\bar{x})$.

$\square$

Формула Клини

Теорема 5

Имеет место равенство $F_{\text{чр}} = F_{\text{выч}}$.

- Класс частично рекурсивных функций задан индуктивным способом и не зависит от устройства каких-либо машин.
- Совпадение класса вычислимых функций с классом частично рекурсивных функций показывает, что класс вычислимых функций «устойчив»: он отражает некоторые содержательные свойства функций, а не какие-то тонкости определения машины Тьюринга.

Лекция 10
Классы P и NP. Проблема выполнимости

Класс P

- Пусть A, B — произвольные алфавиты.
  Рассматриваем частичные функции вида $f\colon A^* \to B^*$.
- Считаем, что $\Lambda \notin A \cup B$ — пустой символ.
- В этом разделе мы не будем делать различий между вычислимостью и правильной вычислимостью.

Определение

Машина Тьюринга $\mathcal{M}$ с алфавитом $A \cup B \cup \{\Lambda\}$ **вычисляет** функцию $f\colon A^* \to B^*$, если, начиная работу на первом символе слова $w \in A^*$ (остальные символы ленты — Λ) в состоянии q_1, машина:

1. Если $f(w)$ определено, то $\mathcal{M}$ через конечное число тактов останавливается, и в этот момент на ленте представлено слово $f(w)$ (остальные символы ленты — Λ), причём головка машина находится на первом символе этого слова.
2. Если $f(w)$ не определено, то $\mathcal{M}$ не останавливается.

Класс P

Определение

- Пусть $T(n)\colon \mathbb{N}_0 \to \mathbb{N}_0$ — всюду определённая функция.
- Функция f вычислима за время $T(n)$, если существует машина Тьюринга $\mathcal{M}$, которая вычисляет функцию f, и при этом для любого слова w длины n время вычисления не превосходит $T(n)$.
- Функция f вычислима за полиномиальное время (полиномиально вычислима), если существует машина Тьюринга $\mathcal{M}$ и полином $p(n)$ с натуральными коэффициентами такие, что $\mathcal{M}$ вычисляет f за время $p(n)$.

- Если функция f зависит от двух переменных ($f\colon A^* \times B^* \to C^*$), то считаем, что вход $w = x\#y$, где $x \in A^*$, $y \in B^*$, $\# \notin A \cup B$.
- Функция f, вычислимая за время $T(n)$, всегда всюду определена.
- Класс полиномиально вычислимых функций не меняется при изменении допустимого размера алфавита и количества дорожек у машин Тьюринга.

Класс P

Определение

Характеристическая функция множества (языка) $L \subseteq A^*$ — это функция $f_L(x)\colon A^* \to \{0,1\}^*$ такая, что

$$f_L(x) = \begin{cases} 0, & \text{если } x \notin L, \\ 1, & \text{если } x \in L. \end{cases}$$

Определение

Класс P — это множество всех языков, характеристические функции которых вычислимы за полиномиальное время.

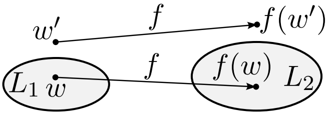
- Язык (множество) — это формализация задачи распознавания.
- Класс P — это класс задач, которые разрешимы «на практике».
- Если задача не принадлежит классу P, то обычно это означает, что для её разрешения требуется неприемлемо много времени.

Полиномиальная сводимость

Определение

- Пусть $L_1 \subseteq A^*$, $L_2 \subseteq B^*$. L_1 **полиномиально сводится** (P-сводится) к L_2 ($L_1 \leqslant_{\mathrm{P}} L_2$), если существует полиномиально вычислимая функция $f \colon A^* \to B^*$ такая, что

$$(\forall w)(w \in L_1 \iff f(w) \in L_2).$$



- Множество L_2 используется как «оракул»: для решения задачи L_1 можно (при помощи функции f) использовать решение L_2.

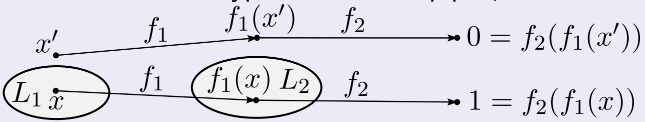
Полиномиальная сводимость

Утверждение

Пусть множество L_1 полиномиально сводится к множеству L_2 и $L_2 \in \mathrm{P}$. Тогда $L_1 \in \mathrm{P}$.

Доказательство

- Пусть f_2 — характеристическая функция множества L_2.
- Поскольку $L_2 \in \mathrm{P}$, существует полином с натуральными коэффициентами $p_2(n)$ такой, что f_2 вычислима за время $p_2(n)$.
- Пусть функция f_1 сводит L_1 к L_2 и вычисляется за время $p_1(n)$, где $p_1(n)$ — полином с натуральными коэффициентами.



- Характеристическая функция множества L_1 есть $f(x) = f_2(f_1(x))$.

Полиномиальная сводимость

Доказательство (продолжение)

- Вычисляем функцию $f(x) = f_2(f_1(x))$. Пусть $n = |x|$.
  - Сначала вычисляем $y = f_1(x)$ за время $p_1(n)$.
  - Длина y не превосходит $n + p_1(n)$, так как на каждом такте машина может добавить символ не более чем в одну пустую ячейку.
  - Далее вычисляем $f(x) = f_2(y)$ за время $p_2(|y|) \leqslant p_2(n + p_1(n))$.
  - Общее время не превосходит $p_1(n) + p_2(n + p_1(n))$ — полином.
- Таким образом, функция $f(x)$ полиномиально вычислима, поэтому $L_1 \in \mathrm{P}$.

$\square$

- Аналогичным образом нетрудно показать, что отношение $\leqslant_{\mathrm{P}}$ рефлексивно и транзитивно.
- Поэтому, если L_1 полиномиально сводится к L_2, то L_1 является «равной по сложности» или «более простой» задачей, чем L_2.

Класс NP

Определение

Недетерминированная машина Тьюринга $\mathcal{M}$ — это набор (A, Q, f, q_1, q_0), где

- $A = \{a_0, \ldots, a_k\}$, $k \geqslant 1$ — алфавит. $a_0 = \Lambda$ — пустой символ.
- $Q \neq \varnothing$ — множество состояний.
- $q_1 \in Q$ — начальное состояние.
- $q_0 \in Q$, $q_0 \neq q_1$ — заключительное состояние.
- $f\colon A \times Q \to 2^{A \times \{L,R,S\} \times Q} \setminus \varnothing$ — программа машины.

- Программу машины можно считать набором команд вида $a_i q_j \to a_r D q_s$, $j \neq 0$. В программе может быть несколько команд с каждой допустимой левой частью:

$$a_i q_j \to a_{r_1} D_1 q_{s_1} \mid a_{r_2} D_2 q_{s_2} \mid \ldots \mid a_{r_l} D_l q_{s_l}.$$

Класс NP

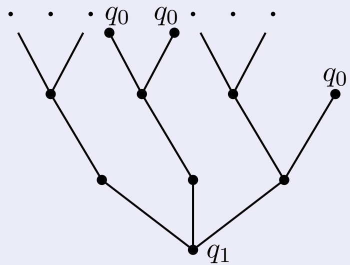
Работа недетерминированной машины

- В начальный момент времени машина находится в состоянии q_1, на ленте записано входное слово $w \in A \setminus \{\Lambda\}$, а головка машины обозревает первый символ этого слова.
- В каждый момент времени машина считывает символ a из обозреваемой головкой ячейки. По этому символу и текущему состоянию q_i машина выбирает команду с левой частью aq_j.
- Если команд с левой частью aq_j несколько, то машина выбирает произвольную.
- После этого машина записывает в текущую ячейку символ b, передвигает головку и переходит в состояние q_j.
- Машина останавливается при переходе в состояние q_0 Такое вычисление называется допускающим (ответ «да»).
- Если этого не происходит, машина работает бесконечно (считаем это ответом «нет»).

Класс NP

Вычисления недетерминированной машины

- На каждом входном слове w недетерминированная машина Тьюринга может отработать несколькими разными способами.
- Все вычисления машины на слове w можно изобразить в виде дерева конфигураций (возможно, бесконечного).
- Ветвление дерева происходит при произвольном выборе команды среди команд с одной и той же левой частью.



Класс NP

Распознавание множеств недетерминированными машинами

- Пусть $\mathcal{M}$ — недетерминированная машина Тьюринга с входным алфавитом A.
- $D(\mathcal{M})$ — это множество всех слов $w \in A^*$ таких, что существует допускающее вычисление машины $\mathcal{M}$ на слове w.

- Иными словами, $D(\mathcal{M})$ — это множество всех слов $w \in A^*$ таких, что в дереве вычислений машины $\mathcal{M}$ имеется хотя бы одна заключительная ветвь (ветвь с заключительным состоянием q_0).
- Недетерминированная машина Тьюринга может выдавать только ответы «да» и «нет». Она может распознавать множества, но не может вычислять функции.

Класс NP

Определение

- Пусть $T(n)\colon \mathbb{N}_0 \to \mathbb{N}_0$ — всюду определённая функция.
- Недетерминированная машина Тьюринга $\mathcal{M}$ распознаёт язык L за время $T(n)$, если $D(\mathcal{M}) = L$, и для любого слова $w \in L$ существует допускающее вычисление $\mathcal{M}$ на слове w длительности не более $T(n)$, где $n = |w|$.
- Недетерминированная машина Тьюринга $\mathcal{M}$ распознаёт язык L за полиномиальное время, если она распознаёт его за время $p(n)$, где $p(n)$ — полином с натуральными коэффициентами.

Определение (основное)

Класс NP — это множество всех языков, распознаваемых на недетерминированных машинах Тьюринга за полиномиальное время.

Альтернативное определение класса NP

Определение (альтернативное)

Класс NP — это класс всех языков L (в произвольных алфавитах A), для которых существует полином $q(n)$ с натуральными коэффициентами, алфавит B и полиномиально вычислимая функция $Q(x, y)\colon A^* \times B^* \to \{0, 1\}^*$ со значениями 0 и 1 такая, что

$$(x \in L) \iff (\exists y)_{|y| \leqslant q(|x|)}(Q(x, y) = 1).$$

- Функция $Q(x, y)$ называется функцией проверки сертификата.
- Слово $y = y(x)$ такое, что $Q(x, y)$ истинно, называется сертификатом для входа x.
- $x \in L$, если существует сертификат для x (имеющий длину, полиномиальную от длины x).

Альтернативное определение класса NP

Утверждение

Основное и альтернативное определения класса NP *равносильны.*

Доказательство

- $\Rightarrow$. Пусть язык $L \subseteq A^*$ распознаётся недетерминированной машиной Тьюринга $\mathcal{M}$ за полиномиальное время $p(n)$.
- Пусть r — максимальное число команд $\mathcal{M}$ с одинаковой левой частью и $B = \{b_1, \ldots, b_r\}$ — алфавит.
- Строим детерминированную машину Тьюринга $\mathcal{M}_Q$ для вычисления $Q(x, y)$:
  - Машина $\mathcal{M}_Q$ имеет две дорожки и моделирует работу $\mathcal{M}$.
  - Первая дорожка хранит содержимое ленты $\mathcal{M}$. В начальный момент — вход x.
  - Вторая дорожка хранит вход $y \in B^*$, указывающий, какие команды машина $\mathcal{M}$ выбирает при недетерминированном вычислении.

Альтернативное определение класса NP

Доказательство (продолжение)

- Машина $\mathcal{M}_Q$ работает следующим образом:
  - Сначала машина $\mathcal{M}_Q$ переписывает y на вторую дорожку.
  - При моделировании каждого такта машина $\mathcal{M}_Q$:
    1. Считывает текущий символ a ленты;
    2. Считывает и стирает очередной символ b слова y;
    3. Определяет выполняемую команду машины $\mathcal{M}$ (если их несколько для данной левой части, то символ b указывает, какую выбрать);
    4. Выполняет выбранную команду.
  - Если машина $\mathcal{M}$ переходит в состояние q_0, то $\mathcal{M}_Q$ завершает вычисление и выдаёт 1.
  - Если слово y закончилось или символ b не указывает ни на одну команду, то $\mathcal{M}_Q$ завершает вычисление и выдаёт 0.
- В качестве $q(n)$ выбираем полином $p(n)$. По построению $\mathcal{M}_Q$ ясно, что заключительная ветвь в вычислении $\mathcal{M}$ на слове x существует тогда и только тогда, когда $(\exists y)_{|y| \leqslant q(|x|)}(Q(x, y) = 1)$.

Альтернативное определение класса NP

Доказательство (продолжение)

- $\Leftarrow$. Пусть для L существует полином $q(n)$ и вычислимая за полиномиальное время $p(n)$ функция $Q(x, y)$ такая, что

$$(x \in L) \iff (\exists y)_{|y| \leqslant q(|x|)}(Q(x, y) = 1).$$

- Строим недетерминированную машину $\mathcal{M}$, работающую следующим образом:
  - Сначала недетерминированно формируется произвольное слово $y \in B^* = \{b_1, \ldots, b_r\}^*$.
  - Для этого на каждом такте машина выбирает одну из команд: дописать символ $b_1, \ldots, b_r$ или завершить формирование y.
  - После этого машина детерминированно вычисляет $Q(x, y)$. Если $Q(x, y) = 1$, то машина останавливается, иначе — зацикливается.
  - На словах из L общее время вычисления хотя бы в одной из ветвей не превосходит $q(n) + p(n + q(n) + 1)$ — полином.

$\square$

Проблема выполнимости

Определение

- Литерал — это формула вида x_k или $\overline{x}_k$.
- Элементарная дизъюнкция (ЭД) — это формула вида $t_{i1} \vee \ldots \vee t_{in_i}$, где все t_{ij} — литералы, а переменные в них различны.
- Конъюнктивная нормальная форма (КНФ) — это формула вида 1 или $D_1 \,\&\, D_2 \,\&\, \ldots \,\&\, D_l$, где все D_i — различные (с точностью до порядка литералов) элементарные дизъюнкции.

Определение

- Пусть F — формула с символами переменных $x_1, \ldots, x_m$, реализующая булеву функцию $f_F(x_1, \ldots, x_m)$, а $\alpha = (a_1, \ldots, a_m) \in \{0, 1\}^m$.
- Набор α выполняет формулу F, если $f_F(a_1, \ldots, a_m) = 1$.
- Формула выполнима, если существует выполняющий её набор.

Проблема выполнимости

Проблема выполнимости (ВЫП или SAT)

- Алфавит: $A = \{(,), x, 0, 1, \neg, \&, \vee\}$.
- **Вход**: КНФ K.
- **Вопрос**: верно ли, что КНФ K выполнима?
- Язык ВЫП состоит из слов в алфавите A^*, которые являются записями выполнимых КНФ.

- При записи КНФ номера переменных записываются в двоичной системе счисления.
- Например, для $K = (x_1 \vee x_2 \vee \overline{x}_3) \,\&\, (\overline{x}_1 \vee x_3)$ имеем

$$(x1 \vee x10 \vee \neg x11) \,\&\, (\neg x1 \vee x11) \in A^*.$$

Проблема выполнимости

Утверждение

ВЫП $\in \mathrm{NP}$.

Доказательство

- Пусть функция $Q(x, y)$ выдаёт 1, если x — КНФ, y — двоичный набор, длина которого равна числу переменных в КНФ x, и набор y выполняет КНФ x.
- Вычисление $Q(x, y)$ можно произвести за полиномиальное время:
  - Проверить корректность КНФ, число переменных и длину набора;
  - Подставить значения из набора на места переменных;
  - Инвертировать значения под отрицаниями;
  - Проверить, во всех ли ЭД есть хотя бы по одной единице.
- Тогда $(x \in \text{ВЫП}) \iff (\exists y)_{|y| \leqslant |x|}(Q(x, y) = 1)$.

$\square$

Лекция 11
Проблема существования клики. NP-полнота.
Теорема Кука

Проблема существования клики

- Рассматриваем простые неориентированные графы (без петель и кратных рёбер).
- Полный граф на k вершинах — это граф с k вершинами, у которого каждая пара вершин соединена ребром.

Определение

- Клика размера k — это полный граф на k вершинах.
- В графе G существует клика размера k, если существует подграф графа G, являющийся кликой размера k.

Проблема существования клики

Проблема существования клики (КЛИКА)

- Алфавит: $A = \{(,\ ),\ [,\ ],\ ;,\ 0,\ 1\}$.
- **Вход**: граф G, натуральное число k.
- **Вопрос**: существует ли в графе G клика размера k?
- Язык КЛИКА состоит из слов в алфавите A^*, которые являются описаниями пар $(G,\ k)$, где G — граф, k — натуральное число, и в G существует клика размера k.

- Считаем, что граф задан списками вершин и рёбер. Номера вершин и число k представлены в двоичном виде.
- Например, для $G = (\{v_1, v_2, v_3, v_4\},\ \{(v_1, v_3),\ (v_3, v_4),\ (v_4, v_1)\})$ и $k = 2$ имеем

$$[1;10;11;100];[(1;11);(11;100);(100;1)];10 \in A^*.$$

Проблема существования клики

Утверждение

КЛИКА $\in \mathrm{NP}$.

Доказательство

- Пусть функция $Q(x,y)$ выдаёт 1, если x — пара (G,k), где G — граф и $k \in \mathbb{N}$, y — список из k номеров вершин G, и граф G имеет клику на вершинах из y.
- Вычисление $Q(x,y)$ можно произвести за полиномиальное время:
  - Проверить корректность описания графа, числа k и списка вершин, проверить число в вершин в списке y;
  - Перебрать все $k(k-1)/2$ пар вершин из y;
  - Для каждой пары (u,v) вершин из y проверить, что в списке рёбер G есть пара (u,v) или (v,u).
- Тогда $(x \in \text{КЛИКА}) \iff (\exists y)_{|y| \leqslant |x|}(Q(x,y)=1)$.

$\square$

Соотношение классов P и NP

Утверждение

$\mathrm{P} \subseteq \mathrm{NP}$.

Доказательство

- Пусть $L \in \mathrm{P}$, т.е. L распознаётся детерминированной машиной Тьюринга за полиномиальное время $p(n)$.
- Дополним эту машину: если после остановки она выдаёт 0, заставляем её вместо этого зациклиться.
- Рассмотрим эту машину как недетерминированную. Она распознаёт язык L за время $p(n)$. Поэтому $L \in \mathrm{NP}$.

$\square$

Соотношение классов P и NP

Содержательный смысл классов P и NP

- P — это класс задач, решение которых требует «не слишком много» времени.
- NP — это класс задач на проверку существования объекта с заданными (полиномиально проверяемыми) свойствами.
- Задачи из NP можно решать перебором, но это требует экспоненциального времени. Неизвестно, можно ли для этих задач придумать алгоритм, избегающий перебора.
- На практике для решения задач из NP применяют SAT-солверы, использующие оптимизированный перебор. Обычно это работает, но нет гарантии быстрой работы во всех случаях.

Соотношение классов P и NP

Проблема соотношения классов P и NP

- Легко видеть, что $\mathrm{P} \subseteq \mathrm{NP}$.
- Вопрос о том, верно ли $\mathrm{P} = \mathrm{NP}$, был поставлен в 1970 г. С. Куком. Это одна из самых известных нерешённых проблем современной математики.
- Большинство специалистов предполагают, что $\mathrm{P} \neq \mathrm{NP}$, но неизвестно, как это можно было бы доказать.
- Этот вопрос имеет большое теоретическое и практическое значение.
- Доказательство $\mathrm{P} = \mathrm{NP}$ позволило бы быстро решать многие прикладные переборные задачи и взламывать ряд кодов.
- Доказательство $\mathrm{P} \neq \mathrm{NP}$ позволило бы получать нижние оценки сложности задач и обосновало бы надёжность ряда криптосистем.

Соотношение классов P и NP

Определение

Множество L является NP-трудным, если к L P-сводится любое множество из класса NP.

Определение

Множество L является NP-полным, если $L \in \mathrm{NP}$ и L NP-трудное.

- NP-полные языки — это «самые сложные» языки класса NP.
- Если какое-то NP-полное множество принадлежит P, то $\mathrm{P} = \mathrm{NP}$.

Теорема Кука

Теорема 6 (С. Кук)

Задача ВЫП *является* NP*-полной.*

- Благодаря этой теореме для решения любой задачи из NP достаточно уметь решать задачу ВЫП (SAT).
- На практике для решения задачи ВЫП используются SAT-солверы, а другие задачи сводятся к ВЫП.

Доказательство

- Ранее было доказано, что ВЫП $\in \mathrm{NP}$. Осталось доказать, что ВЫП NP-трудна.
- Пусть $L \in \mathrm{NP}$ и $L \subseteq A^*$, где $A = \{a_1, \ldots, a_k\}$, $a_0 = \Lambda$.
- По определению NP существует НМТ $\mathcal{M}$ и полином $p(n)$ такие, что $w \in L \iff$ в некотором вычислении на w $\mathcal{M}$ приходит в q_0 через не более $p(|w|)$ тактов.

Теорема Кука

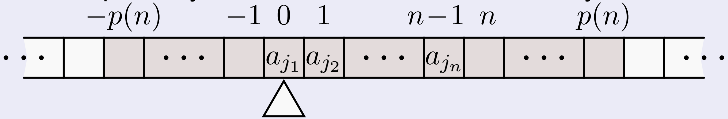
Доказательство (продолжение)

- Будем считать, что в программе машины $\mathcal{M}$ есть команды для заключительного состояния: $a_i q_0 \to a_i S q_0, \ i = \overline{1, r}$.
- $w \in L \iff$ машина $\mathcal{M}$, начиная работу со словом w на ленте, в каком-то вычислении в момент $p(n)$ будет находиться в q_0.
- Покажем, что L P-сводится к ВЫП. Будем строить полиномиально вычислимую функцию $\varphi\colon A^* \to B^*$ такую, что $w \in L \iff F_w = \varphi(w)$ является выполнимой КНФ.
- Конфигурация K_t машины $\mathcal{M}$ в момент времени t представляет собой набор из трёх элементов:
  1. Содержимое ленты;
  2. Положение головки на ленте;
  3. Текущее состояние машины.

Теорема Кука

Доказательство (продолжение)

- Пусть $w = a_{j_1} a_{j_2} \ldots a_{j_n}$ и $p(n) \geqslant n$. Пронумеруем ячейки ленты последовательными целыми числами слева направо, считая нулевой обозреваемую в начале вычисления ячейку.



- При вычислении за время $p(n)$ машина $\mathcal{M}$ не выйдет за пределы области ленты, состоящей из ячеек с номерами от $-p(n)$ до $p(n)$.
- Тогда:
  1. Содержимое ленты — это слово в ячейках от $-p(n)$ до $p(n)$.
  2. Положение головки — это номер ячейки из $\{-p(n), \ldots, p(n)\}$.
  3. Текущее состояние — это номер из $\{0, \ldots, r\}$.
- Конфигурация K_t машины $\mathcal{M}$ полностью определяет все ветви возможных дальнейших вычислений.

Теорема Кука

Доказательство (продолжение)

- Можно записать: $w \in L \iff (\exists K_0)(\exists K_1)\ldots(\exists K_{p(|w|)})$ такие, что выполнены все следующие условия ($n = |w|$):
  1. K_0 — начальная конфигурация для слова w;
  2. $K_{p(n)}$ содержит состояние q_0;
  3. K_{t+1} можно получить из K_t за один такт работы машины $\mathcal{M}$, $t = \overline{0,\ p(n)-1}$.
- Выразим условия на конфигурации с помощью КНФ $F_w = \varphi(w)$.
- Вводим три типа булевых переменных:
  - $x_{i,j}^t$: $(x_{i,j}^t = 1) \iff$ в K_t в ячейке i записан символ a_j;
  - y_i^t: $(y_i^t = 1) \iff$ в K_t головка обозревает ячейку i;
  - z_l^t: $(z_l^t = 1) \iff$ в K_t машина находится в состоянии q_l.
  - Здесь $t = \overline{0,\ p(n)}$, $i = \overline{-p(n),\ p(n)}$, $j = \overline{0,\ k}$, $l = \overline{0,\ r}$.

Теорема Кука

Доказательство (продолжение)

- Строим КНФ F_w, принимающую значение 1 на наборе значений своих переменных, если выполнены все следующие условия:
  1. Набор корректно задаёт последовательность конфигураций $K_0, \dots, K_{p(n)}$;
  2. Конфигурация K_0 является правильной начальной конфигурацией для входа w;
  3. Конфигурация $K_{p(n)}$ сдержит состояние q_0;
  4. Для всякого $t \in \{0, \dots, p(n) - 1\}$ конфигурация K_{t+1} может быть получена из K_t согласно программе $\mathcal{M}$ за один такт работы.
- Итоговая КНФ будет конъюнкцией КНФ F_1, F_2, F_3, F_4, реализующих указанные условия по отдельности.

Теорема Кука

Доказательство (продолжение)

- Условие 1: «Корректная последовательность конфигураций».
- При каждом t должны выполняться все следующие условия:
  - *В каждой ячейке один символ*: для любого i ровно одна переменная $x^t_{i,j}$ (при различных j) принимает значение 1;
  - *Головка обозревает одну ячейку*: ровно одна переменная y^t_i (при различных i) принимает значение 1;
  - *Машина находится в одном состоянии*: ровно одна переменная z^t_l (при различных l) принимает значение 1.
- Чтобы выразить это условие с помощью КНФ, введём вспомогательную функцию.

Теорема Кука

Доказательство (продолжение)

- Обозначим
$$h(v_1, \ldots, v_s) = (v_1 \vee \ldots \vee v_s) \mathbin{\&} \mathop{\&}_{\substack{i,j=\overline{1,s} \\ i<j}} (\overline{v}_i \vee \overline{v}_j).$$
- Функция h принимает значение 1, если ровно одна из её переменных содержит 1.
- Для этой функции выписана КНФ. Её ранг (число символов переменных) равен s^2.

Теорема Кука

Доказательство (продолжение)

- Теперь выпишем КНФ для условия 1:

$$F_1 = \mathop{\&}_{t=0}^{p(n)} \Bigg(\Bigg(\mathop{\&}_{i=-p(n)}^{p(n)} h(x_{i,0}^t, \ldots, x_{i,k}^t) \Bigg) \& \\ \& \, h(y_{-p(n)}^t, \ldots, y_{p(n)}^t) \,\&\, h(z_0^t, \ldots, z_r^t) \Bigg)$$

- В этой КНФ $(p(n)+1)((2p(n)+1)(k+1)^2+(2p(n)+1)^2+(r+1)^2)$ символов переменных, и длина её записи полиномиальна от n.
- Поэтому её можно построить за полиномиальное от n время. КНФ F_1 зависит только от чисел $n=|w|$, $p(n)$, k, r.

Теорема Кука

Доказательство (продолжение)

- Условие 2: «Правильная начальная конфигурация».
- При $t = 0$ должны выполняться все следующие условия:
  - В ячейках $0, \ldots, n-1$ символы слова w, остальные ячейки пусты;
  - Головка обозревает ячейку 0;
  - Машина находится в состоянии q_1.
- Выразим это условие с помощью КНФ:

$$F_2 = x^0_{0,\,j_1} \mathbin{\&} \ldots \mathbin{\&} x^0_{n-1,\,j_n} \mathbin{\&} \\ \mathbin{\&} \left(\mathop{\&}_{i=-p(n)}^{-1} x^0_{i,0} \right) \mathbin{\&} \left(\mathop{\&}_{i=n}^{p(n)} x^0_{i,0} \right) \mathbin{\&} y^0_0 \mathbin{\&} z^0_1.$$

- Ранг этой КНФ $2p(n) + 3$, длина её записи полиномиальна от n.
- Поэтому её можно построить за полиномиальное от n время.

Теорема Кука

Доказательство (продолжение)

- Условие 3: «Заключительная конфигурация содержит q_0».
- Это условие элементарно выражается с помощью КНФ:

$$F_3 = z_0^{p(n)}.$$

- Ранг этой КНФ равен 1, длина её записи полиномиальна от n.
- Поэтому её можно построить за полиномиальное от n время.

Лекция 12
Завершение доказательства теоремы Кука.
Проблемы 3-ВЫП и 2-ВЫП

Теорема Кука

Доказательство (продолжение)

- Условие 4: «K_{t+1} может быть получена из K_t за один шаг».
- Пусть для каждой левой части $a_j q_l$ программа машины $\mathcal{M}$ имеет набор команд $a_j q_l \to a_{\sigma_p(j,l)} D_p(j,l) q_{\tau_p(j,l)}, \; p = \overline{1, \, c(j,l)}$.
- Считаем, что $D_p(j,l) \in \{-1, \, 0, \, 1\}$.
- Распишем условие 4. При каждом t, i, j, l:
  - Пусть в момент t головка обозревает i-ю ячейку, в ней находится символ a_j и машина находится в состоянии q_l.
  - Тогда существует такое p, что в момент $t+1$ в i-й ячейке будет символ $a_{\sigma_p(j,l)}$, головка будет обозревать ячейку $i + D_p(j,l)$, а машина будет в состоянии $q_{\tau_p(j,l)}$.
  - Если в момент t головка не обозревает i-ю ячейку, то в момент $t+1$ в ней будет тот же символ, что и в момент t.

Теорема Кука

Доказательство (продолжение)

- Выразим условие 4 с помощью булевой формулы:

$$F_4' = \mathop{\&}_{t=0}^{p(n)-1} \mathop{\&}_{i=-p(n)}^{p(n)} \mathop{\&}_{j=0}^{k} \mathop{\&}_{l=0}^{r} \Bigg(\Bigg(x_{i,j}^t \,\&\, y_i^t \,\&\, z_l^t \rightarrow$$

$$\rightarrow \bigvee_{p=0}^{c(j,l)} x_{i,\sigma_p(j,l)}^{t+1} \,\&\, y_{i+D_p(j,l)}^{t+1} \,\&\, z_{\tau_p(j,l)}^{t+1} \Bigg) \,\&\, \Big(\overline{y}_i^t \rightarrow (x_{i,j}^{t+1} \sim x_{i,j}^t) \Big) \Bigg).$$

- Часть формулы во внешних скобках зависит от не более $3 + (k+1) + 3 + (r+1) = 8 + k + r$ переменных (не зависит от n).
- Перепишем эту часть формулы в виде совершенной КНФ. Она будет иметь не более $(8 + k + r)2^{8+k+r}$ символов переменных.
- Получим КНФ F_4 с $p(n)(2p(n) + 1)(k + 1)(r + 1)(8 + k + r)2^{8+k+r}$ символами переменных — длина записи полиномиальна от n.

Теорема Кука

Доказательство (продолжение)

- КНФ F_4 можно построить за полиномиальное от n время.
- Наконец, получаем КНФ $F_w = F_1 \mathbin{\&} F_2 \mathbin{\&} F_3 \mathbin{\&} F_4$. Она строится по слову w и машине $\mathcal{M}$ за полиномиальное от $n = |w|$ время.
- Данная КНФ принимает значение 1, если набор значений переменных «изображает» последовательность конфигураций «успешного» вычисления $\mathcal{M}$ (в котором она останавливается).
- Поэтому КНФ F_w выполнима $\iff$ существует успешное вычисление $\mathcal{M}$ $\iff$ $w \in L$.
- Таким образом, произвольный язык L P-сводится к ВЫП.
- В силу этого задача ВЫП NP-трудна, а значит и NP-полна.

$\square$

Проблема 3-выполнимости

Определение

3-КНФ — это КНФ, в которой каждая элементарная дизъюнкция имеет не более трёх литералов.

Проблема 3-выполнимости (3-ВЫП)

- Алфавит: $A = \{(,), x, 0, 1, \neg, \&, \vee\}$.
- **Вход**: 3-КНФ K.
- **Вопрос**: верно ли, что 3-КНФ K выполнима?
- Язык 3-ВЫП состоит из слов в алфавите A^*, которые являются записями выполнимых 3-КНФ.

- При записи КНФ номера переменных записываются в двоичной системе счисления.

Проблема 3-выполнимости

Теорема 7

Задача 3-ВЫП *является* NP*-полной.*

- Эта теорема также была доказана С. Куком.

Доказательство

- Задача 3-ВЫП является частным случаем ВЫП. Проверка того, что КНФ является 3-КНФ, полиномиальна, поэтому 3-ВЫП $\in \mathrm{NP}$.
- В силу теоремы Кука, чтобы доказать NP-трудность 3-ВЫП, достаточно доказать, что ВЫП полиномиально сводится к 3-ВЫП.
- Пусть $K = D_1 \,\&\, \ldots \,\&\, D_k$ — произвольная КНФ. Преобразуем её в 3-КНФ K' с сохранением выполнимости / невыполнимости.
- Преобразуем каждую ЭД D_i в КНФ F_i. Если D_i имеет не более 3 литералов, то $F_i = D_i$.

Проблема 3-выполнимости

Доказательство (продолжение)

- Иначе $D_i = (t_1 \vee t_2 \vee \ldots \vee t_m)$, $m > 3$, где t_i — литералы. Строим

$$F_i = (t_1 \vee t_2 \vee y_1) \,\&\, (\overline{y}_1 \vee t_3 \vee y_2) \,\&\, (\overline{y}_2 \vee t_4 \vee y_3) \,\&\, \ldots$$
$$\ldots \,\&\, (\overline{y}_{m-4} \vee t_{m-2} \vee y_{m-3}) \,\&\, (\overline{y}_{m-3} \vee t_{m-1} \vee t_m).$$

- Здесь $y_1, y_2, \ldots, y_{m-3}$ — переменные, отсутствующие в КНФ K. Для разных КНФ F_i используем непересекающиеся наборы переменных y_j.
- Получаем $K' = F_1 \,\&\, \ldots \,\&\, F_k$.
- Очевидно, ранг F_i не превосходит $3m$. Поэтому длина записи K' полиномиальна от длины записи K, а построение K' требует полиномиального от длины K времени.

Проблема 3-выполнимости

Доказательство (продолжение)

- Покажем, что если F_i выполнима, то и D_i выполнима.
- Пусть $\alpha = (a_1, \ldots, a_n;\ b_1, \ldots, b_{m-3})$, где a_i — значения переменных КНФ K, а b_j — значения новых переменных y_j.
- Пусть $F_i(\alpha) = 1$. Тогда
  - Если $b_1 = 0$, то $t_1(\alpha) \vee t_2(\alpha) = 1$, т.е. $D_i(\alpha) = 1$.
  - Если $b_{m-3} = 1$, то $t_{m-1}(\alpha) \vee t_m(\alpha) = 1$, т.е. $D_i(\alpha) = 1$.
  - Пусть $b_1 = 1$ и $b_{m-3} = 0$. Тогда существует k: $b_k = 1$ и $b_{k+1} = 0$. Имеем $\bar{b}_k \vee t_{k+2}(\alpha) \vee b_{k+1} = 1$, т.е. $t_{k+2}(\alpha) = 1$. Тогда $D_i(\alpha) = 1$.
- Таким образом, если F_i выполнима, то и D_i выполнима.

Проблема 3-выполнимости

Доказательство (продолжение)

- Теперь покажем, что если D_i выполнима, то и F_i выполнима.
- Пусть $\alpha = (a_1, \ldots, a_n)$ и $D_i(\alpha) = 1$.
- Тогда существует такое k, что $t_k(\alpha) = 1$.
- Построим набор $\beta = (\alpha;\ b_1, \ldots, b_{m-3})$ такой, что $F_i(\beta) = 1$.
  - Если $k \in \{1,\ 2\}$, то выбираем $b_1 = \ldots = b_{m-3} = 0$.

    ЭД $t_1 \vee t_2 \vee y_1$ обращается в 1 из-за $t_k(\beta) = 1$, а остальные ЭД F_i содержат $\overline{y}_j(\beta) = 1$.
  - Если $k \in \{m-1,\ m\}$, то выбираем $b_1 = \ldots = b_{m-3} = 1$.

    ЭД $\overline{y}_{m-3} \vee t_{m-1} \vee t_m$ обращается в 1 из-за $t_k(\beta) = 1$, а остальные ЭД F_i содержат $y_j(\beta) = 1$.
  - Иначе выбираем $b_1 = \ldots = b_{k-2} = 1$ и $b_{k-1} = \ldots = b_{m-3} = 0$.

    ЭД $\overline{y}_{k-2} \vee t_k \vee y_k$ обращается в 1 из-за $t_k(\beta) = 1$, а остальные ЭД F_i содержат $y_j(\beta) = 1$ ($j \leqslant k-2$) или $\overline{y}_l(\beta) = 1$ ($l \geqslant k-1$).

Проблема 3-выполнимости

Доказательство (продолжение)

- Итак, F_i выполнима тогда и только тогда, когда выполнима D_i.
- Значит, K' выполнима тогда и только тогда, когда выполнима K.

$\square$

Проблема 2-выполнимости

Определение

2-КНФ — это КНФ, в которой каждая элементарная дизъюнкция имеет не более двух литералов.

Проблема 2-выполнимости (2-ВЫП)

- Алфавит: $A = \{(,), x, 0, 1, \neg, \&, \vee\}$.
- **Вход**: 2-КНФ K.
- **Вопрос**: верно ли, что 2-КНФ K выполнима?
- Язык 2-ВЫП состоит из слов в алфавите A^*, которые являются записями выполнимых 2-КНФ.

- При записи КНФ номера переменных записываются в двоичной системе счисления.

Проблема 2-выполнимости

Теорема 8

Задача 2-ВЫП *принадлежит классу* P*.*

Доказательство

- Построим полиномиальный алгоритм решения задачи 2-ВЫП.
- Пусть $K(x_1, \ldots, x_n)$ — 2-КНФ, содержащая только символы переменных $x_1, \ldots, x_n$.
- Если $n = 1$, то K имеет вид 1, x_1, $\overline{x}_1$ или $x_1\overline{x}_1$. В первых трёх случаях K выполнима, а в последнем невыполнима.
- Пусть $n \geqslant 2$. Покажем, что можно исключить из КНФ K переменную x_n с сохранением выполнимости / невыполнимости.
- Имеем $K = K' \,\&\, (x_n \vee t_1) \ldots (x_n \vee t_k) \,\&\, (\overline{x}_n \vee t'_1) \ldots (\overline{x}_n \vee t'_m)$, где K' — 2-КНФ без x_n и $\overline{x}_n$, а все t_i и t'_i — литералы или нули.

Проблема 2-выполнимости

Доказательство (продолжение)

- КНФ $K(x_1, \ldots, x_n)$ выполнима $\iff$ выполнима формула
$$F = K(x_1, \ldots, x_{n-1}, 0) \vee K(x_1, \ldots, x_{n-1}, 1).$$
- В формуле F множитель K' можно вынести за скобки. Тогда получим
$$F = K' \,\&\, (t_1 \ldots t_k \vee t'_1 \ldots t'_m).$$
- Используя тождество $x \vee yz = (x \vee y) \,\&\, (x \vee z)$, преобразуем
$$F = K' \,\&\, (t_1 \ldots t_k \vee t'_1 \ldots t'_m) = K' \,\&\, \mathop{\&}_{\substack{i=\overline{1,k}\\ j=\overline{1,m}}} (t_i \vee t'_j).$$
- Если $k = 0$ или $m = 0$, то $F = K'$.

Проблема 2-выполнимости

Доказательство (продолжение)

$$F = K' \mathbin{\&} \mathop{\Large\&}_{\substack{i=\overline{1,k}\\ j=\overline{1,m}}} (t_i \vee t'_j)$$

- Совершаем простейшие поглощения. Если есть скобка $0 \vee 0$, то заменяем КНФ на $x_1\overline{x}_1$. Иначе устраняем константы и дубликаты, применяя тождества $1 \cdot x = x$, $0 \vee x = x$, $x \vee x = x$ и $x \cdot x = x$.
- Получили 2-КНФ. Поскольку различных ЭД $t_1 \vee t_2$ не более $(2n)^2$, ранг полученной 2-КНФ не превосходит $2 \cdot (2n)^2 = 8n^2$.
- Последовательно исключаем из КНФ K переменные $x_n, \ldots, x_2$ и сводим задачу к проверке выполнимости КНФ с одной переменной x_1, которая уже рассмотрена ранее.

Проблема 2-выполнимости

Доказательство (продолжение)

- На каждом шаге мы получаем КНФ ранга не более $8n^2$, т.е. КНФ с длиной записи, полиномиальной от длины записи K.
- Поэтому каждый шаг требует полиномиального от длины записи КНФ K времени, а всего шагов n.
- Таким образом, приведённый алгоритм проверки выполнимости задачи 2-ВЫП является полиномиальным.

$\square$

- Итак, задача 3-ВЫП NP-полна, а задача 2-ВЫП полиномиально разрешима.

Литература

1. Лекции С. С. Марченкова: Плейлист на YouTube
2. Марченков С. С. Избранные главы дискретной математики. — М.: МАКС Пресс, 2016. — 136 с.
   https://mk.cs.msu.ru/images/2/25/ИзбрГлавыДискрМатем_2015.pdf
3. Яблонский С. В. Введение в дискретную математику. — М.: Наука, 1986. — 384 с.
4. Алексеев В. Б. Введение в теорию сложности алгоритмов. — М.: Издательский отдел ф-та ВМиК МГУ, 2002. — 82 с.
   https://mk.cs.msu.ru/images/c/c4/KNIGA1.pdf
5. https://docs.python.org/3/library/re.html
6. https://ru.wikipedia.org/wiki/Регулярные_выражения