Лекция 15. Понятие алгоритма. Требования к алгоритмическим процедурам. Рекурсивные функции. Понятие машины Тьюринга. Теорема Тьюринга. Принципы построения Машины Тьюринга.

Алгоритмы и вычислимость

Под **алгоритмом** понимается точное предписание, которое задает вычислительный в широком смысле процесс, называемый в этом случае алгоритмическим. Процесс начинается с произвольного исходного данного из некоторой совокупности допустимых для данного алгоритма исходных данных и направлен на получение результата, полностью определяемого этим исходным данным.

Школьные методы умножения “столбиком” и деления “углом” многозначных десятичных чисел, метод исключения неизвестных при решении системы линейных алгебраических уравнений, правило дифференцирования сложной функции - примеры вычислительных алгоритмов.

Само слово “алгоритм” происходит от algoritmi - имени (в латинском написании) арабского математика IX в. аль-Хорезми. В средневековой Европе алгоритмом называлась десятичная позиционная система счисления и искусство счета в ней, поскольку именно благодаря латинскому переводу (XII в.) трактата аль-Хорезми Европа познакомилась с позиционной системой счисления.

Вообще говоря, не предполагается, что результат будет обязательно получен: процесс применения алгоритма к конкретному возможному исходному данному (т.е. алгоритмический процесс, развертывающийся начиная с этого данного) может также оборваться безрезультатно (в этом случае говорят о **безрезультатной остановке**) или не закончиться вовсе. В случае, если процесс заканчивается (соответственно не заканчивается) получением результата, говорят, что алгоритм **применим** (соответственно **неприменим**) к рассматриваемому исходному данному.

2. Основные требования к алгоритмам.

А. Как сказано выше, алгоритм обрабатывает данные. Требуется уточнить понятие о данных, которые должны быть четко определены.

В теории алгоритмов фиксируют конечные наборы исходных объектов в виде конечного алфавита исходных символов (цифр, букв и т.д.) и конечного набора средств построения других объектов из элементарных; типичным средством являются индуктивные определения. В процессе работы алгоритма образуются промежуточные данные, которые также подвергаются алгоритмической обработке.

Б. Алгоритм состоит из отдельных *элементарных шагов*, или *действий*,

причем множество различных шагов, из которых составлен алгоритм, конечно. Последовательность шагов алгоритма детерминирована, т.е. после каждого шага указывается, какой шаг делать дальше, либо дается команда *остановки*, после которой работа алгоритма считается законченной.

В. От алгоритма требуется *результативность*, т.е. остановка после конечного числа шагов (зависящего от исходных данных) с указанием того, что считать результатом. В частности, чтобы показать, что некоторый алгоритм

решает определенную задачу, необходимо убедиться, что алгоритм останавливается после конечного числа шагов (как говорят, *сходится*) для всех интересующих нас исходных данных этой задачи.

Г. Рассмотрение алгоритма как последовательности элементарных шагов сопряжено с ограниченностью изменений при каждом шаге, иначе говоря, *локальностью* преобразований. Это значит, что преобразование исходного или промежуточного объекта складывается из изменений в отдельных его частях: для многозначных чисел - в соседних разрядах, для слов в алфавите - в некотором подслове ограниченной длины, для графа - в некоторой окрестности отдельной вершины (подграфе ограниченных размеров). В этих случаях можно ввести правила локальных преобразований небольшого (по сравнению с возможными размерами объекта) количества типов.

Рекурсивные функции

1. ***Понятие примитивно рекурсивной функции*.**

Будем рассматривать функции $y = f(x_1,..., x_n)$, $x_1,..., x_n$, $y \in N^0$, где $N^0 = \{0, 1,..., n,...\}$ - множество натуральных (целых неотрицательных) чисел; таким образом, и аргументы, и функция принимают натуральные значения.

Следующие функции называются **исходными**:

1) $Z(x) = 0$ - нуль-функция (“zero”), т.е. константа 0;

2) $N(x) =$ “$x + 1$” - функция следования (“next”), обозначение: x’;

3) $I_k(x1,..., xn) = x_k$ - селекторные функции, где $k = 1, 2,..., n$.

Функция N(x) выражает число, следующее в натуральном ряду за x. Она представляет собой функцию одной переменной, поэтому запись ее с помощью знака “+” служит только для пояснения (что такое сумма - еще не определено, см. ниже). Каждая из функций (x) равна одному из своих аргументов: индекс k выделяет k-ю переменную.

Из исходных функций мы будем образовывать многие другие функции с помощью операций, которые можно называть функциональными операторами.

Говорят, что функция h получается применением **оператора суперпозиции S** к функциям $f, g_1,..., g_n$ (обозначение: $h = S(f, g_1,..., g_n)$), если

$$h(x_1,..., x_m) = f(g_1(x_1,..., x_m),..., g_n(x_1,..., x_m)).$$

С понятием суперпозиции (или подстановки) мы встречались ранее неоднократно.

Примеры. Целочисленные константы вводятся суперпозициями исходных функций Z(x) и N(x): $N(Z(x)) = N(0) = 1$; $N(N(Z(x))) = N(1) = 2$ и т.д. $Z(f(x)) = 0$ для любой функции f(x). $I_1(3, 2)$ следует рассматривать как сокращенное обозначение выражения $I_1(N(N(N(Z(x)))), N(N(Z(x))))$, равного 3; $I_2(3, 2) = 2$. Из функции f(x, y) можно получить функцию g(x, y) = f(y, x) (перестановка переменных) следующей суперпозицией: $g(x, y) = f(I_2(f(x, y)), \quad I_1(f(x, y)))$.

Говорят, что функция f от (n+1) переменных получается применением **оператора примитивной рекурсии R** к функциям g от n переменных и h от (n+2) переменных (обозначение: $f = R(g, h)$), если

1) $f(x_1,..., x_n, 0) = g(x_1,..., x_n)$,

2) $f(x_1,..., x_n, k+1) = h(x_1,..., x_n , k, f(x_1,..., x_n, k)), k \in N^0$.

Рекурсия в этом определении ведется по последней переменной функции f и выражает встречавшийся ранее индуктивный способ определения: зная значение функции при k = 0, мы можем вычислить значение функции для k = 1, затем для k = 2 и т.д.

Примеры. 1) Для n = 1.

$f(x, 1) = x + x + 0 = 2x,$
$f(x, 2) = 2x + x + 3 = 3x + 3,$
$f(x, 3) = (3x + 3) + x + 3\cdot 2 = 4x + 9,$
$f(x, 4) = (4x + 9) + x + 3\cdot 3 = 5x + 18,$
$f(x, 5) = (5x + 18) + x + 3\cdot 4 = 6x + 30.$
.....

Упражнение. Докажите методом математической индукции, что

$f(x, n) = (n + 1) x + 3\cdot C_n^2$

2) Для n = 2.

$f(x, y, 1) = (x + y) \cdot 1,$
$f(x, y, 2) = (x + y) \cdot (x + y) = (x + y)2,$
$f(x, y, 3) = (x + y) \cdot (x + y)2 = (x + y)3$
.....

Нетрудно заметить, что $f(x, y, n) = (x + y)n$.

3) Для n = 0.

$f(1) = 0 + 2\cdot 0 + 1 = 1,$
$f(2) = 1 + 2\cdot 1 + 1 = 4,$
$f(3) = 4 + 2\cdot 2 + 1 = 9,$
$f(4) = 9 + 2\cdot 3 + 1 = 16.$
.....

Методом математической индукции легко показать, что $f(n) = n^2$, поскольку $f(n + 1) = n^2 + 2 n + 1$.

2. Примитивно рекурсивные функции.

Функция называется **примитивно рекурсивной**(сокращенно **п/р**), если она может быть получена из исходных применением конечного числа операторов суперпозиции и примитивной рекурсии.

Отметим, что п/р функции определены при всех $x \in N^0$.

Рассмотрим некоторые примеры примитивно рекурсивных функций.

Функции, выражающие основные арифметические действия сложения и умножения, являются п/р.

А) Сумма $S(x, y) = x + y$ может быть определена так:

1) $S(x, 0) = x$.

Точнее, в обозначениях определения 1, $S(x, 0) = I_1(x)$,

2) $S(x, y + 1) = N(S(x, y)) = (S(x, y))'$, т.е. сумма $x + (y + 1)$ равна числу, следующему за $(x + y)$: $(x + y) + 1$.

В) Произведение $P(x, y) = x \cdot y$ может быть определено так:

1) $P(x, 0) = Z(x)$, т.е. $x \cdot 0 = 0$

2) $P(x, y + 1) = S(P(x, y), x)$, т.е. $x \cdot (y + 1) = x \cdot y + x$

Поскольку операции вычитания, деления и возведения в степень не всегда выполнимы в области натуральных чисел, вводятся их всюду определенные аналоги.

Усеченной разностью называется функция $x \div y = \begin{cases} x - y, \text{если} & x \geq y \\ 0, \text{если} & x \prec y \end{cases}$

Функция $x \div y$ принимает натуральные значения при всех x, y. Обозначим через $\delta(x)$ функцию одной переменной $\delta(x) = x \div 1$;

Положим $x^y = 1$, если $x = y = 0$.

Можно показать, что функции $x \div y$, $\delta(x)$, x^y, $|x - y|$ - примитивно рекурсивны. Также являются п/р функции $[x / y]$ - целая часть частного и rest(x, y) - остаток от деления x на y, если положить $[x / 0] = x$, $rest(x, 0) = x$ для всех x.

3. ***Примитивно рекурсивные предикаты. Условный оператор*****.**

Предикат $P(x_1,...,x_n)$ называется примитивно рекурсивным, если является п/р его характеристическая функция

$$\chi_p(x_1,\ldots,x_n) = \begin{cases} 1, \text{если } P(x_1,\ldots,x_n) \text{ истинно} \\ 0, \text{если } P(x_1,\ldots,x_n) \text{ ложно} \end{cases}$$

Функция $\chi_p(x_1,...,x_n)$ рассматривается как функция от натуральных аргументов $x_1,..., x_n$, принимающая натуральные значения 0 или 1.

Пример. Предикаты $x_1 = x_2$, $x_1 < x_2$ являются п/р предикатами.

Свойство 1. Если $P(x_1,...,x_n)$, $Q(x_1,..., x_n)$- п/р предикаты, то $\neg P$, $P \,\&\, Q$, $P \vee Q$, $P \oplus Q$, $P \sim Q$, $P \to Q$ тоже являются п/р предикатами.

Пример. Предикат $x_1 \leq x_2$, равный $(x_1 < x_2) \vee (x_1 = x_2)$, является п/р предикатом.

Свойство 2. Пусть п/р предикаты $P_1(x_1,..., x_n),..., P_k(x_1,..., x_n)$ таковы, что

$P_i \,\&\, P_j = 0$ тождественно при любых $i \neq j$,

$P_1 \vee P_2 \vee ... \vee P_k = 1$ тождественно, т.е. на любом наборе переменных $x_1,..., x_n$ равен 1 один и только один из предикатов $P_1,..., P_k$.

Пусть также $g_1(x_1,..., x_n),..., g_k(x_1,..., x_n)$ - п/р функции.

Тогда функция $f(x_1,\ldots x_2) = \begin{cases} g_1(x_1,\ldots x_2), \textit{если } P_1(x_1,\ldots x_2) = 1 \\ g_k(x_1,\ldots x_2), \textit{если } P_k(x_1,\ldots x_2) = 1 \end{cases}$ является примитивно рекурсивной.

Функция f совпадает с одной из функций $g_1,..., g_k$ в зависимости от того, какой из предикатов $P_1,..., P_k$ равен 1. При $k = 2$ говорят, что функция f получается

из функций g_1, g_2 применением **условного оператора**. Можно показать, что, например, функции min(x, y), max(x, y) - примитивно рекурсивны.

4. ***Оператор минимизации*****.**

Свойство 1. Если $f(x_1,..., x_n, y)$ - п/р функция, то ограниченная сумма $g(x_1,..., x_n, a) = \sum_{y<a} f(x_1,..., x_n, y)$, при $f(x_1,..., x_n, 0) = 0$, тоже является п/р функцией.

Свойство 2.Если $f(x_1,..., x_n, y)$-п/р функция, то ограниченное произведение $g(x_1,..., x_n, a) = \prod_{y<a} f(x_1,..., x_n, y)$, при $f(x_1,..., x_n, 0) = 1$, тоже является п/р функцией.

Свойство 3. Если $P(x_1,..., x_n, y)$ - п/р предикат, то ограниченный квантор $Q(x_1,..., x_n, a) = \forall_{y<a} y\ P(x_1,..., x_n, y)$, при $P(x_1,..., x_n, 0) = 1$, тоже является п/р предикатом.

Свойство можно рассматривать как обобщение конъюнкции для конечного числа логических сомножителей.

Свойство 4. Если $P(x_1,..., x_n, y)$ - п/р предикат, то ограниченный квантор $Q(x_1,..., x_n, a) = \exists_{y<a} y\ P(x_1,..., x_n, y)$, при $P(x_1,..., x_n, 0) = 0$, тоже является п/р предикатом.

Свойство можно рассматривать как обобщение дизъюнкции для конечного числа логических слагаемых.

Результатом применения **ограниченного оператора минимизации (μ-оператора)** к предикату $P(x_1,..., x_n, y)$ называется функция $f(x_1,..., x_n, z)$, равная такому значению $y \in \{0, 1,..., z\}$, при котором $P(x_1,..., x_n, y) = 1$ и $\forall_{t<y} t\ P(x_1,..., x_n, t) = 0$; если же $P(x_1,..., x_n, y) = 0$ для всех $y \in \{0, 1,..., z\}$, то полагают $f(x_1 ,..., x_n, z) = z + 1$, чтобы функция f была всюду определенной. Обозначение: $f(x_1,..., x_n, z) = \mu_{y\le z} P(x_1,..., x_n, y)$.

В частности, при n = 0 функция $\mu_{y\le z} P(y)$ - это константа $c < z$ такая, что $P(c) = 1$ и $P(t) = 0$, если $t < c$.

5. ***Частично рекурсивные функции*****.**

Итак, из простейших - исходных - функций с помощью операторов суперпозиции и примитивной рекурсии можно получить огромное разнообразие функций, включая основные функции арифметики, алгебры и анализа (с поправкой на целочисленность). Поэтому все эти функции естественно считать вычислимыми.

Остановимся на вопросе о целочисленности примитивно рекурсивных функций. Можно убедиться, что это не является ограничением. Так, дробное число X/Y можно представить словом a/b, где a и b - коды числителя и

знаменателя, а знак операции деления кодируется особым символом. Таким же образом можно, вводя специальное кодирование для символов арифметических процедур, записывать словами любые формулы, а их, в свою очередь, можно кодировать числами.

Возникает вопрос, все ли функции, интуитивно причисляемые к вычислимым, являются п/р? Ответ - отрицательный: можно, например, построить функцию, которая растет быстрее, чем любая п/р функция. Отсюда можно заключить, что описанные средства построения вычислимых функций нуждаются в расширении. Такое расширение достигается включением в число допустимых неограниченного μ-оператора. Однако при этом, в отличие от п/р, могут получиться не всюду определенные функции:

если $f(x_1,..., x_n) = \mu_y (g(x_1,..., x_n, y) = 0,$

то, вычисляя значения g на наборах $(x_1,..., x_n, 0)$, $(x_1,..., x_n, 1)$ и т.д. до тех пор, пока не получим значения 0, мы вычислим значение функции f только в том случае, если такое y существует.

Результатом применения неограниченного оператора минимизации (μ-оператора) к предикату $P(x_1,..., x_n, y)$ называется функция $f(x_1,..., x_n)$, равная такому значению $y \in N^0$, при котором $P(x_1,..., x_n, y) = 1$ и $P(x_1,..., x_n, t) = 0$ для всех $t < y$; если же такого значения y нет, то $f(x_1,..., x_n)$ считается не определенной. Пишут $f(x_1,..., x_n) = my\ P(x1,..., xn, y)$.

То же определение можно записать по-другому:

$f(x_1,..., x_n) = \mu_y P(x_1,..., x_n, y) = \min\limits_{P(x_1,...x_n,t)=1} \{0,1,\ldots,t,\ldots,z\}$

Результатом применения неограниченного оператора минимизации (μ-оператора) к функции $g(x_1,..., x_n, y)$ по переменной y называется функция $f(x_1,..., x_n, z)$, равная $\mu_y (g(x_1,..., x_n, y) = z)$.

То же в другой записи: $f(x_1,..., x_n, z) = \mu_y g(x_1,..., x_n, y) = \min\limits_{g(x_1,...x_n,t)=z} \{0,1,\ldots,t,\ldots,z\}$

Функция называется частично рекурсивной (ч/р), если она может быть получена из исходных применением конечного числа операторов S (суперпозиции), R (примитивной рекурсии) и неограниченных μ - операторов.

Оказывается, что понятие частично рекурсивной функции является в некотором смысле алгоритмически достаточным.

Тезис Черча: всякая эффективно вычислимая функция является частично рекурсивной.

Частично рекурсивная функция называется общерекурсивной (о/р), если она определена всюду на N^0.

Свойство 5. Всякая п/р функция является о/р.

П/р, ч/р и о/р функции называются рекурсивными функциями (р. ф.).

Машины Тьюринга

1. **Машина Тьюринга (МТ или м/т)** задается тройкой $T = (A, Q, P)$, где $A = \{a_1,..., a_n\}$ - внешний алфавит, часто выбирают $A = \{0, 1\}$; $Q = \{q_0, q_1,..., q_m\}$ -

внутренний алфавит, элементы которого обозначают состояния управляющего устройства, обычно q_0 - заключительное состояние, q_1 - начальное состояние; P - **программа**, состоящая из команд вида $q_ia_j \rightarrow q_ka_lS$, где S - один из символов сдвига: L (налево), R (направо), E (на месте); $1 \le q_i \le m$, $0 \le q_k \le m$, a_j, $a_l \in A$.

Пара символов до стрелки называется левой частью команды; тройка символов после стрелки - правой частью. Для каждых i, j программа имеет не более одной команды с левой частью q_ia_j.

Машина функционирует в дискретном времени t = 0, 1, 2,... следующим образом.

Имеется бесконечная в обе стороны лента, разбитая на ячейки (т.е. множество ячеек, упорядоченное по типу Z - множества целых чисел). В начальный момент в конечном числе ячеек записаны символы из A, в остальных ячейках записан пустой символ, обозначаемый через 0. Машина Тьюринга имеет читающую-пишущую головку, которая в любой момент дискретного времени обозревает некоторую ячейку (т.е. находится в ячейке и воспринимает написанный в ней символ) - рис. 8.1.

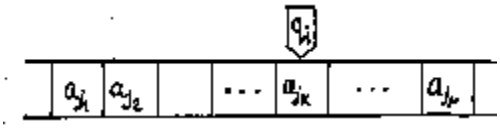


Рис 8.1

МТ имеет управляющее устройство (автомат), которое в каждый момент времени находится в одном из состояний $q_i \in Q$; из Q выделяется подмножество Q_0 заключительных состояний, которое может быть и пустым; обычно $Q_0 = \{q_0\}$, т.е. Q содержит одно заключительное состояние.

Шаг работы МТ. Если МТ находится в состоянии qi и в обозреваемой ячейке записан символ a_j, то выполняется (единственная!) команда с левой частью, образованной этой парой: $q_ia_j \rightarrow q_ka_lS$; МТ переходит в состояние q_k, в ячейке на место a_j записывается a_l и головка сдвигается по ленте согласно S: либо на одну ячейку влево или вправо, либо остается на месте. Таким образом, совершается детерминированный переход: символ a_j воспринимается как входной, а a_l - как выходной для автомата. Затем совершается следующий детерминированный шаг по той же схеме. В начальный момент МТ находится в состоянии q_1; если в процессе работы МТ переходит в одно из заключительных состояний, то она останавливается (команд, в левой части которых присутствует q_0, в программе нет). МТ останавливается также, если отсутствует команда, которую нужно выполнить (т.е. с левой частью, соответствующей текущему полному состоянию МТ): это называется безрезультатной остановкой.

2. **Конфигурацией** (машинным словом) называется слово К вида Uq_iV, где U - слово на ленте левее головки, V - слово правее слова U, начиная с символа, находящегося в той же ячейке, где расположена головка. Стандартная начальная конфигурация – q_1V, а стандартная заключительная - q_0V. Это значит, что в начале работы головка находится в начальном состоянии в самой левой из заполненных ячеек; заполненные ячейки могут не составлять связного куска, а перемежаться незаполненными участками; заключительная конфигурация:

головка в заключительном состоянии также находится в самой левой из заполненных ячеек.

Если при выполнении некоторой команды машины Т из конфигурации $К_i$ получается конфигурация $К_j$, то пишут $Т:К_i \to Кj$. Если $T: K_{i_1} \to K_{i_2} \to \cdots \to K_{i_n}$, то пишут Т: $K_{i_1} \Rightarrow K_{i_n}$. Наконец, если $К_1$ - начальная, а $К_n$- заключительная конфигурации, то пишут $Т:К_1 \mapsto К_n$, а если $Т:К_1 \to К_2 \to ... \to К_n \to ...$ (т.е. машина Т ни в какой момент времени не переходит в заключительное состояние), то пишут $Т:К1\mapsto$. Детерминированная последовательность $K_{i_1}, K_{i_2}, \cdots, K_{i_n} \cdots$ представляет **протокол работы машины Т**. Важно подчеркнуть, что как в начальный, так и в каждый из последующих моментов работы м/т конфигурация на ленте остается конечной (слово в алфавите $А \cup Q$).

Говорят, что машина Т **применима** к слову А, если Т, начав работу на слове А, остановится через конечное число шагов; если при этом получается слово В, то пишут Т(А) = В. Если Т не останавливается, то Т **неприменима** к слову А. В последнем случае совокупность всех конфигураций в процессе бесконечной работы может быть как ограниченной (тогда она периодична: ситуация **зацикливания**), так и неограниченной. М/т считается применимой к множеству слов $\{A_i\}$, если она применима к каждому слову A_i.

Пример 1. Пусть внешний алфавит А = {0, 1}, выражение a^X означает слово aa...a - X раз; аналогично B^X, где В - слово, означает слово BB...B - X раз.

Рассмотрим М/т СЛ(“сдвиг слова 1^X влево”):$q_1001^X0 \mapsto q_001^X0$ можно реализовать следующей программой:

	комментарий
$q_10 \to q_20R$	
$q_20 \to q_31R$	начало сдвига
$q_31 \to q_31R$	цикл движения головки вправо
$q_30 \to q_40L$	начало обратного хода
$q_41 \to q_50L$	конец сдвига
$q_51 \to q_51L$	цикл движения головки влево
$q_50 \to q_00E$	останов

Приведем протокол применения м/т СЛ к начальному слову 111, т.е. при X = 3. Жирным шрифтом выделены изменяющиеся символы:

…	q_10	0	1	1	1	0	…
…	0	q_20	1	1	1	0	…
…	0	**1**	q_31	1	1	0	…
…	0	1	1	q_31	1	0	…
…	0	1	1	1	q_31	0	…
…	0	1	1	1	1	q_30	…
…	0	1	1	1	q_41	0	…
…	0	1	1	q_51	0	0	…
…	0	1	q_51	1	0	0	…
…	0	q_51	1	1	0	0	…
0	q_50	1	1	1	0	0	…
…	q_00	1	1	1	0	0	…

Пример 3. М/т ГЛ (“перевод головки влево”): $01^{X}q_{1}0 \mapsto q_{0}01^{X}0$ имеет следующую программу:

$$q10 \to q20L$$
$$q21 \to q21L$$
$$q20 \to q00E$$

Пример 4. М/т К1 (“копирование слова 1^{X}”): $q101^{X}0$ $q001^{X}01^{X}0$ имеет следующую программу:

$$q10 \to q20R$$

$q20 \to q70L$	$q21 \to q30R$
$q30 \to q40R$	$q31 \to q31R$
$q40 \to q51L$	$q41 \to q41R$
$q50 \to q60L$	$q51 \to q51L$
$q60 \to q21R$	$q61 \to q61L$
$q70 \to q00E$	$q71 \to q71L$

Можно записывать команды в сокращенном виде, опуская в правой части символы, совпадающие с символами в левой части команды. Так, предыдущая программа К1 будет выглядеть следующим образом:

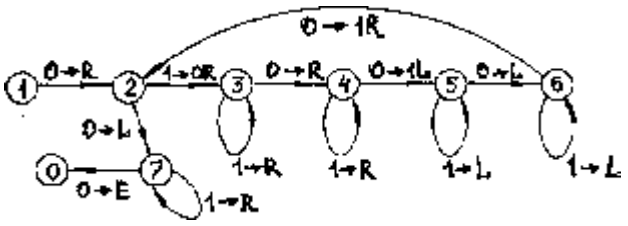
$$q_{1}0 \to q_{2}R$$

$q_{2}0 \to q_{7}L$	$q_{2}1 \to q_{3}0R$
$q_{3}0 \to q_{4}R$	$q_{3}1 \to q_{3}R$
$q_{4}0 \to q_{5}1L$	$q_{4}1 \to q_{4}R$
$q_{5}0 \to q_{6}L$	$q_{5}1 \to q_{5}L$
$q_{6}0 \to q_{2}1R$	$q_{6}1 \to q_{6}L$
$q_{7}0 \to q_{0}E$	$q_{7}1 \to q_{7}L$

Программу м/т можно задавать также таблицей переходов и графом переходов подобно автомату (которым и является управляющее устройство м/т) - надо лишь указывать, кроме выходного символа, символ сдвига. Таблица переходов м/т представляет собой табличное задание программы м/т: строки соответствуют состояниям управляющего устройства, а столбцы - символам внешнего алфавита; в клетках таблицы в полном или сокращенном виде записываются правые части соответствующих команд м/т. Пример предыдущей программы К1 - в таблице 81 и на рис. 8.3.

,

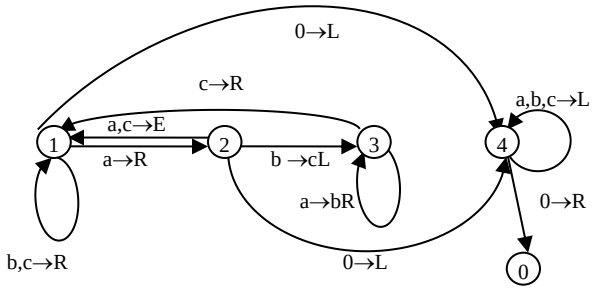
	0	1
q_{1}	$q_{2}R$	
q_{2}	$q_{7}L$	$q_{3}0R$
q_{3}	$q_{4}R$	$q_{3}R$
q_{4}	$q_{5}1L$	$q_{4}R$
q_{5}	$q_{6}L$	$q_{5}L$
q_{6}	$q_{2}1R$	$q_{6}L$
q_{7}	$q_{0}E$	$q_{7}L$



Рассмотрим следующую М/т.

	a	b	c	0
q_1	q_2R	R	R	q_4L
q_2	q_1	q_3cL	q_1	q_4L
q_3	bR		q_1R	
q_4	L	L	L	q_0R

Построим граф переходов.



Составим протокол действия данной М/т на следующую начальную конфигурацию: aababa.

номер шага	результат	номер шага	результат
1	$q_1$aababa	11	abcbcq_1a
2	a$q_2$ababa	12	abcbca$q_2$0
3	a$q_1$ababa	13	abcbcq_4a
4	aa$q_2$baba	14	abcb$q_4$ca
5	a$q_3$acaba	15	abc$q_4$bca
6	ab$q_3$caba	16	ab$q_4$cbca
7	abc$q_1$aba	17	a$q_4$bcbca
8	abca$q_2$ba	18	$q_4$abcbca
9	abc$q_3$aca	19	$q_4$0abcbca
10	abcb$q_3$ca	20	$q_0$abcbca

Приведенный пример демонстрирует отличие м/т от конечного автомата: наличие ленты, на которой могут записываться слова неограниченной длины, обеспечивает потенциально неограниченную внешнюю память м/т. Это значительно расширяет возможности м/т по сравнению с конечным автоматом. Так, преобразования входной информации машинами Тьюринга не могут быть

выполнены автоматом из-за конечной памяти последнего. Ниже рассматриваются некоторые возможности м/т.

3. Вычисление функций машинами Тьюринга.

Будем представлять число m как 01^m0, а кортеж чисел $(m_1,..., m_n)$ как $01^{m_1}01^{m_2}0...01^{m_n}0$.

Говорят, что машина T **правильно вычисляет** функцию $f(x_1,..., x_n)$ с областью определения $D \subseteq N^0 \times N^0 \times ... \times N^0$ (сокращенно f(X)), если для $X \in D$ верно, что $T: q_1 01^{x_1} 01^{x_2} 0 \ldots 01^{x_n} 0 \mapsto q_0 01^{f(X)} 0$, а для $X \notin D$ верно, что $T: q_1 01^{x_1} 01^{x_2} 0...01^{x_n} 0$, т.е. неприменима.

В частности, для функций одной переменной X исходное слово $q_1 01^X 0$ перерабатывается в слово $q_0 01^{f(X)} 0$.

Функция называется **правильно вычислимой**, если есть МТ, которая ее правильно вычисляет.

Пример 1. Функция Z(X) (т.е. константа 0) правильно вычислима.

М/т Z: $q_1 01^X 0 \mapsto q_0 00$; Таблица:

	0	1
q_1	$q_2 1R$	
q_2	$q_3 L$	$q_2 0R$
q_3	L	$q_0 0$

Пример 2. Функция N(x) (“next”) правильно вычислима.

М/т N: $q_1 01^X 0 \mapsto q_0 01^{X+1} 0$; Таблица:

	0	1
q_1	$q_2 R$	
q_2	$q_3 1L$	R
q_3	$q_0 0$	L

Опишем вкратце, как построить м/т для сложения и умножения двух чисел. При сложении слово $q_1 01^X 01^Y 0$ требуется переработать в $q_0 01^{X+Y} 0$. Для этого достаточно сдвинуть слово 1^Y на одну позицию влево, т.е. заменить разделитель 0 на 1 и удалить самую правую единицу.

Если к слову $01^X 0$ (опускаем символ, обозначающий положение головки) применить копирование, а затем сложить полученные два одинаковых числа, то получится удвоение числа X. Его можно рассматривать как прибавление к числу X слагаемого X. Если теперь задана пара чисел $01^X 01^Y 0$, и прибавлять X к первому числу Y раз, уменьшая каждый раз число Y на 1, то в результате получим произведение чисел X и Y.

Говорят, что предикат $P(x_1,..., x_n)$ является вычислимым, если его характеристическая функция $\chi_p(x_1,..., x_n)$ правильно вычислима.

4. Операции над машинами Тьюринга.

Пусть T1 = (A,Q_1,P_1), T2 = (A,Q_2,P_2) - две м/т с общим алфавитом A; $Q_1 = \{q_0', q_1', \ldots q_r'\}$, $Q_2 = \{q_0'', q_1'', \ldots q_s''\}$. **Композицией машин T_1 и T_2** называется м/т T_1T_2 = (A, Q, P); $Q = \{q_0'', q_1', \ldots q_r' \; q_1'', \ldots q_s''\}$, $P = P_1' \cup P_2$, где P_1' получается из P_1 заменой q_0' на q_0'' (т.е. вместо окончания работы T_1 начинает работать T_2).

Различные композиции машин Тьюринга позволяют реализовать комбинации функциональных операторов. Этим устанавливается связь рекурсивных функций и машин Тьюринга.

Свойство 1. Если T_1: $q_1K_1 \mapsto q_0K_2$, а T_2: $q_1K_2 \mapsto q_0K_3$, то T_1T_2:$q_1K_1 \mapsto q_0K_3$. Это означает, что композиция машин последовательно выполняет работу первой и второй.

Замечание. Если T_1:$q_1K_1 \mapsto q_0K_2$, но T_2:$q_1K_2 \mapsto$, то T_1T_2:$q_1K_1 \mapsto$; если T_1:$q_1K_1 \mapsto$, то сразу T_1T_2:$q_1K_1 \mapsto$ независимо от T_2 (т.е. неприменимость на любом из двух этапов влечет неприменимость композиции).

Из этого, в частности, следует следующее свойство.

Свойство 2. Если м/т T_1 правильно вычисляет функцию f: $N^0 \mapsto N^0$, м/т T_2 правильно вычисляет функцию g: $N_0 \mapsto N_0$, то композиция T_1T_2 правильно вычисляет суперпозицию g(f) этих функций.

Теорема 1. Если функции f, g_1, g_2,..., g_n правильно вычислимы, то и функция h = S(f, g_1, g_2,..., g_n) правильно вычислима.

Теорема 2. Пусть функция f(x) задана оператором рекурсии R: f(0) = c, f(x+1) = h(x, f(x)). Тогда, если h(x, y) правильно вычислима, то и f(x) правильно вычислима.

Теорема 3. Если P(x_1,..., x_n, y) - вычислимый предикат, то функция f(x_1,..., x_n) = μ_y P(x_1,..., x_n, y) правильно вычислима.

Поскольку правильно вычислимы исходные п/р функции и реализуемы машиной Тьюринга операторы S, R и μ, то справедлива теорема 4.

Теорема 4 (теорема Тьюринга). Все частично рекурсивные функции правильно вычислимы.

Верно и обратное.

Теорема 5. Всякая функция, вычислимая машиной Тьюринга, частично рекурсивна.