Вычисление

Можно дать как «классическое», так и более широкое, «общее» определение вычисления. Первое подчёркивает функции и остановку, второе охватывает интерактивные, распределённые и квантовые процессы. Здесь будут рассматриваться только эффективные вычисления, для которых существует алгоритм, который может быть запущен на машине Тьюринга. Исключением является только непрерывность.

Классическое определение

Вычисление (*computation*) – это процесс применения конечного набора определенных **правил** (*rules*) к входным данным, который либо завершается с получением выходных данных, либо продолжается неограниченно (дивергирует). Такое определение допускает частичные функции, где для некоторых входов результат может быть не определен. Иногда под вычислением подразумевается только такой процесс, который завершается.

Полная характеристика вычислительной системы в данный момент времени называется **конфигурацией** (*configuration*) или **состоянием** (*state*). В классических вычислениях состояния обладают следующими свойствами:

1. Дискретность (*Discreteness*) – множество возможных состояний является счетным (конечным или счетно-бесконечным).
2. Конечность описания (*Finite Describability*) – каждое состояние представимо конечной последовательностью символов из конечного алфавита.
3. Определенность (*Definiteness*) – в любой момент времени система находится ровно в одном состоянии $s \in S$. Нет возможности "частичного нахождения" в нескольких состояниях одновременно.
4. Наблюдаемость (*Observability*) без возмущения – состояние принципиально наблюдаемо без изменения системы. Измерение состояния не вносит случайность.

Алгоритмом называется определенный набор правил, обладающий следующими свойствами:

1. Конечность (*finiteness*): набор правил должен быть конечным.

2. Однозначность (*unambiguity*): правила должны однозначно определять следующий шаг для каждой ситуации. При этом правило может содержать явно описанный источник случайности (*randomized algorithms*).
3. Механичность: каждое правило описывает простую операцию, которая может быть выполнена без творчества или произвола.
4. Локальность исполнения: для выполнения следующего шага достаточно данных текущего состояния.

В теоретической информатике вычисление формализуется через модели вычислений (*models of computation*), такие как машины Тьюринга (*Turing machines*), λ-исчисление (*lambda calculus*) или рекурсивные функции (*recursive functions*), которые определяют математически точные рамки того, что может быть вычислено в смысле алгоритмической разрешимости (*algorithmic decidability*).

Программой называется конкретная запись алгоритма в выбранном языке для выбранной модели.

Общее определение

В общем случае вычисление (и алгоритм) могут иметь более широкое определение. Приведем здесь одно, наиболее общее определение:

Вычисление – это эволюция информационного состояния вычислительной системы, управляемая набором правил (алгоритмом), удовлетворяющих тем же свойствам, что и классические правила, но здесь может использоваться бесконечное множество правил, но важно, чтобы имелось их конечное описание. При этом на любом этапе система может получать входные данные и генерировать выходные данные.

Состояние вычислительной системы может быть более разнообразным, чем в классическом случае:

- Состояния допускают непрерывность (*continuity*) и могут принадлежать любому математическому пространству, например, квантовые состояния в гильбертовом пространстве.
- Состояния могут допускать частичную неопределенность (*uncertainty*) или быть вероятностными, например, при состоянии суперпозиции.
- Состояние может быть неразложимо (*non-separability*) на состояния компонентов, например, при квантовой запутанности.

- Состояние может быть принципиально ненаблюдаемым (*unobservability*) без изменения системы, в частности, это может нарушить состояние суперпозиции. Кроме того, состояния могут быть принципиально уникальными, не допуская копирование.
- В распределенных системах состояния отдельных компонентов могут быть рассинхронизированы.
- Размерность пространства состояний может меняться со временем.

Эволюция определяет конечную или бесконечную **трассу состояний** (*state trace*) или конфигураций, в которых пребывает система. Обычно предполагается при этом дискретность времени, но в общем случае, время может быть непрерывным. В распределенных системах трасса может образовывать частичный порядок (история без единой глобальной линии времени). В частном случае мы имеем линейную упорядоченность состояний трассы. Часто различают внутренние состояния и входные данные, и трассу образуют не только внутренние состояния, но и связанная с ним история наблюдений (входящих данных). Данное определение так же охватывает и квантовые вычисления.

Примеры вычислений, которые не являются классическими >

Интерактивные системы, например, работа операционной системы или веб-сервера являются вычислительными системами в общем смысле, но не в классическом, так как они принимают и генерируют данные в режиме на различных этапах.

Потоковая обработка данных так же непрерывно принимает данные.

В реактивных системах управления система непрерывно реагирует на изменения в окружающей среде, входные данные поступают асинхронно и непредсказуемо.

При распределенных вычислениях трасса состояний образует частичный порядок, а не линейную последовательность, нет единого глобального времени. Система может продолжать работу даже при отказе части узлов.

Квантовые вычисления не являются классическими в силу того, что в них допускаются состояния суперпозиции. Кроме того, допускаются правила, переводящие систему в такое неопределенное состояние.

Реализуемость на машинах Тьюринга >

Все рассмотренные свойства могут быть эмулированы на машинах Тьюринга, за исключением непрерывности – в этом случае мы можем произвести лишь приближенное вычисление с заданной точностью. При эмулировании квантовых вычислений помимо приближенности, возникает экспоненциальное замедление: n кубитов требуют $O(2^n)$ памяти и времени, нарушается истинный параллелизм (его не удастся эмулировать). Кроме того, в вероятностной машине Тьюринга (ВМТ) источник случайности может быть в общем случае источником принципиальной случайности, а не просто псевдослучайности.

Интерактивные вычисления удобнее вычислять на интерактивных машинах Тьюринга (ITM), но во всяком случае, их можно эмулировать и на обычной машине Тьюринга, если заранее зафиксировать всю последовательность взаимодействий с окружением.

Физические процессы как вычисление >

Все физические процессы могут рассматриваться как вычислительный процесс (в общем смысле):

- полная характеристика системы в момент t – состояние;
- физические законы образуют правила;
- эволюция системы производится детерминировано или стохастически по этим правилам.

Потенциальные проблемы общего определения

Могут возникнуть трудности в определении информационного состояния. Также могут возникнуть неопределённые или неоднозначные причинно-следственные связи. Частичный порядок трассы может нарушаться, образуя циклы. Правила могут в общем случае не иметь конечного описания. Да и вообще в целом могут обнаружиться всякие свойства, принципиально не подходящие даже под столь общее определение вычисления.

Вычислимые по Тьюрингу функции

Пусть $f : \mathbb{N}^k \to \mathbb{N}$ – некоторая функция, всюду определенная (*total*) на $\mathbb{N}^k$. Функция f является **вычислимой** (*computable*), если существует машина Тьюринга M, которая для всякого входа $x \in \mathbb{N}^k$ останавливается и выдает $f(x)$.

Пусть $f : \mathbb{N}^k \rightharpoonup \mathbb{N}$ – некоторая функция, частично определенная (*partial*) на $\mathbb{N}^k$. Функция f является **частично вычислимой** (*partial computable*), если существует машина Тьюринга M, которая

- останавливается и выдает $f(x)$ для всех $x \in \mathbb{N}^k$, для которых функция определена;
- не останавливается, если функция f не определена для точки $x \in \mathbb{N}^k$.

Эквивалентные определения будут получены, если использовать функции $f : \Sigma^* \to \Sigma^*$ и $f : \Sigma^* \rightharpoonup \Sigma^*$, где Σ – конечный алфавит, а Σ^* – множество конечных последовательностей (слов) из элементов Σ, включая пустое слово. В частном случае, $\Sigma = \{0, 1\}$. Это позволяет формально анализировать вычислимость, рассматривая SISO-функции (*string in string out*).

Тезис Чёрча-Тьюринга

Тезис Тьюринга формализует интуитивное представление эффективной вычислимости, заявляя, что машина Тьюринга является моделью универсального вычислительного устройства: для любой вычислимой функции существует машина Тьюринга, которая вычисляет функцию ❐. Все, что не может быть вычислено на машине Тьюринга, считается невычислимым.

Тезис Чёрча–Тьюринга дополнительно указывает на эквивалентность машин Тьюринга, λ-исчисления Чёрча, рекурсивных функций Клини-Гёделя и других моделей вычисления. Эквивалентность этих моделей доказана, в то время как сам тезис о том, что эффективно вычислимо лишь что вычислимо на машине Тьюринга, является лишь тезисом, связывающим формальные математические понятия с интуитивным представлением.

Тезис Чёрча–Тьюринга–Дойча гласит, что любая конечно реализуемая физическая система может быть смоделированная универсальным вычислительным устройством. Конечная реализуемость подразумевает конечное число ресурсов и степеней свободы.

Тьюринг-полные системы

Система с правилами манипуляции данными является **Тьюринг-полной** (*Turing-Complete* – TC) или **вычислительно универсальной** (*computationally universal*), если на нем может быть симулирована универсальная машина Тьюринга, т.е. любая машина Тьюринга. Вычислительная универсальность может быть определена через любую другую Тьюринг-полную систему.

Практически, достаточно трех вещей:

1. Неограниченная по размеру память.
2. Ветвление по условию.
3. Неограниченный цикл или рекурсия без гарантии завершения. Если циклы заранее ограничены константой или размером входа, полнота теряется, должен быть цикл (`while`), число шагов которого определяется данными.

Строго говоря, реальный компьютер не является TC в силу ограниченности памяти, однако Тьюринг полная система является моделью компьютера.

> **Example**
>
> **Теоретические модели**: λ-исчисление, частично-рекурсивные функции, машины Поста, тег-системы и т.д.
>
> **Клеточные автоматы**: Игра Жизнь Конвея, Элементарный клеточный автомат с правилом 110.
>
> **Языки программирования** общего назначения: C/C++, Java, Python, JavaScript, Rust.

Вычислимость задач

В контексте вычислимости задача задается тройкой (X, Y, R), где

- X – множество допустимых входов – **пространство экземпляров** (*instance space*),
- Y – множество допустимых выходов – **пространство решений** (*solution space*),

- $R \subseteq X \times Y$ – отношение корректности: $(x, y) \in R$, если y является корректным решением для экземпляра x. Входы и выходы должны быть эффективно представимы некоторой кодировкой в виде последовательности Σ^* (в частности, $\{0,1\}^*$).

Пусть частичные ($\rightharpoonup$) сюръекции $\delta_X : \Sigma^* \rightharpoonup X$ и $\delta_Y : \Sigma^* \rightharpoonup Y$ – эффективные представления входов и выходов. В силу сюръективности, образы декодеров δ_X и δ_Y совпадают с X и Y соответственно: $\mathrm{Ran}(\delta_X) = X$, $\mathrm{Ran}(\delta_Y) = Y$, но в силу частичности, не для всякой последовательности Σ^* обязательно существует образ в X (или Y). Кодер при этом для формализации вычислимости не обязателен.

Вычислимость задачи сводится к вычислимости некоторой функции, обычно, частичной. Это может быть задача, связанная с разрешимостью или полуразрешимостью множества. Тогда задача сводится к характеристической или полухарактеристической функции. Это может быть задача поиска $y \in Y$ по заданному $x \in X$, такое, что $R(x, y)$. Тогда задача сводится к функции-селектора от x, которая возвращает $f(x)$ такое, что $R(x, f(x))$. Кроме того, задача непосредственно может быть представлена виде функции.

Невычислимость функций

Существование невычислимых функций доказывается тем обстоятельством, что множество всевозможных машин Тьюринга фиксированной модели является счетным, в то время как бесконечное множество всюду определенных функций $f : \mathbb{N} \to \mathbb{N}$ несчетно. Следовательно, существует множество функций, невычислимых машинами Тьюринга.

Несчетность множества всюду определенных функций $f : \mathbb{N} \to \mathbb{N}$ доказывается диагональным методом:

	1	**2**	**3**	**4**	**…**
f_1	$f_1(1)$	$f_1(2)$	$f_1(3)$	$f_1(4)$	$\dots$
f_2	$f_2(1)$	$f_2(2)$	$f_2(3)$	$f_2(4)$	$\dots$
f_3	$f_3(1)$	$f_3(2)$	$f_3(3)$	$f_3(4)$	$\dots$
f_4	$f_4(1)$	$f_4(2)$	$f_4(3)$	$f_4(4)$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Несчетность множества частичных функций так же может быть доказана диагональным методом, но для аргументов j, для которых функция f_i не определена, вместо $f_i(j)$ в таблице будет символ неопределенности $\perp$.

Пусть в рамках некоторой модели машин Тьюринга определяется спецификация для решения задачи вычисления функций $f : \mathbb{N} \to \mathbb{N}$ E. Программы для машин Тьюринга могут быть записаны в виде последовательностей четверок, например: текущее состояние, считываемый символ, действие, следующее состояние. В зависимости от модели, это могут быть, например, пятерки – кортежи длины 5.

Набор правил должен быть конечным, поэтому, всякая программа может быть представлена в виде конечной последовательности символов. Это позволяет произвести Геделеву нумерацию, т.е. перенумеровать все машины Тьюринга данной модели числами из подмножества $\mathbb{N}$. Таким образом доказывается счётность множества всех машин Тьюринга данной модели. Так как каждой машине Тьюринга M_i соответствует конкретная функция $f_i : \mathbb{N} \to \mathbb{N}$, то и множество вычислимых функций является счётным ❐.

❐ Диагональная функция

Диагональная функция

Геделева нумерация машин Тьюринга позволяет продемонстрировать пример невычислимой по Тьюрингу функции путем самореферентности. Определим **диагональную функцию** (*diagonal function*):

$$d(n) = \begin{cases} 2 & \text{if } f_n(n) \text{ is defined and} = 1 \\ 1 & \text{otherwise} \end{cases}$$

Можно представить, что мы бежим по диагонали таблицы и функция d возвращает 2 если значение ячейки 1 и возвращает 1 в противном случае.

Очевидно, что d всюду определена на $\mathbb{N}$, однако она не вычислима по Тьюрингу: d не совпадает ни с одной функцией $f_1, f_2, f_3, \ldots$, которые соответствуют машинам Тьюринга $M_1, M_2, M_3, \ldots$

T Theorem

Диагональная функция d невычислима по Тьюрингу.

Доказательство

Предположим, что d – одна из вычислимых по Тьюрингу функций, скажем f_m. Тогда, для каждого натурального числа n, $d(n)$ так и $f_m(n)$ возвращают одно и то же число (либо не определены при n). Рассмотрим случай $n = m$:

$$f_m(m) = d(m) = \begin{cases} 2 & \text{if } f_m(m) \text{ is defined and} = 1 \\ 1 & \text{otherwise} \end{cases}$$

Возникает противоречие: если $f_m(m)$ не определена, то она определена и равна 1, если же она определена, но не равна 1, то она равна единице, а если она определена и равна 1, то она равна 2. Так как предположение, что функция d появляется в списке $f_1, f_2, \ldots, f_m, \ldots$, приводит к противоречию, мы можем заключить, что предположение о вычислимости d по Тьюрингу ложное.

Сведение вычислимости d к проблеме остановки >

Согласно тезису Тьюринга, так как d не вычислима по Тьюрингу, она является невычислимой в принципе. Почему? Несмотря на то, что не существует машины Тьюринга, которая могла бы вычислить d, мы по крайней мере можем вычислить несколько его первых значений. Как мы выше отмечали функции f_1, f_2, f_3, являются пустыми функциями, поэтому $d(1) = d(2) = d(3) = 1$. Может показаться, что мы действительно можем вычислить $d(n)$ для любого натурального числа n, если мы не ограничены по времени.

Конечно, мы можем теоретически рассчитать, какую последовательность четвёрок определяет машина M_n, выяснив, какое натуральное число ему соответствует и произведя разложение на простые множители. Однако в практическом плане это может оказаться невозможным для больших n, так как это потребует огромного времени вычисления. Но предположим, что мы не ограничены во времени.

Также не сложно будет следовать операциям, определенным машиной M_n, если определена начальная конфигурация, и если M_n в итоге остановится, то мы об этом узнаем. Таким образом если мы запустим машину M_n с входным значением n и она остановится, то мы можем увидеть, остановилась ли она в нестандартной конфигурации, оставив $f_n(n)$ неопределенной, или остановится в стандартной конфигурации с выходным значением $f_n(n) = 1$ или же со значением $f_n(n) \neq 1$. Из этого мы сделаем вывод $d(n) = 1$ или $d(n) = 2$.

Но возможен и еще один случай, при котором $d(n) = 1$, а именно, если M_n никогда не завершится. Если при заданной начальной конфигурации M_n никогда не завершится, то как мы можем это выяснить за конечное время? Это существенный вопрос: определить, остановится ли когда-нибудь или нет машина M_n, которая стартует считывая крайнюю левую не пустую клетку сплошного блока единиц на ленте (и пустой в остальных областях). Таким образом вычислимость диагональной функции сводится к проблеме остановки: является ли вычислимой функция $h(n)$, которая возвращает 1, если машина Тьюринга M_n завершится при входе n, и возвращает 0, если она будет работать бесконечно. Если такая функция является вычислимой, то вычислимо и d, что противоречит тезису Тьюринга: это значит, что существуют вычислимые функции, которые не входят в $f_1, f_2, f_3, \ldots$

Неразрешимость проблемы остановки

В простых случаях, в частности, машин M_1, M_2, M_3 ❐ легко определить, остановится ли машина Тьюринга или нет при заданном входе. Однако, эффективного алгоритма, которая вычисляла бы, остановится машина Тьюринга на некотором произвольном входе или нет, не существует. Таким образом, проблема остановки неразрешима.

Существование такого алгоритма противоречит тезису Чёрча-Тьюринга. Если такой алгоритм существует, то существует машина Тьюринга H, которая позволяет вычислить, остановится ли вычисление диагональной функции d на произвольном входе n или нет. Существование такой машины Тьюринга делает d вычислимой функцией, в то время как здесь было доказано, что d не входит в список $f_1, f_2, f_3, \ldots$ вычислимых по Тьюрингу функций. Если d вычислимо, то существуют вычислимые функции, но не вычислимые по Тьюрингу и тезис Чёрча-Тьюринга ошибочен.

Однако невозможность существования алгоритма, вычисляющего функцию остановки, доказывается непосредственно, независимо от тезиса Чёрча-Тьюринга.

Theorem

Пусть $h : \mathbb{N} \times \mathbb{N} \to \{1, 2\}$ – **функция остановки** (*halting function*), которая возвращает 1, если машина Тьюринга M_n остановится на входе m, т.е. $h(n, m) = 1$, и возвращает 2, если она не остановится.

Функция остановки h невычислима по Тьюрингу (проблема остановки не разрешима).

Доказательство >

Докажем невычислимость функции остановки строго, от противного, сведя к противоречию предположение о существовании машины Тьюринга, разрешающую проблему остановки для машин Тьюринга, вычисляющих функции $f : \mathbb{N} \to \mathbb{N}$.

Для доказательства ведем в рассмотрение две специальные машины Тьюринга.

Копирующая машина (*copying machine*) C. Если дана пустая лента, на которой записан один блок из n единиц, и машина стартует считывая крайнюю левую единицу, то машина в итоге остановится и на ленте будет записано два блока, разделенных одной пустой клеткой и содержащих по n единиц, а машина будет указывать на крайнюю единицу. Предположим, что состояния машины C пронумерованы от 1 до p. Тогда q_1 – стартовое состояние, а q_p – состояние остановки.

Дрожащая машина (*dithering machine*) D. Стартуя при тех же условиях, что и C, дрожащая машина в конечном счете остановится, если $n > 1$, но никогда не остановится, если $n = 1$. Такая машина описывается последовательностью из двух состояний:

$$1, 3, 4, 2, 3, 1, 3, 3$$

или в четверках:

$$q_1 0 S_0 q_3,\ q_1 1 R q_2,\ q_2 0 L q_1,\ q_2 1 L q_3$$

Стартуя на 1 в состоянии 1, машина шагает вправо и переходит в состояние 2. Если после этого машина оказывается на 1, головка смещается обратно влево и переходит в состояние остановки 3. Если же после первой операции машина оказывается на пустой клетке, то она смещается обратно влево и переходит в исходное состояние 1: машина зацикливается, двигаясь туда-сюда.

Теперь предположим, что у нас есть машина H, которая вычисляет функцию остановки $h(m, n)$. Состояния H пронумерованы от 1 до k: q'_1 – стартовое состояние, а q'_k – состояние остановки. Комбинируем машины C и H для того,

чтобы получить машину, вычисляющую функцию остановки $g(n)$ для машины M_n при входе n. Переобозначим номера состояний машины H от p до $r = p - 1 + k$, уберем штриховку, и запишем их после состояний $q_1, q_2, \ldots, q_{p-1}$, заменив состояние остановки q_p машины C начальным состоянием q'_p машины H. Получим машину CH, состояния которой пронумерованы от 1 до r (где q_r – состояние остановки). Программа машины CH представляет собой последовательность четверок программы C, вслед за которыми идут четверки программы H. Изначально машина C подразумевала остановку при переходе в состояние p, но в комбинированной машине переход в состояние p начинает операции машины H. Таким образом новые комбинированные инструкции будут сначала проходить через операции машины C, копируя входное число n, и потом, когда машина C завершит свою часть работы, на ленте будет записано уже два числа n, и запустятся операции машины H. Машина CH будет вычисляет функцию $g(n) = h(n, n)$. Тем самым, можно сказать, мы произвели диагонализацию: $g(n)$ возвращает 1 или 2 в соответствии с тем, останавливается ли машина M_n при входе n, т.е. при вычислении функции $f_n(n)$.

Теперь комбинируем машину CH с машиной D, перенумеровав три его состояния в $r, r+1, r+2$ и записав их после состояния q_{r-1} машины CH, заменив состояние остановки q_r стартовым состоянием машины D. В результате получим машину CHD, которая проходит через операции машины CH и затем через операции машины D. Состоянием остановки этой машины будет q_{r+2}.

Таким образом, если машина Тьюринга, идущая под номером n (ранее, мы обозначили ее как M_n), останавливается, когда стартует со значением n, т.е. если $h(n, n) = g(n) = 1$, то машина CHD не остановится, если будет стартовать со значением n. Если же M_n не останавливается, когда стартует со значением n, т.е. если $h(n, n) = g(n) = 2$, то машина CHD останавливается (стартуя с n).

Покажем, что такая машина как CHD не может существовать. Пусть m – номер машины CHD, т.е. имеем дело с машиной M_m (ей соответствует функция f_m). Что произойдет, если запустить эту машину со стартовым значением m (вычислить $f_m(m)$)? Машина CHD остановится в том и только в том случае, если машина M_m (т.е. она же сама) *не* остановится, стартуя со значением m. Машины C и D заведомо существуют. Противоречивость в

существовании машины CHD показывает, что не может быть такой машины как H.

Полуразрешимость проблемы остановки >

Проблема остановки однако является полуразрешимой: можно построить машину Тьюринга, которая принимает на вход n, m и

- останавливается, выдавая на выходе, скажем, 1, если машина Тьюринга M_n на входе m останавливается,
- не не останавливается, если M_n не останавливается при m.

Иначе говоря, множество $K = \{(n, m) : M_n(m) \downarrow\}$ – полуразрешимо. Но дополнение $\bar{K} = \{(n, m) : M_n(m) \uparrow\}$ не является полуразрешимым. Это естественно: если бы $\bar{K}$ было полуразрешимым, то K и $\bar{K}$ были бы разрешимыми множествами ❐.

❐ Универсальная проблема остановки

Универсальная проблема остановки

Проблема остановки касается существования машины Тьюринга, которая вычисляет задачу остановки при заданных n и m, т.е. для конкретной машины M_n при конкретном входе m. Можно озадачиться вопросом: остановится ли машина M_n на любом входе $m \in \mathbb{N}$? Такая задача приводит к **универсальной проблеме остановки** (*universal halting problem*), которая не является даже полуразрешимой (не говоря уже о разрешимости). Универсальная задача остановки M_n на любом $m \in \mathbb{N}$ тождественна вопросу: является ли функция $\varphi_n(x)$, вычисляемая машиной M_n, всюду определенной?

Машина Тьюринга, которая вычисляет, остановится ли машина M_n на любом входе $m \in \mathbb{N}$, не может существовать хотя бы потому, что проблема остановки является частью универсальной проблемы остановки. Для полуразрешимости множества всюду определенных функций, требуется проверить остановку на бесконечном наборе входов. Такую проверку нельзя провести за конечное число шагов. Другая проблема возникает для полуразрешимости множества частичных функций: здесь, выяснив, что хотя бы для одного m функция $\varphi_n(m)$ не определена, можно было бы определить ее частичность. Тогда, если

функция φ_n частичная, то рано или поздно мы наткнемся на такое m, при котором $\varphi_n(m)$ не определена. Однако, здесь мы упираемся в проблему остановки на каждом m. Множество $\bar{K} = \{(n, m) : M_n(m) \uparrow\}$ не является полуразрешимым ❐, поэтому, в общем случае, мы не можем вычислить является ли φ_n неопределенной при m.

Разрешимость

Множество $A \subseteq \mathbb{N}$ называется **разрешимым** (*decidable*) **рекурсивным**, если существует всюду определенная вычислимая характеристическая функция $\chi_A(x)$. Множество, которое не является разрешимым, называется **неразрешимым** (*undecidable*). Разрешимое множество иногда называют **вычислимым** (*computable set*).

Эквивалентные определения ›

Множество $A \subseteq \mathbb{N}$ разрешимо, если существует машина Тьюринга, которая на всех входах $x \in \mathbb{N}$ останавливается и отвечает `true/false` на вопрос $x \in A$.

Множество $A \subseteq \mathbb{N}$ разрешимо, если его характеристическая функция является общерекурсивной функцией.

Классическое понятие разрешимости применимо к любым счетным носителям: вместо $\mathbb{N}$ может быть любое счетное множество, с фиксированной эффективной нумерацией (кодировкой). В частности, можно говорить о разрешимости отношения R на $\mathbb{N}^k$.

Отношение $R \subseteq \mathbb{N}^k$ является **разрешимым**, если существует всюду определенная вычислимая характеристи00ческая функция (предикат) $\psi(x_1, \ldots, x_k)$, которая возвращает 1, если $(x_1, \ldots, x_k) \in R$, и 0 в противном случае.

Если дополнение $\bar{A} = \mathbb{N} \setminus A$ множества A разрешимо, то разрешимо и A: класс разрешимых множеств замкнут относительно дополнения.

Полуразрешимость

Множество $A \subseteq \mathbb{N}$ называется **полуразрешимым** (*semi-decidable*) или **рекурсивно перечислимым** (*recursively enumerable* – r.e.), если существует алгоритм, который на входе $x \in A$ останавливается и

выдает `true`. Для элементов $x \notin A$ алгоритм не останавливается или выдает `false`. Иногда используются понятия *computably enumerable* (c.e.) и *recognizable*.

Эквивалентно: множество $A \subseteq \mathbb{N}$ полуразрешимо, если существует алгоритм, который перечисляет элементы (возможно, с повторами), или, что то же самое: если существует частично рекурсивная (частично вычислимая) функция $f : \mathbb{N} \rightharpoonup \mathbb{N}$ такая, что $\mathrm{Im}(f) = A$ ❐. Иными словами, можно сказать, что существует алгоритм, который в процессе работы последовательно выдает все элементы A и только их.

Эквивалентно: множество $A \subseteq \mathbb{N}$ полуразрешимо, если его полухарактеристическая функция вычислима для всех $x \in A$.

Если и множество $A \subseteq \mathbb{N}$ и его дополнение $\bar{A} = \mathbb{N} \setminus A$ полуразрешимы, то A и $\bar{A}$ – разрешимые множества.

Полу разрешимое (рекурсивно перечислимое) отношение определяется аналогично. Отношение $R \subseteq \mathbb{N}^k$ является **полуразрешимым**, если существует предикат $\psi(x_1, \ldots, x_k)$, вычислимый для всех $(x_1, \ldots, x_k) \in R$.