Алгоритмы Маркова

Теория алгоритмов оказала существенное влияние на развитие ЭВМ и практику программирования. В теории алгоритмов были предугаданы основные концепции, заложенные в аппаратуру и языки программирования ЭВМ. Языки символьной обработки информации (РЕФАЛ, ПРОЛОГ) берут начало от нормальных алгоритмов Маркова и систем Поста.

Андрей Андреевич Марков

- Андрей Андреевич Марков (1903-1979), советский математик. Сын известного русского математика А.А. Маркова.
- Окончил Восьмую Петроградскую Гимназию в 1919 году. Окончил Ленинградский Университет в 1924 году. Окончил аспирантуру в Астрономическом Институте (Ленинград) в 1928 году.

- Ученая степень доктора физико-математических наук присвоена без защиты диссертации в 1935 году.
- В 1933-1955 годах работал в Ленинградском университете (с 1936 г. - профессор).
- С 1936 г. по 1942 г. и с 1944 г. по 1953 г. заведовал кафедрой геометрии Ленинградского Государственного Университета.
- В 1939-1972 работал в Математическом институте им.Стеклова АН СССР.
- До июля 1942 года находился в блокадном Ленинграде.
- С 1959 г. зав. кафедрой математической логики Московского университета.
- Основные труды по топологии, топологической алгебре, теории динамических систем, теории алгоритмов и конструктивной математике.
- Доказал неразрешимость проблемы гомеоморфизма в топологии, создал школу конструктивной математики и логики в СССР, автор понятия нормального алгоритма.
- Награжден орденом "Знак почета" (1945), орденом Ленина (1954), орденом Трудового Красного Знамени (1963), медалью "За доблестный труд" (1945) и медалью "За оборону Ленинграда" (1946). Премия им. Чебышева АН СССР (1969).

Алгоритмы Маркова

В 1954 году советский математик А.А.Марков предложил другую алгоритмическую схему, эквивалентную машине Тьюринга, в которой данные преобразуются на основе других принципов.

В алгоритмической схеме Маркова нет понятия ленты и осуществляется непосредственный доступ к различным частям преобразуемого слова.

Марков назвал эту алгоритмическую схему нормальным алгоритмом.

Алфавитом будем называть всякое непустое конечное множество символов, а сами символы алфавита – буквами.

Например, алфавитами являются {1},{0,1},{a,b},{1,*},{1,-},{1,*,□,-}

Словом в алфавите А называется всякая конечная последовательность букв алфавита А.

Например,словами в алфавите {a,b} являются a ,b, aa, abba, bbba, baba.

Пустая последовательность букв называется пустым словом и обозначается через **λ**. Для любого алфавита пустое слово есть слово в этом алфавите.

Будем говорить , что имеется вхождение слова P в слово R, если слово R имеет вид R_1PR_2.

Если P и Q – слова в алфавите А,то выражения

P →Q и P→·Q будем называть формулами подстановки в алфавите А. Заметим,что каждое слово P и Q может быть пустым словом.

Формула подстановки P →Q называется простой.

Формула подстановки P →·Q называется заключительной.

Схема алгоритма:

Формат команды (строки) следующий

$P_i \square (\cdot) Q_j$,

где

Pi – последовательность символов, которая ищется в слове

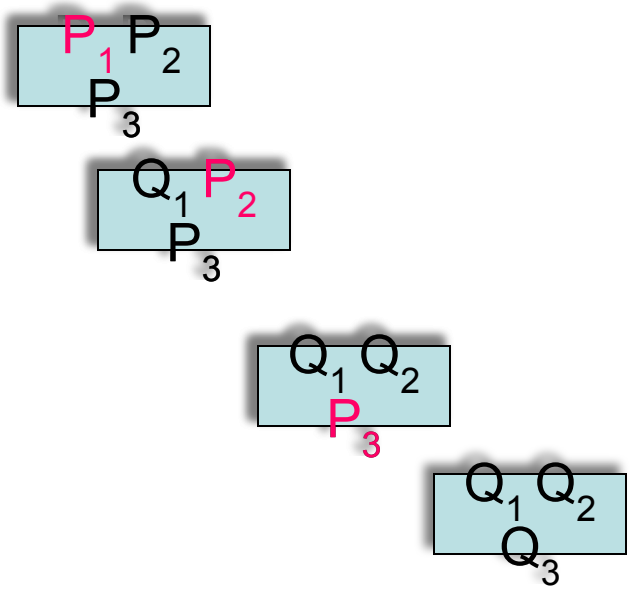
$\square$ - знак перехода к операции записи

Qj - последовательность символов, которая записывается вместо найденной

(•) - знак принудительного окончания алгоритма (необязательный параметр)

$$\begin{cases} P_1 \to (\cdot) Q_1 \\ P_2 \to (\cdot) Q_2 \\ \text{M} \\ P_r \to (\cdot) Q_r \end{cases}$$

Нормальный алгоритм над А преобразует слова в алфавите А следующим образом. Работа данного нормального алгоритма над словом R состоит из отдельных шагов,в результате которых получаются слова $R=R_1, R_2, R_3, \ldots, R_i, R_{i+1}, \ldots$ (1)



$P_i \square (\cdot) Q_i$

Суть упорядоченного использования правил состоит в том, что каждое переработанное слово вновь поступает в "начало" работы алгоритма и вновь проверяется на возможность применения правил подстановки.

Следует обратить внимание, что порядок следования правил подстановки в схеме алгоритма существенно влияет на результат, в то время как для МТ он не существенен.

Работа нормального алгоритма над словом заканчивается в двух случаях:

1) существует такое слово R_i из (1),что ни одна левая часть формул подстановки из нормальной схемы не имеет вхождений в слово R_i

2) существует такое слово R_j из (1),что слово R_{j+1} получается из R_j с помощью заключительной формулы подстановки.

В первом случае результатом работы алгоритма над словом R объявляется слово R_i .Во втором - слово R_{j+1}.

В остальных случаях работа алгоритма не заканчивается,т.к последовательность (1) будет бесконечной, и тогда говорят, что данный нормальный алгоритм не применим к слову R.

Пример1.

Тождественный нормальный алгоритм над А – это нормальный алгоритм над А, который применим к каждому слову в алфавите А и результатом работы которого является это же слово.

Такой алгоритм может быть задан алфавитом В=А (не содержащим → и ·) и нормальной схемой {→·

Пример 2.

Нормальный алгоритм над А «левого присоединения» слова Q(фиксированного) – это нормальный алгоритм над А, применимый к каждому слову R в алфавите А, и результатом работы которого над словом R является слово QR.

Такой алгоритм может быть задан алфавитом В=А и нормальной схемой {→·Q

Заметим, что самое левое вхождение является пустым словом.

Пример 3.

Нормальный алгоритм над алфавитом {a,b} «правого присоединения» слова aba – это нормальный алгоритм,применимый к каждому слову в алфавите {a,b} , и результатом работы которого над словом R будет слово Raba.

Зададим его алфавитом B={a,b,c} и нормальной схемой

ca → ac
cb → bc
c → · aba
→ c

abaa
cabaa
acbaa
abcaa
abaca
abaac
abaaaba

Здесь c выполняет роль рабочего символа

Пример 4.

Рассмотрим алгоритм, который перерабатывает всякое слово P в алфавите A,содержащее хотя бы одно вхождение буквы b ,в слово, которое получается вычеркиванием в P самого левого вхождения буквы b.

Пусть A есть алфавит {b,c}.

Рассмотрим схему подстановки:

$$\{b \rightarrow \bullet\Lambda$$

Пример 5.

Нормальный алгоритм удвоения – это нормальный алгоритм над A, преобразующий каждое слово R в алфавите в слово RR.

Пусть A={a,b}. Зададим нормальный алгоритм над {a,b} алфавитом {a,b,c,d} и нормальной схемой

$$ca \to adac$$
$$cb \to bdbc$$
$$daa \to ada$$
$$dab \to bda$$
$$dba \to adb$$
$$dbb \to bdb$$
$$d \to \Lambda$$
$$c \to \cdot$$
$$\Lambda \to c$$

Здесь аналогично заводятся два рабочих символа c и d.
c по последней формуле подстановки как бы навешивается на пустое слово.

Пояснение: da – это дубликат символа a, db – дубликат символа b.
Алгоритм сначала заводит дубликаты каждого символа исходного слова,
а затем переставляя местами дубликаты символов и сами символы, собирает все дубликаты в конце слова. Заметим, что дубликаты не могут переставляться с дубликатами и символы не могут переставляться с символами.

Т.о c и d являются вспомогательными атрибутами,хранящими и обрабатывающими в процессе выполнения алгоритма промежуточные состояния возможных подслов,которые впоследствии удваивают каждый символ.Т.о каждому входному символу присваивается свой рабочий символ

Работа предыдущего нормального алгоритма над словом bab состоит из следующей последовательности слов :

bab →
cbab →
bdbcab →
bdbadacb →
bdbadabdbc →
badbdabdbc →
badbbdadbc →
babdbdadbc →
babbdadbc →
babbadbc →
babbabc →
babbab .

(каждое подчеркнутое подслово заменяется определенным правилом подстановки)

Пример 6.

Алгоритм, состоящий из одной строки, вида 0 □ *

будучи примененным к слову в алфавите {0,1}, заменит все нули на звездочки.

В свою очередь алгоритм 0 □ * •

будучи примененным к слову в алфавите {0,1}, заменит на звездочку первый встреченный ноль.

Пример 7.

Довольно сложная для реализации на машинах Тьюринга задача сортировки слова по возрастанию, решается при помощи алгоритма Маркова намного быстрее и проще. Допустим в алфавите {0,1,2} :

20 □ 02

10 □ 01

21 □ 12

Построим алгоритм для вычисления функции U(N)=N+1;

S={0,1,2,3,4,5,6,7,8,9}; V={*,+}.

Схема имеет вид;

Перегоняем служебный символ * в конец слова n, чтобы отметить последнюю цифру младших разрядов.

Увеличиваем на единицу, начиная с цифр младших разрядов.

Сложность этого алгоритма, выраженная в количестве выполненных правил подстановки, будет равна:

- (k+1)+(m+1),
- где k - количество цифр в N, m - количество 9, которые были увеличены на 1.

1 → 1
2 → 2
3 → 3
4 → 4
5 → 5
6 → 6
7 → 7
8 → 8
9 → 9
*** → +**

0+ → 1
1+ → 2
2+ → 3
3+ → 4
4+ → 5
5+ → 6
6+ → 7
7+ → 8
8+ → 9
9+ → +0
+ → 1

Вычисление числовой функции с помощью нормального алгоритма.

Целое неотрицательное число m будем изображать словом из m+1 едениц. Набор чисел $m_1,m_2\ldots,m_n$ будем обозначать словом $1^{m(1)+1} * 1^{m(2)+1} *\ldots* 1^{m(n)+1}$.

Пример 8.

Построим нормальный алгоритм М ,вычисляющий числовую функцию f(x)=x+1.

Нормальный алгоритм зададим алфавитом {1} и нормальной схемой {1→·11.

Он применим к каждому слову в алфавите {1},и его работа при вычислении f(m) для любого числа m состоит из двух слов $1^{m+1},1^{m+2}$. (1^m=11….1- m раз)

Пример 9.

Построим нормальный алгоритм К, вычисляющий числовую функцию S(x,y)=x+y.

Он будет применим ко всем словам вида $1^{m+1} * 1^{n+1}$,и результатом его работы будет слово 1^{m+n+1} .

К зададим алфавитом {1,*}

и нормальной схемой {* 1 →·.

Пример 10.

Дано произвольное двоичное слово. Надо убрать из него два первых знака.

Рассмотрим алгоритм вида:

00 □ •

01 □ •

10 □ •

11 □ •

Если даны слова например «001011» или «01011101» , то алгоритм действительно выполнит указанную задачу .

Но в слове 1100101 выбросятся два нуля, которые вовсе не являются первыми символами слова.

В этом случае существующий алфавит надо расширить вспомогательными буквами. Они вводятся с помощью формулы λ □ α (α – вспомогательная буква) или, что более корректно, пары формул

λ 0 □ α 0

λ 1 □ α 1

Применив такие продукции к слову λ 1100101 λ получим:

λ α 1100101 λ

α 0 □ β

α 1 □ β **λ β 100101 λ**

β 0 □ •

β 1 □ • **λ 00101 λ**

Теорема 1:
всякая вычислимая по Тьюрингу функция f является вычислимой по Маркову.

Пример: пусть существует следующая машина Тьюринга, которая печатает на чистой ленте последовательность 001001001001….

Aλ □ 0RB
Bλ □ 0RC
Cλ □ 1RA

Работа алгоритма происходит следующим образом (через запятую указаны пара: символ-состояние) A, 0 B, 0 0C, 001A

Построит алгоритм Маркова, для чего к внешнему алфавиту {0,1} добавляем внутренний алфавит {A,B,C...}

A □ 0B
B □ 0C
C □ 1A

Доказательство:

Пусть функция $f(x_1,\ldots,x_n)$ вычислима по Тьюрингу и ее вычисляет машина Тьюринга T с алфавитом A.

Пусть C- расширение алфавита A. Покажем, что существует нормальный алгоритм $B_{T,C}$ над C, вполне эквивалентный относительно C машине Тьюринга T.

Это означает,что для любых натуральных чисел $k_1,\ldots,k_n$ найдутся такие слова R_1 и R_2 (возможно, пустые) в алфавите $\{S_0\}$ (S_0 – изображение пустой ячейки ленты МТ),что

$B_{T,C}(k1*\ldots*kn)= R_1\, f(k1,\ldots,kn) R_2$.

Пусть C= A U $\{q_{k(0)},\ldots,q_{k(m)}\}$, где $q_{k0},\ldots,q_{k(m)}$-внутренние состояния T и $q_{k0}=q_{0.}$.

Построим схему для искомого алгоритма $B_{T,C}$:

1) выпишем сначала для всех команд вида q_iS_i : $S_kH\,q_r$ машины Тьюринга T подстановки $q_iS_i\rightarrow q_rS_k$.

2) Затем для каждой команды вида q_iS_i : $S_k\,L\,q_r$ выпишем все возможные подстановки вида $S_nq_iS_i\rightarrow q_rS_nS_{k,,}$ где S_nє A, и формулу подстановки $q_iS_i\rightarrow q_rS_0S_{k...}$

3) Далее для каждой команды q_jS_i: $S_k\,R\,q_r$ выпишем все возможные подстановки $q_jS_iS_n\rightarrow S_kq_rS_n$,где S_n єC, и формулу подстановки $q_jS_i\rightarrow S_kq_rS_{0.}$.

4) Далее выпишем всевозможные формулы подстановки $q_{k(i)}\rightarrow\Lambda$, где $q_{k(i)}$ єC и формулу подстановки $\Lambda\rightarrow q_0$.

Полученная таким образом схема определяет некоторый нормальный алгоритм $B_{T,C}$ над C, и легко показать ,что $B_{T,C}(P)\approx T(P)$ для любого слова P ,т.е $B_{T,C}$ – есть искомый алгоритм Маркова.

Пусть G_1-нормальный алгоритм над {1,*,S_0},стирающий все вхождения S_0 перед первым вхождением 1 или * во всяком слове в алфавите {1,*,S_0}.Такой алгоритм задается схемой1

Пусть также G_2-нормальный алгоритм над {1,*,S_0}, который стирает все вхождения S_0 после последнего вхождения 1 или * во всяком слове в алфавите {1,*,S_0}.G_2 можно задать схемой2:

$$\begin{cases} \alpha S_0 \to \alpha \\ \alpha 1 \to \cdot 1 \\ \alpha * \to \cdot * \\ \alpha \to \cdot \Lambda \\ \Lambda \to \alpha \end{cases} \qquad \begin{cases} \alpha * \to * \alpha \\ \alpha 1 \to 1 \alpha \\ \alpha S_0 \to \alpha \\ \alpha \to \cdot \Lambda \\ \Lambda \to \alpha \end{cases}$$

Положим теперь $G = G_2 \circ G_1 \circ B_{T,C}$.

Для любых натуральных $k_1,\ldots,k_n$ имеем

$B_{T,C}(k_1 * \ldots * k_n) \approx R_1 f(k_1,\ldots,k_n) R_2$ где R_1 и R_2- некоторые слова в $\{S_0\}$.

Поэтому

$G_1(R_1 f(k_1,\ldots,k_n) R_2) \approx f(k_1,\ldots,k_n) R_2$ и

$G_2(f(k_1,\ldots,k_n) R_2) \approx f(k_1,\ldots,k_n)$.

Получаем ,что f есть вычислимая по Маркову функция, которую вычисляет нормальный алгоритм G.

Теорема 2: Всякая вычислимая по Маркову функция вычислима по Тьюрингу.

Доказательство:

Пусть B – нормальный алгоритм в алфавите A, не содержащем S_0 и σ.Тогда существует МТ $M_{(AU\{S(0),\sigma\})}$ в алфавите $AU\{S_0,\sigma\}$, такая что

для всякого слова W в A машина M применима к W тогда и только тогда, когда к W применим алгоритм B, и при этом M(W) имеет следующий вид

$$S_0^m B(W) S_0^n$$

где m и n-целые неотрицательные числа.

Значения алгоритмов M и B формально различны, т.к на ленте МТ S0 есть по существу символ «пустоты», а в нормальном алгоритме S0 есть буква равноправная с любой другой буквой.

Пусть $A=\{S_1,S_2\ldots S_k\}$.

Пусть $P\rightarrow(\cdot)Q$ – произвольная формула подстановки. Построим систему команд МТ, действие которой состоит в замещении самого левого вхождения слова P в произвольное слово W (если такие вхождения вообще имеются) словом Q.

Если $P\neq\Lambda$,то пусть $P=b_0\ldots b_r$

Рассмотрим следующую систему команд

$q_0\ S_i\ R\ q_0$ $(S_i \epsilon A, S_i \neq b_0)$
$q_0\ b_0\ \sigma\ q_0$
$q_0\ \sigma\ R\ q_2$
$q_2\ b_1\ R\ q_3$
$q_2\ S_i\ S_i\ q_{r+2}$ $(S_i \epsilon\ AU\{S_0\}\ ,\ S_i \neq b_1)$
$q_3\ b_2\ R\ q_4$
$q_3\ S_i\ S_i\ q_{r+2}$ $(S_i \epsilon\ AU\{S_0\}\ ,\ S_i \neq b_2)$
……
$q_r\ b_{r-1}\ R\ q_{r+1}$
$q_r\ S_i\ S_i\ q_{r+2}$ $(S_i \epsilon\ AU\{S_0\}\ ,\ S_i \neq b_{r-1})$
$q_{r+1}\ b_r\ R\ q_{r+4}$
$q_{r+1}\ S_i\ S_i\ q_{r+2}$ $(S_i \epsilon\ AU\{S_0\}\ ,\ S_i \neq b_r)$
$q_{r+2}\ S_i\ L\ q_{r+2}$ $(S_i \epsilon\ AU\{S_0\}\ )$
$q_{r+2}\ \sigma\ b_0\ q_{r+3}$
$q_{r+3}\ b_0\ R\ q_0$
$q_0\ S_0\ L\ q_{r+5}$
$q_{r+5}\ S_i\ L\ q_{r+5}$ $(S_i \epsilon A)$
$q_{r+5}\ \sigma\ b_0\ q_{r+5}$
$q_{r+5}\ S_0\ R\ q_y$

где индекс Y- некоторое целое число,большее любого из индексов,которые еще будут употреблены.

Эта система команд следующим образом действует на слово W:

если W не содержит вхождений слова P, то действие этой системы команд заканчивается конфигурацией $q_Y W$.

Если же W содержат вхождения слова P и $W=W_1 P W_2$, где W_1 и W_2 определяются самым левым вхождением слова P в W,то работа системы команд закончится конфигурацией $W_1 P q_{r+4} W_2$.

Приведенную систему команд следует расширить дополнительными командами,с помощью которых выделенное вхождение слова P заменялось бы на Q.

Пусть $Q=c_0 \ldots c_s$. Возможны три случая:

1) s=r, т.е P и Q – слова одной длины. В этом случае добавим команды:

q_{r+4} S_i L q_{r+7} ($S_i \epsilon$ AU{S_0})

q_{r+7} b_r c_r q_{r+8}

q_{r+8} c_r L q_{r+9}

q_{r+9} b_{r-1} c_{r-1} q_{r+10}

q_{r+10} c_{r-1} L q_{r+11}

……

…

q_{3r+7} b_0 c_0 q_{3r+8}

q_{3r+8} S_i L q_{3r+8} ($S_i \epsilon$A)

q_{3r+9} S_0 R q_u

Применяя эти команды к W_1 P q_{r+4} W_2 мы получим $q_u W_1$Q W_2.

Где

$$u = \begin{cases} 0, формула подстановки P \to (\cdot) Q - простая \\ 1, формула - заключительная \end{cases}$$

2) s<r , т.е. Q короче P

Добавим команды:

q_{r+4}	S_i	L	q_{r+7}
q_{r+7}	b_r	c_s	q_{r+8}
q_{r+8}	c_s	L	q_{r+8}
	…..		
$q_{r+7+2s+2}$	b_{r-s-1}	S_0	$q_{r+7+2s+2}$
$q_{r+7+2s+2}$	S_0	L	$q_{r+7+2s+3}$
$q_{r+7+2s+3}$	b_{r-s-2}	S_0	$q_{r+7+2s+3}$
$q_{r+7+2s+3}$	S_0	L	$q_{r+7+2s+4}$
	…….		
q_{2r+s+8}	b_0	S_0	q_{2r+s+8}

Применение этих команды к $W_1 P\, q_{r+4}\, W_2$ приводит к конфигурации $W_1 q_{2r+s+8} S_0^{r-s} Q\, W_2$.

Теперь предусмотрим команды,с помощью которых можно было бы слово W_1 подвинуть на ленте на r-s квадратов вправо,чтобы получить слово $W_1 Q W_2$.Пусть M –целое число,больше всех встречающихся выше индексов при q_i и $S_{i,,}$,например M=3r+9

Добавляем команды:

Начиная с конфигурации $W_1 q_{2r+s+8} S_0^{r-s} Q W_2$,с помощью этих команд приходим при некотором положительном p к конфигурации $(S_0)^p q_u W_1 Q W_2$

3) s>r, т.е слово Q длиннее слова P. Этот случай рассматривается аналогично.

$$
\begin{cases}
q_{2r+s+8} S_0 L q_M \\
q_M S_j \delta q_{M+j} \\
q_{M+j} \delta R q_{M+j} \\
q_{M+j} S_0 R q_{M+j} \\
q_{M+j} S_l L q_{2M+j} \\
q_{2M+j} S_0 S_j q_{2M+j} \\
q_{2M+j} S_j L q_{3M+j} \\
q_{3M+j} S_0 L q_{3M+j} \\
q_{3M+j} \delta S_0 q_{4M+j} \\
q_{4M+j} S_0 L q_{5M+j} \\
q_{5M+j} S_0 R q_{6M+j} \\
q_{6M+j} S_0 R q_{6M+j} \\
q_{6M+j} S_l S_l q_n \\
\text{где} \\
u = \begin{cases} 1, \text{если} P \to (\cdot) Q - \text{заключительная} \\ 0, \text{если} P \to (\cdot) Q - \text{простая} \end{cases} \\
q_{5M+j} S_l S_l q_M
\end{cases}
$$

Пусть теперь задан произвольный нормальный алгоритм G в алфавите $A=\{S_1,..S_k\}$, не содержащий S_0 и σ, и схема алгоритма G есть

$$P_1\rightarrow(\cdot)Q_1,.., P_h \rightarrow(\cdot)Q_h.$$

Определим машину Тьюринга M следующим образом.

1). Воспроизведем всю предыдущую конструкцию для $P_1\rightarrow(\cdot)Q_1$.

2). Перейдем ко второй подстановке. Построим список команд по образцу, который изложен выше. Эти полученные команды будут начинать действие после того, как слово, полученное первой группой команд, окажется лишенным вхождения слова P_1 .

С помощью этих новых команд находящееся на ленте слово будет испытываться на наличие в нем вхождений слова P_2. При этом имеются 2 возможности:

А) Самое левое из них будет замещено на Q_2 и машина перейдет в состояние q_1, если $P_2 \rightarrow (\cdot) Q_2$ заключительная подстановка, либо в состояние q_0 ,если – простая формула подстановки.

Б) Вхождений P_2 нет. Машина работающая по командам $P_2 \rightarrow (\cdot) Q_2$ в конечном итоге оставляет без изменения находившееся на ленте слово. Теперь начинают действовать команды $P_3 \rightarrow (\cdot) Q_3$, которые строятся аналогично.

3). Т.о достраиваем всю систему команд машины Тьюринга М, которая имитирует работу нормального алгоритма В в том смысле, что для любого слова W в А машина М применима к W тогда и только тогда, когда к W применим В и М(W) имеет вид $(S_0)^m B(W)(S_0)^n$, где m и n-целые неотрицательные числа.

Заключительные замечания

В машине Тьюринга предусматривается управление последовательностью доступа к различным элементам обрабатываемого слова (сдвиг и влево и вправо). Алгоритмическая схема Маркова жестко закрепляет последовательность доступа (каждый раз ищется первое вхождение левой части очередной формулы подстановки).

Аналогично и последовательность действий в машине Тьюринга управляется за счет смены состояний, а в алгоритмической схеме Маркова реализуется жесткая схема управления (последовательный перебор формул подстановки, а после каждого применения не завершающей формулы возврат к самой первой формуле).

В то же время элементарная операция по преобразованию информации в машине Тьюринга очень проста (замена буквы в ячейке ленты), а в алгоритмической схеме Маркова весьма мощная (замена любого слова на любое другое слово).