Математическая логика, алгоритмические проблемы, вычислительная сложность

Михаил Николаевич Рыбаков

Тверской государственный университет

январь 2020 года

Структура курса

- Математическая логика

  классическая логика высказываний; классическая логика предикатов; логические законы; выразимость и невыразимость условий; неклассические логики; конструктивные и неконструктивные законы

- Алгоритмические проблемы

  алгоритмы; машины Тьюринга; вычислимые функции; разрешимые и неразрешимые проблемы; неразрешимость логики предикатов; разрешимые и неразрешимые теории; рекурсивно перечислимые теории; арифметическая иерархия множеств

- Вычислительная сложность

  сложность алгоритмов; меры сложности; классы сложности; сложные задачи; некоторые теоремы; открытые проблемы теории вычислительной сложности

Математическая логика

Классическая логика высказываний: язык

Исходные символы языка логики высказываний:

- пропозициональные переменные $p_1, p_2, p_3, \ldots$
- логические связки $\wedge$, $\vee$, $\to$, $\leftrightarrow$, $\neg$, $\bot$, $\top$
- скобки

Формулы языка логики высказываний:

- $\bot$, $\top$, а также переменные $p_1, p_2, p_3, \ldots$ являются элементарными (атомарными) формулами
- если A и B — формулы, то $(A \wedge B)$, $(A \vee B)$, $(A \to B)$, $(A \leftrightarrow B)$, $\neg A$ — тоже формулы
- никаких других формул нет

Замечание. При записи формул некоторые скобки опускают, считая силу связок по убыванию: $\neg$, $\wedge$, $\vee$, $\leftrightarrow$, $\to$.

Прочтение связок

Конъюнкция $A \wedge B$:

- A и B
- A, да и B
- A, а B
- A, но B
- A, а и B
- A, но и B
- A вместе с B
- как A, так и B
- A, в то время как B
- A, хотя и B
- не только A, но и B
- A, несмотря на то, что B

Дизъюнкция $A \vee B$:

- A или B
- A или B или оба,
- A, если не B
- A и/или B

Отрицание $\neg A$:

- не A
- не верно, что A
- A не имеет места
- A не верно.

Прочтение связок

Импликация $A \to B$:

- если A, то B
- из A следует B
- B следует из A
- коль скоро A, то и B
- в случае A имеет место B
- для A необходимо B
- для B достаточно A
- B, если A
- A, только если B
- A имплицирует B

Эквивалентность $A \leftrightarrow B$:

- A тогда и только тогда, когда B
- A если и только если B
- A в том и только в том случае, если B
- A в том и только в том случае, когда B
- если A, то B, и обратно
- A, если B, и B, если A
- для A необходимо и достаточно B
- A эквивалентно B
- A равносильно B

Семантика: таблицы истинности

A	B	$A \wedge B$	$A \vee B$	$A \to B$	$A \leftrightarrow B$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

A	$\neg A$	$\bot$	$\top$
0	1	0	1
1	0	0	1

Тождественно истинная формула — это формула, которая *на всех* наборах принимает значение 1.

Тождественно ложная формула — это формула, которая *на всех* наборах принимает значение 0.

Выполнимая формула — это формула, которая *на некотором* наборе принимает значение 1.

Построение формулы по таблице истинности

p	q	r	$F(p,q,r)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Построение формулы по таблице истинности

p	q	r	$F(p,q,r)$	
0	0	0	0	$\neg p \wedge \neg q \wedge \neg r$
0	0	1	1	$\neg p \wedge \neg q \wedge r$
0	1	0	0	$\neg p \wedge q \wedge \neg r$
0	1	1	0	$\neg p \wedge q \wedge r$
1	0	0	0	$p \wedge \neg q \wedge \neg r$
1	0	1	1	$p \wedge \neg q \wedge r$
1	1	0	1	$p \wedge q \wedge \neg r$
1	1	1	1	$p \wedge q \wedge r$

Построение формулы по таблице истинности

p	q	r	$F(p,q,r)$	
0	0	0	0	$\neg p \wedge \neg q \wedge \neg r$
0	0	1	1	$\neg p \wedge \neg q \wedge r$
0	1	0	0	$\neg p \wedge q \wedge \neg r$
0	1	1	0	$\neg p \wedge q \wedge r$
1	0	0	0	$p \wedge \neg q \wedge \neg r$
1	0	1	1	$p \wedge \neg q \wedge r$
1	1	0	1	$p \wedge q \wedge \neg r$
1	1	1	1	$p \wedge q \wedge r$

$$A = (\neg p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge r) \vee (p \wedge q \wedge \neg r) \vee (p \wedge q \wedge r)$$

Построение формулы по таблице истинности

p	q	r	$F(p,q,r)$	
0	0	0	0	$\neg p \wedge \neg q \wedge \neg r$
0	0	1	1	$\neg p \wedge \neg q \wedge r$
0	1	0	0	$\neg p \wedge q \wedge \neg r$
0	1	1	0	$\neg p \wedge q \wedge r$
1	0	0	0	$p \wedge \neg q \wedge \neg r$
1	0	1	1	$p \wedge \neg q \wedge r$
1	1	0	1	$p \wedge q \wedge \neg r$
1	1	1	1	$p \wedge q \wedge r$

$$A = (\neg p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge r) \vee (p \wedge q \wedge \neg r) \vee (p \wedge q \wedge r)$$

$$B = \neg(\neg p \wedge \neg q \wedge \neg r) \wedge \neg(\neg p \wedge q \wedge \neg r) \wedge \neg(\neg p \wedge q \wedge r) \wedge \neg(p \wedge \neg q \wedge \neg r)$$

Логические законы

Равносильные формулы — это формулы, у которых совпадают таблицы истинности.
Обозначение: $A \equiv B$.

- $p \wedge p \equiv p$ — идемпотентность конъюнкции
- $p \vee p \equiv p$ — идемпотентность дизъюнкции
- $p \wedge q \equiv q \wedge p$ — коммутативность конъюнкции
- $p \vee q \equiv q \vee p$ — коммутативность дизъюнкции
- $p \wedge \bot \equiv \bot,\ p \wedge \top \equiv p,\ p \vee \bot \equiv p,\ p \vee \top \equiv \top,\ \bot \to p,\ p \to \top$
- $p \wedge \neg p \to q$ — закон Дунса Скота
- $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$ — ассоциативность конъюнкции
- $p \vee (q \vee r) \equiv (p \vee q) \vee r$ — ассоциативность дизъюнкции
- $(p \wedge q) \vee q \equiv q,\ p \wedge (p \vee q) \equiv p$ — законы поглощения
- $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ — дистрибутивность конъюнкции относительно дизъюнкции
- $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ — дистрибутивность дизъюнкции относительно конъюнкции

Логические законы

Равносильные формулы — это формулы, у которых совпадают таблицы истинности.
Обозначение: $A \equiv B$.

- $p \to (q \to p)$ — *verum ex quolibet* (истина из чего угодно)
- $(p \to q) \to ((q \to r) \to (p \to r))$ — закон силлогизма
- $(p \to (q \to r)) \to ((p \to q) \to (p \to r))$
- $p \wedge q \to p,\ p \to p \vee q$
- $(p \vee q) \wedge (p \vee \neg q) \equiv p$
- $p \to (q \to p \wedge q)$ — введение конъюнкции
- $(p \to (q \to r)) \to (p \wedge q \to r)$ — соединение посылок
- $(p \wedge q \to r) \to (p \to (q \to r))$ — разделение посылок
- $\left.\begin{array}{l} \neg(p \vee q) \equiv \neg p \wedge \neg q \\ \neg(p \wedge q) \equiv \neg p \vee \neg q \end{array}\right\}$ — законы Де Моргана
- $(p \to q) \equiv \neg p \vee q$
- $(p \to q) \equiv \neg(p \wedge \neg q)$

Логические законы

Равносильные формулы — это формулы, у которых совпадают таблицы истинности.
Обозначение: $A \equiv B$.

- $((p \to q) \to p) \to p$ — закон Пирса
- $(\neg p \to p) \to p$ — закон Клавия
- $p \vee \neg p$ — *tertium non datur* («третьего не дано»)
- $p \to p$ — закон тождества
- $\neg(p \wedge \neg p)$ — закон (отсутствия) противоречия
- $\neg p \to (p \to q)$ — *ex falso quolibet* (из ложного что угодно)
- $(p \to q) \equiv (\neg q \to \neg p)$ — закон контрапозиции
- $(p \to q) \wedge p \to q$ — *modus ponens*
- $(p \to q) \wedge \neg q \to \neg p$ — *modus tollens*
- $p \equiv \neg\neg p$ — закон двойного отрицания

Исчисление высказываний

Аксиомы (в языке с $\wedge$, $\vee$, $\to$, $\bot$):

(A1) $p_0 \to (p_1 \to p_0)$

(A2) $(p_0 \to (p_1 \to p_2)) \to ((p_0 \to p_1) \to (p_0 \to p_2))$

(A3) $p_0 \wedge p_1 \to p_0$

(A4) $p_0 \wedge p_1 \to p_1$

(A5) $p_0 \to (p_1 \to p_0 \wedge p_1)$

(A6) $p_0 \to p_0 \vee p_1$

(A7) $p_1 \to p_0 \vee p_1$

(A8) $(p_0 \to p_2) \to ((p_1 \to p_2) \to (p_0 \vee p_1 \to p_2))$

(A9) $\bot \to p_0$

(A10) $p_0 \vee (p_0 \to \bot)$

Правила вывода:

Modus ponens: из формул A и $A \to B$ получается B

Подстановка: из формулы A получается $\mathbf{s}A$,

где $\mathbf{s}$ — функция, заменяющая в A вхождения переменных некоторыми фиксированными формулами.

Выводимость формул

Вывод (в классическом исчислении) — это список формул, каждая из которых — аксиома или получена из предыдущих формул списка по одному из правил вывода.

Формула A называется выводимой, если она является последней в некотором выводе.
Обозначение: $\vdash A$.

Вывод из множества гипотез Γ — это конечный список формул, каждая из которых является выводимой в исчислении высказываний, или принадлежит Γ, или получена из предыдущих формул списка по правилам, отличным от подстановки.

Формула A выводима из гипотез Γ, если A является последней в некотором выводе из Γ.
Обозначение: $\Gamma \vdash A$

Замечание. Таким образом, подстановку можно применять только к формулам, полученным без использования Γ.

Пример вывода

Покажем, что $A \wedge B \vdash B \wedge A$:

(1)	$A \wedge B$	гипотеза
(2)	$A \wedge B \to A$	подстановка в аксиому (A3)
(3)	$A \wedge B \to B$	подстановка в аксиому (A4)
(4)	A	modus ponens, (1), (2)
(5)	B	modus ponens, (1), (3)
(6)	$B \to (A \to B \wedge A)$	подстановка в аксиому (A5)
(7)	$A \to B \wedge A$	modus ponens, (5), (6)
(8)	$B \wedge A$	modus ponens, (4), (7)

Теоремы о дедукции и о полноте

Теорема о дедукции

$\Gamma \vdash A \to B \iff \Gamma, A \vdash B$

Пример использования:

$$\begin{aligned} & \vdash (A \to B) \to ((B \to C) \to (A \to C)) \\ \iff & A \to B \vdash (B \to C) \to (A \to C) \\ \iff & A \to B, B \to C \vdash A \to C \\ \iff & A \to B, B \to C, A \vdash C \end{aligned}$$

Последняя выводимость обосновывается легко:
$A, A \to B, B, B \to C, C.$

Теорема о полноте

$\vdash A \iff A$ тождественно истинна

Другие семантики классической логики высказываний

- Алгебры вида $\langle 2^X; \cap, \cup, \bar{\cdot} \rangle$

Теорема о полноте

$\vdash A \iff$ значение A всегда равно X

- Булевы алгебры $\langle B; \wedge, \vee, \neg, 0, 1 \rangle$

Теорема о полноте

$\vdash A \iff$ значение A всегда равно 1

Предикаты и отношения

Начнём с примеров:

- [верно, что] x и y — родственники
- [верно, что] x и y находятся в отношении родства
- [верно, что] число x больше числа y
- [верно, что] числа x и y находятся в отношении «больше»

Функция $P : D^n \to \{0, 1\}$ называется n-арным предикатом на множестве D.

Если $R \subseteq D^n$, то R называется n-арным отношением на D.

Замечание. Мы можем не различать n-арный предикат P и n-арное отношение R на множестве D, если для любых $a_1, \ldots, a_n \in D$ выполнено следующее:

$$P(a_1 \ldots, a_n) = 1 \iff (a_1, \ldots, a_n) \in R.$$

Классическая логика предикатов: язык

Исходные символы языка логики предикатов:

- связанные предметные переменные $x_1, x_2, x_3, \ldots$
- свободные предметные переменные $a_1, a_2, a_3, \ldots$
- функциональные буквы $f_1, f_2, f_3, \ldots$ (арностей $m_1, m_2, m_3, \ldots$)
- предикатные буквы $P_1, P_2, P_3, \ldots$ (арностей $k_1, k_2, k_3, \ldots$)
- логические связки $\wedge$, $\vee$, $\to$, $\bot$
- кванторные символы $\forall$, $\exists$
- скобки и запятая

Замечание. Наборы предикатных и функциональных букв могут варьироваться; в общем случае язык логики предикатов определяют в некоторой *сигнатуре*.

Классическая логика предикатов: язык

Термы:

- свободная переменная является термом
- если $t_1, \ldots, t_n$ — термы, f — n-арная функциональная буква, то $f(t_1, \ldots, t_n)$ — тоже терм

Примеры: $\sin x^2$, $(x + y) \cdot z^3$, $1 + x + x^2 + x^3 + x^4 + x^5$

Формулы языка логики предикатов:

- $\bot$ — элементарная (атомарная) формула
- если $t_1, \ldots, t_n$ — термы, P — n-арная предикатная буква, то $P(t_1, \ldots, t_n)$ — элементарная (атомарная) формула
- если A и B — формулы, то $(A \wedge B)$, $(A \vee B)$, $(A \to B)$ — тоже формулы
- если A — формула, x — связанная переменная, то $\forall x\, A[a/x]$ и $\exists x\, A[a/x]$ — тоже формулы (запись $A[a/x]$ означает, что в A каждое вхождение свободной переменной a заменено вхождением связанной переменной x)

Примеры термов и формул

- термы:
  $a \cdot b$, c, a^2

  формулы:
  $a \cdot b = c$, $\exists z(a \cdot b = z)$, $\forall x \forall y \exists z(x \cdot y = z)$
  $c = a^2$, $\exists x(c = x^2)$, $\forall y \exists x(y = x^2)$

- термы:
  $f(a, b)$, c, $g(a)$

  формулы:
  $E(f(a, b), c)$, $\exists z\, E(f(a, b), z)$, $\forall x \forall y \exists z\, E(f(x, y), z)$
  $c = g(a)$, $\exists x\, E(c, g(x))$, $\forall y \exists x\, E(y, g(x))$

Кванторы

- $\exists x$ — квантор существования по переменной x
- $\forall x$ — квантор всеобщности по переменной x

Кванторы вида $\exists x$ и $\forall x$ называют кванторами по предметным переменным, или кванторами *первого порядка*.

Замечание. Кванторы второго порядка: $\exists P$, $\exists f$, $\forall P$, $\forall f$.

Прочтение кванторов

- $\forall x A(x)$:
  - для любого x (имеет место) $A(x)$
  - $A(x)$ при произвольном x
  - для всех x (верно) $A(x)$
  - $A(x)$, каково бы ни было x
  - для каждого x (верно) $A(x)$
  - всегда имеет место $A(x)$
  - каждый обладает свойством A
  - свойство A присуще всем
  - всё удовлетворяет A
  - любой объект является A
  - всякая вещь обладает свойством A

Прочтение кванторов

- $\exists x A(x)$:
  - для некоторых x (имеет место) $A(x)$
  - для подходящего x (верно) $A(x)$
  - существует x, для которого (такой, что) $A(x)$
  - имеется x, для которого (такой, что) $A(x)$
  - найдется x, для которого (такой, что) $A(x)$
  - у некоторых вещей есть признак $A(x)$
  - хотя бы для одного x (верно) $A(x)$
  - кто-нибудь относится к (есть, является) $A(x)$
  - по крайней мере один объект есть A

Прочтение кванторов

- $\neg\forall x A(x)$:
  - не для каждого x (верно) $A(x)$
  - не при всяком x (верно) $A(x)$
  - $A(x)$ оказывается истинным не для всех x
  - не все обладают свойством A
  - не все суть A
  - A не всегда верно
- $\forall x\neg A(x)$:
  - для всякого x неверно $A(x)$
  - $A(x)$ всегда ложно
  - ничто не обладает свойством $A(x)$
  - все суть не $A(x)$

Прочтение кванторов

- $\neg\exists x A(x)$:
  - не существует x, такого, что $A(x)$
  - нет такого x, что $A(x)$
  - нет никакого x, что $A(x)$
  - $A(x)$ не выполняется ни для какого x
  - ничто не обладает свойством A
  - ничто не есть A
  - ни для какого x не верно A
  - не верно, что для некоторых x (верно) A
- $\exists x \neg A(x)$:
  - для некоторого x не (верно) $A(x)$
  - что-то не обладает свойством A
  - что-то есть не A

Модель языка логики предикатов

$M = (D, I)$ — модель языка логики предикатов, если

- D — непустое множество (носитель модели);
- I — интерпретация предикатных и функциональных букв в D:
  - если P — n-арная предикатная буква, то $I(P) = P_M$ — n-арный предикат на D;
  - если f — n-арная функциональная буква, то $I(f) = f_M$ — n-арная функция на D.

Примеры. Пусть язык логики предикатов содержит $=$, $\cdot$ и $+$.
Модели:

- $M_1 = (\mathbb{R}; =, \cdot, +)$
- $M_2 = (\mathbb{C}; =, \cdot, +)$
- $M_3 = (\mathbb{N}; =, \cdot, +)$
- $M_4 = (\mathbb{Z}; =, \cdot, +)$
- $M_5 = (\mathbb{Z}_2; =, \cdot, +)$
- $M_6 = (Mat_{5\times 5}\mathbb{R}; =, \cdot, +)$
- ...

Значения термов в модели

Пусть $M = (D, I)$ — модель языка логики предикатов. Для каждого $a \in D$ добавим к языку константу $\underline{a}$ (т.е. 0-арную функциональную букву). Для каждого терма t получившегося языка определим его значение t_M в модели M:

- если $a \in D$, то $\underline{a}_M = a$;
- если $t = f(t_1, \ldots, t_n)$, то $t_M = f_M((t_1)_M, \ldots, (t_n)_M)$.

Пример. Пусть $M = (\mathbb{R}; =_M, \cdot_M, +_M)$, $t = (\underline{2} + \underline{2}) \cdot \underline{2}$.
Тогда $t_M = ((\underline{2} + \underline{2}) \cdot \underline{2})_M = (2 +_M 2) \cdot_M 2.$ $= 8$

Истинность формул в модели

Формула A называется замкнутой, если она не содержит свободных переменных.

Для замкнутой формулы A и модели $M = (D, I)$ определим отношение $M \models A$ — «в M истинна формула A»:

- $M \not\models \bot$;
- $M \models P(t_1, \ldots, t_n)$, если $P_M((t_1)_M, \ldots, (t_n)_M) = 1$;
- $M \models A \wedge B$, если $M \models A$ и $M \models B$;
- $M \models A \vee B$, если $M \models A$ или $M \models B$;
- $M \models A \to B$, если $M \models A$ влечёт, что $M \models B$;
- $M \models \exists x\, A[a/x]$, если существует такой $x \in D$, что $M \models A[a/\underline{x}]$;
- $M \models \forall x\, A[a/x]$, если для любого $x \in D$ верно, что $M \models A[a/\underline{x}]$.

Пусть A — формула со свободными переменными $a_1, \ldots, a_n$. Тогда $M \models A$, если $M \models A[a_1/\underline{x}_1, \ldots, a_n/\underline{x}_n]$ для любых $x_1, \ldots, x_n \in D$.

Формула A тождественно истинна, если она истинна *в любой* модели.
Формула A тождественно ложна, если она ложна *в любой* модели.
Формула A выполнима, если она истинна *в некоторой* модели.

Пример

Пусть $A = \forall x \exists y\, P(x, y)$.

1. $M_1 = (\mathbb{R}; >)$.
Получаем, что $M_1 \models A$.

2. $M_2 = (\mathbb{N}; >)$.
Получаем, что $M_2 \not\models A$.

3. $M_3 = (\mathbb{N}; <)$.
Получаем, что $M_2 \models A$.

4. Пусть $B = \forall x(P(x) \to P(x))$.
Тогда $M \models B$ для любой модели M, т.е. B тождественно истинна.

Логические законы

Формулы A и B называются равносильными, если формула $A \leftrightarrow B$ тождественно истинна.
Обозначение: $A \equiv B$.

- Взаимодействие кванторов с отрицанием:
  - $\neg\forall x\, A[a/x] \equiv \exists x\, \neg A[a/x]$
  - $\neg\exists x\, A[a/x] \equiv \forall x\, \neg A[a/x]$
- Перестановка одноимённых кванторов:
  - $\forall x \forall y\, A[a/x, b/y] \equiv \forall y \forall x\, A[a/x, b/y]$
  - $\exists x \exists y\, A[a/x, b/y] \equiv \exists y \exists x\, A[a/x, b/y]$
  - $\forall x \exists y\, P(x, y) \not\equiv \exists y \forall x\, P(x, y)$ будьте осторожны!

Логические законы

Формулы A и B называются равносильными, если формула $A \leftrightarrow B$ тождественно истинна.
Обозначение: $A \equiv B$.

- Взаимодействие кванторов с конъюнкцией и дизъюнкцией:
  - $\forall x\,(A[a/x] \wedge B[a/x]) \equiv \forall x\,A[a/x] \wedge \forall x\,B[a/x])$
  - $\exists x\,(A[a/x] \vee B[a/x]) \equiv \exists x\,A[a/x] \vee \exists x\,B[a/x])$

  - $\forall x\,(P(x) \vee Q(x)) \not\equiv \forall x\,P(x) \vee \forall x\,Q(x))$ будьте осторожны!
  - $\exists x\,(P(x) \wedge Q(x)) \not\equiv \exists x\,P(x) \wedge \exists x\,Q(x)$ будьте осторожны!

  - $\forall x\,(A \vee B[a/x]) \equiv A \vee \forall x\,B[a/x]$
  - $\exists x\,(A \wedge B[a/x]) \equiv A \wedge \exists x\,B[a/x]$

Равносильность: будьте осторожны!

Формулы A и B называются равносильными, если формула $A \leftrightarrow B$ тождественно истинна.
Обозначение: $A \equiv B$.

- $A \equiv \quad B$, если $\forall M\,(M \models A \leftrightarrow B)$
- $A \equiv' B$, если $\forall M\,(M \models A \iff M \models B)$

Справедливо следующее:

- $A \equiv B$ влечёт, что $A \equiv' B$
- из того, что $A \equiv' B$, не следует, что $A \equiv B$ (контрпример: $A = P(a)$, $B = \forall x\, P(x)$)

Анекдот

$\exists x\,(P(x) \to \forall y\,P(y))$ — тождественно истинная формула!

Разоблачение:

$\forall P \exists x\,(P(x) \to \forall y\,P(y))$ — соответствует формуле

$\exists x \forall P\,(P(x) \to \forall y\,P(y))$ — соответствует ситуации

Вопрос: что можно, а что нельзя выразить на языке логики предикатов?

Обобщённые кванторы

$$\exists! x\, P(x) = \exists x\, (P(x) \wedge \forall y\, (P(y) \to y = x))$$

$$\exists_{=0} x\, P(x) = \neg \exists x\, P(x)$$

$$\exists_{=1} x\, P(x) = \exists x\, (P(x) \wedge \forall y\, (P(y) \to y = x))$$

$$\exists_{=2} x\, P(x) = \exists x \exists y\, (x \neq y \wedge P(x) \wedge P(y) \wedge \forall z\, (P(z) \to z = x \vee z = y))$$

$\exists_{=k} x\, P(x) = ??$ — упражнение

$$\exists_{\leqslant k} x\, P(x) = \exists_{=0} x\, P(x) \vee \ldots \vee \exists_{=k} x\, P(x)$$

$$\exists_{\geqslant k} x\, P(x) = \exists x_1 \ldots \exists x_k\, (\bigwedge_{i=1}^{k} P(x_i) \wedge \bigwedge_{i \neq j} (x_i \neq x_j))$$

$\exists_{=\infty} x\, P(x) = ??$ — не выразить в языке первого порядка

Ограниченные кванторы

$$\forall \varepsilon > 0\ \exists \delta > 0\ \forall x \in \dot{O}(a, \delta)\ f(x) \in O(A, \varepsilon)$$

$$A = \lim_{x \to a} f(x)$$

Ограниченные кванторы

- всеобщности:
  «для любого x, такого, что $P(x)$, верно A»
  $\forall x\,(P(x))\,A(x) = \forall x\,(P(x) \to A(x))$
- существования:
  «существует x, такой, что $P(x)$, для которого верно A»
  $\exists x\,(P(x))\,A(x) = \exists x\,(P(x) \wedge A(x))$

Свойства ограниченных кванторов:

- $\neg\forall x\,(P(x))\,A(x) = \exists x\,(P(x))\,\neg A(x))$
- $\neg\exists x\,(P(x))\,A(x) = \forall x\,(P(x))\,\neg A(x))$

$$\forall \varepsilon(\varepsilon > 0)\ \exists \delta(\delta > 0)\ \forall x(x \in \dot{O}(a, \delta))\ f(x) \in O(A, \varepsilon)$$

$$A = \lim_{x \to a} f(x): \quad \forall \varepsilon(\varepsilon > 0 \to \exists \delta(\delta > 0 \wedge \forall x(x \in \dot{O}(a, \delta) \to f(x) \in O(A, \varepsilon))))$$

$$A \neq \lim_{x \to a} f(x): \quad \exists \varepsilon(\varepsilon > 0 \wedge \forall \delta(\delta > 0 \to \exists x(x \in \dot{O}(a, \delta) \wedge f(x) \notin O(A, \varepsilon))))$$

Выразимые условия

Предикат $P(x_1, \ldots, x_n)$, заданный в модели $M = (D, I)$, называется выразимым в языке логики предикатов, если существует такая формула $A(a_1, \ldots, a_n)$ этого языка, для любых $x_1, \ldots, x_n \in D$

$$P(x_1, \ldots, x_n) = 1 \iff M \models A[\underline{x}_1, \ldots, \underline{x}_n].$$

Термин: Элементарные условия — условия, выразимые на языке логики предикатов.

Функция $f(x_1, \ldots, x_n)$, заданная в модели $M = (D, I)$, называется выразимой в языке логики предикатов, если существует такая формула $A(a_1, \ldots, a_n, b)$ этого языка, что для любых $x_1, \ldots, x_n, y \in D$

$$f(x_1, \ldots, x_n) = z \iff M \models A[\underline{x}_1, \ldots, \underline{x}_n, \underline{z}].$$

Примеры выразимых и невыразимых условий

- вычитание через сложение: $x - y = z \iff x = z + y$
- деление через умножение: $x : y = z \iff x = z \cdot y$
- умножение на 2 через сложение: $2x = y \iff y = x + x$
- умножение на 3 через сложение: $3x = y \iff y = 2x + x$
- умножение на k через сложение: $kx = y \iff y = (k-1)x + x$
- умножение через сложение: $x \cdot y = z$ не выражается!

Примеры выразимых и невыразимых условий

- коммутативность f: $\forall x \forall y\, f(x,y) = f(y,x)$
- ассоциативность f: $\forall x \forall y \forall z\, f(x, f(y,z)) = f(f(x,y), z)$
- идемпотентность f: $\forall x\, f(x,x) = x$
- серийность R: $\forall x \exists y\, R(x,y)$
- рефлексивность R: $\forall x\, R(x,x)$
- иррефлексивность R: $\forall x\, \neg R(x,x)$
- симметричность R: $\forall x \forall y\, (R(x,y) \to R(y,x))$
- антисимметричность R: $\forall x \forall y\, (R(x,y) \wedge R(y,x) \to x = y)$
- транзитивность R: $\forall x \forall y \forall z\, (R(x,y) \wedge R(y,z) \to R(x,z))$
- интранзитивность R: $\forall x \forall y ((\exists n \geqslant 2\, R^n(x,y)) \to \neg R(x,y))$
  не выражается!
- связность R: симметричность + $\forall x \forall y \exists n \geqslant 0\, R^n(x,y)$
  не выражается!

Примеры выразимых и невыразимых условий

- R^{-1} через R: $xR^{-1}y := yRx$
- рефлексивное замыкание R^{ref} через R: $xR^{ref}y := xRy \vee x = y$
- $R_1 \circ R_2$ через R_1 и R_2: $x(R_1 \circ R_2)y := \exists z(xR_1z \wedge zR_2y)$
- R^2 через R: $xR^2y := x(R \circ R)y$
- R^3 через R: $xR^3y := x(R^2 \circ R)y$
- R^k через R: $xR^ky := x(R^{k-1} \circ R)y$
- транзитивное замыкание R^+ (через R): не выражается!
- рефлексивно-транзитивное замыкание R^*: не выражается!

Примеры выразимых и невыразимых условий

- не более 1 элемента: $\forall x \forall y\, x = y$
- не более 2 элементов: $\forall x \forall y \forall z\, (x = y \vee y = z \vee z = x)$
- не более k элементов: упражнение
- не менее 1 элемента: $\exists x\, x = x$
- не менее 2 элементов: $\exists x \exists y\, x \neq y$
- не менее 3 элементов: $\forall x \forall y \forall z\, (x \neq y \vee y \neq z \vee z \neq x)$
- не менее k элементов: упражнение
- конечное множество элементов: не выражается!
- бесконечное множество элементов: не выражается!
- чётное число элементов: не выражается!
- нечётное число элементов: не выражается!
- число элементов, (не)кратное k: не выражается!
- равенство: только как конгруэнтность!

Исчисление предикатов

Аксиомы (в языке $\wedge$, $\vee$, $\to$, $\bot$, $\forall$):

- аксиомы классической логики высказываний
- формулы вида $\forall x\, A[a/x] \to A[a/t]$

Правила вывода:

- *modus ponens*: из формул A и $A \to B$ получается B
- *обобщение*: из формулы A получается $\forall x\, A[a/x]$
- *подстановка*: из формулы A получается $\mathbf{s}A$,
  где $\mathbf{s}$ — функция, заменяющая в A вхождения атомарных формул, отличных от $\bot$, некоторыми формулами

Понятия вывода формулы и вывода формулы из множества гипотез определяются как и для исчисления высказываний.

Теории первого порядка

Теория — это любое множество формул.

Теория называется непротиворечивой, если в ней не выводится $\bot$.

Примеры теорий первого порядка:

- классическая логика предикатов (в некоторой сигнатуре)
- теория (логика) конечных моделей (в некоторой сигнатуре)
- теория одной фиксированной модели (например, арифметика)
- теория групп (например, в сигнатуре $\{=, \cdot\}$ или $\{=, +\}$); теория абелевых групп, теория конечных групп, теория полугрупп, и т.п.
- теория колец, теория полей (в сигнатуре $\{=, \cdot, +\}$)
- истинностная арифметика, арифметика Пеано, арифметика Пресбургера, и т.д.
- теории множеств **ZF**, **NBG** и их расширения аксиомой выбора, континуум-гипотезой и т.п. (в сигнатуре $\{\in\}$)
- математические теории — всё, что основано на теории множеств

Теоремы о полноте

Замечание. Понятие вывода в теории соответствует понятию доказательства в этой теории.

Теорема о полноте

Теория T непротиворечива $\iff$ T имеет модель

Теорема о компакности

Теория T непротиворечива $\iff$ любое конечное подмножество теории T непротиворечиво

Теорема о полноте исчисления предикатов

Формула A выводима в исчислении предикатов $\iff$ A тождественно истинна.

Конструктивность и неконструктивность: примеры

Число x называется *алгебраическим*, если оно является корнем некоторого многочлена с целыми коэффициентами. Числа, не являющиеся алгебраическими, называются *трансцендентными*.

Существование трансцендентных чисел:

- множество $\mathbb{R}$ несчётно, множество $\mathbb{A}$ счётно, следовательно $\mathbb{R} \setminus \mathbb{A}$ не пусто.
- числа π и e являются трансцендентными.

Существование таких иррациональных a и b, что a^b рационально:

- если $\sqrt{2}^{\sqrt{2}} \in \mathbb{Q}$, то берём $a = \sqrt{2}$, $b = \sqrt{2}$
- если $\sqrt{2}^{\sqrt{2}} \notin \mathbb{Q}$, то берём $a = \sqrt{2}^{\sqrt{2}}$, $b = \sqrt{2}$

Интуиционистская (конструктивная) логика

Считаем, что формулы строятся в языке $\{\wedge, \vee, \to, \bot\}$.
Оцениваем формулы не через истинность, а **через доказуемость**:

- $\bot$ не доказуема
- $A \wedge B$ доказуема, если A доказуема и B доказуема
- $A \vee B$ доказуема, если A доказуема или B доказуема, причём указано, какой именно дизъюнктивный член доказуем
- $A \to B$ доказуема, если имеется алгоритм, который любое доказательство формулы A преобразует в доказательство формулы B

Как следствие:

- $\neg A$ доказуема, если имеется алгоритм, который любое доказательство формулы A преобразует в доказательство противоречия ($\neg A = A \to \bot$)

Пример: $p \to p$
Контрпример: $p \vee \neg p$

Примеры неконструктивных классических законов

- $p \vee \neg p$
- $\neg\neg p \to p$ ($A \to \neg\neg A$, $\neg\neg\neg A \to \neg A$ — констр.)
- $\neg p \vee \neg\neg p$
- $(p \to q) \vee (q \to p)$
- $(\neg p \to \neg q) \to (q \to p)$ ($(A \to B) \to (\neg B \to \neg A)$ — констр.)
- $\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$
- $\neg(p \vee q) \leftrightarrow (\neg p \wedge \neg q)$
- все классические законы, позволяющие выражать любую из связок $\{\wedge, \vee, \to, \bot\}$ через другие

Интуиционистское (конструктивное) исчисление

Аксиомы (в языке с $\wedge$, $\vee$, $\to$, $\bot$):

(A1) $p_0 \to (p_1 \to p_0)$

(A2) $(p_0 \to (p_1 \to p_2)) \to ((p_0 \to p_1) \to (p_0 \to p_2))$

(A3) $p_0 \wedge p_1 \to p_0$

(A4) $p_0 \wedge p_1 \to p_1$

(A5) $p_0 \to (p_1 \to p_0 \wedge p_1)$

(A6) $p_0 \to p_0 \vee p_1$

(A7) $p_1 \to p_0 \vee p_1$

(A8) $(p_0 \to p_2) \to ((p_1 \to p_2) \to (p_0 \vee p_1 \to p_2))$

(A9) $\bot \to p_0$

(A10) $p_0 \vee (p_0 \to \bot)$ удаляем

Правила вывода: modus ponens, подстановка

Теорема Гливенко

$\mathbf{CL} \vdash A \iff \mathbf{Int} \vdash \neg\neg A$

Алгоритмические проблемы

Для чего нужно понятие алгоритма?

- Чтобы находить решения задач.
- Чтобы строить алгоритмы решения задач (писать программы).
- Чтобы иметь возможность доказывать, что задача имеет алгоритм решения.
- Чтобы находить оптимальный алгоритм решения задачи.
- Чтобы иметь возможность доказывать, что задача не имеет алгоритма решения.

Цель введения понятия алгоритма — описание класса вычислимых функций, а также функций, которые вычислимыми не являются.

Что такое алгоритм?

Алгоритм (алгорифм) — конечная совокупность точно заданных правил решения произвольного класса задач или набор инструкций, описывающих порядок действий исполнителя для решения некоторой задачи.

Требования к алгоритму:

- дискретность (выполнение должно происходить «по шагам»)
- детерминированность (каждый следующий шаг однозначно определён)
- понятность (выполнение команд доступно для исполнителя)
- ясность условий начала и конца работы, а также результата каждого «шага»
- массовость (применимость к различным входным данным)
- результативность

Математические формализации понятия алгоритма

Были предложены различные формализации:

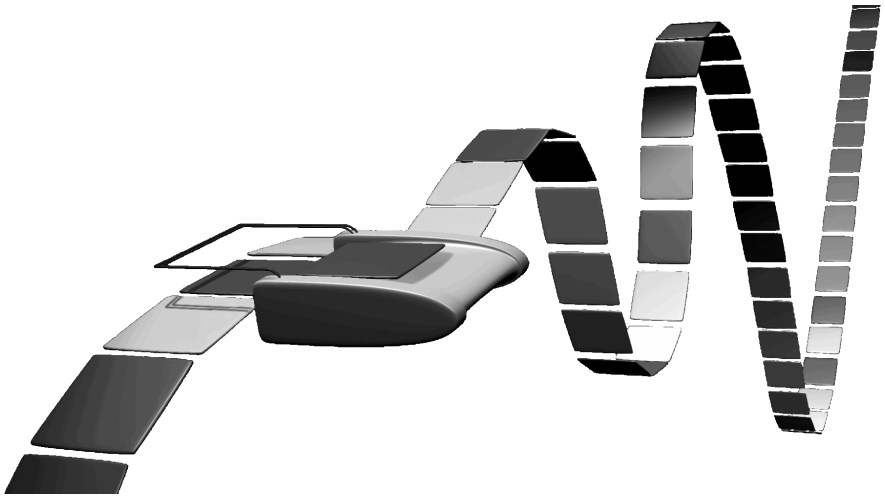
- машины Тьюринга
- машины Минского
- машины Поста
- нормальные алгорифмы Маркова
- частично-рекурсивные функции (Курт Гёдель)
- лямбда-исчисление (Алонзо Чёрч)

Тезис Чёрча–Тьюринга: это гипотеза, постулирующая эквивалентность между интуитивным понятием алгоритмической вычислимости и строго формализованными понятиями частично рекурсивной функции и функции, вычислимой на машине Тьюринга.

Описание машины Тьюринга

Машину Тьюринга обычно ассоциируют с лентой, состоящей из ячеек, в которых могут быть записаны символы. По ленте двигается головка; она может:

- считывать символ,
- записывать символ,
- смещаться.



Лента ограничена слева, в крайней ячейке записан символ # (граничный маркер), который запрещено стирать.
Справа лента потенциально бесконечна.
У машины имеется внутренняя память — конечное множество состояний.
Управление происходит с помощью команд (инструкций).
Команда имеет вид $qa \to q'a'\Delta$, где a, a' — символы, q, q' — состояния, $\Delta \in \{-1, 0, +1\}$ — сдвиг.

Определение машины Тьюринга

Машина Тьюринга — это набор $M = (\Sigma, Q, q_0, q_e, P)$, где

- Σ — внешний (терминальный) алфавит; $\#, \square \in \Sigma$;
- Q — внутренний алфавит (множество состояний); $q_0, q_e \in Q$;
- q_0 — начальное состояние;
- q_e — заключительное состояние;
- P — программа.

Программа P состоит из инструкций вида $qa \to q'a'\Delta$, где

- $a, a' \in \Sigma$,
- $q, q' \in Q$,
- $\Delta \in \{-1, 0, +1\} = \{L, S, R\}$,

при этом $a = \#$ тогда и только тогда, когда $a' = \#$, и нет инструкций вида $q\# \to q'\#L$; плюс, P не содержит нескольких инструкций с одинаковой левой частью.

Работа машины Тьюринга

Пусть $M = (\Sigma, Q, q_0, q_e, P)$ — машина Тьюринга, $x = x_1 \dots x_n$ — слово в алфавите Σ, не содержащее $\#$ и $\square$.

Работа машины M на слове x происходит следующим образом.

Начало работы:

- слева на ленту помещается $\#x_1 \dots x_n$;
- головка обозревает ячейку с символом $\#$;
- машина находится в состоянии q_0.

Шаг работы:

- пусть M находится в состоянии q и обозревает символ a;
- находим инструкцию с левой частью «qa»: $qa \to q'a'\Delta$;
- заменяем a на a', переходим в состояние q', делаем сдвиг Δ.

Конец работы:

- M пришла в состояние q_e.

Результат работы: то, что записано на ленте.

Пример машины Тьюринга

Задача: построить машину Тьюринга, которая в слове над алфавитом $\{a, b\}$ дублирует в конце последний символ (если это слово не является пустым).

$q_0\# \to q_0\#R$	сдвигаемся вправо
$q_0 a \to q_0 aR$	сдвигаемся вправо
$q_0 b \to q_0 bR$	сдвигаемся вправо
$q_0\square \to q_1\square L$	q_1 — дошли до конца слова, сдвигаемся влево
$q_1 a \to q_2 aR$	q_2 — запомнили a
$q_1 b \to q_3 bR$	q_3 — запомнили b
$q_1\# \to q_e\#S$	конец (нечего дублировать)
$q_2\square \to q_e aS$	конец (продублировали a)
$q_3\square \to q_e bS$	конец (продублировали b)

Вычислимые функции

Обозначение: $\mathcal{A}^*$ — множество всех слов в алфавите $\mathcal{A}$.

Функция $f : \mathcal{A}^* \to \mathcal{B}^*$ вычислима по Тьюрингу, если существует такая машина Тьюринга M, что для любого слова $x \in \mathcal{A}^*$

- $M(x)$ определено $\iff$ $f(x)$ определено;
- если $f(x)$ определено, то $M(x) = f(x)$.

Замечание. Машины Тьюринга можно использовать для вычисления функций многих аргументов (в том числе числовых). Для этого достаточно на вход подавать слова, являющиеся записями (кодами) наборов аргументов. На выходе нужно ожидать запись (код) значения вычисляемой функции.

Существование невычислимых функций

Утверждение. *Невычислимые функции существуют.*

Доказательство.
Заметим, что множество всех машин Тьюринга счётно.
Следовательно, множество вычислимых функций тоже счётно.
Заметим, что множество функций вида $f : \mathcal{A}^* \to \mathcal{A}^*$ несчётно.
Значит, множество невычислимых функций непусто. □

Замечание. Приведённое доказательство не является конструктивным, в частности, оно не даёт примера невычислимой функции.

Кодирование машин Тьюринга

Наблюдение. Программу машины Тьюринга можно рассматривать как слово над некоторым алфавитом. Считаем, что $\square = a_0$, $\# = a_1$, $q_e = q_1$, и тогда программу любой машины Тьюринга можно закодировать в алфавите $\{a, q, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \to, L, S, R, *\}$, где $*$ — разделитель.

Обозначение. Пусть $\varkappa M$ — код машины M в алфавите $\{a, q, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \to, L, S, R, *\}$.

Становится возможным:

- подавать $\varkappa M$ на вход другим машинам Тьюринга;
- подавать $\varkappa M$ на вход машине M;
- анализировать код машины M с помощью машин Тьюринга;
- ставить задачи машинам Тьюринга о машинах Тьюринга.

Универсальная машина Тьюринга

Наблюдение. Заметим, что закодировать можно не только программу машины Тьюринга, но и входные слова.

Обозначение. Пусть $\varkappa x$ — код машины слова x в алфавите $\{a, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Машина Тьюринга M_0 называется универсальной, если для всякой машины M и всякого слова x

- $M_0(\varkappa M * \varkappa x)$ определено $\iff$ $M(x)$ определено;
- если $M(x)$ определено, то $M_0(\varkappa M * \varkappa x) = \varkappa(M(x))$.

Другими словами, универсальная машина Тьюринга — это машина, способная моделировать работу любой другой машины Тьюринга на любом входе.

Теорема

Универсальная машина Тьюринга существует

Пример: компьютер.

Разрешимые множества

Множество слов (язык) L в некотором алфавите $\mathcal{A}$ называется разрешимым, если характеристическая функция χ_L множества L вычислима.

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

Здесь $x \in \mathcal{A}^*$.

Утверждение. *Если L_1 и L_2 разрешимы, то $L_1 \cap L_2$, $L_1 \cup L_2$, $L_1 \setminus L_2$ тоже разрешимы.*

Примеры разрешимых множеств

- По числам a, b, c установить, является ли c суммой (разностью, произведением, частным) чисел a и b
- По формуле логики высказываний A установить, является ли A тождественно истинной (тождественно ложной, выполнимой)
- По простому графу G установить, является ли G связным (эйлеровым, гамильтоновым, планарным, и т.п.)
- По натуральному числу x установить, является ли x простым (составным, чётным, квадратом, кубом, и т.п.)
- По трёхчлену $ax^2 + bx + c$ с действительными коэффициентами установить, имеет ли он действительный корень
- По многочлену $f(x)$ с целыми коэффициентами установить, имеет ли он рациональный корень
- По матрице A_f квадратичной формы f установить, является ли f положительно определённой
- По матрице A установить, имеет ли A обратную

Проблема самоприменимости

Машина Тьюринга M называется самоприменимой, если M даёт какой-нибудь результат, начав работать на собственном коде, т.е. если $M(\varkappa M)$ определено.

Пример: $q_0\# \to q_e\#S$ — программа самоприменимой машины.
Контрпример: $q_0\# \to q_0\#S$ — программа несамоприменимой машины.

Положим $L_0 = \{\varkappa M : M$ — самоприменимая машина Тьюринга$\}$.

Неразрешимость проблемы самоприменимости

Теорема. *Множество L_0 неразрешимо.*

Доказательство.
Предположим, что множество L_0 разрешимо, т.е. существует такая машина Тьюринга M_1, что для любой машины M

$$M_1(\varkappa M) = \begin{cases} 1, & \text{если } M \text{ самоприменима,} \\ 0, & \text{если } M \text{ несамоприменима.} \end{cases}$$

Переделаем M_1: пусть она вместо ответа «1» зацикливается.
Получим M_2:

$$M_2(\varkappa M) = \begin{cases} \text{не опр.}, & \text{если } M \text{ самоприменима,} \\ 0, & \text{если } M \text{ несамоприменима.} \end{cases}$$

Вопрос: является ли M_2 самоприменимой?
Если да, то $M_2(\varkappa M_2)$ не определено, т.е. M_2 несамоприменима.
Если нет, то $M_2(\varkappa M_2) = 0$, т.е. M_2 самоприменима.
Получили противоречие. Значит, L_0 неразрешимо. $\square$

Теорема Успенского–Райса

Свойство P машин Тьюринга называется нетривиальным, если существуют машины, обладающие им, и существуют машины, не обладающие им, т.е. $\exists M\, P(M) \wedge \exists M\, \neg P(M)$.

Машины Тьюринга M_1 и M_2 называются эквивалентными, если они вычисляют одну и ту же функцию.

Свойство P машин Тьюринга называется инвариантным, если из того, что M_1 и M_2 эквивалентны, следует, что они одновременно либо обладают, либо не обладают свойством P, т.е. $\forall M_1 \forall M_2\, (M_1 \text{ экв. } M_2 \wedge P(M_1) \to P(M_2))$.

Теорема Успенского–Райса

Любое нетривиальное инвариантное свойство машин Тьюринга алгоритмически неразрешимо.

Следствия из теоремы Успенского–Райса

Следующие проблемы алгоритмически неразрешимы:

- по машине M выяснить, вычисляет ли M данную вычислимую функцию f
- по машине M выяснить, остановится ли M на данном входе x
- по машине M выяснить, вычисляет ли M всюду определённую функцию
- по машине M выяснить, вычисляет ли M где-нибудь определённую функцию
- по машине M выяснить, вычисляет ли M нигде не определённую функцию
- по машинам M_1 и M_2 выяснить, эквивалентны ли они

Теорема Чёрча

Теорема Чёрча

Проблема тождественной истинности формул логики предикатов алгоритмически неразрешима.

Идея доказательства.

1. Берём проблему «M останавливается на пустом входе». Эта проблема алгоритмически неразрешима.

2. Описываем алгоритм, который по произвольной машине M строит формулу $A(M)$, которая описывает работу M на пустом входе и утверждает, что M остановится.

3. Доказываем: $M()$ определено $\Longleftrightarrow$ $A(M)$ тождественно истинна.

4. Если бы проблема тождественной истинности была алгоритмически разрешима, то мы могли бы решать и проблему остановки машин на пустом входе, а это невозможно.

Теорема Чёрча

Теорема Чёрча

Проблема тождественной истинности формул логики предикатов алгоритмически неразрешима.

Идея доказательства.

Нам потребуются предикатные буквы $=$, $<$, а также:

- $C(x, y)$ — на шаге x машина M обозревает ячейку номер y
- $Q_k(x)$ — на шаге x машина M находится в состоянии q_k
- $S_k(x, y)$ — на шаге x ячейка y содержит символ a_k
- $Next(x, y) = (x < y \wedge \neg\exists z\,(x < z \wedge z < y))$
- $Min(x) = \forall y\,(x \neq y \to x < y)$

Теорема Чёрча

Теорема Чёрча

Проблема тождественной истинности формул логики предикатов алгоритмически неразрешима.

Идея доказательства.

Начальная конфигурация машины M описывается формулой:

$$\begin{aligned} Start \quad = \quad & \exists x \, (Min(x) \wedge C(x,x) \wedge Q_0(x) \wedge S_1(x,x) \wedge \\ & \wedge \forall y \, (x \neq y \rightarrow S_0(x,y))) \end{aligned}$$

Теорема Чёрча

Теорема Чёрча

Проблема тождественной истинности формул логики предикатов алгоритмически неразрешима.

Идея доказательства.

На каждом шаге M обозревает единственную ячейку, находится в одном состоянии, в ячейках записано по одному символу:

$$U_C = \forall x \exists y \left(C(x,y) \wedge \forall z \left(C(x,z) \to z = y \right) \right)$$

$$U_Q = \forall x \bigvee_k \left(Q_k(x) \wedge \bigwedge_{j \neq k} \neg Q_j(x) \right)$$

$$U_S = \forall x \forall y \left(\bigvee_k S_k(x,y) \wedge \bigwedge_{j \neq k} \neg S_j(x,y) \right)$$

Теорема Чёрча

Теорема Чёрча

Проблема тождественной истинности формул логики предикатов алгоритмически неразрешима.

Идея доказательства.

Описание инструкции $I = q_i a_j \to q_k a_l S$:

$$\begin{aligned} Instr_I \quad = \quad & \forall x \forall y \Big(Q_i(x) \wedge C(x,y) \wedge S_j(x,y) \to \\ & \exists z \left(Next(x,z) \wedge Q_k(z) \wedge C(z,y) \wedge S_l(z,y) \wedge \right. \\ & \left. \wedge \forall u \left(u \neq y \to \bigwedge_t (S_t(x,u) \to S_t(z,u)) \right) \right) \Big). \end{aligned}$$

Теорема Чёрча

Теорема Чёрча

Проблема тождественной истинности формул логики предикатов алгоритмически неразрешима.

Идея доказательства.

Описание инструкции $I = q_i a_j \to q_k a_l R$:

$$\begin{aligned} Instr_I \quad = \quad & \forall x \forall y \Big(Q_i(x) \wedge C(x,y) \wedge S_j(x,y) \to \\ & \exists z \exists v \big(Next(x,z) \wedge Next(y,v) \wedge Q_k(z) \wedge C(z,v) \wedge S_l(z,y) \wedge \\ & \wedge \forall u \, (u \neq y \to \bigwedge_t (S_t(x,u) \to S_t(z,u))) \big) \Big). \end{aligned}$$

Теорема Чёрча

Теорема Чёрча

Проблема тождественной истинности формул логики предикатов алгоритмически неразрешима.

Идея доказательства.

Описание инструкции $I = q_i a_j \to q_k a_l L$:

$$\begin{aligned} Instr_I \quad = \quad & \forall x \forall y \Big(Q_i(x) \wedge C(x,y) \wedge S_j(x,y) \to \\ & \exists z \exists v \big(Next(x,z) \wedge Next(v,y) \wedge Q_k(z) \wedge C(z,v) \wedge S_l(z,y) \wedge \\ & \wedge \forall u \, (u \neq y \to \bigwedge_t (S_t(x,u) \to S_t(z,u))) \big) \Big). \end{aligned}$$

Теорема Чёрча

Теорема Чёрча

Проблема тождественной истинности формул логики предикатов алгоритмически неразрешима.

Идея доказательства.

Опишем работу M на пустом входе:

$$Comp(M) = Start \wedge U_C \wedge U_Q \wedge U_S \wedge \bigwedge_{I \in P} Instr_I.$$

Остановка: $Stop = \exists x\, Q_1(x)$.

Тогда $A(M) = Comp_M \to Stop$.

Получаем: $M()$ определено $\iff$ $A(M)$ тождественно истинна. $\square$

Примеры разрешимых и неразрешимых теорий

проблема выполнимости предикатных формул	неразрешима
теория модели $(\mathbb{N}; =, \cdot, +, 0)$, т.е. арифметика	неразрешима
арифметика Пеано	неразрешима
арифметика Пресбургера	разрешима
логика одноместных предикатов	разрешима
теория равенства	разрешима
логика одноместных предикатов с равенством	разрешима
теория групп	неразрешима
теория графов	неразрешима
теория простых графов	неразрешима
проблема равенства в полугруппах	неразрешима
теория конечных моделей	неразрешима
теории множеств **ZF**, **NBG**, $+AC$, $+CH$	неразрешимы
теории моделей $(\mathbb{R}; =, <)$ и $(\mathbb{Q}; =, <)$	разрешимы

Рекурсивно перечислимые множества

Множество (язык) L называется рекурсивно перечислимым, если существует перечиляющий его алгоритм (машина Тьюринга) M, т.е. $L = \{M(\varkappa n) : n \in \mathbb{N}\}$.

Утверждение. *Всякое разрешимое множество является рекурсивно перечислимым.*

Утверждение. *Всякая конечно аксиоматизируемая теория является рекурсивно перечислимой.*

Утверждение. *Если L_1 и L_2 рекурсивно перечислимы, то $L_1 \cap L_2$ и $L_1 \cup L_2$ тоже рекурсивно перечислимы.*

Теорема Поста

Теорема Поста

Множество разрешимо $\Longleftrightarrow$ оно и его дополнение рекурсивно перечислимы.

Следствие. *Если T — неразрешимая конечно аксиоматизируемая теория, то множество не-теорем теории T не является рекурсивно перечислимым.*

Следующие теории не являются рекурсивно перечислимыми:

- теория конечных моделей
- арифметика, т.е. теория модели $(\mathbb{N}; =, \cdot, +, 0)$
- логика второго порядка

Вопрос: существуют ли множества, устроенные ещё более сложно *алгоритмически*?

Арифметическая иерархия множеств

Пусть $\Sigma_0^0 = \Pi_0^0 = \{P : P$ — разрешимый предикат на $\mathbb{N}\}$

$\Sigma_{n+1}^0 = \{\exists \bar{x}\, P : P \in \Pi_n^0\}, \quad \Pi_{n+1}^0 = \{\exists \bar{x}\, P : P \in \Sigma_n^0\}, \quad \Delta_n^0 = \Sigma_n^0 \cap \Pi_n^0$

Пример: $\Sigma_3^0 = \{\exists \bar{x} \forall \bar{y} \exists \bar{z}\, P : P$ — разрешимый предикат на $\mathbb{N}\}$.

Классы Σ_n^0, Π_n^0 и Δ_n^0 обладают следующими свойствами:

- Σ_n^0, Π_n^0 замкнуты относительно объединения и пересечения
- $A \in \Sigma_n^0 \iff \bar{A} \in \Pi_n^0$
- $\Sigma_n^0 \subset \Sigma_{n+1}^0, \quad \Pi_n^0 \subset \Pi_{n+1}^0$
- $\Sigma_n^0 \subset \Sigma_{n+1}^0, \quad \Pi_n^0 \subset \Pi_{n+1}^0$
- $\Sigma_n^0 \cup \Pi_n^0 \subset \Delta_{n+1}^0$, где $n \geqslant 1$

Известно следующее:

- $\Sigma_0^0 = \Pi_0^0 = \Delta_0^0 = \Delta_1^0$ — класс разрешимых предикатов
- Σ_1^0 — класс рекурсивно перечислимых предикатов

Теорема Тарского

Теорема Тарского

Арифметика, т.е. теория модели $(\mathbb{N}; =, \cdot, +, 0)$, не является арифметическим множеством.

Вычислительная сложность

Вычислительная сложность

Для чего нужна теория сложности?

- Чтобы оценивать сложность алгоритмов.
- Чтобы находить оптимальный алгоритм решения задачи.
- Чтобы оптимизировать имеющийся алгоритм.
- Чтобы оценивать сложность задач.
- Чтобы иметь возможность доказывать, что задача не имеет *более простого* алгоритма решения.

Цель теории вычислительной сложности — классификация задач, разделение их на классы с «эффективными» алгоритмами решения и классы «трудно» решаемых задач.

Пример: выполнимость булевых формул

Булева формула $A(p_1, \ldots, p_n)$ называется выполнимой, если сущесвует такой набор $b_1, \ldots, b_n$ булевских значений переменных $p_1, \ldots, p_n$, что $A(b_1, \ldots, b_n) = 1$.

Проблема выполнимости состоит в том, что по произвольной булевой формуле A нужно определить, выполнима ли A.

Проблема выполнимости *разрешима*: достаточно построить таблицу истинности для A и проверить, получилось ли в ней хоть одно значение «1».

Пример: выполнимость булевых формул

Замечено: студенты легко строят таблицу истинности для формул от 1–6 переменных.

Пусть A зависит от 100 переменных. Сколько времени нужно современному персональному компьютеру, чтобы гарантированно установить выполнимость такой формулы?

Уточнение: считаем, что каждая переменная встречается в формуле A от 1 до 10 раз.

Обычные варианты ответов студентов:

- ну... не знаю
- меньше секунды
- доли секунды
- пусть будет две секунды, но я не уверен
- не... это долго... пусть будет пять минут

Пример: выполнимость булевых формул

Частота процессора современного ПК: 10 ГГц, т.е. 10^{10} оп/сек
Дадим ему чуть-чуть побольше: 1 ТГц, т.е. 10^{12} оп/сек

Формула содержит 100 перменных, т.е. не менее 99 операций.
Будем считать, что их 100.

Тогда время работы алгоритма (без учёта «накладных расходов»):

$$
\begin{aligned}
t &= 2^{100} \cdot 100 : 10^{12}\ (\text{сек}) = (2^{10})^{10} \cdot 100 : 10^{12}\ (\text{сек}) = \\
&= 1024^{10} \cdot 100 : 10^{12}\ (\text{сек}) > (10^3)^{10} \cdot 100 : 10^{12}\ (\text{сек}) = \\
&= 10^{20}\ (\text{сек}) > \\
&> 10^{18}\ (\text{мин}) > \\
&> 10^{16}\ (\text{час}) > \\
&> 10^{14}\ (\text{дней}) > \\
&> 10^{11}\ (\text{лет}) = 100\ (\text{млрд лет})
\end{aligned}
$$

Модели вычислений и меры сложности

В качестве модели вычислений выберем многоленточную машину Тьюринга:

- на одной ленте записаны входные данные;
- вычисления производятся на нескольких лентах;
- на последней из них формируется результат.

Такая модель близка к устройству компьютеров.

Меры сложности, которые рассмотрим:

- затраты времени (временна́я сложность);
- затраты памяти (ёмкостная сложность).

Функции сложности

Пусть M — машина Тьюринга (определённая на любом входе), x — произвольное слово в некотором алфавите $\mathcal{A}$ (содержащемся в алфавите машины M).

Определим временну́ю и ёмкостную сложность M на x:

- $t_M(x)$ — число шагов вычисления, которые совершает M, работая на x;
- $s_M(x)$ — число ячеек, которые дополнительно задействует M, работая на x.

Функции $t_M : \mathcal{A}^* \to \mathbb{N}$ и $s_M : \mathcal{A}^* \to \mathbb{N}$ называются функциями временно́й и ёмкостной сложности машины M.

Определим временну́ю и ёмкостную сложность M в наихудшем случае:

- $t_M(n) = \max\limits_{|x| \leqslant n} t_M(x)$; здесь $t_M : \mathbb{N} \to \mathbb{N}$;
- $s_M(n) = \max\limits_{|x| \leqslant n} s_M(x)$; здесь $s_M : \mathbb{N} \to \mathbb{N}$.

Оценка функций сложности

На практике функции $t_M : \mathbb{N} \to \mathbb{N}$ и $s_M : \mathbb{N} \to \mathbb{N}$ вычислять довольно сложно. Но это и не нужно: вместо вычисления функций используют их оценку через «о-большое» (при $n \to \infty$):

$$f \in O(g), \text{ если } \exists c\, \exists n_0\, \forall n \geqslant n_0\ f(n) \leqslant c \cdot g(n).$$

Примеры.

- Сортировка последовательности длины n
  - методом пузырька: $O(n^2)$ сравнений;
  - метод слияния: $O(n \cdot \log_2 n)$ сравнений.
- Вычисление определителя матрицы $n \times n$
  - по формуле «сумма произведений»: $O(n \cdot n!)$ произведений и $O(n!)$ сложений.
  - «методом Гаусса»: $O(n^3)$ произведений и $O(n^3)$ сложений.

Заметим, что

$$\lim_{n\to\infty} \frac{n \cdot \log_2 n}{n^2} = 0 \text{ и } \lim_{n\to\infty} \frac{n^3}{n!} = 0,$$

и можно выбрать более эффективный алгоритм.

Задачи принадлежности

Формально, под задачей будем понимать множество (язык) L слов в некотором алфавите.

Содержательно, задачу L понимаем как задачу принадлежности к L, поэтому вместо L иногда пишут «$x \in L?$».

Задачи принадлежности: примеры

- **SAT** (т.е. «$A \in$ **SAT**?»): по булевой формуле A выяснить, выполнима ли A;
- **ЭЙЛЕРОВ ГРАФ**: по графу $G = (V, E)$ выяснить, является ли он эйлеровым;
- **ГАМИЛЬТОНОВ ГРАФ**: по графу $G = (V, E)$ выяснить, является ли он гамильтоновым;
- **КЛИКА**: по графу $G = (V, E)$ и числу $k \in \mathbb{N}$ выяснить, содержит ли G клику размера k (клика — полный вершинный подграф, т.е. в котором между любыми двумя вершинами имеется ребро);
- **ПРОСТОЕ ЧИСЛО**: по числу $k \in \mathbb{N}$ (в десятичной записи) выяснить, является ли k простым;
- **ВЗАИМНАЯ ПРОСТОТА**: по числам $k, m \in \mathbb{N}$ (в десятичной записи) выяснить, являются ли k и m взаимно простым.

Задачи принадлежности: важное замечание

Если изменить форму записи исходных объектов для задачи, задача останется или изменится?

Пример.
Алгоритм, основанный на построении таблиц истинности, решает проблему **SAT** за время, ограниченное снизу функцией 2^n, где n — число переменных в формуле.

Заметим, что, во-первых, любую булеву фукнцию можно выразить через $\wedge$ и $\neg$, а во-вторых, $A \wedge B = A \cdot B$ и $\neg A = A + 1$. Таким образом, каждую булеву функцию можно выразить через $+$, $\cdot$, 1, т.е. представить в виде полинома Жегалкина.

Известно: полином Жегалкина для булевой функции определяется единственным образом.

Следствие: A выполнима $\iff$ её полином Жегалкина отличен от 0.

Следствие: проблема выполнимости булевых формул в виде полинома Жегалкина решается за один шаг.

Классы DTIME и DSPACE

Теорема. Справедливы следующие отношения:

- $s_M(x) \leqslant t_M(x)$; как следствие, $s_M(n) \leqslant t_M(n)$;
- $t_M(x) \leqslant 2^{c \cdot s_M(x)}$; как следствие, $t_M(n) \leqslant 2^{c \cdot s_M(n)}$.

Пусть имеется функция $f : \mathbb{N} \to \mathbb{N}$.

Определим классы:

- DTIME(f) — класс задач, решаемых *детерминированными* машинами Тьюинга, временна́я сложность которых находится в классе $O(f)$;
- DSPACE(f) — класс задач, решаемых *детерминированными* машинами Тьюинга, ёмкостная сложность которых находится в классе $O(f)$.

Пример: **SAT** $\in$ DTIME($n \cdot 2^n$), **SAT** $\in$ DSPACE(n).

Классы P, EXPTIME и другие

В теории вычислительной сложности важную роль играют следующие классы:

- $\mathrm{P} = \bigcup_{k=0}^{\infty} \mathrm{DTIME}(n^k)$ — класс задач, решаемых *полиномиально по времени*;
- $\mathrm{EXPTIME} = \bigcup_{k=0}^{\infty} \mathrm{DTIME}(2^{n^k})$ — класс задач, решаемых *экспоненциально по времени*;
- $\mathrm{PSPACE} = \bigcup_{k=0}^{\infty} \mathrm{DSPACE}(n^k)$ — класс задач, решаемых *с полиномиальными затратами памяти*;
- $\mathrm{EXPSPACE} = \bigcup_{k=0}^{\infty} \mathrm{DSPACE}(2^{n^k})$ — класс задач, решаемых *с экспоненциальными затратами памяти*.

Класс P

$$\mathrm{P} = \bigcup_{k=0}^{\infty} \mathrm{DTIME}(n^k).$$

Класс P *принято считать* классом «быстро» решаемых задач.

Примеры полиномиально решаемых задач (не только задач принадлежности):

- целочисленное сложение, вычитание, умножение, деление, нахождение остатка от деления;
- умножение матриц; решение систем линейных уравнений;
- выяснение связности графа, эйлеровости графа, полуэйлеровости графа;
- сортировка конечного множества;
- поиск нужного слова в тексте;
- построение покрывающего граф дерева (остова графа) наименьшей стоимости;
- нахождение кратчайшего пути между вершинами графа;
- задача линейного программирования.

Об иерархии классов

Справедливы следующие соотношения:

- $P \subseteq PSPACE \subseteq EXPTIME \subseteq EXPSPACE$;
- $P \neq EXPTIME$;
- $PSPACE \neq EXPSPACE$.

Доказательство неравенства классов получается за счёт разделяющих задач.

Пример: P $\neq$ EXPTIME

Пусть L состоит из кодов таких машин Тьюринга M, что M остановится на своём коде $\varkappa M$ не более чем через $2^{|\varkappa M|}$ шагов.

Утверждение. $L \in \mathrm{EXPTIME}$, $L \notin \mathrm{P}$.

Доказательство.
Включение $L \in \mathrm{EXPTIME}$ следует из определения L.
Предположим, что $L \notin \mathrm{P}$, т.е. существует такая машина M_1, что

$$M_1(\varkappa M) = \begin{cases} 1, & \text{если } \varkappa M \in L, \\ 0, & \text{если } \varkappa M \notin L \end{cases}$$

и при этом $t_{M_1}(n) \leqslant p(n)$, где $p(n)$ — полином.
Преобразуем M_1 в M_2:

$$M_2(\varkappa M) = \begin{cases} \text{не опр.}, & \text{если } \varkappa M \in L, \\ 0, & \text{если } \varkappa M \notin L, \end{cases}$$

причём можно считать, что $2^{|\varkappa M_2|} > p(|\varkappa M_2|)$.
Но тогда вопрос: $M_2(\varkappa M_2) = ??$ $\square$

Как быть с PSPACE?

Заметим, что

- $\mathrm{P} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXPTIME}$;
- $\mathrm{P} \neq \mathrm{EXPTIME}$;
- имеются задачи — **SAT**, **ГАМИЛЬТОНОВ ГРАФ**, **КЛИКА** и др. — которые принадлежат классу PSPACE.

Вопрос: решаются ли эти задачи *полиномиально по времени*?

Замечание: Оценка $t_M(x) \leqslant 2^{c \cdot s_M(x)}$ не даёт ответ.

Недетерминированные алгоритмы

Недетерминированная машина Тьюринга определяется так же, как и «обычная» машина Тьюринга, с той разницей, что в определении отсутствует требование детерминированности, т.е. разрешается, чтобы программа содержала несколько различных инструкций с одинаковой левой частью.

Это приводит к тому, что на входе x недетерминированная машина Тьюринга может иметь несколько вычислений (в том числе с разными результатами).

Замечание. «Обычные» машины Тьюринга называют детерминированными; но при этом они являются и недетерминированными тоже.

Недетерминированные алгоритмы: примеры

Задача: для данного конечного множества G, операции $+$ и элемента 0 докажите, что $(G, +, 0)$ не является группой.

Алгоритм: проверьте, что не выполняется хотя бы одно из следующих условий:

- $\forall x, y \in G \;\; x + y \in G$;
- $\forall x, y, z \in G \;\; x + (y + z) = (x + y) + z$;
- $\forall x \in G \;\; x + 0 = x$;
- $\forall x \in G \; \exists y \in G \;\; x + y = 0$.

Задача: для данного булевой формулы A от n переменных докажите, что A выполнима.

Алгоритм: найдите набор значений переменных формулы A, на котором A принимает значение 1.

Результат работы недетерминированного алгоритма

Пусть M — недетерминированная машина Тьюринга, каждое вычисление которой (на любых входных данных) заканчивается либо с результатом 1, либо с результатом 0.

Пусть x — слово; считаем, что M принимает x, если *существует* вычисление машины M на слове x, заканчивающееся результатом 1.

Замечание: если *каждое* вычисление M на слове x заканчиваются результатом 0, то M не принимает x.

Классы NTIME и NSPACE

Значения функций $t_M(x)$ и $s_M(x)$ для недетерминированной машины M и слова x определяются так же, как и для детерминированных машин, с той разницей, что берётся максимум по всем вычислениям M на x.

Функции $t_M : \mathbb{N} \to \mathbb{N}$ и $s_M : \mathbb{N} \to \mathbb{N}$ определяются как и раньше.

Пусть имеется функция $f : \mathbb{N} \to \mathbb{N}$.

Определим классы:

- NTIME(f) — класс задач, решаемых *недетерминированными* машинами Тьюинга, временна́я сложность которых находится в классе $O(f)$;
- NSPACE(f) — класс задач, решаемых *недетерминированными* машинами Тьюинга, ёмкостная сложность которых находится в классе $O(f)$.

Пример: **SAT** $\in$ NTIME(n^2), **SAT** $\in$ NSPACE(n).

Классы NP, NEXPTIME и другие

- $\mathrm{NP} = \bigcup\limits_{k=0}^{\infty} \mathrm{NTIME}(n^k)$ — класс задач, решаемых *недетерминированно и полиномиально по времени*;
- $\mathrm{NEXPTIME} = \bigcup\limits_{k=0}^{\infty} \mathrm{NTIME}(2^{n^k})$ — класс задач, решаемых *недетерминированно и экспоненциально по времени*;
- $\mathrm{PSPACE} = \bigcup\limits_{k=0}^{\infty} \mathrm{NSPACE}(n^k)$ — класс задач, решаемых *с полиномиальными затратами памяти*;
- $\mathrm{EXPSPACE} = \bigcup\limits_{k=0}^{\infty} \mathrm{NSPACE}(2^{n^k})$ — класс задач, решаемых *с экспоненциальными затратами памяти*.

Отношения между классами

Верны следующие отношения:

- $P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq NEXPTIME \subseteq EXPSPACE$;
- $P \neq EXPTIME$;
- $EXPTIME \neq NEXPTIME$;
- $PSPACE \neq EXPSPACE$.

Следующие проблемы не решены:

- $P = NP?$
- $P = PSPACE?$
- $NP = PSPACE?$
- $NP = EXPTIME?$
- $PSPACE = EXPTIME?$
- ...

Проблема равенства P и NP

Проблемы тысячелетия, определённые Математическим институтом Клэя (Кэмбридж) в 2000 году:

- гипотеза Пуанкаре (решена Г. Перельманом в 2010 году)
- равенство классов P и NP
- гипотеза Ходжа
- гипотеза Римана
- теория Янга – Миллса
- существование и гладкость решения уравнений Навье – Стокса
- гипотеза Бёрча – Свиннертон-Дайера

За решение каждой из этих проблем Математическим институтом Клэя назначена премия в 1 миллион долларов США.

Проблема равенства P и NP

Некоторые аспекты значимости проблемы равенства P и NP:

- эту проблему уже долго не могут решить
- многие важные для приложений задачи находятся в классе NP, при этом
  - для них неизвестны полиномиальные алгоритмы
  - для некоторых из них доказано, что если $P \neq NP$, то полиномиальных алгоритмов решения этих задач не существует
- области, содержащие такие задачи в огромном количестве:
  - математическая логика
  - графики, графы, гиперграфы
  - математическое программирование
  - формальные языки и обработка строк
  - игры и головоломки
  - шифрование и защита информации
- наличие содержательных интерпретаций

Классы дополнительных задач

Для класса задач C определим класс дополнительных задач $\mathrm{co}C = \{L : \bar{L} \in C\}$.

Пример: задача выполнимости булевых формул находится в NP, значит, задача невыполнимости булевых формул находится в coNP.

Утверждение. $\mathrm{coDTIME}(f) = \mathrm{DTIME}(f)$, $\mathrm{coDSPACE}(f) = \mathrm{DSPACE}(f)$.

Следствие. $\mathrm{coP} = \mathrm{P}$, $\mathrm{coPSPACE} = \mathrm{PSPACE}$, $\mathrm{coEXPTIME} = \mathrm{EXPTIME}$, $\mathrm{coEXPSPACE} = \mathrm{EXPSPACE}$.

Следующая проблема не решена:

- $\mathrm{coNP} = \mathrm{NP}$?

Следствие. Если $\mathrm{coNP} \neq \mathrm{NP}$, то $\mathrm{P} \neq \mathrm{NP}$.

Альтернирующие алгоритмы: примеры

Задача: для данного конечного множества G, операции $+$ и элемента 0 докажите, что $(G, +, 0)$ является группой.

Алгоритм: проверьте, что выполняется каждое из следующих условий:

- $\forall x, y \in G \;\; x + y \in G$;
- $\forall x, y, z \in G \;\; x + (y + z) = (x + y) + z$;
- $\forall x \in G \;\; x + 0 = x$ или $\forall x \in G \;\; 0 + x = x$;
- $\forall x \in G \; \exists y \in G \;\; x + y = 0$.

Задача: для данного булевой формулы A от n переменных докажите, что A эквивалентна логической константе (тождественно истинна или тождественно ложна).

Алгоритм: проверьте, что на каждом наборе значений переменных формулы A эта формула принимает значение 1 или что на каждом наборе значений переменных формулы A эта формула принимает значение 0.

Альтернирующие машины Тьюринга

Альтернирующая машина Тьюринга M определяется так же, как и недетерминированная машина Тьюринга, с той разницей, что множество состояний машины M разбивается на две группы: a-состояния и e-состояния.

Как и раньше, вычисление M на x — это последовательностей конфигураций, первая из которых является начальной с x на ленте, а каждая следующая получается из предыдущей выполнением одной инструкции.

Альтернирующие машины Тьюринга

Принимающая конфигурация в вычислении M на x:

- заключительная конфигурация считается принимающей, если на ленте записан результат 1;
- конфигурация с e-состоянием считается принимающей, если из неё за один шаг M *может прийти* в принимающую конфигурацию;
- конфигурация с a-состоянием считается принимающей, если из неё за один шаг M *каждый раз приходит* в принимающую конфигурацию.

Полагаем что машина M принимает слово x, если начальная конфигурация M на x является принимающей.

Классы ATIME и ASPACE

Значения функций $t_M(x)$ и $s_M(x)$ для альтернирующей машины M и слова x определяются так же, как и для недетерминированных машин: берётся максимум по всем вычислениям M на x.

Функции $t_M : \mathbb{N} \to \mathbb{N}$ и $s_M : \mathbb{N} \to \mathbb{N}$ определяются как и раньше.

Пусть имеется функция $f : \mathbb{N} \to \mathbb{N}$. Определим классы:

- ATIME(f) — класс задач, решаемых *альтернирующими* машинами Тьюинга, временна́я сложность которых находится в классе $O(f)$;
- ASPACE(f) — класс задач, решаемых *альтернирующими* машинами Тьюинга, ёмкостная сложность которых находится в классе $O(f)$.

Пример: **QBF** $\in$ ATIME(n^2), **QBF** $\in$ ASPACE(n).

Пояснение: Задача **QBF** — это задача выяснения по булевой формуле с кванторами A, является ли A истинной. Булева формула с кванторами — это формула, при построении которой, помимо булевых связок, используются кванторы вида $\exists p$ и $\forall p$.

Теорема Савича

Теорема Савича

$\mathrm{NSPACE}(f) \subseteq \mathrm{ATIME}(f^2) \subseteq \mathrm{DSPACE}(f^2)$

- $\mathrm{AP} = \bigcup\limits_{k=0}^{\infty} \mathrm{ATIME}(n^k)$;
- $\mathrm{AEXPTIME} = \bigcup\limits_{k=0}^{\infty} \mathrm{ATIME}(2^{n^k})$;
- $\mathrm{DPSPACE} = \bigcup\limits_{k=0}^{\infty} \mathrm{DSPACE}(n^k)$; $\mathrm{NEXPSPACE} = \bigcup\limits_{k=0}^{\infty} \mathrm{NSPACE}(2^{n^k})$;
- $\mathrm{NPSPACE} = \bigcup\limits_{k=0}^{\infty} \mathrm{NSPACE}(n^k)$; $\mathrm{DEXPSPACE} = \bigcup\limits_{k=0}^{\infty} \mathrm{DSPACE}(2^{n^k})$.

Следствие. $\mathrm{NPSPACE} = \mathrm{DPSPACE} = \mathrm{AP}$,
$\mathrm{NEXPSPACE} = \mathrm{DEXPSPACE} = \mathrm{AEXPTIME}$.

Полиномиальная сводимость задач

Задача «$x \in L_1$?» полиномиально сводится к задаче «$y \in L_2$?», если существует такая полиномиально вычислимая (по времени) функция f, что для всякого слова x

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

Обозначение: $L_1 \leqslant_p L_2$.

Замечание. Если $L_1 \leqslant_p L_2$, то сложность задачи «$x \in L_1$?» можно грубо оценить сверху как сложность задачи вычисления f плюс сложность «$y \in L_2$?».
Значит,

- если L_2 решается «просто», то и L_1 решается «просто»;
- если L_1 решается «трудно», то и L_2 решается «трудно».

Полиномиальная сводимость задач: примеры

- $2 \cdot \mathbb{Z} \leqslant_p 2 \cdot \mathbb{Z} + 1$
- $2 \cdot \mathbb{Z} + 1 \leqslant_p 2 \cdot \mathbb{Z}$
- $\overline{\textbf{SAT}} \leqslant_p \textbf{CL}$
- $\textbf{CL} \leqslant_p \overline{\textbf{SAT}}$
- $\textbf{SAT} \leqslant_p \textbf{QBF}$
- $\overline{\textbf{SAT}} \leqslant_p \textbf{QBF}$
- $\textbf{CL} \leqslant_p \textbf{Int}$
- $\textbf{QBF} \leqslant_p \textbf{Int}$
- $\textbf{Int} \leqslant_p \textbf{QBF}$
- $\textbf{SAT} \leqslant_p \textbf{КЛИКА}$
- $\textbf{КЛИКА} \leqslant_p \textbf{SAT}$

Трудные и полные задачи в классе

Пусть C — некоторый класс сложности.

Задача «$x \in L$?» называется C-трудной, если к ней полиномиально сводится любая задача из класса C.

Задача «$x \in L$?» называется C-полной, если

- она C-трудна;
- она принадлежит классу C.

Теорема Кука – Левина

Теорема Кука – Левина

Задача **SAT** является NP-полной

Следствие. *Задача* $\overline{\textbf{SAT}}$ *является* coNP*-полной.*

Следствие. $\mathrm{P} = \mathrm{NP} \iff \textbf{SAT} \in \mathrm{P}$.

Другие полные задачи

- **SAT**, **КЛИКА**, **ГАМИЛЬТОНОВ ГРАФ**, **ЗАДАЧА О РЮКЗАКЕ**, **ЗАДАЧА О КОММИВОЯЖЁРЕ**, ... — NP-полны
- **CL**, $\overline{\textbf{SAT}}$, ... — coNP-полны
- **QBF**, **Int**, **LTL** ... — PSPACE-полны
- **PDL**, **CTL**, ... — EXPTIME-полны
- **IPDL**, **CTL***, **ATL***, ... — 2EXPTIME-полны

Следствие. NP = PSPACE $\iff$ **QBF** $\leqslant_p$ **SAT** $\in$ NP.

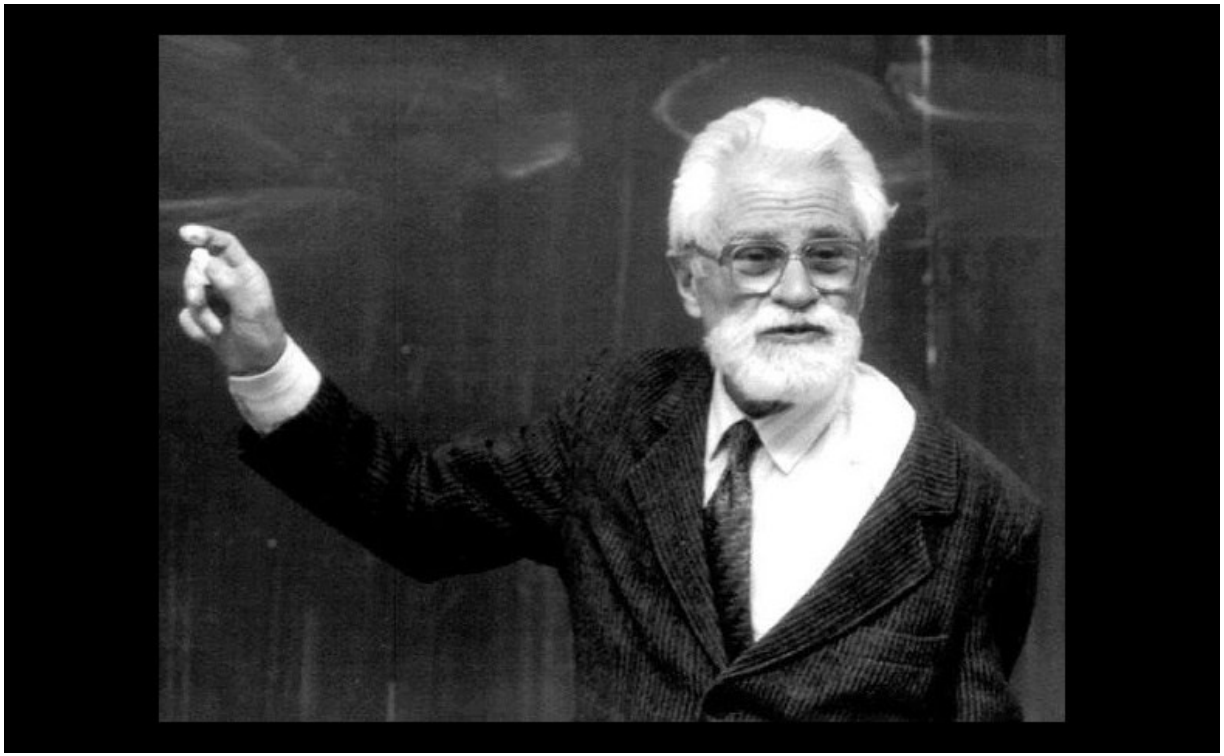
Следствие. NP = PSPACE $\iff$ **Int** $\leqslant_p$ **CL** $\in$ coNP.

Следствие. PSPACE = EXPTIME $\iff$ **CTL** $\leqslant_p$ **QBF** $\in$ PSPACE.

Логики ТвГУ

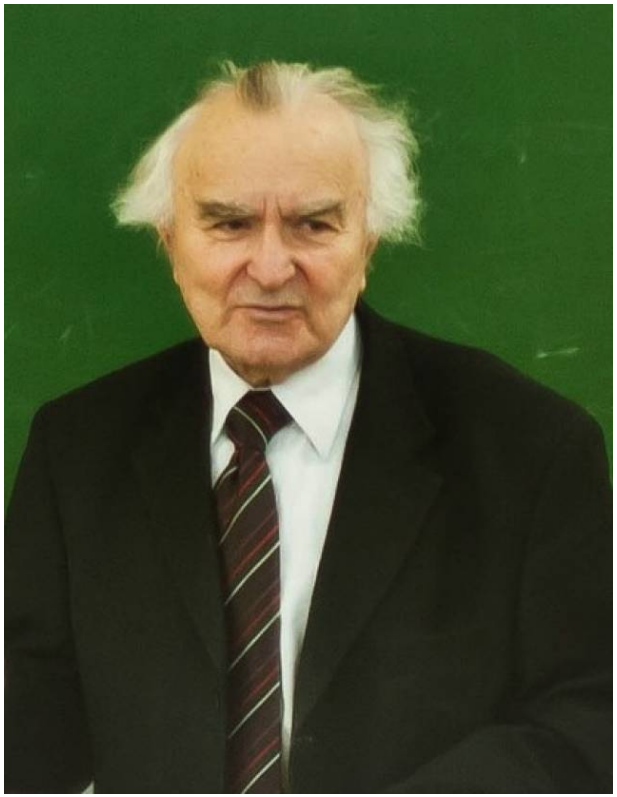
Логики ТвГУ

Алексей Всеволодович Гладкий (1928–2018гг.)



Исследователь в области математической логики и лингвистики. Автор ряда книг, в том числе «Элементы математической лингвистики» (совместно с И.А.Мельчуком, М., 1969), «Формальные грамматики и языки» (М., 1973), «Математическая логика» (М., 1998), «Введение в современную логику. Учебное пособие» (М., 2001). Доктор физико-математических наук, профессор.

Михаил Абрамович Тайцлин (1936–2013гг.)



Советский и российский математик, специалист в области математической логики и теоретической информатики. Доктор физико-математических наук, профессор, заведовал кафедрой информатики факультета ПМиК в ТвГУ.
Основные работы — по алгебре, математической логике, теоретическому программированию (более 100 научных работ).

Среди алгебраических результатов — создание структурной теории конечно порождённых коммутативных полугрупп, решение серии алгоритмических проблем и проблемы изоморфизма для коммутативных полугрупп. Внёс вклад в исследование категоричных квазимногообразий.

Михаил Иосифович Дехтярь (1946–2018гг.)

Доктор физико-математических наук, профессор, работал на кафедре информатики факультета ПМиК в ТвГУ.

Область научных интересов: теоретическая информатика, сложность вычислений, вычислительная лингвистика, биоинформатика, колмогоровская сложность, эффективные алгоритмы для конкретных задач, искусственный интеллект, логическое программирование, дедуктивные и активные базы данных, временные и вероятностные базы данных.

Александр Васильевич Чагров (1957–2016гг.)

Доктор физико-математических наук, профессор, работал на математическом факультете ТвГУ.

Известен своими исследованиями и результатами в области модальных и суперинтуиционистских логик. Решил много вопросов в области неклассических логик, в том числе связанных с алгоритмической и сложностной проблематикой.

Автор книги «Modal Logic» (совместно с М.В.Захарьящевым, Oxford University Press, 1997).

Является автором более 135 научных публикаций.

Макс Иосифович Канович

Доктор физико-математических наук, профессор, работал на математическом факультете ТвГУ.
Известен своими работами в области математической логики и её приложений в теоретической информатике.

С 2015 года работает в Высшей школе экономики, совмещая с работой в Лондонском Университетском колледже (University College London).

Логики, работающие в ТвГУ: ПМиК

Сергей Михайлович Дудаков, доктор физико-математических наук, доцент. Основные темы научной работы: теория вычислительной сложности, теория языков первого порядка, теория баз данных, логическое программирование.

Дмитрий Ольгердович Дадеркин, кандидат физико-математических наук, доцент. Сфера научных интересов: теоретическая информатика, моделирование бизнес-процессов.

Борис Николаевич Карлов, кандидат физико-математических наук. Сфера научных интересов: формально-логические языки, искусственный интеллект, мультиагентные системы.

Логики, работающие в ТвГУ: матфак

Лилия Алексеевна Чагрова, кандидат физико-математических наук, доцент. Известна работами в области модальных и суперинтуиционистских логик. Доказала алгоритмическую неразрешимость основных вопросов теории соответствий (теорема Чагровой).

Игорь Анатольевич Горбунов, кандидат физико-математических наук. Известен результатами в области нормальных и квазинормальных модальных логик. Построил примеры логик, не имеющих независиомой аксиоматизации. Постоил пример разрешимой логики, логика конечных моделей которой неразрешима.

Михаил Николаевич Рыбаков, кандидат физико-математических наук, доцент. Получил ряд результатов об алгоритмической (и сложностной) выразительности различных неклассических логик и теорий нулевого и первого порядка, а также некоторых их фрагментов.

Что ещё имеется?

Какие ещё темы можно затронуть?

Логика

- Классическая логика: углублённый курс. Теория моделей.
- Интуиционистская (конструктивная) логика: семантика возможных миров, дедуктивные и алгоритмические свойства.
- Модальные логики. Обзорный курс: синтаксис, семантика, дедуктивные и алгоритмические свойства, выразимость свойств отношений.
- «Компьютерные» логики. Обзорный курс: динамические логики, логики линейного и ветвящегося времени, логики знания и другие.
- Исчисления.
- Нечёткие множества, нечёткие логики.
- ...
- Релевантные логики? Логика второго порядка? «Женская логика»?

Алгебра

- Линейная алгебра: линейные и евклидовы пространства, линейные преобразования пространств. Вводный курс.
- Алгебраические структуры: моноиды, группоиды, полугруппы, группы, кольца, поля. Вводный курс.
- Универсальная алгебра. Вводный курс.
- ...
- Реляционные алгебры, базы данных?

Алгоритмы

- Теория алгоритмов: углублённый курс.
- Разрешимые и неразрешимые теории: углублённый курс.
- Теория вычислительной сложности: углублённый курс.
- Эффективные алгоритмы.
- Различные формализации понятия алгоритма и их эквивалентность.
- Степени неразрешимости.
- ...
- Объектно ориентированное программирование?

Разное

- Теория графов. Задачи на графах и их сложность.
- Существование математического объекта: конструктивные и неконструктивные обоснования.
- Игры как средство решения математических вопросов.
- Формальные языки: автоматные, контекстно-свободные, разрешимые, рекурсивно перечислимые и другие.
- Арифметические теории и теории множеств. Теоремы Гёделя о неполноте. Теорема Тарского о невыразимости истины.
- Нестандартный анализ.
- ...
- Дескриптивные языки, дескриптивные логики, базы знаний?

Благодарность

Спасибо за внимание!

Быть может, имеются вопросы? Задавайте!

Автор выражает особую благодарность:

Александру Васильевичу Чагрову

Лилии Алексеевне Чагровой

Борису Семёновичу Галюкшову

Михаилу Абрамовичу Тайцлину

Игорю Анатольевичу Горбунову

Дмитрию Петровичу Шкатову

Александру Степановичу Карпенко

Валентину Борисовичу Шехтману

Льву Дмитриевичу Беклемишеву

Владлену Анатольевичу Тиморину

Юлии Владимировне Чемариной