О теореме Успенского-Райса

А. И. Грюнталь[1]

[1]НИЦ «Курчатовский институт» – НИИСИ, Москва, Россия, grntl@niisi.ras.ru

Аннотация. В статье содержится подробное доказательство теоремы Успенского-Райса. Вводятся основные понятия, относящиеся к вычислимым функциям. Используется вычислительная модель «Машина с неограниченными регистрами». Изложение носит замкнутый и элементарный характер.

Ключевые слова: вычислимые функции, программа, машина с неограниченными регистрами

1. Введение

Предлагаемая статья содержит подробное (как автор надеется) доказательство теоремы Успенского-Райса. Эта теорема имеет важное общенаучное значение. Понимание её смысла и доказательства представляет собой существенный элемент образования программистов. Поэтому полное и простое изложение доказательства этой теоремы представляется целесообразным.

Теорема Успенского-Райса – это математическое утверждение, и поэтому все понятия, которые применяются в формулировке теоремы или используются при доказательстве, должны быть математически определены.

Однако само понятие программы точным не является. Например, стандарт языка программирования C содержит несколько тысяч страниц и не подлежит формализации.

Можно вместо математически точных понятий использовать их аналоги из программирования. Но тогда как сама теорема, так и её доказательство теряют математическую строгость и становятся излишне лаконичными. Настолько, что восстановить точно доказательство становится затруднительным.

По этой причине формулировка основных понятий, относящихся к теореме Успенского-Райса, требует определения вычислительной модели, допускающей точные формулировки и доказательства. Существует несколько примеров таких вычислительных моделей. Самые известные – это машина Тюринга [2], [8], частично-рекурсивные функции [3], машина с неограниченными регистрами [1], [4], алгорифмы Маркова [5], машина Поста [6].

Формально говоря, это означает, что существуют различные по семантическому смыслу теоремы Успенского-Райса. Однако можно показать, что все эти вычислительные модели в известном смысле эквивалентны [6]. Несмотря на различные исходные понятия и различные системы определений сами доказательства теоремы для различных моделей по существу одинаковы.

Дадим предварительные определения, относящиеся к теореме Успенского-Райса. Уточнённые определения будут приведены ниже. Через N обозначим множество целых неотрицательных чисел (натуральных чисел). Пусть f – функция, область определения которой представляет собой подмножество N (или всё N), принимающая значение во множестве N. Функция f называется вычислимой, если существует программа p с одним входным параметром x из N, которая обладает следующими свойствами:

– если x принадлежит области определения функции f, то на входе x программа p выводит значение f(x) и останавливается;

– если x не принадлежит области определения функции f, то на входе x программа p зацикливается.

Про программу p мы будем говорить, что она вычисляет функцию f. Ясно, что существует бесконечное количество программ, вычисляющих одну и ту же функцию.

Под свойством программы мы будем понимать свойство функции, которую эта программа вычисляет. Отсюда следует, что две программы, вычисляющие одну и ту же функцию, обладают одинаковыми свойствами. Примером свойства будет следующая характеристика программы – функция, которую вычисляет программу, тождественно равна нулю. Пример характеристики, которая свойством не является, – программа содержит три команды.

Теорема Успенского-Райса (в очень приблизительной формулировке) утверждает, что наличие у программы определённого свойства нельзя определить по тексту программы. Точнее, имеется в виду следующее. Не существует программы, которая, получив на вход текст другой (исследуемой) программы, может определить: обладает ли исследуемая программа этим свойством или не обладает?

Доказательство теоремы Успенского-Райса,

как и сама теорема, имеется во многих источниках. Одним из основных источников по этой проблематике является [1].

Данные выше формулировки нуждаются в уточнении. Перечислим лишь некоторые из обстоятельств, требующих уточнения.

Если мы говорим о свойствах программ как о свойствах функций, которые программы вычисляют, то произвольная программа должна либо останавливаться, выводя при этом некоторое значение, или зацикливаться. Однако существуют программы, которые останавливаются и ничего не выводят, или которые останавливаются в результате возникновения исключительной ситуации. Такие программы никакую функцию не вычисляют.

Следующее обстоятельство состоит в том, что программы надо каким-то способом идентифицировать. Если фиксирован язык программирования, то программы можно идентифицировать, например, в лексикографическом порядке. Однако такая идентификация должна быть корректной в том смысле, что каждому номеру должна соответствовать программа. Для «обычного» языка программирования это не так – произвольная последовательность символов программой не является.

Поэтому необходима как формализация введённых понятий, так и формализованное изложение самих доказательств. Для настоящего изложения выбрана вычислительная модель, называемая машиной с неограниченными регистрами. Автор часто использовал конструкции из [4].

Раздел математики, к которому относится теорема Успенского-Райса, называется «теория алгоритмов». Книга [5] представляет собой фундаментальный обзор по теории алгоритмов. Профессиональный учебник по теории алгоритмов - это [3]. Начальные сведения можно получить в книге [7], ясно и доступно написанной.

2. Определение программы

Мы будем рассматривать вычислительную модель, которая называется «машина с неограниченными регистрами». Дадим основные определения. Машина с неограниченными регистрами содержит бесконечную последовательность регистров. В каждом регистре может храниться произвольное натуральное (целое неотрицательное) число.

Программой называется конечный последовательно занумерованный набор команд. Номер команды в программе будем называть *индексом* команды. Индексы команд программы начинаются с 1.

Выполнение программы состоит в том, что команды программы последовательно выполняются в порядке их индексации, начиная с первой команды. Порядок выполнения команд может быть изменён в результате выполнения команды условного перехода.

Выполнение команды, отличной от команды условного перехода, состоит в том, что команда изменяет значение (содержимое) одного из регистров R1, R2, … . Последовательность значений регистров называется конфигурацией.

Перед началом выполнения программы лишь конечное количество значений регистров отлично от нуля. Значение всех остальных регистров равно 0. Конфигурация, заданная перед началом выполнения программы, называется начальной.

Команды бывают четырёх типов: обнуления Z(n), прибавления единицы S(n), пересылки T(m,n) и условного перехода J(m,n,q). Опишем действие этих команд.

При выполнении команды Z(n) регистру Rn присваивается значение 0. Выполнение команды S(n) состоит в том, что значение регистра Rn увеличивается на 1. В результате выполнения команды T(m,n) значение регистра Rn становится равным значению регистра Rm. Значения всех остальных регистров, кроме регистра Rn, остаются прежними. Команда условного перехода J(m,n,q) предназначена для изменения порядка выполнения команд. Если значение регистра Rn равно значению регистра Rm, то следующей после команды J(m,n,q) будет выполняться команда с индексом q. Если значения регистров Rn и Rm различны, то после команды J(m,n,q) будет выполнена команда, находящаяся в программе вслед за командой J(m,n,q). Число q называется адресом перехода.

Числа m, n и q называются аргументами команды. Команды одного типа с различными аргументами считаются различными. Например, команды Z(1) и Z(2) различны.

Программа заканчивает работу (останавливается) в случае, если вслед за текущей должна быть выполнена команда, которая в программе отсутствует.

Одно из основных свойств машины с неограниченными регистрами состоит в том, что при любой начальной конфигурации программа либо останавливается, либо зацикливается. Если программа останавливается, то конфигурация, которая получилась после остановки, называется завершающей.

Пример. Программа, состоящая из одной команды J(1,1,1), всегда зацикливается. Программа Z(1) останавливается при произвольной начальной конфигурации.

Ниже под программой мы будем понимать

программу для машины с неограниченными регистрами. Также мы будем пользоваться термином «алгоритм». Под алгоритмом мы понимаем огрублённое описание последовательности вычислений, на основании которого можно написать программу.

3. Ввод и вывод

Аналогом ввода и вывода для программ машины с неограниченными регистрами является процедура чтения начальной конфигурации и формирования завершающей конфигурации. Особенность программ машины с неограниченными регистрами состоит в том, что программа использует столько регистров, сколько ей потребуется для вычислений.

Пусть $x_1, \ldots, x_n$ – последовательность из n натуральных чисел. Конфигурацию $x_1, \ldots, x_n, 0, 0, \ldots$ будем обозначать через $\langle x_1, \ldots, x_n \rangle$.

Если на начальной конфигурации $\langle x_1, \ldots, x_n \rangle$ программа p останавливается, то через $p(x_1, \ldots, x_n)$ будем обозначать значение регистра R1 после того, как программа p остановилась. В этом случае значение регистра R1 называется результатом выполнения (или выходом) программы p на начальной конфигурации (или на входе) $\langle x_1, \ldots, x_n \rangle$.

Пример. Программа

1 J(2,3,5)
2 S(1)
3 S(3)
4 J(1,1,1)

преобразует начальную конфигурацию <x,y> в завершающую конфигурацию <x+y,y,y>.

В качестве упражнения можно написать программы умножения чисел, вычитания, деления с остатком, возведения в целую степень.

Уточним определение вычислимой функции для машины с неограниченными регистрами. Функция f от n переменных называется *вычислимой*, если существует такая программа p, что выполнены следующие два условия:

1) для последовательности $x_1, \ldots, x_n$, для которой функция f определена, выполняется равенство $f(x_1, \ldots, x_n) = p(x_1, \ldots, x_n)$;

2) для последовательности $x_1, \ldots, x_n$, для которой функция f не определена, программа p на конфигурации $\langle x_1, \ldots, x_n \rangle$ зацикливается.

4. Композиция программ

Программным модулем (или просто модулем) называется последовательно индексированный набор команд. В отличие от программ, индексация команд в модуле может начинаться с произвольного положительного числа (не обязательно с единицы). Минимальный и максимальный индексы команд, образующих модуль, обозначаются через m_{min} и m_{max}. Ещё одно требование к модулю состоит в том, что адрес перехода q команды J(m,n,q) модуля должен удовлетворять неравенству $m_{min} \leq q$. Примером программного модуля является программа.

Программный модуль выполняет команды начиная с команды с индексом m_{min} (первой командой модуля). Программный модуль заканчивает выполнение, если следующей за текущей командой должна быть выполнена команда, индекс которой больше m_{max}.

При выполнении программный модуль начинает изменять конфигурацию, которая была при выполнении первой команды модуля. Эта конфигурация называется начальной конфигурацией модуля. Конфигурация, которая получается при окончании работы модуля, называется завершающей конфигурацией модуля.

Из двух модулей можно составить новый модуль. Для этого нужно, чтобы индекс последней команды предшествующего модуля был на единицу меньше индекса команды последующего модуля (тогда команды результирующего модуля будут последовательно индексированы).

Таким же способом, последовательно располагая модули друг за другом, из нескольких модулей можно составить один модуль. Процедура получения модуля из нескольких называется композицией.

Программный модуль называется нормализованным, если адреса перехода q всех команд J(m,n,q) модуля удовлетворяют условию $q \leq m_{max} + 1$.

Если программный модуль нормализованный, то после окончания выполнения модуля следующей будет выполнена первая команда последующего модуля.

Произвольный модуль может быть нормализован. Для этого команды J(m,n,q) модуля, у которых $q > m_{max} + 1$, надо заменить на команды $J(m,n, m_{max} + 1)$. Нормализация модуля не влияет на завершающую конфигурацию модуля.

Обычно программные модули получаются из программы сдвигом на несколько индексов «вниз», то есть индекс каждой команды увеличивается на одно и тоже для всех команд число $s \geq 1$. Для того, чтобы программный модуль, сдвинутый на s индексов, выполнялся так же, как и программа до сдвига, надо модифицировать все команды условного перехода J(m,n,q), заменив адрес перехода q на число q + s. В этом случае переходы будут происходить на те же команды, что и до сдвига.

Мы будем пользоваться следующими обозначениями. Если F – программа, то через F_{+s} будем обозначать программу, которая получается

из программы F сдвигом на s индексов вниз и заменой адресов перехода q на q + s.

5. Нумерация программ

Опишем формулы, устанавливающие взаимно однозначное соответствие между программами и натуральными числами.

Сначала определим номера команд. Номер команды с будем обозначать через #c. Для команд обнуления и прибавления единицы положим

$\#Z(n) = 4n$ и $\#S(n) = 4(n - 1) + 1$.

Из этих формул видно, что номера команд обнуления – это целые положительные числа, делящиеся на 4, а номера команд прибавления единицы – это целые положительные числа, которые при делении на 4 дают в остатке 1.

Для определения номеров команд пересылки и условного перехода определим функции N_2 и N_3. Положим

$N_2(k,l) = 2^{k-1}(2l - 1)$

и

$N_3(k,l,m) = N_2(k, N_2(l,m)) = 2^{k-1}(2^l(2m - 1) - 1)$.

Номера команд пересылки и условного перехода определяются формулами

$\#T(k,l) = 4(N_2(k,l) - 1) + 2$

и

$\#J(k,l,m) = 4(N_3(k,l,m) - 1) + 3$.

Номера этих команд представляют собой все целые положительные числа, которые при делении на 4 дают в остатке 2 и 3 соответственно.

Существует программа, которая по номеру команды определяет тип команды и её аргументы. Утверждение достаточно очевидное, однако мы поясним его на примере.

Тип команды будем обозначать целыми числами, лежащими в диапазоне от 0 до 3. Эти значения обозначают соответственно команды обнуления, прибавления единицы, пересылки и условного перехода.

Для определения типа команды, надо номер команды разделить на 4 с остатком. Затем, в зависимости от остатка, то есть от типа команды, надо вычислить аргументы команды. Например, если номер команды с равен 38, то остаток от деления #c на 4 будет 2. Следовательно, команда с номером 38 – это команда пересылки. Вычтя 2 из #c и разделив на 4, получим $N_2(k,l) = 2^{k-1}(2l - 1) = 10$. Разделив два раза $N_2(k,l)$ на 2 получим в остатке 1. Следовательно, k = 2. Поэтому 2l – 1 = 5 и l = 3.

Теперь определим номер программы. Пусть p – программа и $a_1, \ldots, a_n$ – последовательность номеров команд этой программы. Последовательность $b_1, \ldots, b_n$, элементы которой определяются формулой

$$b_k = a_1 + \ldots + a_k, \quad (1)$$

где k = 1, …, n, называется *последовательностью накопленных номеров* команд программы p. Так как номера команд – положительные числа, то последовательность накопленных сумм монотонно возрастает.

Из формулы (1) следует, что различным последовательностям номеров команд соответствуют различные последовательности накопленных номеров и что для каждой монотонно возрастающей последовательности положительных чисел существует единственная последовательность номеров команд, которая её порождает.

Номером #p программы p называется число

$$\#p = -1 + 2^{b_1-1} + \ldots + 2^{b_n-1}. \quad (2)$$

Поскольку последовательность накопленных номеров монотонно возрастающая, то представление номера программ в виде (2) однозначно.

Ясно также, что произвольное целое неотрицательное число представимо в виде (2). Поэтому целые неотрицательные числа находятся во взаимно однозначном соответствии программами машины с неограниченными регистрами.

Пример. Пусть программа p состоит из одной команды S(1). Тогда

$\#S(1) = a_1 = 1$, $b_1 = a_1 = 1$, $\#p = -1 + 2^{b_1-1} = 0$.

Опишем алгоритм программы, которая по номеру программы #p и индексу команды m в программе p определяет тип команды и её аргументы.

Для определения номера команды a_m надо последовательно определять накопленные номера $b_1, b_2, \ldots$. Накопленные номера вычисляются с помощью формулы (2) путём последовательного деления с остатком на 2 числа #p + 1. Определение очередного накопленного номера b_m может быть выполнено с использованием ограниченного, не зависящего от n, количества регистров. Поэтому это действие может быть выполнено с помощью программы машины с неограниченными регистрами.

Затем определяем номер команды $a_m = b_m - b_{m-1}$. Для определения номера a_m достаточно хранить текущий и предыдущий накопленные номера. Как было отмечено выше, существует программа, которая по номеру команды вычисляет её тип и аргументы. Поэтому, определив номер команды, можно вычислить эти данные.

Отметим, что наличие взаимно однозначного соответствия между натуральными числами и программами позволяет рассматривать программы машины с неограниченными регистрами как натуральные числа.

6. Универсальная функция

Определим функцию двух переменных Ф, аргументами которой являются целые неотрицательные числа. Через φ_x обозначим программу с номером x согласно определённой выше нумерации.

Если на входе $\langle y \rangle$ программа φ_x останавливается, то положим $\Phi(x,y) = \varphi_x(y)$. Если на входе $\langle y \rangle$ программа φ_x зацикливается, то функция Ф на паре аргументов (x,y) не определена. Функция Ф называется универсальной функцией (для вычислимых функций одного переменного).

Покажем, что универсальная функция вычислима. Вычислимость означает, что существует такая программа F, для которой на входе $\langle x, y \rangle$ выполняется равенство $\Phi(x,y) = F(x,y)$ в случае, если пара (x,y) принадлежит области определения функции Ф. Если же пара (x,y) не принадлежит области определения функции Ф, то программа F на входе $\langle x, y \rangle$ зацикливается.

Опишем алгоритм программы F. Для программы F нам потребуется целочисленное кодирование конфигураций. Мы используем кодирование отличное от кодирования, которое применялось для нумерации программ. Другой метод кодирования выбран потому, что действие команд машины с неограниченными регистрами на коды конфигураций для нового метода кодирования описывается проще, чем действие на коды, полученные прежним методом кодирования.

Определим этот метод кодирования. Обозначим через q_i простое число с номером i. Примеры: $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, и т.д. Согласно основной теореме арифметики произвольное целое положительное число $n \geq 2$ однозначно представимо в виде

$$n = p_1^{r_1} \times \ldots \times p_m^{r_m}, \qquad (3)$$

где r_i– натуральные числа и $r_m > 0$.

Пусть $r_1, \ldots, r_m$ – конечная последовательность натуральных чисел и $r_m > 0$. Поставим в соответствие конфигурации, порождённой последовательностью $r_1, \ldots, r_m$, число n согласно формуле (3). Отметим, что каждое такое n больше 1. Если конфигурация состоит из одних нулей, то положим n = 1. Число n называется номером конфигурации.

Опишем алгоритм, который по номеру конфигурации n и номеру k определяет число r_k. Первый шаг алгоритма состоит в определении простого числа q_k. Поскольку нас не интересует алгоритмическая сложность вычислений, то для определения простоты числа можно воспользоваться перебором и делением с остатком. Таким методом находим последовательно простые числа, пока не найдём простое число с номером k, то есть число q_k.

На следующем шаге алгоритма найдём r_k. Это можно сделать путём деления числа n на q_k с остатком. Если в результате деления с номером s остаток становится отличным от 0, то $r_k = s - 1$.

Теперь приведём алгоритм программы F, вычисляющий функцию Ф.

Вход для программы F – это конфигурация $\langle x, y \rangle$. В случае, когда пара (x,y) принадлежит области определения функции Ф, после завершения программы F на входе $\langle x, y \rangle$ значение регистра R1 должно быть равным $\varphi_x(y)$. В противном случае программа F на входе $\langle x, y \rangle$ зацикливается.

Программа F производит покомандный разбор φ_x и выполняет (интерпретирует) эти команды. При интерпретации каждой из команд φ_x программы F преобразуют не регистры R1, R2, …, которые используются самой программой F, а номер конфигурации, который хранится в переменной conf. В начале работы программы F переменная conf инициализируется значением 2^y, что равно номеру конфигурации $\langle y \rangle$.

Приведём псевдокод программы F. Пояснения помещаются в круглых скобках. В фигурных скобках указываются действия, которые последовательно должны быть совершены при выполнении определённого условия. Знак == используется для обозначения равенства двух величин. Знак := используется для обозначения того, что величине в левой части от этого знака присваивается значение, находящееся в правой части.

Начало алгоритма

Вход: $\langle x, y \rangle$

По номеру x определяем n (n – количество команд программы φ_x)

i := 1 (i – индекс текущей команды программы φ_x)

conf := 2^y (conf – номер текущей конфигурации)

(конфигурация инициализируется начальным значением y,0,0,…)

Начало цикла

Если i > n, то переходим на первую команду программы после окончания цикла

Определяем тип и аргументы команды с индексом i программы φ_x

(тип команды записывается в переменную type)

(для команд Z,S,T и J переменная type принимает значения 0,1,2,3)

(аргументы команд записываются в переменные m, n и q)

Если type == 0, то

{Находим p_m и r_m в разложении conf на простые сомножители

conf := conf/($p_m^{r_m}$)

i := i + 1

Переходим на начало цикла}

Если type == 1, то

{Находим p_m в разложении conf на простые сомножители

conf := conf× p_m

i := i + 1

Переходим на начало цикла}

Если type == 2, то

{Находим r_m в разложении conf на простые сомножители

Находим p_n и r_n в разложении conf на простые сомножители

conf := $(conf/(p_n^{r_n}))\times(p_n^{r_m})$}

i := i + 1

Переходим на начало цикла}

Если type == 3, то

Находим r_m в разложении conf на простые сомножители

Находим r_n в разложении conf на простые сомножители

i := i + 1

Если $r_m == r_n$, то i := q

Переходим на начало цикла}

Конец цикла

(Попадаем сюда, если вышли из цикла)

Находим r_1 в разложении conf на простые сомножители.

В R1 записываем r_1

Конец алгоритма

Итак, мы описали алгоритм программы F, которая вычисляет универсальную функцию Φ. В силу того, что алгоритм использует только конечное количество регистров, которое не зависит от начальной конфигурации $\langle x, y\rangle$, алгоритм может быть реализован в виде программы на языке машины с неограниченными регистрами.

7. Диагональная конструкция

Построим пример функции, которая не является вычислимой. Пусть Φ определённая выше универсальная функция и x – натуральное число. В случае, если пара (x,x) принадлежит области определения функции Φ, положим $\zeta(x) = \Phi(x,x) + 1$. Если пара (x,x) не принадлежит области определения функции Φ, то положим $\zeta(x) = 0$.

Функция ζ не является вычислимой. Предположим, что это не так. Тогда существует такое натуральное n, что программа φ_n вычисляет эту функцию. Если на входе n программа φ_n останавливается, то должно выполняться равенство

$\varphi_n(n) = \Phi(n,n) + 1$ или $\varphi_n(n) = \varphi_n(n) + 1$,

что невозможно. Если же на входе n программа φ_n зацикливается, то равенство $\varphi_n(n) = \zeta(n)$ также невозможно, поскольку левая часть этого равенства не определена, а правая равна 0. Итак, мы доказали, что функция ζ не является вычислимой.

Теперь определим вычислимую функцию d. Для тех x, для которых пара (x,x) принадлежит области определения функции Φ, положим $d(n) = \Phi(n,n)$. Если же пара (x,x) не принадлежит области определения функции Φ, то функция d для аргумента x не определена. Так определённую функцию d будем называть диагональной.

Покажем, что функция d вычислима. Пусть F – программа, вычисляющая функцию Φ. Через F_{+1} обозначим модуль, который получается из программы F сдвигом на 1. Программа G, состоящая из команды T(1,2) и модуля $F_{+1,}$ вычисляет функцию d.

Начальная конфигурация для программы G – это $\langle x\rangle$. Команда T(1,2) превращает конфигурацию $\langle x\rangle$ в конфигурацию $\langle x, x\rangle$. Модуль F_{+1} вычисляет функцию d на паре аргументов (x,x), то есть выводит число Φ(x,x), если x принадлежит области определения функции d, и зацикливается в другом случае. Следовательно, программа F вычисляет функцию d.

Диагональная функция не является всюду определённой. Это следует, например, из того, что программа, состоящая из одной команды J(1,1,1), зацикливается.

8. Разрешимые множества

Пусть S – некоторое множество натуральных чисел N. Характеристической функцией множества S называется всюду определённая функция, которая на элементах множества S принимает значение 1, а на элементах из N, не принадлежащих S, принимает значение 0. Множество S называется разрешимым, если характеристическая функция этого множества вычислима.

Покажем, что область определения D диагональной функции d не является разрешимым множеством.

Предположим, что характеристическая функция χ множества D вычислима и что G – программа, вычисляющая функцию χ. Без ограничения общности можно считать, что программа G не изменяет и не использует в вычислениях значение регистра R2. Кроме того, программа G при завершении обнуляет все регистры, кроме регистров R1 и R2. Результат выполнения программы G содержится в R1.

Обозначим через G_{+1} модуль, который получается из программы G сдвигом на единицу. Через F обозначим программу, которая вычисляет функцию Φ.

Покажем, что из предположения вычислимости функции χ следует существование программы, вычисляющей функцию ζ. Приведём алгоритм этой программы.

Начало алгоритма

Вход: $\langle x\rangle$

T(1,2) (запоминаем x в регистре R2)

G_{+1} (в R1 записывается 0 или 1)

J(1,3,q) (значение R3 равно 0, значение q больше индекса команды S(1))

T(2,1) (конфигурация становится равной $\langle x, x\rangle$)

F_{+s} (s такое, что модуль F_{+s} находится сразу за командой T(2,1))

S(1) (значение регистра R1 становится равным Φ(x,x) + 1)

Конец алгоритма

Итак, в предположении, что множество D разрешимо, мы написали программу, вычисляющую ζ. Так как функция ζ не является вычислимой, то предположение о том, что множество D разрешимо неверно.

Лемма. Пусть f – вычислимая функция двух переменных. Тогда существует такая всюду определённая вычислимая функция одного переменного к, что $f(x,y) = \varphi_{к(x)}(y)$.

Точнее, если пара (x,y) принадлежит области определения функции f, то выполнено приведённое выше равенство. Если же пара (x,y) не принадлежит области определения функции f, то программа $\varphi_{к(x)}$ на входе $\langle y\rangle$ зацикливается.

Доказательство. Сначала дадим набросок доказательства. Пусть F – программа, которая вычисляет функцию f. Если в программе F зафиксировать значение первой переменной, то получится программа, зависящая только от одного входного параметра. Каждая такая программа имеет свой номер, который зависит от значения фиксированной переменной. Эта зависимость и представляет собой искомую функцию к.

Опишем процедуру построения функции к подробнее. Для каждого натурального a построим программу X_a, которая обладает следующим свойством: если на паре аргументов (a,y) функция f определена, то программа X_a на конфигурации $\langle y\rangle$ выводит в R1 значение функции f(a,y); если же на этой паре аргументов функция f не определена, то программа X_a зацикливается. Приведём алгоритм программы X_a.

Начало алгоритма

T(1,2) (конфигурация $\langle y\rangle$ превращается в $\langle y, y\rangle$)

Z(1) (конфигурация превращается в $\langle 0, y\rangle$)

S(1) (эта команда выполняется a раз, конфигурация становится равной $\langle a, y\rangle$)

F_{a+2} (модуль вычисляет функцию f(a,y))

Конец алгоритма

Результат выполнения модуля F_{a+2} на конфигурации $\langle a, y\rangle$ совпадает с результатом выполнения программы F на той же конфигурации. Поэтому если функция f определена на паре аргументов (a,y), то результат выполнения модуля F_{a+2}, а значит и программы X_a будет равен f(a,y). В противном случае программа F на этой конфигурации зацикливается.

Определим номер программы X_a. Номера первых a + 2 команд программы X_a составляют последовательность 10, 4, 1, …, 1. Количество чисел 1 в этой последовательности равно a. Далее следуют команды модуля F_{a+2}.

Обозначим номера команд программы F через a_1, …, a_n. Команды программы F с индексом i, отличные от команд перехода, совпадают с командами модуля F_{a+2} с индексом i + a + 2. Поэтому номера этих команд в программе F и в модуле F_{a+2} совпадают.

Найдём разность номеров для команд перехода модуля F_{a+2} и программы F. Непосредственное вычисление показывает, что

$\#J(m,n,k+q) - \#J(m,n,q) = 2^{m+n+2}q$.

В нашем случае q = a + 2. Поэтому номера команд модуля F_{a+2}, начиная с команды с номером a + 3, образуют последовательность

$d_1 = a_1 + c_1 q$, …, $d_n = a_n + c_n q$.

Для команд Z(n), S(n) и T(m,n) коэффициенты c_i равны 0, а для команд J(m,n,k) коэффициенты c_i равны $2^{m+n+2}q$.

В соответствии с этими формулами можно написать программу, которая последовательно вычисляет номера команд программы X_a. Параллельно с вычислением номеров программы X_a можно, согласно формулам (1), последовательно вычислять накопленные номера b_1, …, b_{n+a+2} команд программы X_a, и частичные значения суммы для определения номера программы по формуле (2). Полученное в результате этих вычислений значение и будет номером к(a) программы X_a.

Итак, мы показали, что значение к(a) вычислимо зависит от a. Согласно определению программа X_a совпадает с программой $\varphi_{к(a)}$. Следовательно, $\varphi_{к(a)}$ вычисляет функцию f(a,y). *Лемма доказана*.

9. Доказательство теоремы Успенского-Райса

Вначале приведём точную формулировку теоремы Успенского-Райса. Мы рассматриваем программы, входом которых является натуральное число. Программы p_1 и p_2 назовём эквивалентными, если они вычисляют одну и ту же функцию. Точнее, для любого общего входа $\langle x\rangle$ программы p_1 и p_2 останавливаются (каждая из них), либо обе программы зацикливаются. Если программы останавливаются, то выполняется равенство $p_1(x) = p_2(x)$. Все программы разбиваются на классы эквивалентности согласно введённому определению эквивалентности.

Свойством назовём множество классов эквивалентности программ. Будем говорить, что программа p обладает свойством A, если класс эквивалентности программы p принадлежит A. Множество номеров программ, которые обладают свойством A, будем обозначать через S(A).

Свойство A называется нетривиальным, если оно не совпадает с множеством всех классов эквивалентности и не пусто.

Пример свойства: класс эквивалентности состоит из программ, которые зацикливаются на произвольном входе.

Свойство A называется разрешимым, если множество номеров программ S(A), которые этим свойством обладают, разрешимо.

Теорема (*Успенского-Райса*). Произвольное нетривиальное свойство программ неразрешимо.

Доказательство. Пусть A – нетривиальное свойство. Через B обозначим дополнительное свойство, состоящее из классов эквивалентности, которые не принадлежат A.

Будем предполагать, что программа, которая зацикливается на любом входе, обладает свойством A. Пусть Q – программа, обладающая свойством B. Программа Q существует, поскольку свойство A нетривиально. Через q обозначим функцию, которую вычисляет программа Q. В случае, когда Q останавливается на конфигурации $\langle y \rangle$, должно выполняться равенство q(y) = Q(y). Если программа Q зацикливается на конфигурации $\langle y \rangle$, то функция q при значении аргумента y не определена.

Обозначим через D область определения диагональной функции d. Напомним, что D не совпадает с множеством N и что D не является разрешимым множеством. Программу, вычисляющую функцию d, обозначим через P.

Определим функцию f двух аргументов x и y. Если x принадлежит D, и y принадлежит области определения функции q, то положим f(x,y) = q(y). В остальных случаях функция f на паре аргументов (x,y) не определена.

Покажем, что функция f вычислима. Программа F, которая вычисляет функцию f, представляет собой композицию программы P и модуля $\overline{Q}$. Модуль $\overline{Q}$ состоит из команды T(2,1) и модуля Q_{+s}. Сдвиг s выбран так, чтобы первая команда модуля Q_{+s} имела индекс на единицу больше, чем команда T(2,1). Без ограничения общности можно предполагать, что программа P не изменяет и не использует значение регистра R2.

Программа F вычисляет f. Действительно, пусть $\langle x, y \rangle$ – начальная конфигурация. Если x принадлежит D, то программа P завершает выполнение и начинает выполняться модуль $\overline{Q}$. Этот модуль сначала переписывает в R1 значение y, а затем запускается модуль Q_{+s}, который вычисляет функцию q. Если x не принадлежит D, то программа P на этом входе зацикливается.

Поскольку f – вычислимая функция двух аргументов, то согласно лемме существует такая вычислимая функция к, что для каждого y, для которого функция q определена, выполняется равенство $q(y) = \varphi_{к(x)}(y)$. Если q не определена на y, то программа $\varphi_{к(x)}$ на входе $\langle y \rangle$, зацикливается.

Это означает, что при x из D программа с номером к(x) эквивалентна программе Q. Поэтому к(x) принадлежит S(B). Если же x не принадлежит D, то программа $\varphi_{к(x)}$ зацикливается на произвольном входе. Следовательно, эта программа обладает свойством A, и к(x) принадлежит S(A).

Предположим, что множество S(A) разрешимо. Тогда существует вычислимая функция χ, принимающая на S(A) значение 1, а на S(B) значение 0.

Функция χ∘к будет всюду определённой. Также эта функция будет вычислимой. Действительно, пусть программа K вычисляет функцию к, а программа X вычисляет функцию χ. (Здесь через K мы обозначили прописную букву каппа, а через X - прописную букву хи).

Обозначим через X_{+s} модуль, который получается из программы X сдвигом, чтобы команды модуля X_{+s} начинались сразу за командами программы K. Тогда композиция программы K и модуля X_{+s} будет вычислять функцию χ∘к. Здесь мы делаем обычное предположение о нормализованности программы K.

По построению функция χ∘к будет характеристической вычислимой функцией множества D, что невозможно, поскольку множество D неразрешимо. Следовательно, предположение о том, что множество S(A) разрешимо, неверно. *Теорема доказана*.

10. Заключение

Автор благодарит С.Ф. Сопрунова за тщательное чтение начального варианта статьи и ценные замечания, существенно улучшившие изложение, Ю.В Кузнецова и М.В. Михайлюка за внимание к этой публикации и благожелательные и конструктивные комментарии.

On Uspensky-Rice Theorem

Andrey Gryuntal

Abstract. This paper contains detailed proof of the Uspensky-Rice theorem. Basic definitions related to computable functions are introduced. The computing model of an unlimited register machine is used. The presentation is of thorough and elementary nature.

Keywords: computable functions, program, unlimited register machine

Литература

1. Н.К. Верещагин, А. Шень, Вычислимые функции, М., МЦНМО, 2017.
2. С. Пилипенко, Дискретная математика 2. Конспект, 2020-2021, https://hse-tex.me/course-2/discrete-math-02-lecture-notes.pdf.
3. В.А. Успенский, Лекции о вычислимых функциях, М., Государственное издательство физико-математической литературы, 1960.
4. Ю.Ю. Кочетков, Вычислимые функции, http://kirill-andreyev.narod2.ru.
5. В.А. Успенский, А.Л. Семёнов, Теория алгоритмов: основные открытия и приложения, М., «Наука», Главная редакция физико-математической литературы, 1987.
6. А.Е. Ромащенко, https://users.mccme.ru/anromash/courses/lecture-notes-logic-computability-2013.pdf
7. В.А. Успенский, Машина Поста, Популярные лекции по математике, выпуск 54, издание второе, М., «Наука», главная редакция физико-математической литературы, 1988.
8. Конспект лекций О.Б. Лупанова по курсу «Введение в математическую логику», М., МГУ им. М.В. Ломоносова, механико-математический факультет, 2007.