Понятие универсальной машины Тьюринга — теоретической модели современного компьютера

- До сих пор мы придерживались той точки зрения, что различные алгоритмы осуществляются в различных машинах Тьюринга, отличающихся своими функциональными схемами.
- Однако можно построить *универсальную* тьюрингову машину, способную в известном смысле выполнять любой алгоритм, а значит, способную выполнить работу любой тьюринговой машины.

Алгоритм подражания

Для того чтобы лучше уяснить себе, как это делается, представим себе следующий эксперимент.

Пусть на ленту машины подана начальная информация U, и предположим, что некоторому человеку предложено указать, как будет перерабатывать машина эту информацию и во что она переработает ее окончательно. Если этот человек знаком с принципами работы тьюринговых машин, то достаточно ему сообщить, кроме этой начальной информации U, еще функциональную схему машины. Тогда человек, ***подражая*** работе машины и выписывая нужные конфигурации, сможет получить тот же результат, что и машина. Но это как раз и означает, что такой человек способен выполнять работу любой тьюринговой машины, если ему только задана ее функциональная схема. Сам же процесс подражания машине в соответствии с ее функциональной схемой может быть регламентирован в виде точного предписания, которое можно сообщить человеку, не имеющему ни малейшего понятия о машинах Тьюринга. Если человека, располагающего таким предписанием, которое естественно называть ***алгоритмом подражания***, снабдить функциональной схемой какой-либо машины Тьюринга и, кроме того, некоторой начальной конфигурацией, изображенной на ленте, то он окажется способным в точности подражать работе соответствующей машины и в результате выдаст тот же результат. Подобный алгоритм подражания можно было бы задать хотя бы в виде следующей системы указаний:

Указание 1. Обозревайте на ленте ячейку (единственную), под которой подписана буква.

Указание 2. Отыщите в таблице (т.е. в функциональной схеме) столбец, обозначенный такой же буквой, какая подписана под обозреваемой ячейкой.

Указание 3. В найденном столбце обозревайте ту тройку букв, которая расположена на пересечении со строкой, обозначенной такой же буквой, какая вписана в обозреваемой ячейке.

Указание 4. Замените букву из обозреваемой ячейки первой буквой из обозреваемой тройки.

Указание 5. Если в обозреваемой тройке второй буквой является ***!***, то остановитесь: процесс окончен.

Указание 6. Если в обозреваемой тройке второй буквой является ***Н***, то замените букву, подписанную под обозреваемой ячейкой, третьей буквой из обозреваемой тройки.

Указание 7. Если в обозреваемой тройке второй буквой является ***Л***, то сотрите букву, подписанную под обозреваемой ячейкой, и левее её запишите третью букву из обозреваемой тройки.

Указание 8. Если в обозреваемой тройке второй буквой является ***П***, то сотрите букву, подписанную под обозреваемой ячейкой, и правее её запишите третью букву из обозреваемой тройки.

Указание 9. Переходите к указанию 1.

Но оказывается, что вместо человека, действующего в соответствии с алгоритмом, можно поставить некоторую тьюрингову машину; это и будет **универсальная тьюрингова машина**, способная подражать работе любой другой тьюринговой машины. Иными словами, это означает, что алгоритм подражания, словесно описанный нами выше посредством системы девяти указаний, может быть надлежащим образом задан в виде некоторой тьюринговой функциональной схемы (универсальной схемы).

Заметим в начале, что в описанном нами алгоритме подражания в качестве исходных данных (исходной информации) фигурируют функциональная схема подражаемой машины и соответствующая начальная конфигурация. Эта исходная информация перерабатывается алгоритмом в окончательную конфигурацию, изображающую результат, выдаваемый подражаемой машиной. Универсальная машина должна делать то же самое. Однако здесь приходится учитывать следующие два обстоятельства:

1. Непосредственная подача функциональной схемы подражаемой машины и соответствующей конфигурации на ленту универсальной машины в качестве исходной информации невозможна. Действительно, в универсальной машине, как и во всякой тьюринговой машине, информация изображена буквами, расположенными на ленте *одномерно*, т.е. в одной строке, образуя одно или несколько слов во внешнем алфавите машины. В то же время функциональные схемы мы до сих пор задавали посредством «двумерных» таблиц, в которых буквы располагаются несколькими строками; аналогично обстоит дело с конфигурациями, в которых буквы состояний записаны под буквами внешнего алфавита (под лентой).

2. Универсальная машина (как и всякая тьюрингова машина) может располагать лишь фиксированным конечным внешним алфавитом. Между тем она должна быть приспособлена к приему в качестве исходной информации всех возможных схем и конфигураций, в которых могут встречаться буквы из разнообразных алфавитов со сколь угодно большим числом различных букв.

Универсальная машина

Мы должны в первую очередь позаботиться о выработке надлежащего способа задания функциональных схем и конфигураций, который соответствовал бы указанным выше особенностям любой отдельно взятой тьюринговой машины, а именно *одномерности* информации и *конечности* алфавита.

К описанию такого способа мы сейчас и переходим:

1. Вместо того чтобы изобразить схему в виде двумерной таблицы, насчитывающей *k* строк и *m* столбцов, распишем одну за другой *mk* пятерок букв из этой таблицы так, что первый символ пятерки указывает строчку таблицы, второй – столбец таблицы, а последующие три – символы той строки, которая располагается в таблице на пересечении указанной строки с указанным столбцом.

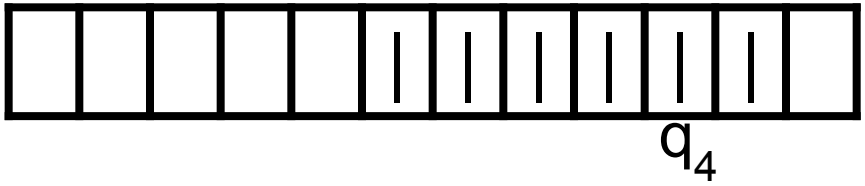
   Рассмотрим, например, функциональную схему алгоритма Евклида (таблица 1):

	q_1	q_2	q_3	q_4	q_5
Λ	Λ$Пq_4$	Λ$Лq_3$	Λ$Пq_1$	Λ$Лq_5$	Λ$Нq_5$
|	α$Нq_2$	β$Нq_1$	|$Пq_1$	|$Лq_1$	|$Нq_5$
α	α$Лq_1$	α$Пq_2$	|$Лq_3$	Λ$Пq_4$	α$Нq_5$
β	β$Лq_1$	β$Пq_2$	Λ$Лq_3$	|$Пq_4$	β$Нq_5$

Т.о. вместо этой схемы возникает одномерная строка символов

$$\Lambda\, q_1\, \Lambda\, \Pi q_4\, \Lambda\, q_2\, \Lambda\, Лq_3\, \ldots \qquad (\Omega)$$

Очевидно, по этой строке можно при желании однозначным образом восстановить первоначальную таблицу. Поступая аналогично, можно при рассмотрении конфигураций условиться о том, чтобы букву состояний писать не под обозреваемой буквой, а непосредственно правее её. Рассмотрим пример конфигурации:



Т.о. эта конфигурация перейдет в строку:

$$|\,|\,|\,|\,|\,q_4\,|$$

Очевидно, по такому одномерному изображению конфигурации можно при желании однозначным образом восстановить её первоначальный вид.

2. Для характеристики функциональной схемы и конфигураций вовсе не существенны специфические начертания букв внешнего алфавита и алфавита состояний, которые в ней фигурируют. Например, если всюду в таблице 1 или в соответствующей ей строке заменить букву β буквой b, то от этого ничего в наших рассмотрениях не изменится. Важно лишь то, чтобы различные объекты были заданы различными символами и чтобы можно было различить буквы состояний от букв внешнего алфавита.

Конечно, можно было выбрать другие буквы, отличные от ***Л, П, Н*** и для обозначения сдвигов (влево, вправо, отсутствие сдвига), однако при этом должно быть четко оговорено, какой именно буквой такой сдвиг обозначается. Здесь сказывается тот факт, что каждая из трех букв обозначает вполне определенное действие, которое не заменимо другим.

Учитывая эти обстоятельства, заменим в строке Ω каждую отдельную букву некоторой последовательностью из единиц и нулей (*кодовой группой*) так, чтобы различные буквы заменялись различными кодовыми группами, но одна и та же буква заменялась бы всюду, где бы она ни встречалась, одной и той же кодовой группой. В результате такой замены, например, строка Ω перейдет в некоторую строку Ω'. Для того чтобы по Ω' можно было восстановить Ω, способ кодирования (отнесения кодовых групп буквам) должен удовлетворять следующим условиям:

1) чтобы строку Ω' можно было бы однозначным образом разбить на отдельные кодовые группы;

2) чтобы можно было распознавать, какие кодовые группы отнесены каждой из букв ***Л, П, Н*** в отдельности, и чтобы можно было различить кодовые группы, отнесенные буквам состояний от тех, которые отнесены буквам внешнего алфавита.

Эти два условия будут наверняка соблюдены при следующем способе кодирования:

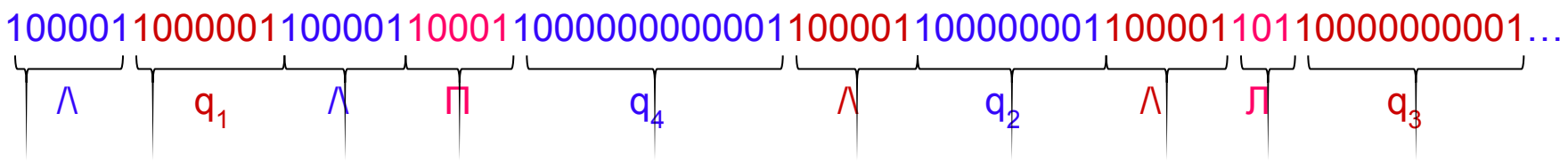
1. В качестве кодовых групп берутся $3+k+m$ различных слов вида 100…01 (между единицами – сплошь нули).

   Тогда разбиение строки Ω' на кодовые группы производится однозначным образом и легко, путем выделения последовательностей нулей, заключенных между двумя единицами.

2. Сопоставление кодовых групп исходным буквам осуществляется согласно следующей таблице кодирования:

	Буква	Кодовая группа	
	Л	101	
	Н	1001	
	П	10001	
Внешний алфавит	S_1	100001 4 нуля	Четное число нулей, большее чем 2
	S_2	10000001 6 нулей	
	…….		
	S_k	10..01 2(k+1) нулей	
Алфавит состояний	q_1	1000001 5 нулей	Нечетное число нулей, большее чем 5
	q_2	100000001 7 нулей	
	…….		
	q_m	10..01 2(m+1)+1 нулей	

При такой системе кодирования в нашем случае Ω' выглядит так:



Подобную строчку из единиц и нулей, составленную для функциональной схемы или для конфигурации, будем называть *шифром функциональной схемы или шифром конфигурации* соответственно. По заданному шифру сама схема или конфигурация в своем первоначальном виде легко восстанавливается; поэтому задание схемы или конфигурации можно всегда осуществить посредством их шифров. Разумеется, вместо единицы и нулей можно было бы брать любые другие два знака, например *a,b*.

Теперь уже нетрудно сообразить, как изменить формулировки указаний 1-9 из первоначального описания алгоритма подражания для того, чтобы получить алгоритм, который перерабатывает шифр схемы подражаемой машины и шифр начальной конфигурации в шифр результирующей конфигурации.

Указание 1. Обозревайте в шифре конфигурации кодовую группу (единственную), расположенную непосредственно левее кодовой группы с нечетным числом нулей.

Указание 2 и 3. Отыщите в шифре схемы пару соседних кодовых групп, одинаковую с парой кодовых групп в шифре конфигурации, в которой первая группа является обозреваемой.

Указание 4. Замените в шифре конфигурации обозреваемую кодовую группу на первую кодовую группу из тройки кодовых групп в шифре схемы, стоящей после пары соседних кодовых групп из указаний 2 и 3.

Указание 5. Если в обозреваемой тройке кодовых групп в шифре схемы второй является группа, которая обозначает знак остановки **!**, например 1000001, то УМТ останавливается и полученный шифр конфигурации будет результатом работы УМТ.

Указание 6. Если в обозреваемой тройке кодовых групп из шифра схемы второй является группа 1001, то в шифре конфигурации замените кодовую группу с нечетным числом нулей третьей кодовой группой из этой обозреваемой тройки.

Указание 7. Если в обозреваемой тройке кодовых групп из шифра схемы второй является группа 10001, то в шифре конфигурации замените кодовую группу с нечетным числом нулей на справа стоящую от неё группу с четным числом нулей, а вместо данной кодовой группы с четным числом нулей запишите третью кодовую группу с нечетным числом нулей из обозреваемой тройки в шифре схемы.

Указание 8. Если в обозреваемой тройке кодовых групп из шифра схемы второй является группа 101, то в шифре конфигурации замените кодовую группу с нечетным числом нулей на слева стоящую от неё группу с четным числом нулей, а вместо этой группы с четным числом нулей запишите третью кодовую группу из обозреваемой тройки в шифре схемы.

Указание 9. Переходите к указанию 1.

Дальнейшее рассмотрение этого алгоритма позволяет сводить каждую операцию над кодовыми группами к цепи стандартных операций, осуществимых в тьюринговой машине (замена одного знака другим, сдвиг на один шаг и т.п.). При этом, кроме знаков 1 и 0, из которых построены шифры, будут участвовать и другие буквы, например буква, отделяющая один шифр от другого, буквы, играющие роль временных пометок при просмотре единиц и нулей и другие.

Пример: рассмотрим сложение двух чисел. На ленту машины подается два числа (2 и 1), разделенных знаком *, и функциональная схема МТ, которая выполняет сложение.

Функциональная схема имеет вид:

	q_0	q_1	q_2
|	Λ П q_2	| Л q_1	| П q_2
Λ	Λ П q_1	Λ П q_0	| Н q_1
*	Λ !	* Л q_1	* П q_2

Кодирование внешнего алфавита и алфавита состояний в данном случае будет выглядеть следующим образом:

|	100001
Λ	10000001
*	1000000001
q_0	1000001
q_1	100000001
q_2	10000000001
!	1000000000001

Рассмотрим начальную конфигурацию УМТ (здесь подразумевается, что каждый символ находится в одной ячейке ленты, многоточие означает, что кодирование функциональной схемы выполнено не до конца, символ / отделяет шифр функциональной схемы от шифра начальной конфигурации):

10000110000011000000110001100000000011000011000000001

10000110110000000110000110000000001100001100011000000001…/
10000110000011000011000000001100001

Далее после выполнения указаний 1-4 получим конфигурацию:

10000110000011000000110001100000000011000011000000001

10000110110000000110000110000000001100001100011000000001…/
10000001100000110000110000000001100001

После выполнения указаний 5-8 получим конфигурацию вида:

1000011000001100000011000110000000011000011000000001

1000011011000000011000011000000000110000110001100000000001…/
1000000110000110000000001100000000001100001

Переходим к указанию 1 и т.д. В конце мы получим результирующую конфигурацию:

1000011000001100000011000110000000011000011000000001

1000011011000000011000011000000000110000110001100000000001…/
1000000110000001100000011000000000001100001100001100001

В конечном счете, в результате такой детализации алгоритм подражания описывается функциональной схемой машины Тьюринга ***M***, которая *универсальна* в следующем смысле. Если какая-либо машина ***A*** решает некоторую задачу, то и ***M*** решает эту задачу при условии, что кроме шифра исходных данных этой задачи на ее ленту будет подан шифр схемы машины ***A***.

Тем самым установлена теорема:

Теорема. *Существуют универсальные машины Тьюринга.*

В связи с этим фактом всевозможные функциональные схемы (или их шифры) можно толковать двояко:

1) схема описывает логический блок *специальной* машины Тьюринга, реализующей соответствующий алгоритм (это и есть точка зрения, которая проводилась первоначально);
2) схема описывает программу, подаваемую на ленту универсальной машины для реализации соответствующего алгоритма.

Заметим в заключении, что современные вычислительные электронные машины и строятся как раз как универсальные машины, в запоминающее устройство которых наряду с исходными данными поставленной задачи вводится также и программа ее решения.

Разделение памяти на внешнюю и внутреннюю характерно и для вычислительных машин. Однако в отличие от машины Тьюринга, в которой внешняя память бесконечна (лента бесконечна), в любой реальной вычислительной машине внешняя память конечна.

Разумеется, это принципиальное различие между реальной вычислительной машиной и машиной Тьюринга, представляющей собой некоторую абстрактную, идеализированную машину, является неустранимым. Вместе с тем важно отметить, что в реальной вычислительной машине внешнюю память можно неограниченно увеличивать без изменения конструкции машины.