1 Дополнения к теме "Рекурсивные и рекурсивно-перечислимые множества"

На прошлой лекции мы говорили о рекурсивных и рекурсивно-перечислимых языках. Приведем еще ряд утверждений, которые помогают понять смысл этих понятий.

Утверждение 1. *Язык L является рекурсивно-перечислимым $\Leftrightarrow \exists$ МТ M, печатающая все его элементы.*

Доказательство. Прежде всего определим, что значит "МТ печатает элементы (слова) языка". Итак, МТ, печатающая элементы языка, - эта такая МТ, которая никогда не останавливается, но последовательно, через запятую печатает на ленте слова этого языка.
$\Rightarrow$ Пусть у нас есть некоторый язык L, который является рекурсивно-перечислимым. По определению это означает, что существует МТ M, такая, что она распознает все слова L, то есть $\forall w \in L$ M$(w) = 1$.
В этом случае задача печати всех элементов множества решается параллельным перебором. Нами уже было выяснено ранее, что существует параллельная МТ(или универсальная МТ), то есть такая МТ, на которой мы можем запускать параллельно экземпляры другой МТ с различными входными данными. Воспользуемся этим фактом.
Пусть N - паралелльная МТ, $w_1, w_2, w_3, ...$ - всевозможные слова над алфавитом Σ. Организуем следующий процесс:

1. Запустим на N экземляр МТ M с входными данными w_1.
2. Позволим $M(w_1)$ проработать один такт, запустим экземпляр МТ M с входными данными w_2.
3. Проведем один такт работы $M(w_1)$, один такт $M(w_2)$, запустим экземпляр M с входными данными w_3.

И так далее. Если на такте t для некоторого w M(w) завершит работу, напечатаем w на выходной ленте.
По определению программы для МТ программа является обязательно конечной последовательностью команд, алфавит Σ также по определению конечен и, соответственно, слов над данным алфавитом не более, чем счетно. Следовательно, очевидно, что $\forall w \in L \; \exists t$ - такт МТ N, такой, что на такте t w будет напечатано на выходной ленте. Отсюда следует, что N и есть искомая МТ.
$\Leftarrow$ У нас есть МТ M, такая, что она печатает все слова языка L. Нам необходимо показать, что существует МТ N, такая, что N

распознает все элементы L.
Берем $\forall w \in$ L. Запускаем на универсальной МТ машину M, программируем ее так, чтобы она останавливалась и напечатала 1, когда машина M напечатает на входной ленте элемент w. Так как M печатает все элементы языка L, мы можем быть уверены в том, что в случае, если $w \in$ L, оно будет напечатано $\Rightarrow$ универсальная МТ остановится и напечатает для такого w 1 $\Rightarrow$ она и будет искомой машиной N, существование которой доказывает, что язык L является рекурсивно-перечислимым. $\square$

Утверждение 2. *Язык L является рекурсивным* $\Leftrightarrow \exists$ *МТ, которая печатает его элементы в соответствии со следующим порядком:* $w_1 < w_2 \Leftrightarrow \begin{cases} |w_1| < |w_2| \\ w_1 \prec w_2 \text{ (в лексикографическом порядке)} \end{cases}$

Доказательство. Доказательство, по сути, аналогично доказательству предыдущего утверждения.
$\Rightarrow$ L является рекурсивным $\Rightarrow \exists$ МТ M, такая, что
$\begin{cases} \forall w \in A \Rightarrow M(w) = 1 \\ \forall w \notin A \Rightarrow M(w) = 0 \end{cases}$

Упорядочим слова над языком Σ в соответствии с приведенным порядком. Заметим, что здесь нам даже не понадобиться запускать M параллельно.
Достаточно просто взять универсальную МТ N и последовательно запускать на ней машину M для каждого из слов последовательности, написав для N такую программу, чтобы $\forall w$ N печатала на выходной ленте w в том случае, если $M(w) = 1$, и не печатала в противном случае. Очевидно, что таким образом МТ N напечатает все слова языка L в нужном нам порядке.
$\Leftarrow$ Пусть L - некоторый язык, для которого существует МТ M, такая, что она печатает все слова L на ленте в нужном нам порядке.
Пусть N - МТ, которая умеет запускать МТ M, останавливается и печатает 1, когда ее входные данные появляются на ленте машины M, останавливается и печатает 0, если на ленте M входные данные напечатаны не были.

1. Пусть $w \in$ L. Тогда оно будет напечатано машиной M на ленте $\Rightarrow$ машина N остановится и напечатает 1.

2. Пусть $w \notin$ L. Машина M печатает на ленте элементы языка L в обозначенном выше порядке $\Rightarrow$ если машина M напечатала h, такое, что $w < h$ (в соответствии с обозначенным

порядком), мы можем наверняка сказать, что $w \notin \mathrm{L} \Rightarrow$ в этот момент можем остановить машину N и напечатать 0.

Исходя из пунктов 1 и 2, можем сказать, что мы нашли машину N, такую, что ее существование доказывает тот факт, что L является рекурсивным языком по определению. □

Почему вторая часть доказательства предыдущего предыдущего утверждения так важна?
Заметим следующий факт. В силу того обстоятельства, что для выполнения условия рекурсивной перечислимости языка L необходимо и достаточно только лишь, чтобы существовала МТ, печатающая слова L в произвольном порядке. Для того же, чтобы выполнялось также условие рекурсивности языка нужно, чтобы этот порядок был алгоритмически вычислим.

2 Верификация программ

Теперь обратимся к тому, какое значение имеет сам факт существования рекурсивно-перечислимых, но не рекурсивных языков.

Определение 1. *Верификация - формальная процедура, доказывающая, что программа работает (основана на анализе алгоритма).*

Приложением проблемы останова в данном случае будет тот факт, верификация невозможна (по крайней мере, для любого алгоритма). К примеру, невозможно для данной МТ доказать, что она останавливается.

Определение 2. *Обозначим P_0 - проблему останова. В этом случае P_0 = {(M, X) / M - останавливается на входных данных X }*

Определение 3. *Обозначим P_1 = {N / N останавливается на любых входных данных}*

Утверждение 3. P_1 - *алгоритмически неразрешима (то есть невозможен алгоритм, по данной МТ определяющий, останавливается ли она на всех входных данных)*

Доказательство. Для доказательства данного утверждения воспользуемся методом сведения. На прошлой лекции мы уже доказали, что проблема P_0 - проблема останова - является алгоритмически неразрешимой.

Покажем теперь, что проблема P_0 сводится к P_1.
Но для начала, естественно, нам нужно определить, что значит, что P_0 сводится к P_1, а также выяснить, что нам дает знание этого факта.

Определение 4. $P_0 \prec_R P_1 \Rightarrow \exists$ *f:* $\Sigma_1^* \to \Sigma_2^*$ - *рекурсивная функция(то есть должен быть алгоритм), такая, что (M,X)* $\in P_1 \Leftrightarrow N \in P_0$

Заметим, что в данном случае, если P_1 - алгоритмически разрешима, то и P_0 будет алгоритмически разрешима. Почему это так?
Возьмем некоторые входные данные (M, X). Нам нужно определить, принадлежит ли (M, X) P_0 или нет. Как мы можем это сделать, если P_1 - разрешима? Мы просто строим при помощи рекурсивной функии f новые входные данные N, но уже для задачи P_1. Так как P_1 - алгоритмически разрешима, при помощи алгоритма ее решения мы можем одназначно сказать, принадлежит N P_1 или нет. А так как (M, X)$\in P_0 \Leftrightarrow$ N$\in P_1$, то и для (M, X) мы сможем однозначно заключить, принадлежит (M, X) P_0 или нет.
Соответственно, в том случае, если P_0 - алгоритмически неразрешима, то P_1 - алгоритмически неразрешима и подавно.
Соответственно, для доказательства того факта, что P_1 алгоритмически неразрешима, достаточно показать, что P_0 сводится к P_1. Построим сведение.
Итак, пусть у нас есть (M, X) $\in \Sigma_1^*$. Построим машину N. Для этого неформально зададим программу для данной машины:

1. Стереть все данные с ленты
2. Написать X
3. Запустить M

Очевидно, что в данном случае N остановится $\Leftrightarrow$ M остановится на входных данных X. Получается, что мы построили сведение проблемы P_0 к проблеме $P_1 \Rightarrow P_1$ - алгоритмически неразрешима и точно так же, как проблема останова является рекурсивно-перечислимым, но не рекурсивным языком. □

Из данного утверждения автоматически следует невозможность верификации, то есть не существует такого алгоритма, который мог бы по уже имеющемуся алгоритму определить, что он остановится на любых входных данных.

Утверждение 4. *Проблема эквивалентности МТ A и B является алгоритмически неразрешимой.*

Доказательство. Введем промежуточную проблему $P_2 = \{$(A, B) | A и B останавливаются одновременно на одних и тех же данных$\}$.
Очевидно, что проблема эквивалентности сводится к P_2, так как если мы можем по двум МТ сказать, что они эквивалентны, то можем и определять, остановятся ли они одновременно на одних и тех же данных.
Также очевидно, что если мы по двум машинам A и B можем сказать, остановятся ли они одновременно на выходных данных X, то мы можем любую МТ A сравнить с МТ, которая никогда не останавливается и тем самым определить, останавливается ли A на любых входных данных, то есть проблема P_2 сводится к проблеме P_1.
Тогда Проблема эвивалентности $\prec_R P_2 \prec_R P_1 \Rightarrow$ Проблема эквивалентности является алгоритмически неразрешимой. □

Итак, в общем случае невозможно определить, что две МТ эквивалентны, что некоторая МТ останавливается на любых входных данных, а также что некоторая МТ останавливается на входных данных X. Но важно понимать, что это справедливо лишь в общем случае. Если будет ограничена семантика языка, на котором пишутся МТ, некоторые утверждения о корректности программ доказать будет можно. Если же язык, на котором написаны ваши программы, достаточно семантически богат, для них это сделать нельзя. При этом для достаточного семантического богатсва языка достаточно, чтобы в его рамках можно было прибавлять единицу, вычитать единицу и сравнивать с 0.
Таким образом, суть верификации заключается в том, чтобы изложить программу на каком-ниудь простом формализме, например, конечном автомате, а потом доказать ее корректность. Это сделать можно.

3 Диофантовы уравнения

Проблемы Гильберта - список из 23 кардинальных проблем математики, представленный Дэвидом Гильбертом на II Международном Конгрессе математиков в 1900 году. На данный момент 16 из них уже решены, и одной из решенных проблем является проблема №10, которая формулируется следующим образом:

"Указать метод, позволяющий определить, имеет ли диофантово уравнение решение в целых числах"

Определение 5. *Диофантово уравнение - это уравнение вида* $P(x_1, ..., x_n) = 0$, *где* $P(x_1, ..., x_n)$ *- полином с целыми коэффициентами.*

При этом следует заметить, что формулировал Гильберт эту проблему именно таким образом, то есть он не сомневался в том, что метод решения данной задачи, действительно, существует. Однако оказалось, что это не так.

Определение 6. *Если* $P(x_1, ..., x_n)$ *- некий полином с целыми коэффициентами,* $\{(a_1, ..., a_n)|P(x_1, ..., x_n) = 0\}$ *- множество всех его решений - называется диофантовым множеством*

Теорема 1. *(Дэвис, Патнем, Робинсон, Матиясевич) Множество является диофантовым* $\Leftrightarrow$ *множество является рекурсивно-перечислимым*

Почему этот результат так важен?
Действительно, на прошлой лекции мы доказали, что есть рекурсивно-перечислимые, но не рекурсивные множества, выяснили, что по данной МТ не можем определить, останавливается ли она на некоторых входных данных X. Но эта проблема оставалась лишь в рамках теории алгоритмов и никого, кроме ученых, работающих в этой области, особенно не интересовала. Диофантовы уравнения же имеют практическое применение, используются достаточно широко и, соответственно, уметь решать их и определять по данному уравнению, если ли у него решение, нам необходимо. Соответственно, ДПРМ-теорема показывает нам, что проблема существования рекурсивно-перечислимых, но не рекурсивных множеств "выплескивается"из своего родного домена - теории алгоритмов - и начинает затрагивать другие области математики.

4 Теоремы Геделя

Рассмотрим сначала простую формулировкую Первой теоремы Геделя. Итак, пусть A - утверждение **"A нельзя доказать"**.
В этом случае у нас есть два варианта:

1. A - доказуемо. Это означает, что в рамках нашей теории мы доказали ложное утверждение $\Rightarrow$ в базовой аксиоматике нашей теории есть противоречие $\Rightarrow$ в рамках данной теории можно доказать все, что угодно.

2. A - не доказуемо. Тогда A является истинным. Но тогда в аксиоматике той теории, которая содержит A, есть истинное и не доказуемое утверждение $\Rightarrow$ данная теория не является полной.

Соответственно, любая теория, в аксиоматике которой можно сформулировать такое утверждение, как утвеждение A (что возможно, например, в аксиоматике формальной арифметики), то такая теория будет либо противоречивой, либо неполной.

Теорема 2. *(Первая теорема Геделя, о неполноте) Система аксиом формальной арифметики либо противоречива, либо неполна.*

Доказательство. 1. Геделизация

Процесс геделизации заключается в том, чтобы каждой последовательности символов в нашей теории присвоить свой номер. Занумеруем все символы, из которых складываются утверждения в нашей теории. Тогда каждую последовательность символов мы можем преобразовать в последовательность номеров этих символов. Затем выберем некоторое основание и рассмотрим полученную последовательность цифр как число, записанное в некоторой системе счисления. Таким образом каждая последовательность символов в нашей теории получит свой номер, который мы назовем геделевым номером.

Если F - последовательность символов, то ее номер будем обозначать #F. Теперь утверждения об утверждениях (в том числе и утверждение A, сформулированное выше) будут просто утверждениями о числах. И, формулируя утверждение A, мы просто говорим о том, что у некоторого числа есть некоторое свойство.

2. Введем предикат P(x, y) = "x является доказательством y". Запишем его в аксиоматике формальной арифметики.

Здесь мы не будем подробно останавливаться на том, что такое аксиоматика формально арифметики, просто скажем, что это система аксиом, описывающих сложение, умножение и математическую индукцию. Нам понадобится лишь тот факт, что в рамках формальной арифметики можно складывать с единицей и вычитать единицу, этого будет достаточно. Также без доказательства мы примем, что существует MT, способная проверять, что x является доказательством y, то есть печатающая 1 в том случае, если x (то есть последовательность символов с номером x) является

доказательством y (то есть утверждения с номером y), и 0, если не является.
Как мы уже выяснили ранее, существование МТ автоматически означает существование соответствующей ММ.
Итак, для полученной ММ входными данными будут: x в регистре R_1, y в регистре R_2. В виде выхода ММ приходит в состояние q_y, если x является доказательством Y, в состояние q_n, если нет.
Рассмотрим единичную команду для ММ:
$q_i : R_p + +; q_j$
Зададим эту команду при помощи предикатов:
$Q_r(t)$ = "ММ находится в состоянии q_r в такт №t"
$R_l(t, x)$ = "R_l = x в такт №t"
Тогда описание команды будет выглядеть следующим образом:
$Q_i(t)\&R_p(t, x) \Rightarrow Q_j(t)\&R_p(t + 1, x + 1)$
Заметим, что все остальные команды для ММ можно выразить сходным образом точно так же, как и дополнительные предикаты, необходимые, чтобы гарантировать корректность состояния ММ, например, предикат о том, что в одном регистре не может быть два числа одновременно:
$\forall t, x, y R_i(t, x) = R_i(t, y) \Rightarrow Ex(x, y)$, где Ex(x,y) = "x=y"
Далее все предикаты связываются в одну большую формулу при помощи конъюнкции.
Осталось записать, что ММ остановится в состоянии q_y. Делается это так:
$\exists t\ q_y(t)$.
В итоге мы получаем некую гигантскую формулу предикатов, эмулирующую ММ. То есть мы выяснили, что P(x, y) - это предикат, который можно выразить при помощи аксиоматики формальной арифметики.

3. Диагонализация
Возьмем предикат q(n, #F) = "n не является доказательством F(#F)". Здесь F(x) - одноместный предикат, а #F - геделев номер F.
Введем предикат p(x)= $\forall y\ q(y, x)$
Заметим, что если #F=x, то F(#F) - не доказуемо.
Запишем отрицание ¬p = ∃¬q(y,x).
Рассмотрим p(#p).
Предположим, что p(#p) - доказуемо ⇒ ∃ y $\neg q(y, \#p)$ ⇒ $\neg p(\#p)$ - доказуемо ⇒ получили противоречие.
Предположим, что ¬ p(#p) - доказуемо. ⇒ ∃ n ¬q(n, p(#p))

$\Rightarrow$ n доказывает p(#p) $\Rightarrow$ снова получили противоречие.

Следовательно, про утверждение p(#p) мы можем сказать, что либо ни оно само, ни его отрицание не доказуемы и тогда система аксиом формальной арифметики неполна, либо что система аксиом формальной арифметки противоречива, если p(#p) либо $\neg$p(#p) - доказуемо. $\square$

Также следует заметить, что Первая теорема Геделя справедлива для любой теории, которая содержит аксиомы формальной арифметики, так как для того, чтобы построить доказательство данной теоремы достаточно, по сути, только уметь прибавлять единицу и вычитать единицу. Так как сложно представить себе какую-либо полезную математическую теорию, в которой нельзя прибавлять единицу, можно сказать, что практически все математические теории либо противоречивы, либо неполны. Примером такого утверждения, которое в рамках теоретико-множественной аксиоматики нельзя ни доказать, ни опровергнуть является континуум-гипотеза.

Теорема 3. *(Вторая теорема Геделя) Непротиворечивость формальной арифметики нельзя доказать средствами формальной арифметики.*

То есть в силу Первой и Второй теорем Геделя, по сути, мы можем говорить о том, что мы не можем быть уверены в непротиворечивости ни одной из тех теорий, которые содержат аксиомы формальной арифметики.