2. Семинар 14.02.2017

2.1. Универсальная МТ

Каждую машину Тьюринга с фиксированным входным алфавитом Σ можно описать конструктивным образом, закодировав множества Q, Π, F и таблицу переходов δ. Описание МТ M будем обозначать $\langle M \rangle \in \{0,1\}^*$. Можно построить машину Тьюринга, которая, получив на вход слово x и описание $\langle M \rangle$, эмулирует работу M над этим словом. Для этого в памяти нужно хранить текущую конфигурацию эмулируемой МТ, и осуществлять переходы согласно таблице, для чего, конечно, можно построить алгоритм. Такую машину будем называть **универсальной МТ**. Заметим, что если МТ M не останавливается на входе x, то универсальная МТ U не останавливается на входе $(x, \langle M \rangle)$, а если M останавливается и возвращает y, то так же делает и U. Можно сказать, что универсальная МТ является интерпретатором для языка программирования, который описывает программы на МТ.

2.2. Вычислимость, перечислимость и разрешимость

Пусть задана функция $f\colon X \to Y$ (возможно, частично определенная), где X, Y — некоторые множества конструктивных объектов (например, Σ^*). Область определения $f(\cdot)$ будем обозначать как $\mathrm{Dom}(f)$, а область значений — $f(X)$ или $\mathrm{Val}(f)$. Частично определенная функция $f(\cdot)$ называется (частично) **вычислимой**, если существует алгоритм, который на каждом входе $x \in \mathrm{Dom}(f)$ возвращает $f(x)$, а на входах не из $\mathrm{Dom}(f)$ алгоритм не останавливается.

Язык (или множество конструктивных объектов) называется **разрешимым** (рекурсивным), если существует алгоритм, который *распознает* этот язык и всегда останавливается. Это эквивалентно тому, что всюду определенная *характеристическая функция* языка L

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

является вычислимой.

Язык (множество) называется **перечислимым** (рекурсивно перечислимым), если существует алгоритм, который печатает (перечисляет) все слова из этого языка, и только их (в произвольном порядке, возможно, с повторениями). Заметим, что такой язык, вообще говоря, может быть бесконечным — тогда наш алгоритм не останавливается. Аналогично разрешимым языкам, оказывается, что язык L перечислим тогда и только тогда, когда его частично определенная *полухарактеристическая функция*

$$\tilde{\chi}_L(x) := \begin{cases} 1, & x \in L; \\ \text{не определена}, & x \notin L; \end{cases}$$

является вычислимой. Иначе можно сказать, что язык перечислим если и только если существует алгоритм, который его принимает, т.е. $L = L(M)$ для какой-то МТ. Действительно, если L перечислим, то можно построить алгоритм, который, получив на вход слово x, начинает перечислять все слова из L, пока не встретит x. Если $x \in L$, то рано или поздно это слово встретиться, поэтому алгоритм примет x, если же $x \notin L$, то алгоритм не останавливается. Пусть теперь L принимается некоторой МТ M. Покажем, что есть алгоритм, перечисляющий L. Для этого будем для $n = 1, 2, \ldots$ с помощью универсальной МТ имитировать первые n шагов M на первых n словах из Σ^* (все слова в алфавите, очевидно, можно перечислять, например, в лексикографическом порядке). Если M приняла какое-то слово за эти n шагов, то печатаем его. Таким образом мы перечислим все слова из языка $L = L(M)$.

Некоторые свойства перечислимых и разрешимых множеств и вычислимых функций.

- Если язык L разрешимый, то его дополнение $\overline{L} := \Sigma^* \setminus L$ тоже разрешимо. Действительно, если функция $\chi_L(x)$ вычислима, то и $\chi_{\overline{L}}(x) = 1 - \chi_L(x)$ тоже вычислима.

- Любой конечный язык разрешим. Алгоритму достаточно просто сравнить входное слово с конечным набором слов, которые он может хранить в своем коде (программе).

- Любой разрешимый язык перечислим. Чтобы вычислить полухарактеристическую функцию, можно построить алгоритм, который вычисляет характеристическую функцию языка L, и если $\chi_L(x) = 0$, то алгоритм зацикливается.

- Язык L разрешим тогда и только тогда, когда L и $\overline{L}$ перечислимы *(теорема Поста)*. В одну сторону это очевидным образом следует из предыдущего пункта. Пусть теперь L и $\overline{L}$ перечислимы. Тогда можно построить следующий алгоритм, разрешающий L: получив на вход слово x, поочередно печатаем слова из L и $\overline{L}$, пока не встретим x (это рано или поздно произойдет, так как мы перечисляем $L \cup \overline{L} = \Sigma^*$). Если x было напечатано алгоритмом, перечисляющим L, то принимаем его, иначе отвергаем.

- Множество F перечислимо $\Leftrightarrow$ F — множество определения вычислимой функции $\Leftrightarrow$ F — множество значений вычислимой функции.

***Теорема* 1.** *Множество F перечислимо тогда и только тогда, когда оно является проекцией некоторого разрешимого множества, т.е. существует разрешимое множество пар $W \in X \times Y$, такое, что $F = \pi_x(W) := \{x \big| \exists y : (x, y) \in W\}$.*

Иначе можно сказать, что для всякого перечислимого F существует всюду определенный вычислимый предикат $R(\cdot, \cdot)$ на парах (x, y) — высказывание, которое может быть истинно ($R = 1$) или ложно ($R = 0$), — и $x \in F$ если и только если можно предоставить такой y *(сертификат для x)*, что верно $R(x, y)$.

Доказательство. Пусть $R(\cdot, \cdot)$ — всюду определенный вычислимый предикат. Построим алгоритм, принимающий множество F: для заданного x будем перечислять все y и вычислять значение $R(x, y)$. Если $R(x, y) = 1$, то $x \in F$, поэтому принимаем x. Если же $R(x, y) = 0$ для любого y, то $x \notin F$, и алгоритм не останавливается. Таким образом, данный алгоритм принимает F.

Пусть теперь F — перечислимо. Тогда есть алгоритм A, который печатает все элементы F. Нам удобно считать, что A не останавливается — даже если F конечно, можно печатать его элементы циклически. Так как работу A можно имитировать на универсальной МТ, то вычислим такой предикат: $R(x, n) =$ "алгоритм A печатает x на шаге n". Как несложно видеть, $x \in F \Leftrightarrow \exists n : R(x, n) = 1$. $\square$

Заметим, что существуют неперечислимые языки: действительно, любой перечислимый язык можно задать с помощью соответствующей МТ, поэтому их счетное число, а всех возможных языков несчетное число (т.к. это все подмножества счетного множества Σ^*).

Можно построить неперечислимое множество и в более явном виде. Для удобства будем рассматривать множество натуральных чисел $\mathbb{N}$ вместо Σ^*. Во-первых, покажем, что существует **универсальное перечислимое множество**, т.е. такое перечислимое множество пар $W \subset \mathbb{N} \times \mathbb{N}$, что для любого перечислимого $V \subset \mathbb{N}$ найдется номер n, при котором $V = W_n := \{x : (n, x) \in W\}$. Чтобы его построить, можно рассмотреть все МТ, решающие проблему проверки принадлежности (это перечислимое множество) и использовать универсальную МТ: для каждого $n = 1, 2, \ldots$ запускать первые n МТ $M_1, \ldots, M_n$ на первых n числах, и делать на них n шагов. Если за эти n шагов МТ M_i приняла число j, то включаем пару (i, j) в W. Тогда W_n есть в точности множество, принимаемое машиной с номером n.

Неперечислимое множество построим с помощью диагонального процесса: $K := \{n : (n, n) \notin W\}$. Тогда K отличается от любого перечислимого множества $V = W_n$ в точке n. Следовательно, K — неперечислимое. Схожим образом можно построить перечислимое неразрешимое множество: достаточно просто взять дополнение K. Очевидно, $\overline{K} = \{n : (n, n) \in W\}$ перечислимо, но не разрешимо по теореме Поста.

2.3. Алгоритмически неразрешимые задачи

Рассмотрим **проблему остановки**: по заданной паре $(x, \langle M \rangle)$ определить, остановится ли МТ M на входе x. Оказывается, не существует алгоритма, который решал бы эту задачу. Действительно, если бы он существовал, то был бы алгоритм, решающий **проблему самоприменимости**: определить, останавливается ли МТ M на входе $\langle M \rangle$. Тогда можно построить МТ T, которая принимает на вход описания $\langle M \rangle$ и останавливается, если M зацикливается на $\langle M \rangle$, иначе T зацикливается. Тогда, рассматривая работу T на входе $\langle T \rangle$, приходим к противоречию. Следовательно, не существует МТ, которая решала бы проблему самоприменимости или проблему остановки. Также можно показать, что **проблема остановки на пустом входе** алгоритмически неразрешима.

Заметим, что язык

$$L_{stop} := \{\langle M \rangle : \text{МТ } M \text{ останавливается на пустом входе}\}$$

является перечислимым (опять же, можно эмулировать конечное число шагов МТ). Аналогично, перечислимы и языки, соответствующие проблеме самоприменимости и проблеме остановки.

Важный прием, который здесь использовался — это *сводимость* одной задачи к другой. Так, мы показали, что если бы проблема остановки была разрешима, то можно было бы построить алгоритм, который решал и бы и проблему самоприменимости. Но она неразрешима, следовательно, проблема остановки тоже неразрешима. То есть, мы *свели* задачу самоприменимости к задаче остановки.

Упражнение 1. Является ли разрешимым язык $L_{<2017}$, состоящий из описаний всех МТ, которые делают меньше 2017 шагов на каждом входном слове?

Решение. Во-первых, если МТ делает меньше 2017 шагов, то она не может сдвинуться больше, чем на 2016 ячеек. В этом случае имеют значение только первые 2017 символов входного слова. Таким образом, язык $L_{<2017}$ состоит в точности из описаний тех МТ, которые делают меньше 2017 шагов на каждом слове x, длина которого $|x| \leq 2017$. Таких слов, разумеется конечное число, поэтому с помощью универсальной МТ можно имитировать первые 2017 шагов заданной МТ M на каждом из этих слов. Если M остановилась на каждом из этих слов, сделав меньше 2017 шагов, то $\langle M \rangle \in L_{<2017}$, иначе $\langle M \rangle \notin L_{<2017}$. Таким образом, язык $L_{<2017}$ *разрешим.* $\square$

2.4. Невычислимые функции

Эту тему на семинаре я не успел рассказать, но идеологически она очень тесно связана с неразрешимыми задачами. Простейший пример невычислимой функции — это характеристическая функция $\chi_L(\cdot)$ какого-нибудь неразрешимого языка L. Другой любопытный пример невычислимой функции, не использующий непосредственно неразрешимость (вернее, неразрешимость проблемы остановки можно доказать из невычислимости данной функции) — это функция Радо $R(\cdot)$, она же busy beaver, то бишь, "занятой бобр". Ее можно определить следующим образом: рассмотрим все МТ с алфавитом $\Pi = \{1, \sqcup\}$, которые имеют n состояний. Тогда $R(n)$ равно максимальному числу единиц, которые будут записаны на ленте после окончания работы одной из этих МТ, запущенных на пустом входе. Т.е. мы устраиваем соревнование, в котором участвуют те МТ с n состояниями, которые останавливаются на пустом входе, и побеждает машина, написавшая больше всего 1. Заметим, что здесь как раз-таки и кроется причина невычислимости этой функции — мы не знаем, какие из машин в конце концов остановятся, а какие нет. Можно показать, что $R(3n+1) \geq 4n$ (см. книгу Вялого, п. 4.2.4). Если $R(\cdot)$ вычислима, то можно построить МТ с $3n + C$ состояниями (где C — некоторая константа), которая сначала печатает $4n$ единиц, а затем, используя их как вход, вычисляет $R(4n)$. Тогда $R(4n) \leq R(3n + C)$, но функция Радо, очевидно, монотонна, что приводит к противоречию при $n > C$.

Интересно, что можно привести пример частично определенной вычислимой функции, которая не имеет всюду определенного вычислимого продолжения. Пусть, например, $f(n)$ равна номеру шага, на котором останавливается МТ с номером n, запущенная на пустом входе. Если же машина не останавливается на пустом входе, то $f(n)$ неопределена. Эта функция является вычислимой, но частично определенной. Рассмотрим ее произвольное продолжение $g(\cdot)$, т.е. такую всюду определенную функцию, что $g(n) = f(n)$ для всех $n \in \mathrm{Dom}(f)$. Допустим, она вычислима. Тогда для любого n можно с помощью универсальной МТ сымитировать первые $g(n)$ шагов МТ с номером n на пустом входе. Если она не остановилась, то $g(n) \neq f(n)$, значит, $f(n)$ не определена, следовательно, эта машина вообще не останавливается. Но тогда мы получили алгоритм, решающий проблему остановки — противоречие.