НОРМАТИВНЫЙ КОНСПЕКТ ПО ДИСЦИПЛИНЕ «ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ В ГУМАНИТАРНОЙ СФЕРЕ»

Часть 4
«АЛГОРИТМИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ. НОРМАЛЬНЫЕ (МАРКОВСКИЕ) АЛГОРИТМЫ»

Введение. Теория нормальных алгоритмов (или алгорифмов, как называл их автор теории) была разработана советским математиком А. А. Марковым (1903-1979) в конце 1940-х и начале 1950-х годов XX века. Эти алгоритмы представляют собой некоторые правила по переработке слов в каком-либо алфавите, так что исходные данные и искомые результаты для алгоритмов являются словами в некотором алфавите.

Марковские подстановки. *Алфавитом* называется любое непустое конечное множество. Его элементы называются *буквами*, а любые последовательности букв – *словами* в данном алфавите. Для удобства рассуждений допускаются *пустые слова* (они не имеют в своем составе ни одной буквы). Пустое слово будем обозначать Λ (лямбда).

Если A и B суть два алфавита, причем каждый символ алфавита A есть также символ алфавита B, но не каждый символ алфавита B есть также символ алфавита A, то алфавит B называется *расширением* алфавита A.

Слова будем обозначать латинскими буквами (P, Q, R) или этими же буквами с индексами. Одно слово может быть составной частью другого слова. Тогда первое называется *подсловом* второго или вхождением во второе. Например, если А есть алфавит русских букв, то можем рассмотреть такие слова: P_1=*параграф*, P_2=*граф*, P_3=*ра*. Слово P_2 является подсловом слова P_1, а P_3 является подсловом P_1 и P_2, причем в P_1 оно входит дважды. Особый интерес для нас сейчас представляет *первое вхождение*.

Марковской подстановкой называется операция над словом R, задаваемая с помощью упорядоченной пары слов (P, Q) и состоящая в следующем: в заданном слове R находят первое вхождение слова P (если таковое имеется) и, не изменяя остальных частей слова R, заменяют в нем это вхождение словом Q. Полученное слово называется *результатом применения* марковской подстановки $P \to Q$ к слову R. Если же первого вхождения слова P в слово R нет (и, следовательно, вообще нет ни одного вхождения P в R), то считается, что марковская подстановка $P \to Q$ *неприменима* к слову R.

Частными случаями марковских подстановок являются подстановки с пустыми словами:

$\to Q$ (если пустое слово записать явно, то $\Lambda \to Q$);

$P \to$ (если пустое слово записать явно, то $P \to \Lambda$);

$\to$ (если пустое слово записать явно, то $\Lambda \to \Lambda$).

Запись $P \to Q$, использующуюся для обозначения марковской подстановки *(P, Q)*, называют *формулой подстановки (P, Q)*. Некоторые подстановки *(P, Q)* будем называть *заключительными*. Для обозначения таких подстановок будем использовать запись $P \to .Q$, называя ее *формулой заключительной подстановки*. Слово P называется *левой частью*, а слово Q называется *правой частью* в формуле подстановки $P \to .Q$ или $P \to Q$.

Нормальные алгоритмы и их применение к словам. Упорядоченный конечный список формул подстановок в алфавите A называется *схемой* (или *записью*) *нормального алгоритма* в алфавите A. Такая схема определяет (детерминирует) алгоритм преобразования слов, называемый *нормальным алгоритмом Маркова*. Дадим его точное определение.

Нормальным алгоритмом (алгоритмом Маркова) в алфавите А называется следующее правило построения последовательности $V_1, V_2, V_3, \ldots$ слов в алфавите A, исходя из данного слова V в этом алфавите:

1. В качестве начального слова последовательности V_1 берется слово V.

2. Среди формул подстановок, составляющих схему нормального алгоритма, находится первая, левая часть которой входит в слово V_1. Если такой формулы нет, то построение последовательности окончено.

3. Если формула подстановки, названная в пункте 2, существует, то ее правая часть подставляется в слово V_1 вместо первого вхождения ее левой части в слове V_1. Получившееся в результате слово V_2 является новым членом последовательности. Если примененная к слову V_1 формула подстановки не была завершающей, то процесс построения последовательности продолжается, в противном случае – завершается.

4. Вообще, пусть для некоторого $i > 0$ слово V_i построено, а процесс построения рассматриваемой последовательности еще не завершился. Если при этом в схеме нормального алгоритма нет формул, левые части которых входили бы в слово V_i, то процесс построения последовательности считается завершившимся, а слово V_i признается последним членом этой последовательности. Если же в схеме имеются формулы подстановок с левыми частями, входящими в V_i, то в качестве слова V_{i+1} берется результат марковской подстановки правой части первой из таких формул вместо первого вхождения ее левой части в слово V_i. Процесс построения последовательности считается завершившимся, если на данном шаге была применена формула заключительной подстановки, и продолжающимся – в противном случае.

Если процесс построения упомянутой последовательности обрывается, то говорят, что рассматриваемый нормальный алгоритм *применим* к слову V. Последний член последовательности W называется *результатом применения* нормального алгоритма к слову V. Говорят, что нормальный алгоритм *перерабатывает* слово V в слово W.

Мы определили понятие нормального алгоритма *в* алфавите A. Если же алгоритм задан в некотором расширении алфавита A, то говорят, что он есть нормальный алгоритм *над* A.

Приведем примеры нормальных алгоритмов.

Пусть $A = \{a, b\}$ – алфавит. Рассмотрим следующую схему нормального алгоритма Σ в алфавите A:

Σ: $a\rightarrow$. (R_1)

$b\rightarrow b$ (R_2)

Нетрудно понять, как работает определяемый этой схемой нормальный алгоритм. Всякое слово V в алфавите A, содержащее хотя бы одно вхождение буквы a, он перерабатывает в слово, получающееся из V вычеркиванием в нем самого левого (первого) вхождения буквы a (применяется формула подстановки R_1 и работа завершается, поскольку эта формула является завершающей). Пустое слово он перерабатывает в пустое слово (ни одна формула подстановки не применяется). Алгоритм Σ не применим к таким словам, которые содержат только букву b (неограниченное количество раз применяется формула подстановки R_2). Например:

$\Sigma(a\underline{a}bab)\rightarrow R_1\rightarrow abab$

$\Sigma(\underline{a}b)\rightarrow R_1\rightarrow b$

$\Sigma(\underline{a}a)\rightarrow R_1\rightarrow a$

$\Sigma(bb\underline{a}b)\rightarrow R_1\rightarrow bbb$

$\Sigma(b\underline{a}ba)\rightarrow R_1\rightarrow bba$

$\Sigma()\rightarrow$

Пусть $A = \{a_0, a_1, ..., a_n\}$ – алфавит. Рассмотрим схему нормального (марковского) алгоритма Ω:

Ω: $a_0\rightarrow$ (R_1)

$a_1\rightarrow$ (R_2)

……

$a_n\rightarrow$ (R_{n+1})

$\rightarrow$. (R_{n+2})

Эта схема определяет нормальный алгоритм Ω, перерабатывающий всякое слово в алфавите А в пустое слово. Например:

$\Omega(a_1a_2a_1a_3\underline{a_0})\rightarrow R_1\rightarrow \underline{a_1}a_2a_1a_3\rightarrow R_2\rightarrow a_2\underline{a_1}a_3\rightarrow R_2\rightarrow \underline{a_2}a_3\rightarrow R_3\rightarrow \underline{a_3}\rightarrow R_3\rightarrow \underline{\Lambda}\rightarrow$
$\rightarrow R_{n+2}\rightarrow \Lambda$

$\Omega(\underline{a_0}a_2a_2a_1a_3a_1)\rightarrow R_1\rightarrow a_2a_2\underline{a_1}a_3a_1\rightarrow R_2\rightarrow a_2a_2a_3\underline{a_1}\rightarrow R_2\rightarrow \underline{a_2}a_2a_3\rightarrow R_3\rightarrow$
$\rightarrow \underline{a_2}a_3\rightarrow R_3\rightarrow \underline{a_3}\rightarrow R_4\rightarrow \underline{\Lambda}\rightarrow R_{n+2}\rightarrow \Lambda$

$\Omega()\rightarrow R_{n+2}\rightarrow \Lambda$

Нормально вычислимые функции и принцип нормализации Маркова. Как мы уже отмечали выше, каждая *массовая задача* может быть переформулирована как описание того, какие слова некоторого алфавита могут быть ее входными данными и какие слова им соответствуют в качестве решений в каждом частном случае. При этом каждому входному слову соответствует единственное выходное слово, хотя одному и тому же выходному слову могут (в принципе) соответствовать разные входные слова. Можно говорить, что условие каждой массовой задачи представляет собой описание некоторой функции, значениями аргумента которой являются слова, и значениями функции тоже являются слова. Такие функции иногда называют *алфавитными функциями*. Тогда можно говорить, что условие каждой массовой задачи представляет собой описание некоторой алфавитной функции.

Функция (алфавитная функция), заданная на некотором множестве слов алфавита A, называется *нормально вычислимой*, если найдется нормальный алгоритм над алфавитом A, перерабатывающий каждое слово V в алфавите A из области определения данной функции в слово $f(V)$.

Создатель теории нормальных алгоритмов советский математик А. А. Марков выдвинул *гипотезу*, названую позже ***принципом нормализации Маркова***: *для нахождения значений функции (алфавитной функции) тогда и только тогда существует какой-нибудь алгоритм, когда функция нормально вычислима*. Учитывая тот факт, что под массовой задачей мы понимаем описание некоторой алфавитной функции, можно сформулировать принцип нормализации Маркова и так: *вообще алгоритм решения массовой задачи существует тогда и только тогда, когда существует нормальный (марковский) алгоритм, решающий эту массовую задачу*.

Сформулированный сейчас принцип, как и тезис Тьюринга, носит внематематический характер и не может быть строго доказан. Он выдвинут на основании математического и практического опыта.

Эквивалентность различных теорий алгоритмов. Можно доказать справедливость следующих ниже утверждений.

Утверждение 1: *Если для решения некоторой задачи существует нормальный (марковский) алгоритм, то существует и машина Тьюринга, решающая эту задачу.* Действительно, в соответствии с принципом нормализации Маркова, из существования нормального (марковского) алгоритма, решающего задачу, следует существование вообще алгоритма, ее решающего. Теперь, согласно тезису Тьюринга, следует признать, что существует и машина Тьюринга, решающая эту задачу.

Утверждение 2: *Если существует машина Тьюринга, решающая некоторую задачу, то существует и нормальный алгоритм для ее решения.* Действительно, в соответствии с тезисом Тьюринга, из существования машины Тьюринга, решающей данную задачу, следует существование вообще алгоритма, ее решающего. Теперь, согласно с принципом нормализации Маркова, следует признать, что существует и нормальный (марковский) алгоритм, решающий эту задачу.

Утверждения 1 и 2 можно объединить.

Утверждение 3: *Для решения некоторой задачи существует нормальный (марковский) алгоритм тогда и только тогда, когда существует машина Тьюринга, решающая эту задачу.*

Из утверждения 3 следует и важный отрицательный результат.

Утверждение 4: *Для каждой задачи не имеется решающего ее нормального (марковского) алгоритма тогда и только тогда, когда не существует машины Тьюринга, ее решающей.*

Все четыре утверждения, сформулированные выше, можно обобщить следующим образом: *классы задач, решаемых машинами Тьюринга и нормальными (марковскими) алгоритмами совпадают*. Это означает, что две названные теории алгоритмов эквивалентны в смысле широты охвата массовых задач.