**Тезис Маркова**: любой вычислительный процесс можно преобразовать в нормальный алгоритм.

**Алгоритмы Маркова**

Основная операция при работе алгоритмов Маркова – это переработка слов в некотором алфавите. Эта переработка заключается в производстве некоторого количества замен определенных последовательностей символов. Эти замены совершаются в СТРОГО определенном порядке, а именно: после каждой замены алгоритм читается с самого начала, а слово анализируется с самого первого символа. В отличие от машин Тьюринга, алгоритмы Маркова выполняются без какого – либо устройства, осуществляющего движения и имеющего внутреннюю память. В данном случае мы можем оперировать только ленточными знаками. Сама лента в этом случае не разделяется на строгие ячейки, а имеет гибкую основу, что позволяет ей растягиваться и сжиматься исходя из того, увеличивается ли в слове число символов или уменьшается.

Формат команды (строки) следующий

{ai} 🡪 {bj} [•],

где

{ai} – последовательность символов, которая ищется в слове

🡪 - знак перехода к операции записи

{bj} - последовательность символов, которая записывается вместо найденной

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

Программа (алгоритм) представляет собой совокупность строк указанного вида, причем порядок строк имеет важнейшее значение. Окончание работы алгоритма происходит в тот момент, когда выполняется строка, содержащая знак принудительной остановки, либо тогда, когда более ни одна строка не может быть выполнена (в слове нет ни одной из искомых последовательностей символов).

Например, 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.

Довольно сложная для реализации на машинах Тьюринга задача сортировки слова по возрастанию, допустим в алфавите {0,1,2}, решается при помощи Маркова весьма изящно:

20 🡪 02

10 🡪 01

21 🡪 12

Некоторые задачи переработки слов нельзя решить без расширения алфавита.

Например, дано произвольное двоичное слово. Надо убрать из него два первых знака.

Рассмотрим алгоритм вида:

00 🡪 •

01 🡪 •

10 🡪 •

11 🡪 •

Если в слове случайно первыми двумя символами были нули (например, «001011»), то алгоритм действительно выполнит указанную задачу. Также работа закончится успешно, если в слове ни разу не встретилось два нуля подряд, а первыми символами оказалась пара 01 (например, «01011101»). Но в слове 1100101 выбросятся два нуля, которые вовсе не являются первыми символами слова. В этом случае существующий алфавит надо расширить вспомогательными буквами, которых нет в начальном слове и которые появляются в ходе вычисления. По сути, это некоторые аналоги внутренней памяти (состояний машин Тьюринга). 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

Дальше мы «тащим» эту букву по слову вправо, стирая и отсчитывая символы

(стерли первый):

α 0 🡪 β

α 1 🡪 β

(стерли второй):

β 0 🡪 •

β 1 🡪 •

Однако если мы расположит эти строки в обычном порядке, а именно:

λ 0 🡪 α 0

λ 1 🡪 α 1

α 0 🡪 β

α 1 🡪 β

β 0 🡪 •

β 1 🡪 •

алгоритм будет работать совсем не так, как хотелось бы. В частности вся его деятельность будет сводиться к созданию бесконечно большого числа символов α поочередно со стиранием символов слова. Это связано с тем, что после выполнения каждой замены управление передается снова первой строке, а слово анализируется с левого символа. Поэтому чаще всего алгоритм пишется как бы «снизу-вверх», т.е. в самом начале ставятся строки, относящиеся к группе «окончание алгоритма», далее «тело программы» и в самом низу блок «инициализация», которая будет выполняться только однажды, а затем управление перейдет к более ранним строкам.

Правильный алгоритм выглядит следующим образом.

β 0 🡪 •

β 1 🡪 •

α 0 🡪 β

α 1 🡪 β

λ 0 🡪 α 0

λ 1 🡪 α 1

Ту же самую задачу можно решить и использовав всего один дополнительный символ:

α00 🡪•

α01 🡪•

α10 🡪•

α11 🡪•

α0 🡪•

α1 🡪•

λ 0 🡪 α 0

λ 1 🡪 α 1