Семинар 2.
Перечислимость, разрешимость, m-сводимость

Составил Р. Делла Пиетра

15.2.20

1 Общие сведения

1.1 Теорема Поста

Множество $A \subset L$ разрешимо $\iff$ A и $L \setminus A$ перечислимы.

$\boxed{\Rightarrow}$: их характеристической функции для A тривиально составляются полухарактеристические для A и $L \setminus A$:

$$\widetilde{\chi}_A(x) = \chi_A(x) \quad \widetilde{\chi}_{L\setminus A}(x) = 1 - \chi_A(x)$$

$\boxed{\Leftarrow}$: строим характеристическую функцию для A таким образом: запускаем перечисляющие алгоритмы для A и $L \setminus A$ одновременно, выполняя по одному такту на каждой. Если $x \in A$, за какое-то конечное время первый алгоритм перечислит x, и наша функция вернёт 1. Иначе $x \in L \setminus A$, второй алгоритм перечислит x тоже через конечное время, и функция вернёт 0. Таким образом, вычислимая характеристическая функция построена.

1.2 Примеры множеств

1.2.1 Разрешимые множества

- Множество простых чисел (за конечное время проверяем на делимость на все числа, меньшие данного)
- Любое конечное множество (конечное количество сравнений данного числа с числами множества)
- $\{n|$ *в десятичной записи числа* π *найдутся не менее* n *подряд семёрок*$\}$

  Если в π есть любое количество семёрок подряд, то это просто $\mathbb{N}$

  Если в π не более N_7 семёрок подряд, то это $\{1, 2, \ldots, N_7\}$

1.2.2 Перечислимые множества

- Множество слов, на которых данная МТ останавливается

  Строим таблицу (i, j) и запускаем МТ на входе i на j тактов. Если МТ останавливается на i, существует некоторое количество тактов, за которые она остановится на этом входе, поэтому как только мы такие (i, j) находим, печатаем i. Если МТ не останавливается на i, то ни она не остановится ни для какого количества тактов, и алгоритм не перечислит i

- Множество описаний МТ, которые останавливаются на пустом входе.

  Идея абсолютно аналогичная, но теперь i — это номер МТ, и вместо запуска фиксированной МТ на разных входах запускаем разные МТ на пустом входе.

2 Задачи на перечислимость и разрешимость

2.1 Разрешим ли язык описаний МТ, делающих не более 2020 шагов на любом входе до остановки?

Если МТ делает не более 2020 шагов, то она сможет просмотреть не более 2021 ячейку, то есть начальная ячейка, 2020 направо и 2020 налево — это все ячейки, которые могут быть задействованы такой МТ. Всего разных слов, умещающихся на этом отрезке ленты, $N = |A|^{4041}$ — конечное количество.
Построим характеристическую функцию для этого языка: $\chi_{L_{2020}}(M)$ запускает M на первом из N входов на не более 2020 тактов и проверяет, произошла ли остановка. Если не произошла, такая МТ не останавливается на каком-то слове за не более 2020 тактов, и можно вернуть 0. Иначе выбираем второе слово из N и повторяем. Если для всех слов останавливается, $M \in L_{2020}$, и возвращаем 1.
Мы построили характеристическую функцию, и она вычислима, т. к. требует в худшем случае $2020 \cdot N$ тактов, что всё ещё константа.

2.2 Вычислима ли функция, проверяющая, что МТ зацикливается на пустом входе?

Пусть такая функция вычислима. Назовём её $f_c(M)$.
Пусть M^* — машина Тьюринга, печатающая один служебный символ ℵ и зацикливающаяся на месте.
Построим алгоритм, разрешающий проблему останова, таким образом:
запускаем $f_c(M \circ M^*)$ (левая МТ выполняется раньше). Всего могут быть 3 исхода:

- $M \circ M^*$ не зацикливается. В этом случае дело не дошло до M^*, то есть M не останавливается на пустом входе.
- $M \circ M^*$ зацикливается и на ленте есть ℵ. Это значит, что M в какой-то момент остановилась, M^* вывела ℵ и зациклилась.
- $M \circ M^*$ зацикливается и на ленте нет ℵ. Это значит, что M зациклилась.

Таким образом, для любой M можно сказать, остановится она или нет на пустом входе, то есть мы разрешили проблему останова. Приходим к противоречию.
Небольшая тонкость возникает с поиском ℵ на ленте, зная, что $M \circ M^*$ зациклилась. Чтобы потратить на это конечное количество тактов, можно поступить одним из двух способов:

- Вспомнив строгое определение зацикленности (повтор конфигурации,
  то есть {содержательной части ленты, положения головки относительно ленты и состояния}) и зная, что $M \circ M^*$ зацикливается, запускаем эту композицию и ищем повторы конфигурации или ℵ.
  Если найден повтор конфигурации до ℵ, то ℵ уже не будет, и зациклилась M.
- $f_c(M \circ M^*) = 1 \implies$ можно отдельно проверить зацикливаемость M вторым вызовом $f_c(M)$. Тогда M^* может просто зацикливаться на месте, печать ℵ избыточна.

2.3 Разрешима ли проблема останова для МТ, работающих на $A = \{\Lambda\}$?

Такая проблема останова оказывается разрешима.
Будем искать зацикленность, как описано выше. В нашем случае лента всегда состоит из бесконечного количества Λ поэтому её можно выкинуть из конфигурации вместе с положением головки. В итоге конфигурация полностью описывается одним лишь состоянием, и если повторится состояние, МТ зациклилась. Сделаем $|Q| + 1$ шаг, и, очевидно, либо МТ завершится, либо конфигурация повторится, и МТ зациклится.

2.4 Задача о предикатах и сертификатах для перечислимого множества

Доказать, что $L \subset \Sigma^*$ перечислим $\iff$ существует вычислимая функция $R(x, y) : \Sigma^* \times \Sigma^* \to \{0, 1\}$: $x \in L \iff \exists y \in \Sigma^* : R(x, y) = 1$.

Для начала разберёмся в смысле R и y. R играет роль предиката: обычно это утверждение о том, что x обладает некоторыми свойствами, которое можно легко проверить, опираясь сертификат (подсказку) y.
Сразу пример: рассмотрим составные числа. Подсказкой к тому, что число составное, будет какой-нибудь делитель этого числа, а R будет соответственно проверять, делится ли x на y.

$\boxed{\Rightarrow}$: L перечислим, тогда построим R опираясь на алгоритм, перечисляющий L: если элемент номер y, выведенный перечисляющим алгоритмом, равен x, то $R(x, y) = 1$, иначе 0.
Легко видеть, что $\forall x \in L \quad \exists y : R(x, y) = 1$ и $\forall x \in \Sigma^* \setminus L \quad \forall y : R(x, y) = 0$.

$\boxed{\Leftarrow}$: Строим табличку (i, j), и считаем в каждой ячейке соответствующий $R(i, j)$. R — вычислимая функция, поэтому подсчёт результата в каждой ячейке занимает конечное время, и мы дойдём до любой ячейки. В этом случае i играет роль элемента (или x), а j роль подсказки (или y). Если $R(x_0, y_0) = 1$, то $x_0 \in L$, и наш перечисляющий алгоритм его выводит. По определению предиката для любого $x \in L$ найдётся подходящий y, и x выведется алгоритмом, и в обратном случае $x \in \Sigma^* \setminus L \implies \forall y \; R(x, y) = 0$, и такой x не выведется.

Эта задача имеет большое значение для нашего курса. Можно просмотреть параллель между перечислимыми и разрешимыми множествами и $\mathcal{P}$ и $\mathcal{NP}$ задачами.