**Проблема самоприменимости, ее неразрешимость**

*Проблема распознавания самоприменимости* алгоритмически неразрешима в классе всех алгоритмов исходного алфавита операторов. [1]

*Проблема распознавания самоприменимости* алгоритмически неразрешима. [2]

*Проблему распознавания самоприменимости* можно сформулировать так: по любому шифру определить, к какому типу относится соответствующий ему алгоритм. [3]

Тогда*проблема распознавания самоприменимости* состоит в следующем: по любому заданному шифру установить, к какому классу относится машина. [4]

Тогда*проблема распознавания самоприменимости* состоит в следующем: по любому заданному шифру установить, к какому классу относится машина, зашифрованная им, - к классу самоприменимых или несамоприменимых. [5]

При доказательстве алгоритмической неразрешимости*проблемы распознавания самоприменимости* был использован метод от противного. С помощью этого метода доказана алгоритмическая неразрешимость множества различных проблем, в том числе и проблема эквивалентности слов для ассоциативных исчислений. [6]

Но теперь ясно, что неразрешима и*проблема распознавания самоприменимости*. [7]

Таким образом, предположение об алгоритмической разрешимости*проблемы распознавания самоприменимости* приводит к логическому противоречию и поэтому неверно. Тем самым показана алгоритмическая неразрешимость этой проблемы. [8]

Таким образом, предположение об алгоритмической разрешимости*проблемы распознавания самоприменимости* приводит к логическому противоречию и является поэтому неверным. Тем самым показана алгоритмическая неразрешимость этой проблемы. [9]

Это обстоятельство приводит к выводу, что алгоритмическая неразрешимость*проблемы распознавания самоприменимости* не является следствием узости современного точного понятия алгоритма. Если бы удалось построить точное понятие алгоритма, включающее в себя некоторые ненормализуемые алгоритмы, то проблема распознавания самоприменимости алгоритмов по-прежнему оставалась бы алгоритмически неразрешимой. [10]

Это обстоятельство приводит к выводу, что алгоритмическая неразрешимость*проблемы распознавания самоприменимости* не является следствием узости современного точного понятия алгоритма. Если бы удалось построить точное понятие алгоритма, включающее в себя некоторые ненормализуемые алгоритмы, то проблема распознавания самоприменимости алгоритмов по-прежнему оставалась бы алгоритмически неразрешимой. [11]

К ним относится*проблема распознавания самоприменимости машин Тьюринга*, проблема остановки алгоритма, проблема распознавания принадлежности числа данному нерекурсивному множеству натуральных чисел. [12]

Алгоритм В заведомо существует, значит, не может быть алгоритма А. Тем самым мы доказали, что*проблема распознавания самоприменимости* алгоритмически неразрешима. Отсюда вовсе не следует, что, какой бы мы ни взяли алгоритм, невозможно распознать, самоприменим ли он. [13]

В теории алгоритмов известны некоторые задачи, о которых доказано, что для их решения не существует алгоритма. Такие задачи называются алгоритмически неразрешимыми. Обычно алгоритмическая неразрешимость новых задач доказывается методом сведения к этим задачам известных алгоритмически неразрешимых задач. Тем самым доказывается, что если бы была разрешима новая задача, то можно было бы решить и заведомо неразрешимую задачу. В самом деле, если удается свести к новой задаче неразрешимую задачу, то любой алгоритм решения новой задачи решал бы и ту задачу, что противоречило бы ее неразрешимости. Применяя метод сведения, обычно ссылаются на искусственные задачи, которые не представляют самостоятельного интереса, но для которых легко непосредственно доказать их неразрешимость. К числу таких задач относится*проблема распознавания самоприменимости*. [14]

Самоприменимость машины Тьюринга

Можно, не изменяя принципам,

 найти сотни достойных решений

Агни Йога (т.1 стр.196)

Какая странность – машина Тьюринга это всегда вычислимость, а нас к бухте «машина Тьюринга» прописали только потому, что у нас на борту есть некая невычислимость. Если ответить коротко, то невычислимость обнаруживается тогда, когда какая-то задача не имеет решения. Теперь подробнее об одном из способов  достижения невычислимости.

Рассмотрим пример, но не тот, что раньше (там задача была элементарна, а таблица машины оказалась достаточно громоздкой), а проще. Нас будет интересовать в качестве примера машина Тьюринга, инвертирующая двоичный код. Скажем, на ленте дана входная запись 011001, после работы машина должна записать на ленте код 100110, т.е. нули должны замениться единичками, а единички – нулями. Таблица такой машины – весьма проста.

|  |  |  |
| --- | --- | --- |
|  | q0 | q1 |
| 0 | 1Rq0 | 0Lq1 |
| 1 | 0Rq0 | 1Lq1 |
| l | lLq1 | lRS |

 Рис.4

Рекомендую убедиться самостоятельно, что машина выполняет предложенную ей задачу. При этом обращаю внимание читателя, что длина двоичной записи из последовательности нулей и единиц может быть произвольной, какой угодно длины. Машина легко «справляется» с подобной задачей и демонстрирует на примере этой задачи вычислимость.

Как из данной в примере машины Тьюринга «извлечь» невычислимость? Первое, что приходит в голову – подать на ленту в качестве входного слова, что-нибудь этакое, что она не сможет воспринять, например, вместо нулей и единичек в качестве входного слова на ленте записать последовательность букв, скажем – abbbaa… (не «ломать» же ради невычислимости машину!). Все! Комбинацию «aq0» в самом начале работы она уже не «понимает», нет такой комбинации. А не понимая, не может выполнить никакой команды: ни записать-стереть символ, ни сдвинуть головку вправо-влево, ни изменить состояние. Машина «парализована», задача для такой машины оказалась невычислимой. Правда, смущает одно обстоятельство. Мы машину ни коим образом не изменяли, а подали на ее входную ленту нечто для нее «непотребное». Это все равно, что залить в бак вместо бензина молоко и желать, чтобы автомобиль поехал. Нет, нужно подать на ленту нечто машине «знакомое», так сказать – взятое из ее существа.

Попробуем провести над машиной странную операцию: развернем нашу двумерную таблицу (например, слева направо и сверху вниз) в одномерную запись. Правильно – у нас получится:

01Rq0q000Lq1q110Rq0q011q1Lq1q1llLq1q0llRSq1

Напоминаю, что символ l обозначает пустую ячейку. А теперь предъявим машине для чтения на ленте эту одномерную запись. Все символы ее взяты не из воздуха, а из «существа» машины.

01Rq0q000Lq1q110Rq0q011q1Lq1q1llq1 q0LllRSq1

Ý

Получается, что запустив машину, мы заставляем ее «читать саму себя». С первым нулем машина справится, заменит его единичкой, и головка сделает шаг вправо. Там в ячейке записана единичка, она заменит ее нулем, и головка сдвинется еще на один шаг вправо. На ленте возникнет следующая ситуация:

10Rq0q000Lq1q110Rq0q011q1Lq1q1llLq1q0llRSq1

    Ý

В этой ситуации машину ожидает большая неприятность: головка читает с ленты символ R. А в левом столбце таблицы рис.4 (столбце входных символов) такого символа нет. Машина этот символ «не понимает». И поэтому, она не «знает», что делать дальше. Куда двигать головку, какой символ стирать-записывать на ленте, в какое состояние переходить: команды «на этот случай жизни» у машины нет. Мы столкнулись с  ранее рассмотренной ситуацией, когда машине Тьюринга предложили для чтения незнакомые символы. Но тут, казалось бы, случай другой: машина «столкнулась» с символом, взятым из ее  же «тела», из ее собственной записи. Если воспользоваться предыдущей аналогией «бензин-молоко», то она до некоторой степени похожа на ситуацию «бензин - тосол» (тосол – это уже нечто автомобильное). Да, символ R принадлежит «телу» машины, но чтобы она могла его «понять», необходимо его вхождение не только в команду-тройку, но и в строку входных символов ее таблицы. Чуть ниже мы попытаемся это сделать, а сейчас назовем ситуацию, когда машине предъявляется для чтения собственная запись режимом самоприменимости.

Что, тупик? В данном примере, да. Основное условие правильной работы машины Тьюринга нарушилось: алгоритм не реализуется, и машина Тьюринга… перестала быть машиной Тьюринга! Виновата в этом введенная нами самоприменимость. Мы пока еще не знаем хороша ли самоприменимость или плоха: об этом в свое время, дальше мы будем говорить достаточно подробно. Пока лишь зафиксируем наше «достижение»: и невычислимость возникла в виде особого режима работы машины Тьюринга, но и сохранилась сама машина с ее способностью моделировать лишь вычислимые процессы (в примере – инвертировать двоичный код), когда режим самоприменимости отключен.

Читатель, наверное, обратил внимание, если текст читал, а не  бегло просматривал, что когда мы развертывали двумерную таблицу машины в одномерную запись, то предложили развертку выполнить слева направо и сверху вниз. Почему, чем эти направления предпочтительней других, например, снизу вверх и справа налево, ведь тоже получится одномерная запись? Или развертывать в шахматном порядке? Вспомним, что одномерная запись таблицы – это тоже «таблица» её самоё. Однако, если двумерная таблица имеет один и тот же вид, то одномерные записи при разных направлениях развертки будут разными. И когда машина, будучи поставленной в режим самоприменимости, будет читать эти разные записи, не будет ли она «думать», что она «знакомится» с разными машинами. Какая из них – она сама?

Нам остается предположить, что дело не в направлении развертки, а в чем-то другом, ибо речь идет все-таки об одной и той же машине. Окинем более внимательным взором таблицу на рис.4. Не трудно представить, что символы левого столбца таблицы (у нас это символы – 0, 1, l), можно трактовать как входные символы ленты, когда машина работает как инвертирующая машина без всякой-там самоприменимости. А символы q0, q1, S, L, R – это символы, принадлежащие самой машине, иными словами – ее внутренние символы. Я чувствую, что наиболее дотошный читатель уже начал догадываться – куда (или откуда) дует ветер. В самом деле, когда машина начинает работать в режиме самоприменимости (кто включает этот режим, сейчас неважно), ей приходится обрабатывать две группы символов: внешние (из внешнего мира!) и внутренние (из своего внутреннего мира!). Здесь совершенно неважно – в каком направлении была осуществлена развертка; важно, что количественное соотношение между внешними и внутренними символами при любом способе развертки остается неизменным. Машина в режиме самоприменимости с легкостью «разбирается» с внешними символами (еще бы – для этого машина и создавалась, чтобы инвертировать единички и нолики!), а встретившись с внутренним символом – «затыкается».

Всякая ли машина Тьюринга несамоприменима? Далеко не всякая. Вот, например, та же инвертирующая машина, которую мы «превратили» в самоприменимую.

|  |  |  |  |
| --- | --- | --- | --- |
|  | q0 | q1 | q2 |
| 0 | 1Rq0 | 0Lq1 | 0Hq2 |
| 1 | 0Rq0 | 1Lq1 | 1Hq2 |
| l | 0Lq1 | lRq2 | lHq2 |
| H | HRq0 | 0Rq0 | HHq2 |
| q2 | q2Rq0 | 1Rq0 | q2Hq2 |
| q0 | q0Rq0 | q0Lq1 | q0Hq2 |
| q1 | q1Rq0 | q1Lq1 | q1Hq2 |
| L | LRq0 | LLq1 | LHq2 |
| R | RRq0 | RLq1 | RHq2 |

Рис. 5

Очевидно, из нашего примера с машиной на рис.4, машину нужно было научить понимать внутренние  символы L,R,H, q0, q1, q2 (здесь для однообразия символики прежнее состояние останова S мы обозначили символом q2). Естественно, при появлении новых входных символов, должны были появиться новые клетки таблицы. Их нужно было заполнить какими-то командами (тройками), чтобы машина «знала», что нужно делать, когда головка читает  новый входной символ. (Мы сочли нужным добавить в таблицу еще один, третий, столбец, соответствующий состоянию останова q2, хотя этого можно было и не делать). Так вот, по поводу слова «превратили». Как заполнять новые клетки, чтобы наша инвертирующая машина стала самоприменимой?

Здесь приходится сообщить «любителям самоприменимости» «принеприятнейшую новость»: доказано, что проблема самоприменимости машины Тьюринга алгоритмически неразрешима. Это означает, что нет алгоритма (способа), с помощью которого можно было однозначно сказать – находящаяся перед вами машина Тьюринга самоприменима или нет. Или иначе – как создать самоприменимую машину Тьюринга? Здесь даже не помогает то обстоятельство, что вы «научите» машину «понимать» ее внутренние символы. Остается единственный способ: «запустить» машину и посмотреть – самоприменима или нет. Самоприменима – значит, остановится. А если что-то долго не останавливается; вам уже надоело ждать? Вы смотрите на часы, вас ждут другие дела. Можно остановить машину принудительно, так и не дождавшись ответа о ее свойствах. Обидно, если вы не дождались всего каких-то пару минут, и пара часов ожидания потеряна напрасно. Нельзя ли сразу узнать, еще до запуска машины – остановится она или нет. Тут новая незадача – проблема останова машины Тьюринга тоже алгоритмически неразрешима: в общем случае по внешнему виду ее таблицы нельзя сразу сказать – остановится или нет.

Вот почему мы выше и написали – «превратили». Заполнили таблицу, руководствуясь какими-то интуитивными соображениями, «запустили», посмотрели. Сразу понятно, что сделать это «в ручную» даже для такой сравнительно небольшой таблицы, как на рис.5, было невозможно: легко видеть, что после развертки такой таблицы ее одномерная запись содержит 135 символов. Естественно, наша машина Тьюринга была промоделирована на обычном компьютере. И то, пока какой-то вариант машины остановился (по всем правилам) – значит, самоприменима – головка сделала свыше 5000 шагов туда-сюда вдоль записи, состоящей из этих 135-ти входных символов.

Ну, вот – какой-то вариант оказался самоприменимым (машина остановилась), значит невычислимости все-таки нет? Не будем спешить: сам вариант «появился» посредством невычислимости! Мне, скажем повезло, что «придумалось» за относительно короткое время. А если бы «не придумалось»? Нельзя исключать из рассмотрения и такой момент: для какой-то специализированной машины Тьюринга ее самоприменимый вариант «создать» нельзя по одной простой причине: такого варианта не существует в принципе. Узнать же, что его не существует, невозможно из-за алгоритмической неразрешимости самоприменимости машины Тьюринга.

Итак, мы на борту имеем: обычную инвертирующую машину Тьюринга, режим, когда она становится несамоприменимой (иными словами, перестает быть машиной Тьюринга, а всего лишь каким-то неработающим преобразователем информации), и наконец, когда она путем определенного усложнения своей таблицы вновь становится самоприменимой, т. е. опять приобретает полное право называться машиной Тьюринга.

Размышляя над «уловкой» капитана нашего корабля, я все время задаю себе вопрос. Почему он ввел какой-то непонятный режим работы машины, заставляя ее «читать саму себя»? Не проще ли было взять любую  известную алгоритмически неразрешимую проблему, и сказать: «пожалуйста, вот вам невычислимость»? Может быть, это был просто нетрадиционный ответ на нетрадиционную же и озорную идею подрезать мухе крылья? Все может быть. Но сдается мне, что тут была заложена более глубокая мысль. Машина Тьюринга была выбрана в качестве основополагающей модели, если угодно – путеводной звезды – на пути к постижению сознания. Значит, нужно «выжать» из этой модели все, что возможно и даже – невозможно. Мы, естественно, знали сетования М.Минского по поводу самоприменимости УМТ – мол, там непреодолимая и раздражающая трудность. Поэтому искать нужно там, где трудно: плохо спрятанное легко найдет любой. Сущность сознания спрятана очень хорошо. Капитан не стал затруднять себя вопросом – кто-то спрятал, или само спряталось? Нужно искать, а там жизнь подскажет – где холодно, где тепло, а где совсем горячо.

НЕРАЗРЕШИМЫЕ АЛГОРИТМИЧЕСКИЕ ПРОБЛЕМЫ  Понятие алгоритмически неразрешимых проблем.  Примеры алгоритмически неразрешимых проблем. Введение В общей теории алгоритмов актуальными являются проблемы существования или отсутствия алгоритма для бесконечной серии однотипных задач, наличие алгоритмически неразрешимых проблем и конкретные алгоритмически неразрешимые проблемы. Вопрос 1. Понятие алгоритмически неразрешимых проблем О.1.1.Алгоритмическая проблема – это проблема, в которой требуется найти единственный метод (алгоритм) для решения бесконечной серии однотипных единичных задач. Такие проблемы так же называют массовыми проблемами. Алгоритмические проблемы возникали и решались в различных областях математики на протяжении всей ее истории, однако некоторые из них долгое время не поддавались решению. Причина этого выяснилась лишь в 30-х годах ХХ века после того, как было выработано рассмотренное нами уточнение понятия алгоритма. Оказалось, что алгоритмические проблемы могут быть неразрешимыми, т.е. искомые в них алгоритмы могут не существовать. Каждое утверждение о неразрешимости той или иной алгоритмической проблемы представляет собой математически строго доказанную теорему о невозможности решения рассматриваемой алгоритмической проблемы с помощью алгоритмов данного класса. Рассмотрим ряд необходимых для дальнейшего понятий о разрешимых и перечислимых множествах. Важность этих понятий для теории алгоритмов состоит, прежде всего, в том, что благодаря им, становятся точными такие понятия, как «конструктивный способ задания множества» и «множество, заданное эффективно». Язык общей теории множеств является универсальным языком математики и всякому утверждению можно придать вид утверждения о множествах, а язык разрешимых и перечислимых множеств является универсальным языком для утверждений о существовании (или отсутствии) алгоритмов решения математических проблем. О.1.2.Множество М называется разрешимым (рекурсивным), если существует алгоритм АМ, который для любого объекта m решает проблему его вхождения во множество М. Алгоритм АМ называется разрешающим алгоритмом для множества М. Эквивалентное, но более точное, определение разрешимого множества можно дать с помощью общерекурсивной характеристической функции. О.1.3. Множество М называется разрешимым, если оно обладает следующей общерекурсивной характеристической функцией: О.1.4. Множество М называется рекурсивно (эффективно) перечислимым, если существует вычислимая функция φ такая, что φ(0), φ(1), φ(2), … есть перечисление (возможно с повторениями) элементов множества М. Это одно из возможных определений рекурсивно перечислимого множества. Такие множества впервые рассматривались Клини в 1939 году с помощью общерекурсивных функций. О.1.5. Множество М называется рекурсивно перечислимым, если оно является областью значений некоторой общерекурсивной функции, т.е. если существует общерекурсивная функция φМ(х) такая, что m∈М    ⇔     ∃х:  φМ(х) = m. Функция φМ(х) называется перечисляющей для множества М. Алгоритм, вычисляющий φМ(х), называется перечисляющим для множества М. Замечание Так как класс вычислимых функций совпадает и с классом частично рекурсивных функций, то иногда рекурсивно перечислимое множество определяют и через частично рекурсивные функции. Т.1.1.Если множества М и Н рекурсивно перечислимы, то рекурсивно перечислимы          М ∪ Н и М ∩ Н. Т.1.2. (теорема Поста) Множество М разрешимо тогда и только тогда, когда оно само и его дополнение рекурсивно перечислимы. Пример 1.  Множество М = {1, 4, 9, …, n2, …} квадратов натуральных чисел перечислимо и разрешимо.  Множество всех тождественно истинных булевых формул разрешимо. Многие математические проблемы состоят в требовании доказать или опровергнуть утверждение о том, что некоторые конкретные множества являются разрешимыми, а отрицательные решения тех или иных проблем состоят в установлении неразрешимости соответствующих им множеств. Т.1.3.Если непустое множество М разрешимо, то оно перечислимо. Обратное утверждение неверно! Этот факт является одним из фундаментальных результатов общей теории алгоритмов. На нем основаны (или могут быть основаны) все известные отрицательные решения алгоритмических проблем. Вопрос 2. Примеры алгоритмически неразрешимых проблем Первые примеры алгоритмически неразрешимых проблем были обнаружены в самой теории алгоритмов. Существование невычислимых по Тьюрингу функций Т.2.1.Существует функция ψ, не вычислимая по Тьюрингу, т.е. не вычислимая ни на одной машине Тьюринга. Принимая во внимание тезис Тьюринга, заключаем, что не существует вообще никакого алгоритма для вычисления значений функции ψ. Это означает, что массовая проблема нахождения значений функции ψ для всевозможных значений аргумента алгоритмически не разрешима. Проблемы распознавания самоприменимости и применимости Предположим, что на ленте машины Тьюринга записана ее собственная функциональная схема в алфавите машины. Если машина применима к такой конфигурации, то будем ее называть самоприменимой, в противном случае – несамоприменимой. Возникает массовая проблема распознавания самоприменимых машин Тьюринга, состоящая в следующем. По заданной функциональной схеме (программе) машины Тьюринга установить, к какому классу относится машина: к классу самоприменимых машин или к классу несамоприменимых машин. Т.2.2.Проблема распознавания самоприменимых машин Тьюринга алгоритмически неразрешима. На основании т.2.2 устанавливается алгоритмическая неразрешимость и некоторых других массовых проблем, возникающих в теории машин Тьюринга. Например, проблема распознавания применимости для машин Тьюринга, которая состоит в следующем. Заданы функциональная схема (программа) какой-нибудь машины Тьюринга и конфигурация в ней. Необходимо установить, применима ли машина к данной конфигурации или нет. Т.2.3.Проблема распознавания применимых машин Тьюринга алгоритмически неразрешима. Проблема определения общерекурсивности алгоритмов Эта проблема заключается в определении того, ко всяким ли допустимым начальным данным применим алгоритм. Т.2.4.Проблема определения общерекурсивности алгоритмов алгоритмически неразрешима. Теорема Райса Эта теорема устанавливает алгоритмическую неразрешимость вообще всякого нетривиального свойства вычислимых функций. Т.2.5. (теорема Райса) Пусть С – любой класс вычислимых функций от одной переменной, нетривиальный в том смысле, что имеются функции, как принадлежащие классу С, так и не принадлежащие этому классу. Тогда не существует алгоритма, который бы по номеру х функции fх определял бы, принадлежит эта функция классу С или нет. Иначе говоря, множество {х: fх ∈С} неразрешимо. Смысл теоремы Райса состоит в том, что по описанию алгоритма ничего нельзя узнать о свойствах функции, которую он вычисляет. В частности, оказывается неразрешимой проблема эквивалентности алгоритмов: по двум заданным алгоритмам нельзя узнать, вычисляют они одну и ту же функцию или нет. Многочисленные и разнообразные алгоритмические проблемы возникают так же при конструктивном истолковании различных разделов математики. К ним, в частности, относятся: 1. Неразрешимость проблемы равенства слов (тождества) для полугрупп (доказано А.А. Марковым и Э.Л. Постом в 1947 году независимо друг от друга).    Суть проблемы заключается в отыскании алгоритма для распознавания по любой паре слов Х и У полугруппы G, эквивалентны они в этой полугруппе или нет, т.е. тождественны определенные ими элементы полугруппы G или нет. Марков и Пост построили конкретные полугруппы, для которых уже невозможен алгоритм, решающий проблему равенства. 2. Неразрешимость классической проблемы тождества теории групп.   Формулируется сходно проблеме тождества для полугрупп с тем отличием, что рассматриваются только такие системы отношений, которые гарантируют существование в рассматриваемой полугруппе обратных элементов для всех образующих (т.е. некоторого алфавита А = {а0, а1, …, аn}). 3. Неразрешимость десятой проблемы Гильберта о диофантовых уравнениях (1901 г.)      Диофантовым уравнением называется уравнение вида Р(х1, х2, …, хn) = 0, где Р является многочленом с целочисленными коэффициентами и целыми показателями степеней. Требовалось найти алгоритм, позволяющий для любого диофантового уравнения, выяснить, имеет ли оно целочисленное решение. В общем случае эта проблема долго оставалась нерешенной, и только в 1970 году советский математик Ю.В. Матиясевич доказал ее  неразрешимость. Заключение Алгоритмическая неразрешимость означает лишь отсутствие единого способа для решения всех единичных задач данной бесконечной серии, в то время как каждая индивидуальная задача серии вполне может быть решена своим индивидуальным способом. Более того, может оказаться разрешимой (своим индивидуальным методом) не только каждая отдельная задача этого класса, но и целые подклассы задач этого класса. Поэтому, если проблема неразрешима в общем случае, нужно искать ее разрешимые частные случаи. Задача в более общей постановке имеет больше шансов оказаться неразрешимой. В тоже время понимание сути проблемы алгоритмической неразрешимости и знание основных алгоритмических неразрешимостей является одним из важных элементов современной математической и логической культуры, особенно для тех, кто имеет профессиональные дела, связанные с компьютерами, программированием и информатикой.  
  
Источник: <http://5fan.info/rnabewpolpolotrbew.html>