Лекція 11

Тема 8. Теорія обчислень

Машини Тьюрінга

План лекції

- **Основні вимоги до алгоритмів**
- **Означення машини Тьюрінга**
- **Обчислення числових функцій на машинах Тьюрінга**
- **Теза Тьюрінга. Приклади алгоритмічно нерозв'язних проблем**

Основні вимоги до алгоритмів

1. Будь-який алгоритм застосовують до початкових даних, і він видає результати. У звичних технічних термінах це означає, що алгоритм має *входи* й *виходи*. Отже, алгоритм застосовують для розв'язування цілого класу задач із різними початковими даними (цю властивість називають *масовістю*). Крім того, під час роботи алгоритму з'являються проміжні результати, використовувані в подальшому. Звідси випливає, що, кожний алгоритм обробляє *дані* — вхідні, проміжні та вихідні. Оскільки ми збираємось уточнити поняття алгоритму, потрібно уточнити й поняття даних, тобто зазначити, яким вимогам мають задовольняти об'єкти, щоб алгоритми могли з ними працювати.

2. Для розміщення даних потрібна *пам'ять*. Її зазвичай уважають однорідною й дискретною, тобто такою, що складається з однакових комірок, причому кожна комірка може містити один символ алфавіту даних. Отже, одиниці виміру об'єму даних і пам'яті узгоджені, при цьому пам'ять може бути нескінченною. Питання про те, чи потрібна одна пам'ять, чи декілька й, зокрема, чи потрібна окрема пам'ять для кожного з трьох типів даних, вирішують по-різному.

3. Алгоритм складається з окремих *елементарних кроків* (або *дій*), причому множина різних кроків, з яких утворено алгоритм, скінченна. Типовий приклад множини елементарних дій – система команд процесора. Зазвичай на елементарному кроці обробляється фіксована кількість символів, проте є команди, які працюють із полями пам'яті зі змінною довжиною.

4. Послідовність кроків алгоритму *детермінована*, тобто після кожного кроку або зазначено, який крок робити далі, або виконується команда зупинки, після чого робота алгоритму вважається закінченою.

5. Природно від алгоритму вимагати *результативності*, тобто зупинки після скінченної кількості кроків (яка залежить від початкових даних) із зазначенням того, що вважати результатом. Зокрема, алгоритм для обчислення функції $f(x)$ має зупинятися після скінченної кількості кроків для будь-якого x із області визначення функції f. У такому разі говорять, що алгоритм *збігається*. Проте перевірити результативність (збіжність) значно важче, ніж вимоги, викладені в пп. 1–4. На відміну від них збіжність переважно не вдається виявити простим переглядом опису алгоритму. Загального ж методу перевірки збіжності, придатного для довільного алгоритму A й довільних початкових даних α, як ми доведемо в подальшому, взагалі не існує.

Означення машини Тьюрінга

Машина Тьюрінга – це математична модель пристрою, який породжує обчислювальні процеси. Її використовують для теоретичного уточнення поняття алгоритму та його дослідження. Названо цю модель ім'ям англійського математика А. Тьюрінга (A. Turing), який запровадив її 1936 р. У кожній машині Тьюрінга є три частини (рис. 1).

1. Стрічка, поділена на комірки.
2. Пристрій керування (ПК).
3. Головка читання-запису (Г).

З кожною машиною Тьюрінга пов'язано два скінченні алфавіти: *алфавіт зовнішніх символів A* та *алфавіт внутрішніх станів* $Q=\{q_0, q_1, \ldots, q_k\}$. Із різними машинами Тьюрінга можуть пов'язуватись різні алфавіти.

Алфавіт зовнішніх символів *A* часто називають *зовнішнім алфавітом*, а його елементи – *буквами*. Один символ з алфавіту *A* називають *порожнім*, зазвичай його позначають Λ. Усі інші букви з алфавіту *A*, крім Λ, називають *непорожніми*. Комірку стрічки, у якій записано букву Λ, називають *порожньою*.

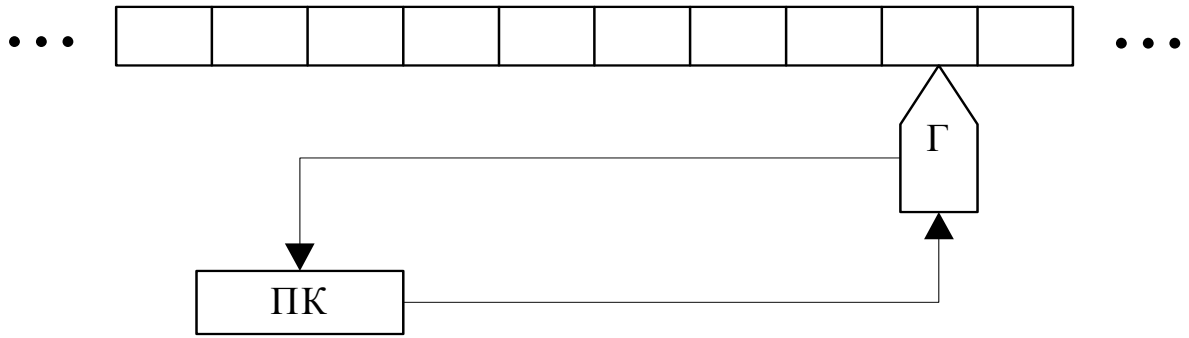


Рис. 1

Машина Тьюрінга працює в часі, який уважають дискретним; його моменти занумеровано: 0, 1, 2, 3,… . У кожний момент часу стрічка містить скінченну кількість комірок. Головка пересувається вздовж стрічки; у кожний момент головка перебуває над певною коміркою стрічки. У такому разі говорять, що головка *зчитує* букву, записану в цій комірці. У наступний момент головка залишається над цією ж коміркою (що позначають *Н*), або пересувається на одну комірку вправо (що позначають *П*), або пересувається на одну комірку вліво (що позначають *Л*). Якщо в даний момент t головка перебуває над крайньою коміркою та зсувається на комірку, якої немає, то автоматично прибудовується нова порожня (тобто з порожньою буквою Λ) комірка, над якою перебуватиме головка в момент t+1. Отже, стрічка *потенційно нескінченна в обидва боки*, тобто до неї завжди можна додати нові комірки як зліва, так і справа.

Алфавіт внутрішніх станів $Q=\{q_0, q_1, \ldots, q_k\}$ – це *внутрішня пам'ять*. На відміну від нескінченної *зовнішньої пам'яті* на стрічці вона скінченна. Елемент q_0 називають *заключним внутрішнім станом*, а елемент q_1 – *початковим внутрішнім станом*. Пересування головки вздовж стрічки залежить від букви, яка зчитується, і від внутрішнього стану машини.

Пристрій керування в кожний момент t залежно від букви, яка зчитується в цей момент зі стрічки, і внутрішнього стану машини виконує такі дії:

1. Змінює букву a_i, яка зчитується в момент t зі стрічки, на нову букву a_j (може бути $a_j = a_i$).
2. Пересуває головку в одному з напрямків *Н*, *П*, *Л*;
3. Змінює внутрішній стан машини q_i в момент t на новий стан q_j, у якому машина перебуватиме в момент t+1 (може бути $q_j = q_i$).

Таку дію пристрою керування називають *командою*, її записують так:

$$q_i a_i \rightarrow a_j D q_j,$$

де q_j – внутрішній стан машини в даний момент; a_i – буква на стрічці, зчитувана в цей момент; a_j – буква, на яку змінюється буква a_i (може бути $a_j = a_i$); символ D – це або *Н*, або *П*, або *Л* і вказує

напрям пересування головки; q_j – внутрішній стан машини в наступний момент часу (може бути $q_j = q_i$).

Вирази $q_i a_i$ та $a_j D q_j$ називають, відповідно, *лівою* та *правою* частинами цієї команди. У лівій частині жодної команди q_0 не зустрічається. Усіх команд, у яких ліві частини попарно різні, скінченна кількість (оскільки множини $Q\backslash\{q_0\}$ та A скінченні); їх сукупність називають *програмою машини*. Зазначимо, що жодні дві команди не можуть мати однакові ліві частини.

Виконання однієї команди називають *кроком*. Обчислення (або робота) машини Тьюрінга – це послідовність кроків одного за одним без пропусків, починаючи з початкового. Роботу машини Тьюрінга повністю визначено, якщо в початковий момент задано:

1) слово на стрічці, тобто послідовність букв, записаних у її комірках (слово одержують читанням цих букв у комірках стрічки зліва направо);

2) положення головки Г;

3) внутрішній стан машини.

Сукупність цих трьох умов (у даний момент часу t) називають *конфігурацією* (у даний момент часу t). Уважатимемо, що в початковий момент внутрішній стан машини – q_1, а головка перебуває над першою зліва коміркою стрічки.

Отже, у початковий момент конфігурація така: на стрічці, що складається з якоїсь кількості комірок (не менше однієї), у кожній комірці записано одну з букв зовнішнього алфавіту A, головка над першою зліва коміркою стрічки, внутрішній стан машини – q_1. Наприклад, якщо в початковий момент на стрічці записано слово $a_1 \Lambda a_2 a_1 a_1$ то початкова конфігурація така:

a_1	Λ	a_2	a_1	a_1

q_1

(під коміркою, над якою перебуває головка, зазначено внутрішній стан машини). Отже, програма й початкова конфігурація повністю визначають роботу машини над словом, яке є на стрічці в початковій конфігурації. Тому вважатимемо машину Тьюрінга заданою, якщо відома її програма.

Означення машини Тьюрінга можна подати у формі, як це було зроблено для скінченних автоматів.

Машиною Тьюрінга називають систему $M = (A, Q, q_0, q_1, \Lambda, \delta)$, у якій A, Q – скінченні множини, Λ – порожня буква, $\Lambda \in A$, q_0 – заключний стан, q_1 – початковий стан, q_0, $q_1 \in Q$, δ – функція *переходів*, визначена на декартовому добутку $(Q \setminus \{q_0\}) \times A$:

$$\delta: (Q \setminus \{q_0\}) \times A \to A \times \{Л, П, Н\} \times Q.$$

Зазначимо, що функція δ – векторна, значення $\delta(q_i, a_i)$ – трійка (a_j, D, q_j) – права частина команди.

Якщо в роботі машини Тьюрінга в момент t виконається команда, права частина котрої містить символ q_0, то в такий момент роботу машини вважають завершеною; тоді говорять, що машина *застосовна* до слова на стрічці в початковій конфігурації. Справді, q_0 не зустрічається в лівій частині жодної команди, тому його називають заключним станом. Результатом роботи машини в такому разі вважають слово, яке буде записано на стрічці в *заключній конфігурації*, тобто в конфігурації з внутрішнім станом q_0.

Якщо ж у роботі машини в жодний момент не зустрінеться заключний стан, то процес обчислень буде нескінченним. У такому разі говорять, що машина *незастосовна* до слова на стрічці в початковій конфігурації.

Зазначимо, що $q_0 \in Q$ для кожної машини Тьюрінга.

Приклад 1. Побудуємо машину Тьюрінга T_1, котра застосовна до всіх слів у алфавіті $\{a, b\}$ та перетворює довільне слово $x_1x_2\ldots x_n$, де $x_i \in \{a, b\}$, $i = 1, \ldots, n$ на слово $x_2\ldots x_nx_1$.

Як зовнішній алфавіт машини візьмемо множину $A = \{\Lambda, a, b\}$; команди разом із коментарями запишемо в два стовпчики. Для запам'ятовування букв алфавіту стрічки потрібно перейти в новий стан.

Команда	**Коментар**
$q_1 a \to \Lambda\, П\, q_2$	Перша буква a запам'ятовується переходом у стан q_2 і стирається.
$q_1 b \to \Lambda\, П\, q_3$	Перша буква b запам'ятовується переходом у стан q_3 і стирається.
$q_2 a \to a\, П\, q_2$ $q_2 b \to b\, П\, q_2$ $q_2 \Lambda \to a\, Н\, q_0$	Проходження головки через слово $x_2 \ldots x_n$ і запис у його кінці букви $x_1 = a$, зупинка.
$q_3 a \to a\, П\, q_3$ $q_3 b \to b\, П\, q_3$ $q_3 \Lambda \to b\, Н\, q_0$	Проходження головки через слово $x_2 \ldots x_n$ і запис у його кінці букви $x_1 = b$, зупинка.

Команди можна записати у вигляді таблиці, яка задає функцію переходів δ: величина, записана в рядку q_i та стовпці a_i – це значення $\delta(q_i, a_i)$. Цю таблицю називають *функціональною схемою Тьюрінга* (табл. 1).

Таблиця 1

	Λ	*a*	*b*
q_1	–	$\Lambda П q_2$	$\Lambda П q_3$
q_2	aHq_0	$aП q_2$	$bП q_2$
q_3	bHq_0	$aП q_3$	$bП q_3$

У таблиці прочерком позначено неістотну команду, тобто таку, яка не зустрінеться під час роботи цієї машини.

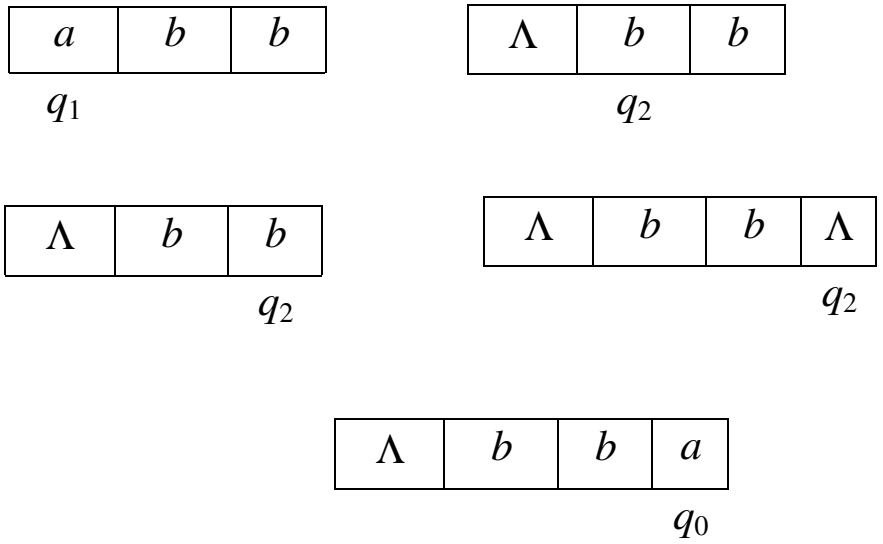


Рис. 2

Розглянемо роботу машини T_1 над словом *abb*. Конфігурації, які при цьому виникають, зображено на рис. 2.

Приклад 2. Побудуємо машину Тьюрінга T_2, яка застосовна до всіх слів в алфавіті $\{a, b\}$ та знімає копію слова на стрічці, тобто перетворює слово $x_1 \ldots x_n$ у початковій конфігурації на слово $x_1 \ldots x_n \Lambda x_1 \ldots x_n$ у заключній конфігурації.

x_1	x_2	…	x_{i-1}	x_i	x_{i+1}	…	x_n	Λ	x_1	x_2	…	x_{i-1}

q_1

Рис. 3

Цю машину можна побудувати, наприклад, так. Нехай задано якесь слово в алфавіті $\{a, b\}$, що складається з n букв ($n \geq 1$). Робота машини складається з n циклів. Для $i \leq n$ на початок i-того циклу конфігурацію зображено на рис. 3. Отже, скопійовано $i-1$ букв слова, і головка зчитує букву x_i. Черговий цикл виконуватиметься в три етапи.

1. Запам'ятовування букви x_i (при цьому вона позначається, тобто фактично замінюється іншою буквою) та пересування головки вправо.

2. Пошук правої крайньої порожньої комірки, у яку записується x_i.

3. Повернення головки вліво до позначеної букви, стирання мітки (тобто відновлення букви x_i) та пересування головки на одну клітку вправо.

Перший етап реалізують такі команди.

Команда	**Коментар**
$q_1a \to a'\Pi q_2$	a' – позначена буква a, q_2 – стан запам'ятовування букви a.
$q_1b \to b'\Pi q_3$	b' – позначена буква b, q_3 – стан запам'ятовування букви b.

Другий етап.

Команда	**Коментар**
$q_2a \to a\,\Pi\,q_2$ $q_2b \to b\,\Pi\,q_2$	Проходження залишку слова в стані q_2.
$q_3a \to a\,\Pi\,q_3$ $q_3b \to b\,\Pi\,q_3$	Проходження залишку слова в стані q_3.
$q_2\Lambda \to \Lambda\Pi q_4$ $q_3\Lambda \to \Lambda\Pi q_5$	Проходження головки через порожню букву Λ між словами; q_4 – копіюватиметься буква a; q_5 – копіюватиметься буква b.
$q_4a \to a\Pi\,q_4$ $q_4b \to b\Pi\,q_4$ $q_5a \to a\Pi\,q_5$ $q_5b \to b\Pi\,q_5$	Проходження початкової частини копії слова.
$q_4\Lambda \to a\,Л\,q_6$ $q_5\Lambda \to b\,Л\,q_6$	Запис копійованої букви.

Третій етап.

Команда	**Коментар**
$q_6 a \to a\, Л\, q_6$ $q_6 b \to b\, Л\, q_6$ $q_6 \Lambda \to \Lambda\, Л\, q_6$ $q_6 a' \to a\, П\, q_1$ $q_6 b' \to b\, П\, q_1$	Рух головки вліво до позначеної букви, стирання мітки, пересування на одну клітку вправо з переходом у початковий стан q_1. Машина готова до виконання чергового циклу копіювання букви.
$q_1 \Lambda \to \Lambda П q_0$	Команда зупинки, копію слова побудовано.

Отже, зовнішній алфавіт машини T_2 – множина $A = \{\Lambda, a, b, a', b'\}$, множина внутрішніх станів $Q = \{q_0, q_1, \ldots, q_6\}$. Нарешті, запишемо команди у вигляді функціональної схеми Тьюрінга, яка задає функцію переходів $\delta(q_i, a_i)$ (табл. 2).

Таблиця 2

	Λ	a	b	a'	b'
q_1	$\Lambda П q_0$	$a' П q_2$	$b' П q_3$	–	–
q_2	$\Lambda П q_4$	$a П q_2$	$b П q_2$	–	–
q_3	$\Lambda П q_5$	$a П q_3$	$b П q_3$	–	–
q_4	$a Л q_6$	$a П q_4$	$b П q_4$	–	–
q_5	$b Л q_6$	$a П q_5$	$b П q_5$	–	–
q_6	$\Lambda Л q_6$	$a Л q_6$	$b Л q_6$	$a П q_1$	$b П q_1$

Зазначимо, що алфавіт зовнішніх символів A в цьому прикладі окрім порожньої букви Λ та букв a і b, у яких записують слово для копіювання, містить іще додаткові букви a', b', використовувані в проміжних обчисленнях. Іноді в зовнішньому алфавіті A явно виділяють підмножину V *вхідних символів*, $V \subset A$, $\Lambda \in A \setminus V$. У такому разі в цьому прикладі $A = \{\Lambda, a, b, a', b'\}$, $V = \{a, b\}$.

Обчислення числових функцій на машинах Тьюрінга

Числовою називають функцію $f(x_1, \ldots, x_n)$, значеннями якої як і значеннями її аргументів є невід'ємні цілі числа. Розглядатимемо часткові числові функції, тобто числові функції, визначені, узагалі кажучи, не для всіх значень аргументів.

Для обчислення числових функцій на машинах Тьюрінга використовують спеціальне кодування чисел. Наприклад, невід'ємне ціле число m можна задавати набором з $(m+1)$ одиниць, який позначатимемо 1^{m+1}. Отже, 0 задаватимемо як 1, 1 – як 11, 2 – як 111 тощо.

Числову функцію $f(x_1, \ldots, x_n)$ називають *обчислюваною за Тьюрінгом,* якщо існує така машина Тьюрінга T, що для довільних цілих невід'ємних чисел $m_1, m_2, \ldots, m_n$, коли $f(m_1, m_2, \ldots, m_n) = m$, то машина T – застосовна до слова $1^{m_1+1}\Lambda 1^{m_2+1}\Lambda\ldots\Lambda 1^{m_n+1}$ і в заключній конфігурації на якомусь відрізку стрічки буде записано слово 1^{m+1}, а решта комірок (якщо такі будуть) виявляться порожніми. Якщо ж $f(m_1, m_2, \ldots, m_n)$ невизначено, то машина T *незастосовна* до слова $1^{m_1+1}\Lambda 1^{m_2+1}\Lambda\ldots\Lambda 1^{m_n+1}$.

Приклад 3. Побудуємо машину Тьюрінга, яка обчислює числову функцію $s(x) = x+1$. Зовнішній алфавіт $A = \{\Lambda, 1\}$, множина станів $Q = \{q_0, q_1\}$, а команди можна задати так: $q_1 1 \to 1 П q_1$, $q_1 \Lambda \to 1 H q_0$. Конфігурації, які виникають у разі обчислення $s(1)$, зображено на рис. 4.

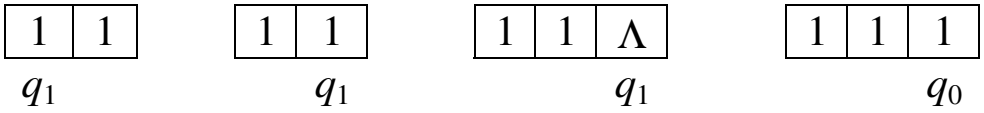


Рис. 4

Приклад 4. Побудуємо машину Тьюрінга, яка обчислює числову функцію $f(x, y)=x + y$. Зовнішній алфавіт $A = \{\Lambda, 1\}$, множина станів $Q = \{q_0, q_1, q_2, q_3, q_4\}$. У процесі роботи машини головка рухається вправо й символ Λ між аргументами замінюється на 1. Далі продовжується рух праворуч до першої комірки із символом Λ, після чого головка починає рухатись вліво й стирає дві останні 1 поспіль.

Команди можна задати так: $q_1 1 \to 1 П q_1$, $q_1 \Lambda \to 1 П q_2$, $q_2 1 \to 1 П q_2$, $q_2 \Lambda \to \Lambda Л q_3$, $q_3 1 \to \Lambda Л q_4$, $q_4 1 \to \Lambda Н q_0$. Конфігурації, що виникають під час обчислення $f(1, 2)$, зображено на рис. 5.

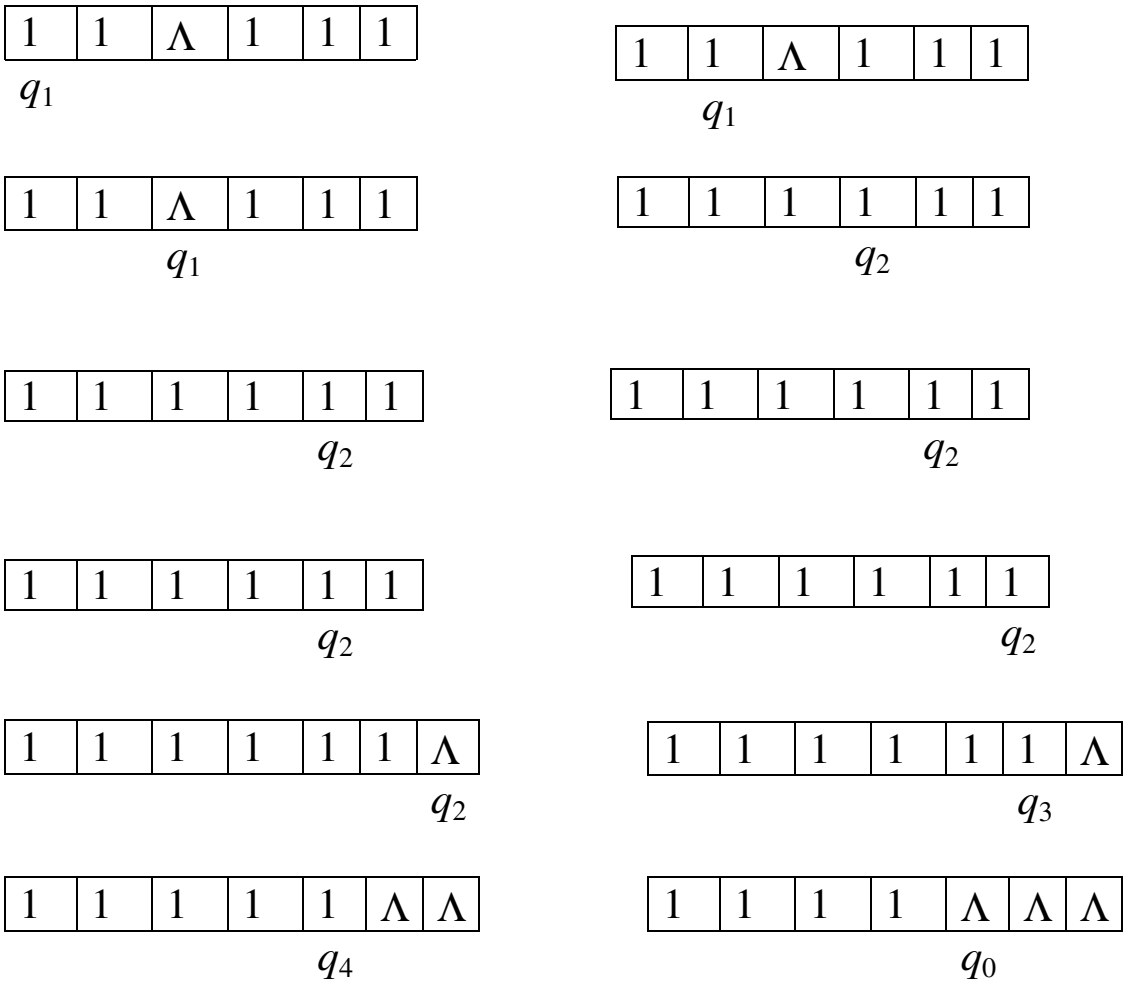


Рис. 5

Теза Тьюрінга. Приклади алгоритмічно нерозв'язних проблем

До цього часу нам удавалося для всіх процедур, які „претендують" на алгоритмічність, будувати машини Тьюрінга, що їх реалізують. Чи правильно це загалом? Ствердна відповідь на це питання міститься в *тезі Тьюрінга*, яку формулюють так: *будь-який алгоритм можна реалізувати машиною Тьюрінга*.

Довести тезу Тьюрінга неможливо, бо саме поняття алгоритму інтуїтивне (неточне). Це не теорема й не аксіома математичної теорії, а твердження, яке пов'язує теорію з тими об'єктами, для яких її побудовано. За своїм характером теза Тьюрінга нагадує гіпотези фізики про адекватність математичних моделей фізичним явищам і процесам. Підтверджує тезу Тьюрінга, по-перше, математична практика, по-друге – та обставина, що опис алгоритму в термінах будь-якої іншої відомої алгоритмічної моделі можна звести до його опису у вигляді машини Тьюрінга.

Теза Тьюрінга дає змогу, по-перше, замінити неточні (інтуїтивні) твердження про існування ефективних процедур (алгоритмів) точними твердженнями про існування машин Тьюрінга, а, по-друге, тлумачити твердження про неіснування машини Тьюрінга як твердження про неіснування алгоритму.

Розглянемо тепер три приклади алгоритмічно нерозв'язних проблем.

Проблема самозастосовності. Закодуємо всі машини Тьюрінга символами в алфавіті {*, 1} так, щоб за системою команд можна було ефективно побудувати код машини та, навпаки, за кодом машини можна було ефективно виписати її програму. Це можна зробити, наприклад, так. Нехай T – будь-яка машина Тьюрінга, зовнішній алфавіт якої $\{\Lambda, a_1, a_2, \ldots, a_m\}$, а множина внутрішніх станів – $\{q_0, q_1, \ldots, q_k\}$. Занумеруємо всі символи, які зустрічаються в командах (окрім $\rightarrow$), числами 0, 1, 2, … так, як це показано в табл. 3.

Таблиця 3

Символ	H	$Л$	$П$	Λ	a_1	…	a_m	q_0	q_1	…	q_k
Номер	0	1	2	3	4	…	m+3	m+4	m+5	…	m+4+k

Число i подаватимемо набором з (i+1) одиниць. Якщо b – один із символів H, $Л$, $П$, Λ, a_1, … , a_m, q_0, q_1, … , q_k, то як $N(b)$ позначимо набір з одиниць, кількість яких відповідає номеру символу b. Наприклад, $N(H)$=1, $N(Л)$=11, $N(П)$=111, $N(\Lambda)$=1111, $N(a_1)$=11111, $N(a_2)$=111111. Тепер команді $q_i a_i \to a_j D q_j$ поставимо у відповідність слово

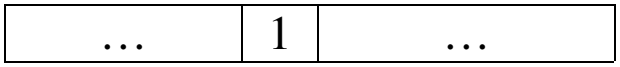
$$C(q_i a_i) = N(q_i) * N(a_i) * N(a_j) * N(D) * N(q_j),$$

яке називають *кодом* цієї *команди*. *Кодом машини* T називають слово

$$W(T) = **C(q_1\Lambda)**C(q_1 a_1)**\ldots**C(q_1 a_m)**C(q_2\Lambda)**C(q_2 a_1)**\ldots$$
$$\ldots**C(q_2 a_m)**C(q_3\Lambda)**\ldots**C(q_k\Lambda)**\ldots**C(q_k a_m)**$$

(тут у визначеному порядку записано коди всіх команд, відокремлені один від одного двома зірочками).

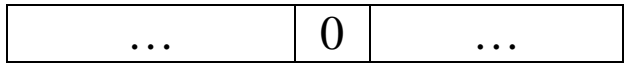
Розглядатимемо машини Тьюрінга, зовнішній алфавіт яких містить символи * та 1. Нехай T – одна з таких машин. Якщо машина T застосовна до слова $W(T)$, тобто до власного коду, то її називають *самозастосовною*, а ні, то *несамозастосовною*. Кожна машина розглянутого типу або самозастосовна, або несамозастосовна.



q_0

Рис. 6

Проблема самозастосовності полягає в тому, що потрібно навести алгоритм для визначення того, чи дана машина Тьюрінга самозастосовна, чи ні. Згідно з тезою Тьюрінга цю задачу можна сформулювати як задачу про побудову машини Тьюрінга, котра застосовна до кодів усіх машин, і заключні конфігурації в разі роботи над кодами самозастосовних і несамозастосовних машин відрізняються. Наприклад, вимагатимемо, щоб у разі роботи над кодом самозастосовної машини заключна конфігурація мала такий вигляд, як на рис. 6, а в разі роботи над кодом несамозастосовної машини – як на рис. 7. На цих рисунках поза зчитуваним символом можуть бути будь-які символи зовнішнього алфавіту.



q_0

Рис. 7

Теорема 1. Проблема самозастосовності алгоритмічно нерозв'язна, тобто не існує машини Тьюрінга, яка розв'язує в наведеному вище розумінні цю проблему.

Доведення. Припустимо протилежне. Нехай існує машина L, яка розв'язує проблему самозастосовності. За машиною L побудуємо машину T, яка має такі властивості:

1) машина T застосовна до кодів несамозастосовних машин;
2) машина T незастосовна до кодів самозастосовних машин.

Машину T будують так. Усі команди машини L оголошують командами машини T, але заключний стан q_0 машини L оголошують незаключним станом машини T. Заключним станом

машини T буде введений новий стан q_0', і додамо до системи команд машини T ще такі дві команди:

$$q_0 0 \to 0 H q_0' \text{ та } q_0 1 \to 1 H q_0 .$$

Очевидно, що машина T має зазначені властивості 1 та 2. Але машина T – одна із двох: або самозастосовна, або несамозастосовна. Нехай машина T самозастосовна, тобто вона застосовна до свого коду (коду самозастосовної машини). Проте це суперечить властивості 2. Якщо ж машина T несамозастосовна, то вона незастосовна до свого коду (коду несамозастосовної машини). Проте це суперечить властивості 1. Отже, машина T не може бути ні самозастосовною, ні несамозастосовною. Одержали суперечність. Оскільки машину T побудовано з машини L цілком конструктивно, то залишається зробити висновок, що машини L не існує. Теорему доведено.

Проблема зупинки. Серед вимог, яким має задовольняти алгоритм, було названо результативність (пункт 5 основних вимог). Найрадикальнішим її формулюванням була б вимога, щоб для будь-якого алгоритму A та даних α можна було визначити, чи приведе робота алгоритму A для початкових даних α до результату. Інакше кажучи, потрібно побудувати такий алгоритм B, що $B(A, \alpha) = 1$, якщо $A(\alpha)$ дає результат, і $B(A, \alpha)=0$, якщо $A(\alpha)$ не дає результату.

Згідно з тезою Тьюрінга цю задачу можна сформулювати так. За машиною T і словом α розпізнати, чи зупиниться машина T, розпочавши роботу над словом α (тобто розпізнати, чи застосовна машина T до слова α). Отже, потрібно побудувати таку машину P, яка була б застосована до всіх слів $W(T)\Lambda\alpha$, де T – довільна машина, α – довільне слово, Λ – порожній символ зовнішнього алфавіту. Окрім того, заключні конфігурації машини P мають бути різними залежно від того, чи застосовна машина T до слова α, чи незастосовна.

Теорема 2. Проблема зупинки алгоритмічно нерозв'язна, тобто не існує машини Тьюрінга P, яка розв'язує в наведеному вище розумінні названу проблему.

Доведення. Нехай існує машина Тьюрінга P, яка розв'язує проблему зупинки, а T – якась машина Тьюрінга. Тоді машина P має бути застосовна до всіх слів $W(T)\Lambda\alpha$, зокрема, коли $\alpha=W(T)$. Отже, машина P має бути застосовною до слова, $W(T)\Lambda W(T)$. У разі роботи над словом $W(T)\Lambda W(T)$ вона зупиниться в конфігурації, зображеній на рис. 6, якщо машина T застосовна до слова $W(T)$ (тобто якщо машина T самозастосовна), і в конфігурації, зображеній на рис. 7, якщо машина T незастосовна до слова $W(T)$ (тобто якщо T несамозастосовна).

Нагадаємо, що машина, яка розв'язує проблему самозастосовності, починає працювати зі словом $W(T)$ на стрічці. Отже, щоб із машини P отримати машину, яка розв'язує проблему самозастосовності, використаємо машину T_2 з прикладу 2 (уважатимемо, що a – це *, b – це 1). Машина T_2 за вказаної домовленості будь-яке слово β в алфавіті {*,1} перетворює в слово $\beta\Lambda\beta$. Тепер машину Тьюрінга, яка розв'язує проблему самозастосовності, одержують із машин T_2 та P: спочатку працює машина T_2, а потім – машина P. Але це суперечить теоремі 1. Отже, машини P не існує. Теорему доведено.

Тлумачачи твердження, пов'язані з алгоритмічною нерозв'язністю, потрібно звернути увагу на те, що в них ідеться про те, що немає єдиного алгоритму, який розв'язує певну проблему. При цьому зовсім не виключено, що її можна розв'язати в часткових випадках, але за допомогою різних процедура для кожного з них. Тому нерозв'язність загальної проблеми зупинки зовсім не знімає потреби доводити збіжність пропонованих алгоритмів, а лише показує, що пошук таких доведень не можна повністю автоматизувати.

Нерозв'язність проблеми зупинки можна інтерпретувати як неіснування загального алгоритму для налагодження програм, точніше, алгоритму, котрий за текстом довільної програми й даними для неї міг би визначити, чи зациклиться програма на цих даних. Якщо врахувати зроблене перед цим зауваження, то така інтерпретація не суперечить тому емпіричному факту, що більшість програм удається налагодити, тобто виявити зациклення, знайти його причину й усунути її. При цьому вирішальну роль відіграють досвід і майстерність програміста.

Доведення теорем у логіці предикатів. Головне застосування логіки предикатів у комп'ютерних інформаційних технологіях – автоматизація пошуку доведень теорем. Ця проблема актуальна для розв'язування задач штучного інтелекту, задач, які виникають у базах даних і знань, під час вирішення питань подання знань і роботи з ними. Проблема доведення теорем у логіці предикатів сильно різниться від такої проблеми в логіці висловлювань. Це пояснюється тим, що в логіці висловлювань наявна лише скінченна кількість інтерпретацій, а в логіці предикатів інтерпретацій певної формули може бути нескінченно багато.

Математична логіка як наука виникла в зв'язку з проблемою пошуку універсальної процедури доведення математичних тверджень. Цією проблемою займались Лейбніц (1646 – 1716 рр.), Пеано (кінець XIX – початок XX ст.) і Д. Гілберт зі своїми учнями. Цей пошук тривав до того часу, коли А. Чорч і А. Тьюрінг незалежно один від одного довели, що:
не існує алгоритму, який перевіряє тотожну істинність формул логіки предикатів (логіки першого порядку).

Це не суперечить факту існування алгоритмів, котрі можуть підтверджувати, що формула логіки предикатів загальнозначуща, якщо вона справді такою є. Проте, якщо формула не загальнозначуща, то ці алгоритми, узагалі говорячи, свою роботу можуть не закінчити. Беручи до уваги результат Чорча–Тьюрінга, доходимо висновку, що це найкраще, на що можна сподіватись. Прикладом такого алгоритму є метод резолюцій для логіки предикатів.

Зазначимо, що важливий для застосувань фрагмент числення предикатів розв'язний: числення одномісних предикатів (тобто числення, яке допускає у формулах тільки одномісні предикатні букви) є розв'язним.

Засновник інформатики і штучного інтелекту

Alan Mathison Turing
Алан Метісон Тьюрінг (23 червня 1912 – 7 червня 1954)