Белоусов Алексей Иванович

кандидат физико-математических наук, доцент

МГТУ им. Н.Э. Баумана, г. Москва

[al\_belous@bk.ru](mailto:al_belous@bk.ru)

**О методике изложения некоторых разделов теории алгоритмов: проблема применимости для нормальных алгорифмов Маркова[[1]](#endnote-1)**

**Аннотация**

*Актуальность рассматриваемой методической задачи обусловлена тем, что в учебных планах студентов-программистов существенное место занимает теория алгоритмов, и требуется отработка методики строгого и в то же время доступного студентам, не специализирующимся собственно в математике, изложения основ этой теории.*

*В статье предлагается методика изложения ключевых положений теории алгоритмов на примере нормальных алгорифмов Маркова. Подробно доказывается алгоритмическая неразрешимость проблемы применимости (или проблемы останова) для нормальных алгорифмов. Кратко излагаются основные результаты, необходимые для понимания основной теоремы. Содержание статьи будет полезно студентам программистских специальностей, а также преподавателям курсов математической логики и теории алгоритмов, курсов проектирования алгоритмов.*

***Ключевые слова:*** *алгоритм, нормальный алгорифм Маркова, алгоритмически неразрешимые проблемы.*

**Введение**

В учебные планы по специальностям, связанным так или иначе с программными технологиями, обязательно входят курсы, посвященные разработке различных алгоритмов. Обстоятельный обзор таких алгоритмов содержится в недавно вышедшей в русском переводе книги [1].

Это, несмотря на возникающие собственно математические вопросы, - практические аспекты теории алгоритмов как таковой. Фундаментом же является классическая теория алгоритмов, представленная многочисленными точными моделями алгоритма (машины Тьюринга, рекурсивные функции, канонические системы Поста, нормальные алгорифмы Маркова). Эта теория должна лежать в основе всех дальнейших разработок, и главный результат здесь – анализ алгоритмических массовых проблем и доказательство неразрешимости некоторых из них. Любой программист должен четко знать границы применения алгоритмических методов, понимать суть алгоритмической неразрешимости, знать, какие алгоритмически неразрешимые проблемы возникают в сфере программирования (например, проблема функциональной эквивалентности программ [2, 3]).

В данной статье рассматривается методика изложения ключевых положений теории алгоритмов на примере нормальных алгорифмов Маркова. Эта модель алгоритма заслуживает особого внимания, так как лежит в основе некоторых языков программирования (например, языка РЕФАЛ [4, 5, 6]), предназначенных для обработки символьной информации, для разработки трансляторов в том числе. Изложение ведется на доступном для понимания студентов-нематематиков (программистов, например) уровне с опорой на основополагающие источники, книгу [7] прежде всего. Следует иметь в виду и очень хорошую книгу Б.А. Кушнера [8]. Мы стараемся излагать все доказательства максимально просто и прозрачно, не нанося существенного ущерба строгости изложения. Собственно, в этом и состоит суть предлагаемой методики: достаточно просто и в то же время строго изложить часто весьма непростые результаты и доказательства. Методика ориентирована на студентов-нематематиков, но тем, которым в своей профессиональной деятельности придется математику использовать.

Предваряя дальнейшее изложение основных положений теории алгоритмов, необходимо дать перечень главных разделов этой теории. Эти разделы таковы:

1. Определение основной модели (в данном случае, нормального алгорифма) и формулировка основной гипотезы, состоящей в том, что предлагаемая модель представляет, в известном смысле, любой алгоритм.

2. Понятие эквивалентных алгоритмов.

3. Теоремы сочетания (основные конструкции из алгоритмов, как то: последовательное применение, независимое применение, условный переход и цикл), согласно которым любое сочетание алгоритмов может быть в конечном счете сведено к основной модели.

4. Универсальный алгоритм (по сути, универсальный интерпретатор в классе программ, определяемых основной моделью).

5. Разрешимые и перечислимые множества.

6. Алгоритмически неразрешимые проблемы.

Материал статьи основан на курсе «Логика и теория алгоритмов», который автор уже многие годы читает на программистских специальностях МГТУ им. Н.Э. Баумана. Студенты подготовлены к освоению курса курсом дискретной математики, опирающимся на базовый учебник [9], в котором для понимания теории алгоритмов особенно важны элементы теории формальных языков.

Цель статьи состоит прежде всего в том, чтобы очертить рамки предлагаемой методики. Поэтому мы ограничиваемся минимумом примеров. Подробному обсуждению примеров и приемов разработки нормальных алгорифмов для решения конкретных задач будет посвящена отдельная статья.

**Концепция алгоритма**

Прежде чем определять алгоритм математически точно, нужно кратко охарактеризовать саму концепцию алгоритма, или то, что называют ***алгоритмом в интуитивном смысле слова***. Здесь методически целесообразно ввести основные обозначения и понятия, используемые в дальнейшем.

Алгоритм можно изначально рассматривать как «черный ящик», осуществляющий преобразование входных данных из некоторого множества  в результаты, составляющие множество . Запишем условно это так

 (1)

и будем говорить: ***алгоритм  типа*** .

Входные данные и результаты алгоритма понимаются как ***конструктивные объекты***. Очень коротко: конструктивный объект есть (неформально) некий объект, который может быть построен за конечное число шагов по определенным правилам («правилам сборки») из неких базовых элементов (наподобие моделей в детском конструкторе).

Всюду в дальнейшем мы будем рассматривать частный случай конструктивного объекта, а именно, ***слово в конечном алфавите*** [9]. Здесь базовые элементы – буквы алфавита, а единственное правило сборки – известная операция соединения слов. Это частный случай, но важнейший. Можно выдвинуть гипотезу, согласно которой любой конструктивный объект может быть представлен (более или менее сложно) в виде слова в конечном алфавите.

Итак, мы можем уточнить идею алгоритма, понимаемого как преобразователь слов в конечных алфавитах.

Не всякий преобразователь слов, однако, будет алгоритмом. Алгоритм характеризуется следующими тремя признаками:

- ***признак детерминированности***: алгоритм есть предписание, определяющее пошаговый процесс преобразования входных слов, в котором каждый шаг определен однозначно, а также однозначно определены условия прекращения процесса;

- ***признак массовости***: алгоритм есть такое предписание, которое единообразно применяется в широком классе входных данных независимо от их индивидуальных особенностей (например, алгоритм вычисления корней квадратного уравнения един для всех уравнений и никак – сам по себе – не зависит от коэффициентов конкретного уравнения);

- ***признак результативности***: алгоритм есть такое предписание, в силу которого процесс преобразования входных данных (слов) завершается для достаточно широкого множества этих данных выдачей результата (также в виде слова в некотором конечном алфавите).

Из перечисленных признаков первый есть важнейший. Процесс работы алгоритма не допускает никакого выбора шага. Здесь методически уместно провести сравнение алгоритмического процесса преобразования входных данных с игровым. Например, шахматная партия тоже есть некий процесс преобразования конструктивных объектов (позиций на доске, известным образом представляемых вербально), но на каждом шаге возможен выбор в рамках правил игры, то есть очередной шаг (ход в игре) не определен однозначно. В процессе работы алгоритма такая ситуация невозможна. В курсе математической логики, читаемом вслед за теорией алгоритмов, студентам объясняется, что подобный же игровой процесс есть формальное доказательство.

Итак, мы можем уточнить рамочную концепцию алгоритма следующим образом.

Имеется входной алфавит , и множество входных данных  есть некоторое множество слов в этом алфавите, то есть язык , где  есть множество *всех* слов в указанном алфавите. Задан выходной алфавит , и множество результатов  есть некоторый язык в выходном алфавите, то есть . Тогда алгоритм  типа , представленный формулой (1), может быть описан в виде

 (2)

О таком алгоритме будем говорить, как об алгоритме типа , указывая лишь входной и выходной алфавиты. Без существенного ограничения общности можно допустить совпадение входного и выходного алфавитов, а также считать множества входных и выходных слов множествами всех слов в своих алфавитах.

Необходимо заметить, что результат, вообще говоря, определен не для каждого входного слова  (и точка под стрелкой в формулах (1) и (2) есть условное обозначение этого обстоятельства). Если результат работы алгоритма  со слово определен, то будем говорить, что *алгоритм примен****и****м к слову* и писать , обозначая сам результат через . Тем самым выходное слово . В противном случае пишем и говорим, что *алгоритм  не применим к слову* . Если , то алгоритм  будем называть ***полным алгоритмом типа***  и писать

 (3)

(без точки под стрелкой).

Теперь введем понятие вербальной (словарной) функции и вычислимой (в интуитивном смысле слова) вербальной функции.

Вербальная (словарная) функция типа есть любое, в общем случае частичное, отображение вида:

 (4)

(точка под стрелкой указывает на частичность).

Здесь разумно привести некоторые примеры.

**Примеры**. 1) Функция удвоения: для любого слова  значение функции  . Эта функция определена для всех слов во входном алфавите, который совпадает с выходным.

2) Характеристическая функция определенного множества слов в заданном входном алфавите :

 .

Здесь мы полагаем, что выходной алфавит .

3) В качестве частичной вербальной функции можно рассмотреть следующую. Определим множества  как множества двоичных кодов натуральных чисел и положим, что  тогда и только тогда, когда . Очевидно, что функция определена только для полных квадратов.

Для вербальной функции  обозначим  область ее определения.

Следующее понятие является в этом разделе основным.

Вербальная функция типа называется ***вычислимой в интуитивном смысле слова***, если существует алгоритм  такой, что для любого слова  имеет место следующее:  тогда и только тогда, когда , и в этом случае .

То есть указанный алгоритм результативен для всех слов из области определения функции, и только для них, и результат его работы с исходным словом совпадает со значением функции на этом слове. Важно подчеркнуть, что в этом определении не фигурирует какая-либо точная математическая модель алгоритма, и это понятие рассматривается в рамках очерченной выше содержательной концепции. И еще важно обратить внимание на то, что в теории алгоритмов само понятие функции трактуется в ином ключе, чем в классической математике: в теории алгоритмов функция есть не только некоторое соответствие из одного множества в другое, но еще и процесс ее вычисления.

Перейдем теперь к рассмотрению точной модели алгоритма в виде нормального алгорифма Маркова. Заметим сразу, что школе А.А. Маркова принят именно термин «алгорифм», и сочетание «нормальный алгоритм» считается недопустимым.

**Нормальный алгорифм Маркова**

Прежде всего необходимо заметить, что в нашем курсе (и, соответственно, в этой статье) нормальный алгорифм рассматривается исключительно как точная модель алгоритма, но не как основа дальнейшего построения математики в рамках конструктивного направления [8]. Можно сказать, что мы строим некоторую теоретико-множественную модель нормального алгорифма, определяя это понятие аналогично похожим понятиям в теории формальных языков (например, понятиям грамматики или автомата [9]).

Начнем с некоторых предварительных определений.

Пусть  некоторый алфавит, а  и  слова в этом алфавите.

Говорят, что ***слово***  ***входит в слово*** , если существуют такие слова  (возможно пустые), что . Например, слова «ход» и «вход» входят в слово «входит». Соответствующее отношение на множестве слов будем обозначать , используя тем самым запись . Очевидны следующие свойства этого отношения: каждое слово входит само в себя (отношение рефлексивно), пустое слово (обозначаемое далее буквой ) входит в любое слово, и если , а , то . Тем самым отношение является предпорядком.

Если , то возникает упорядоченная тройка слов , называемая ***вхождением слова***  ***в слово*** . В теории нормальных алгорифмов вхождение записывают так: , где «звездочка» является «метасимволом», не принадлежащем алфавиту . При этом пустое слово часто не пишется вообще, то есть, например, вхождение слова «вход» в слово «входит» следует записывать так: \*вход\*ит. Слово  называют ***левым***, а слово  ***правым*** ***крылом*** вхождения.

Одно и то же слово может входить в другое слово несколько раз. Например, . Мы имеем тут два вхождения: и . Среди всех вхождений слова в слово выделяют первое, или главное. Это то, которое имеет наименьшую длину левого крыла, то есть самое левое. Из показанных выше двух вхождений первое как раз будет главным.

Дадим теперь определение формулы подстановки.

***Формула подстановки в алфавите***  есть, по определению, упорядоченная пара слов в этом алфавите, записываемая в виде , где «стрелка» есть опять-таки метасимвол, отделяющий ***левую часть формулы*** (слово ) от ***правой*** (слова ). Часто дальше будем говорить просто «формула» вместо «формула подстановки».

Пусть теперь  есть формула , а  . Говорят, что ***формула  применима к слову***  (или является ***подходящей для слова*** ), если ее левая часть входит в слово: . Пусть  - первое вхождение  в . ***Результат применения*** формулы  к слову  есть слово . Это схематически можно показать так:



Говорят в этом случае, что ***словополучено в результате замены первого вхождения левой части формулы  в слово  на правую***.

Такую операцию над исходным словом называют также ***нормальной подстановкой***.

**Основное определение**: ***нормальный алгорифм (НА) в алфавите *** есть упорядоченная тройка , где  - ***схема НА***, представляющая собой упорядоченный набор (кортеж, конечная последовательность) формул подстановки в алфавите , в котором отмечены некоторые формулы, называемые ***заключительными*** и образующие частичный кортеж (подпоследовательность) . Заключительная формула в схеме отмечается точкой после стрелки.

Будем далее, если контекст это позволяет, пользоваться аббревиатурой НА, или говорить просто «алгорифм», имея в виду именно нормальный алгорифм Маркова.

Пример схемы:

 (5)

Здесь предполагается, что это схема НА в алфавите , причем третья сверху формула является заключительной, а левая часть самой нижней формулы есть пустое слово.

Из приведенного примера можно понять синтаксис записи схем нормальных алгорифмов: формулы схемы пишутся в столбик и упорядочиваются сверху вниз, схема окаймляется левой фигурной скобкой и предваряется обозначением НА с двоеточием.

В общем случае схема НА выглядит так:



Квадратные скобки означают необязательность вхождения точки.

Так строится «статическая» часть определения нормального алгорифма. Теперь нужно определить «динамику», то есть точно определить процесс преобразования слов, задаваемый нормальным алгорифмом. Есть смысл сначала дать содержательное описание этого процесса, называемого ***процессом работы НА с исходным словом***.

Пусть задан НА и входное слово (в каком-то алфавите).

Схема НА просматривается сверху вниз в поисках первой применимой ко входному слову формулы. Если таковой не оказывается, то процесс заканчивается и само входное слово считается его результатом (то есть НА вычисляет тогда тождественную функцию). Иначе, найденная формула применяется, как описано выше (операция нормальной подстановки). Если эта формула оказалась простой (не заключительной), то с полученным в результате ее применения словом поступают так же, как с исходным. Если же она оказалась заключительной, то после ее применения процесс заканчивается и слово, полученное в результате ее применения, считается результатом. Таким образом, процесс продолжается до тех пор, пока не произойдет одно из двух событий: 1) ***естественный обрыв***, когда в схеме нет ни одной применимой к текущему слову формулы; в этом случае говорят, что слово не поддается данному НА (или его схеме), и оно считается результатом; 2) к текущему слову была применена заключительная формула, и тогда результатом считается слово, полученное применением этой формулы к текущему. Может оказаться, что ни одном из шагов не выполняется условие останова (прекращения процесса). В этом случае результат не определен, и говорят, что данный НА не применим ко входному слову.

Чтобы дать точное (формальное) определение процесса работы, нужно ввести некоторые обозначения и определения.

1) Будем писать  и говорить, что ***НА  непосредственно просто переводит слово  в слово ***, если , где  - первая (самая верхняя) формула в схеме НА, оказавшаяся простой.

2) Будем писать  и говорить, что ***НА  непосредственно заключительно переводит слово  в слово ***, если , где  - первая (самая верхняя) формула в схеме НА, оказавшаяся заключительной.

3) Будем писать и говорить, что ***НА  просто переводит слово  в слово ***(без слова «непосредственно»!), если существует последовательность слов  в которой для каждого  выполняется .

4) Будем писать  и говорить, что ***НА  заключительно переводит слово  в слово ***, если  или существует такое слово , что и .

Можно сказать, что в пп. (1) и (2) дано формальное определение ***шага процесса работы*** НА со словом, а в пп. (3) и (4) определяется последовательность таких шагов (в том числе и один шаг).

5) Будем писать , когда слово  ***не поддается*** НА .

Теперь мы можем дать строгое определение процесса работы НА со словом.

***Процесс работы НА  со словом *** есть конечная или бесконечная последовательность слов где , причем для каждого   или , если слово  определено в последовательности. Это слово, как и все последующие, считается не определенным, если , то есть последнее слово получено применением заключительной формулы, или , то есть последнее слово не поддается НА и имеет место естественный обрыв процесса работы.

Если процесс работы НА  со словом  конечен, то его последнее слово обозначается  и называется ***результатом работы НА  со словом*** . В этом случае говорят, что ***НА  применим к слову *** и пишут ******. В противном случае говорят, что ***НА***  ***не применим к слову *** и пишут .

Чтобы завершить описание основной модели алгоритма (НА), необходимо определить понятие вербальной функции, вычислимой по Маркову и сформулировать основную гипотезу, утверждающую универсальность предложенной модели.

Вербальная функция типа называется ***вычислимой по Маркову***, если может быть построен НА  такой, что для любого слова  имеет место следующее:  тогда и только тогда, когда , и в этом случае .

Необходимо обратить внимание, что в этом определении вместо слова «существует» (в определении интуитивной вычислимости) стоит «можно построить». Это очень важно: чтобы доказать вычислимость по Маркову, нужно предъявить схему соответствующего НА и доказать, что этот НА вычисляет заданную функцию, то есть доказательство должно быть конструктивным. Конечно, в любом случае, при разработке алгоритма необходимо его точное описание и доказательство корректности, но всё равно тогда возникает некое уточнение интуитивного понимания алгоритма, например, в виде псевдокода или даже программы на языке высокого уровня. То есть всякий раз, когда от интуитивной концепции вычислимости мы переходим к той или иной точной модели, доказательство вычислимости функции обязано быть конструктивным. Это, повторим, важнейший аспект в методике преподавания теории алгоритмов студентам-программистам.

Основная гипотеза теории алгоритмов в версии теории нормальных алгорифмов Маркова называется ***принципом нормализации*** и состоит в следующем:

*Всякая вербальная функция, вычислимая в интуитивном смысле слова, вычислима по Маркову*.

Подчеркнем, что это именно гипотеза, а не теорема, так как в формулировке фигурирует не определенное математически точно понятие функции, вычислимой в интуитивном смысле слова.

На этом описание основной модели закончено.

Рассмотрим некоторые примеры нормальных алгорифмов, важные для дальнейшего изложения.

**Примеры**

1) *Тождественный НА* , задаваемый схемой



любое слово в заданном алфавите  перерабатывает в это же слово за один шаг, то есть

.

2) *Нигде не применимый* НА  , задаваемый схемой



Очевидно, что для любого входного слова процесс работы такого НА с ним будет бесконечным, то есть 

3*) НА левого присоединения фиксированного слова.*

НА , заданный схемой

,

где  - фиксированное слово в алфавите , вычисляет функцию левого присоединения к любому слову в алфавите  данного слова  , то есть



4) *НА правого присоединения фиксированного слова.*

НА , заданный схемой

 (6)

в алфавите  , где , вычисляет функцию правого присоединения данного фиксированного слова  в алфавите к любому слову в алфавите , т.е.



Действительно, к произвольному слову  применима только самая нижняя формула в схеме алгорифма . После ее применения оказывается применимой одна из формул верхней строки, если слово  не пусто. Эти формулы применяются до тех пор, пока «решетка» (#) не окажется последней буквой очередного слова, выводимого из исходного слова . Тогда она посредством применения средней формулы схемы заключительно заменяется на слово .

Заметим, что в этой схеме фигурирует параметр , пробегающий алфавит , и, тем самым, верхняя строка схемы представляет конечное множество формул, число которых равно числу букв алфавита . В схему НА можно вводить параметры, но необходимо, чтобы он пробегал некоторое конечное множество. Схема НА , как нетрудно видеть, обобщает схему (5).

5) *НА удвоения*

Рассмотрим НА *Double*, задаваемый схемой в алфавите *V*1 = *V*∪{α, β}, причем α, β ∉ *V*:

 (7)

Можно показать, что , т.е. этот алгорифм удваивает любое входное слово в алфавите *V*.

Мы в рамках этой работы, ради экономии объема, не анализируем подробно работу приведенных в примерах нормальных алгорифмов. Подробности такого рода будут рассмотрены в отдельной публикации, специально посвященной проблемам технике разработки схем нормальных алгорифмов.

Можно заметить также, что если вместо формулы  поставить формулу , то копия будет инвертирована, т.е. получится слово .

Если схему (7) модифицировать следующим образом

 (8)

то получим алгорифм удвоения через разделитель, то есть .

**Эквивалентность нормальных алгорифмов. Теорема о переводе**

НА  и  типа  над алфавитом  называют ***вполне*** ***эквивалентными относительно алфавита*** , если для всякого слова  имеет место  и если выполняется , то .

В теории нормальных алгорифмов используется выражение ,

называемое ***условным равенством***, смысл которого состоит в том, что левая часть его определена тогда и только тогда, когда определена правая, и в этом случае они обозначают одно и то же слово. Это не что иное, как записанное для слова  требование полной эквивалентности НА  и , т.е. НА  и  вполне эквивалентны относительно алфавита тогда и только тогда, когда .

Рассмотрим теперь некоторые преобразования схем НА, которые дают НА, вполне эквивалентный исходному относительно заданного алфавита.

1) *Замыкание нормального алгорифма*

Так называется НА, схема которого получается из схемы исходного НА путем добавления после всех формул заключительной формулы  . Замыкание НА  обозначается . Легко показать, что эти НА вполне эквивалентны относительно алфавита, в котором задан исходный НА.

Переход к замыканию позволяет впредь при обсуждении применимости НА к слову не рассматривать ситуацию естественного обрыва, то есть считать, что применимость НА к слову означает, что на последнем шаге процесса работы применяется заключительная формула.

2) *Естественное и формальное распространение*

Для НА  в алфавите  можно той же самой схемой задать НА  в более широком алфавите (т.е. подавать на его вход слова в некотором расширении алфавита ). Тогда очевидно, что для любого слова  в алфавите  имеет место  Таким образом, НА  и  вполне эквивалентны относительно алфавита . НА  называют при этом ***естественным распространением*** исходного на более широкий алфавит.

***Формальное распространение***  НА  на более широкий алфавит определяется тем, что к схеме , в начале ее («сверху»), приписываются формулы  для всех букв  из . Тогда, как нетрудно видеть, , но, в отличие от естественного распространения, имеет место , то есть НА не применим ни к одному из слов, не являющимся словом в алфавите .

Следующее, более сложное, преобразование НА связано с так называемой теоремой о переводе, которая ограничивает число дополнительных букв при построении НА, вычисляющего некоторую функцию типа  для некоторого произвольно заданного алфавита (например, в рассмотренных выше НА удвоения и инвертирования потребовались две дополнительные буквы). Оказывается, что именно двумя буквами можно ограничить расширение алфавита в любом случае. При этом доказывается также, что существует такая вербальная функция указанного типа, которую невозможно вычислить посредством НА, не расширяя алфавит.

Чтобы сформулировать соответствующий результат, введем понятие перевода в двухбуквенный алфавит.

Пусть  - исходный алфавит, а алфавит  состоит из двух букв. Для простоты будем считать, что эти два алфавита не пересекаются (хотя это требование не является обязательным).

Тогда назовем переводом буквы  в алфавит  слово . Перевод непустого слова  есть, по определению, соединение переводов его букв:

.

Перевод пустого слова естественно определить как пустое слово. Например, если , и , то . Здесь предполагается, что буквы исходного алфавита нумеруются, как 1-я, 2-я и 3-я.

Часто в качестве двухбуквенного алфавита, в который делается перевод, берется алфавит , в котором «нуль» и «единица» фигурируют только как буквы, но не как числа. Понятие натурального числа появится позже.

Сформулируем теперь без доказательства теорему, которая в данном контексте называется теоремой о переводе, хотя она на самом деле является одним из следствий более общего утверждения, принадлежащего А.А. Маркову и названной им теоремой о переводе [7].

**Теорема 1**. Каков бы ни был нормальный алгорифм  типа  над алфавитом , может быть построен нормальный алгорифм  в алфавите  , вполне эквивалентный относительно алфавита , то есть для любого слова  имеет место условное равенство .

Идея доказательства состоит в том, что все буквы из исходного расширения алфавита заменяются их переводами в алфавит , соответственно преобразуется схема исходного НА и все слова его процесса работы с исходным словом, после чего доказывается полная эквивалентность полученного НА и исходного.

**Теоремы сочетания**

Алгоритм (в интуитивном смысле) может быть «запрограммирован» не только в виде нормального алгорифма – выписыванием его схемы, но через определенные правила комбинации (сочетания) уже построенных нормальных алгорифмов.

Например, следующее предписание определяет точный алгоритм переработки слов в заданном алфавите:

1) применить к исходному слову  нормальный алгорифм ;

2)если слово  определено, то к нему применить нормальный алгорифм ;

3) результатом считать слово , если оно определено.

Это предписание не является априори нормальным алгорифмом, поскольку оно не задано единой схемой такого алгорифма, а оперирует нормальными алгорифмами, словно некоторыми «блоками». Оно определяет один из способов сочетания нормальных алгорифмов, называемый ***композицией*** (иногда ***последовательной композицией***).

Оказывается, что композицию и другие сочетания нормальных алгорифмов, которые будут рассмотрены ниже, можно «запрограммировать» в виде схем нормальных алгорифмов. Теоремы о возможности построения таких схем называю

тся ***теоремами сочетания***.

Мы рассмотрим в этом разделе только два способа сочетания из четырех: композицию и разветвление, так как они необходимы нам для доказательства основной теоремы. Объединение (независимое, «параллельное», применение нормальных алгорифмов ко входному слову) и повторение («моделирующее» обычный программный цикл) обсудим в отдельных публикациях. Теоремы сочетания здесь формулируются без доказательства. Для простоты считаем, что все сочетаемые нормальные алгорифмы заданы в одном и том же алфавите .

Сформулируем теорему композиции.

**Теорема 2** (*о композиции нормальных алгорифмов*). Каковы бы ни были нормальные алгорифмы  и  в алфавите , может быть построен такой нормальный алгорифм  над алфавитом , что для любого слова  имеет место условное равенство .

Нам будет удобно дальше обозначать НА  как , то есть .

Способ сочетания нормальных алгорифмов, называемый ***разветвлением***, задается так:

1) к исходному слову  применить алгорифм ;

2) если слово  существует и равно пустому слову, то к слову  применить алгорифм  и слово  считать общим результатом;

3) если слово  существует и не равно пустому слову, то к слову  применить алгорифм  и считать слово  общим результатом;

4) если слово  не определено, то не определен и общий результат.

Теорема о «программировании» разветвления нормальных алгорифмов в виде схемы некоторого нормального алгорифма называется теоремой о разветвлении и формулируется следующим образом.

**Теорема** **3** (*о разветвлении нормальных алгорифмов*). Каковы бы ни были нормальные алгорифмы  и  в алфавите , может быть построен алгорифм  над алфавитом  такой, что для любого слова  выполняется следующее: 1) если  то ; 2) если , то  при  и  при .

Нормальный алгорифм , который может быть построен согласно теореме о разветвлении, будем обозначать и называть -***разветвлением алгорифмов***  и . Коротко этот алгорифм можно задать такой записью:

 .

В программистских терминах -разветвление можно описать так:

**if**  **then** **else** .

Сформулированные здесь теоремы о композиции и разветвлении будут использованы при доказательстве основного результата о неразрешимости проблемы применимости для нормальных алгорифмов.

**Универсальный нормальный алгорифм**

Приступая к объяснению конструкции универсального НА, необходимо заметить следующее.

Очевидно, что наши действия по применению НА, заданного своей схемой, являются шагами некоторого алгоритма (в интуитивном смысле слова). Следовательно, согласно принципу нормализации, их можно задать некоторым нормальным алгорифмом. Он должен как-то воспринимать на входе любой НА и работать за него. Но для того, чтобы одни алгорифмы использовать в качестве исходных данных для других, необходимо алгорифм как данное, *превратить в конструктивный объект*, то есть представить в виде слова в некотором алфавите. Существует два основных способа такой ***конструктивизации*** нормального алгорифма: посредством ***изображения*** и посредством ***записи***.

Пусть НА  является алгорифмом в алфавите . ***Изображение НА***  есть слово в алфавите , где которое строится по схеме  следующим образом: формулы выписываются в порядке следования их в схеме, причем буква служит разделителем между формулами, каждая стрелка заменяется буквой , а каждая точка – буквой .

Например, алгорифм  левого присоединения слова , заданный схемой

,

будет иметь изображение: .

Алгорифм  правого присоединения непустого слова , заданный схемой в алфавите , где , имеющей вид (в сокращенной записи)

 ,

будет иметь такое изображение: , где предполагается, что .

В общем случае алгорифм, задаваемый схемой 



в алфавите , будет иметь изображение в виде слова .

Вместо изображения нормального алгорифма часто используют ***запись нормального алгорифма*** – слово, определяемое как перевод изображения в алфавит . При построении записи НА в алфавите принято считать, что буквы , фигурирующие в изображении, получают номера  соответственно.

Так, запись алгорифма левого присоединения слова  в алфавите будет иметь вид .

Заметим, что в некоторых источниках буква  в изображении НА пишется и в конце, то есть пишется сразу справа от изображения каждой формулы подстановки.

Запись нормального алгорифма  будем обозначать.

Нетрудно понять, что как по изображению, так и по записи схема нормального алгорифма восстанавливается однозначно. Использование записи удобно в теоретических рассмотрениях, так как позволяет считать все нормальные алгорифмы представленными словами в одном и том же двухбуквенном алфавите (своего рода «двоичным кодом»).

### Теорема 4 *(теорема об универсальном нормальном алгорифме)*. Для любого алфавита и любого НА в этом алфавите может быть построен НА над алфавитом такой, что для любого слова имеет место условное равенство .

Алгорифм , построенный согласно данной теореме, называют ***универсальным нормальным алгорифмом***.

Существует вариант формулировки теоремы об универсальном алгорифме, в которой вместо записи фигурирует изображение «объектного» алгорифма .

Доказательство теоремы 4 чрезвычайно сложно и не может быть здесь рассмотрено. В курсе теории алгоритмов для студентов-программистов эта теорема также только формулируется.

Содержательный смысл теоремы состоит в том, что универсальный алгорифм получает на вход пару «объектный алгорифм – слово», декодирует запись объектного алгорифма, восстанавливая тем самым его схему, и применяет эту схему согласно известным правилам ко входному слову. Тем самым заданный в определении нормального алгорифма *алгоритм* (определенный вне использования нормальных алгоримфов) применения схемы, согласно теореме об универсальном алгорифме сам может быть запрограммирован как *нормальный* алгорифм.

С программистской же точки зрения универсальный алгорифм есть не что иное как *универсальный интерпретатор* в классе программ, записанных в виде схем нормальных алгорифмов. Эту аналогию как раз важно провести в аудитории студентов-программистов. А именно, если пара (объектная программа , данные для нее ), то системная программа-интерпретатор , «декодируя» определенным образом объектную программу, сразу применяет ее к исходным данным. Это можно записать в виде условного равенства:



То есть интерпретатор выполняет функцию универсального алгоритма в некотором класса объектных программ.

### Разрешимые и перечислимые множества (языки)

Язык (множество слов)  называют ***алгорифмически разрешимым***, если может быть построен нормальный алгорифм  над алфавитом  такой, что для каждого  имеет место  и .

Алгорифм , фигурирующий в вышеприведенном определении, называют ***разрешающим алгорифмом*** для языка .

Легко показать, что, например, множество слов, содержащих хотя бы одно вхождение заданного непустого слова, будет алгорифмически разрешимо.

От ситуации возможности построения разрешающего НА для языка следует отличать ситуацию ***полуразрешающего*** НА. Так называется НА , применимый к слову  тогда и только тогда, когда . Принципиальное отличие от разрешающего НА состоит в том, что разрешающий всегда заканчивает работу за конечное число шагов, и мы рано или поздно получим ответ на вопрос о принадлежности слова языку. Что же касается полуразрешающего НА, то неизвестно, остановится он когда-нибудь и, следовательно, будет ли вообще дан ответ на вопрос о принадлежности.

Интуитивно очевидно, что из невозможности построения полуразрешающего НА следует невозможность построения и разрешающего. Докажем это строго.

**Теорема 5**. Если для языка невозможен полуразрешающий НА, то невозможен и разрешающий.

**Доказательство**. Предположим противное, то есть предположим, что для языка не может быть построен полуразрешающий НА, но возможно построение разрешающего. Пусть  - такой НА. Тогда по теореме разветвления (теорема 3), построим НА

.

Ясно, что для любого слова  имеет место , и НА  является полуразрешающим вопреки предположению.

Образно говоря, разрешающий алгорифм можно «испортить», сделав его полуразрешающим.

Определим теперь понятие перечислимого множества (языка). Здесь нам понадобится понятие **конструктивного натурального числа** (КНЧ). По определению, это любое слово в алфавите  вида .

Язык (множество слов)  называют ***алгорифмически перечислимым***, если существует нормальный алгорифм  над алфавитом  такой, что для каждого натурального числа *n* имеет место ,  и для каждого  осуществимо такое натуральное число , что .

В дальнейшем будем говорить просто о разрешимых и перечислимых языках.

Понять смысл определения перечислимого языка будет проще, если вспомнить классическое теоретико-множественное понятие нумерации множества.

Именно, под ***нумерацией множества***  понимается любое сюръективное отображение  множества натуральных чисел на множество . Для простоты будем рассматривать далее только биективные (взаимно однозначные) нумерации. Можно показать, что это не ограничивает существенно общности. Множество  в таком случае есть не что иное, как счетное множество.

Алгорифм  называют при этом ***алгорифмом, перечисляющим множество*** *L* (или просто ***перечисляющим алгорифмом***, если множество подразумевается). Говорят также, что ***алгорифм  перечисляет множество*** *L*. В свете теоретико-множественного понятия нумерации, перечислимое множество есть теоретико-алгоритмический (конструктивный) аналог понятия счетного множества, перечисляющий НА есть нумерация, заданная в виде нормального алгорифма, а осуществимость КНЧ  в конце определения есть возможность задать в виде НА отображение, обратное нумерации (если последняя биективна).

**Пример**. Запишем НА, который перечисляет множество конструктивных целых чисел (КЦЧ). КЦЧ – это слово вида . Напишем сначала формулу обычной нумерации  множества целых чисел:



Соответствующий нормальный алгорифм строим по теореме разветвления:

,

где  - НА проверки четности,

 - НА деления пополам с переходом к противоположному числу ( ),

  - НА прибавления единицы и последующего деления пополам.

Может быть доказана следующая теорема:

**Теорема 6**. Всякое алгорифмически разрешимое множество алгорифмически перечислимо.

Заметим, что обратное неверно, что будет видно из дальнейших рассмотрений.

Легко понять также, что дополнение разрешимого языка также разрешимо.

В контексте проблемы применимости особенно важна характеристика перечислимых языков через области применимости нормальных алгорифмов.

Пусть дан НА  над алфавитом . ***Областью применимости*** НА  называется множество .

**Теорема 7**. Язык  перечислим тогда и только тогда, когда он является областью применимости относительно алфавита  некоторого нормального алгорифма.

Доказательство теорем 6 и 7 мы не приводим (см., например, [8]).

### Проблема применимости как неразрешимая массовая проблема

***Проблема применимости*** для нормальных алгорифмов формулируется как частная и как общая.

Начнем с частной проблемы.

Дан нормальный алгорифм в алфавите  . Можно ли построить НА  над алфавитом  так, чтобы он был применим к любому слову , причем  тогда и только тогда, когда .

Другими словами, НА  должен быть разрешающим для дополнения области применимости НА  относительно алфавита .

Общая проблема применимости формулируется следующим образом: можно ли построить нормальный алгорифм  над алфавитом  такой, что для любых нормального алгорифма  в алфавите  и слова  выполнялось  и , где буква # не принадлежит алфавиту .

Можно сказать, что если в постановке частной проблемы применимости ставится вопрос о «тесте» на применимость заданного НА, то в постановке общей проблемы речь идет уже о возможности построения универсального «теста» на применимость в некотором классе нормальных алгорифмов. Ниже будет доказана основная теорема, в доказательстве которой предлагается способ построения НА с неразрешимой частной проблемой применимости. Так что ответ на оба поставленных выше вопроса оказывается отрицательным.

Для этого рассмотрим сначала другую проблему, которая называется ***проблемой самоприменимости*** и формулируется следующим образом: можно ли построить такой алгорифм  над алфавитом , что для любого алгорифма  в алфавите , где  имеет место и. Заметим, что согласно теореме о переводе в алфавите  может быть задан любой алгорифм типа . Следует заметить также, что при обсуждении проблемы самоприменимости мы предполагаем, что все алгорифмы, заданные в каком-то заранее заданном алфавите заменяются их естественными распространениями на алфавит , чтобы можно было корректно их применять к собственной записи.

Таким образом, алгорифм  должен аннулировать (перерабатывать в пустое слово) записи тех и только тех алгорифмов в алфавите , которые *не применимы к собственной записи*. Такие алгорифмы называют ***несамоприменимыми***. Другими словами, НА  должен быть разрешающим для языка записей несамоприменимых нормальных алгорифмов.

Простыми примерами самоприменимого и несамоприменимого НА могут служить НА  правого присоединения фиксированного слова в заданном алфавите , естественно распространенный на алфавит , где  и НА, служащий формальным распространением  на алфавит  соответственно.

**Лемма 1**. Невозможен нормальный алгорифм  в алфавите , применимый к записи произвольного нормального алгорифма  в этом алфавите тогда и только тогда, когда алгорифм несамоприменим.

**Доказательство**. Пусть такой алгорифм  построен. Так как и он является алгорифмом в алфавите , то может быть поставлен вопрос о его самоприменимости. Если , то поскольку  должен быть применим только к записям несамоприменимых алгорифмов, имеет место . Но тогда, так как  несамоприменимый алгорифм, то должно быть . Полученное противоречие доказывает, что алгорифм  не может быть построен.

Таким образом, если «тест» и «тестируемый» на самоприменимость алгорифм заданы в одном и том же алфавите, то проблема самоприменимости неразрешима, так как нельзя построить для языка записей несамоприменимых алгорифмов даже полуразрешающий НА (см. теорему 5).

Можно подумать, что, позволив «тесту» (НА ) работать в каком угодно расширении алфавита  , проблему можно решить. Но это не так.

**Теорема 8**. Невозможен алгорифм над алфавитом , применимый к записям тех и только тех алгорифмов в алфавите , которые несамоприменимы.

**Доказательство**. Пусть построен алгорифм  над алфавитом  так, что для всякого алгорифма  в алфавите  имеет место . По теореме о переводе тогда может быть построен алгорифм  в алфавите  так, что для любого слова  имеет место условное равенство .

Пусть теперь алгорифм  определен как естественное распространение алгорифма  на алфавит . Тогда оказывается, что этот алгорифм в алфавите  таков, что для любого алгорифма  в алфавите  имеет место , что невозможно в силу доказанной леммы.

Суть изложенного доказательства состоит в том, что алгорифм-«тест» и алгорифм, «тестируемый» на самоприменимость, приведены к одному и тому же алфавиту, и мы попадаем в условия леммы 1.

Таким образом, проблема самоприменимости для нормальных алгорифмов, заданных в произвольном наперед заданном алфавите неразрешима.

Здесь методически целесообразно разъяснить учащимся, что алгоритмическая неразрешимость какой-либо проблемы не означает, что на вопрос, который эта проблема ставит, не может быть дан ответ ни в каком конкретном случае. Вполне можно привести пример конкретного нормального алгорифма, для которого может быть доказана его самоприменимость (или несамоприменимость). Но доказательство не есть алгоритм. *Неразрешимость проблемы состоит именно в невозможности построить* ***алгоритм****, который* ***для любого*** *ее частного случая дает ответ на вопрос, сформулированный в проблеме.* Именно так следует понимать *массовость* любой алгоритмически неразрешимой проблемы.

Опираясь на факт неразрешимости проблемы самоприменимости, можно доказать возможность построения нормального алгорифма с неразрешимой частной проблемой применимости, откуда будет следовать, что и общая проблема подавно неразрешима.

**Теорема 9**. Можно построить нормальный алгорифм  в алфавите  так, что невозможен нормальный алгорифм  над алфавитом , удовлетворяющий условиюдля любого слова .

**Доказательство**. Пусть . По теореме об универсальном алгорифме построим алгорифм  над алфавитом  так, что для любого алгорифма  в алфавите  и любого слова  имеет место .

Теперь построим алгорифм  над алфавитом  так, что для любого слова  . (Это можно сделать в виде композиции алгорифма, перерабатывающего всякое слово  в слово  и алгорифма , то есть  , см. схему(8)). Следующее место в доказательстве является наиболее тонким.

Мы имеем НА  *над* алфавитом , но последний есть расширение алфавита  и, следовательно, по теореме о переводе, может быть заменен вполне эквивалентным ему относительно алфавита  алгорифмом  в двухбуквенном расширении алфавита , то есть ***в*** алфавите , и для любого слова  будем иметь: . Утверждается, что это и есть искомый и указанный в условии теоремы алгорифм  с неразрешимой частной проблемой применимости, то есть  .

Действительно, пусть построен алгорифм  над алфавитом  так, что . Тогда для любого алгорифма  в алфавите  будем иметь

,

То есть НА  оказывается полуразрешающим для проблемы самоприменимости для нормальных алгорифмов в алфавите , что невозможно в силу теоремы 8.

Итак, проблема применимости для нормальных алгорифмов является неразрешимой.

**Теорема 10**. Существует перечислимое множество, не являющееся разрешимым.

Эта теорема является следствием доказанной неразрешимости проблемы применимости. Действительно, в качестве такого множества можно взять область применимости (относительно соответствующего алфавита) любого алгорифма с неразрешимой частной проблемой применимости.

Комментируя этот результат, можно заметить, что нормальный алгорифм с неразрешимой частной проблемой применимости строится как универсальный. Но на вход универсального алгорифма подается «объектный» алгорифм, запись которого может быть сколь угодно сложной. Сложность этого конструктивного объекта не ограничена сверху, возникает как бы случайная (0,1)-последовательность [10]. Естественно поэтому, что проблема применимости оказывается в этом случае неразрешимой. Но формальное доказательство основано на противоречии, обнаруживаемом в доказательстве леммы 1.

**Заключение**

В статье рассмотрена целостная методика преподавания теории алгоритмов на примере нормальных алгорифмов Маркова. Методика ориентирована на студентов, обучающихся по направлениям, связанным с разработкой программных технологий.

Новизна методики состоит в подходе к изложению самой концепции алгоритма, теоретико-множественном подходе к нормальным алгорифмам Маркова, в подборе некоторых примеров и приемов доказательства, где расставлены необходимые акценты; сравнение алгоритмических и игровых процессов (алгоритма и доказательства, в частности). Всё в целом ориентировано на студентов-программистов. С этим же связаны и определенные программистские аналогии (например, сравнение универсального алгорифма с программой-интерпретатором).

Основная цель состояла в изложении доступного для понимания студентов-нематематиков доказательства теоремы о неразрешимости проблемы применимости (останова) для нормальных алгорифмов.

Работа основана на личном опыте автора в преподавании теории алгоритмов. В качестве дальнейших методических задач можно назвать проработку курса математической логики для студентов тех же специальностей. Содержание статьи будет полезным преподавателям и студентам при подготовке к занятиям.

**Ссылки на источники**

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ, 3-е изд.: Пер. с англ.- СПб.: ООО «Диалектика», 2019. – 1328 с.

2. Годлевский А.Б. О некоторых разрешимых и неразрешимых случаях проблемы функционально эквивалентности схем программ над памятью // Тезисы докладов III Всесоюзного симпозиума.- Изд. Кишиневского гос. ун-та, 1974. – С. 174-179

3. Подловченко Р.И. Исследования в теории алгебраических моделей программ с процедурами // Программирование. – 2016. - №1. – С. 5-9

4. Косовский Н.К., Косовская Т.М. Полиномиальный тезис Черча для РЕФАЛ-5-функций, нормальных алгоритмов и их обобщений //Компьютерные инструменты в образовании. – 2010. – С. 12-21

5. Косовский Н.К. Алгоритмы Маркова-Турчина и доказательства полиномиальной эффективности программ на языке РЕФАЛ-5 // Компьютерные инструменты в образовании.- 2012. - №4. – С. 41-49

6. Гурин Р.Ф., Романенко С.А. Язык программирования РЕФАЛ ПЛЮС / Учеб.пособие, Ун-т г. Переславля им. А.К. Айламазяна. – Переславль-Залесский, 2006

7. Марков А.А., Нагорный Н.М. Теория алгорифмов. – М.: Наука, 1984. – 432 с.

8. Кушнер Б.А. Лекции по конструктивному математическому анализу. – М.: Наука, 1973. – 448 с.

9. Белоусов А.И., Ткачев С.Б. Дискретная математика //Учеб. для вузов. – 5-е изд. – М.: Изд. МГТУ им. Н.Э. Баумана, 2015. – 744 с.

10. Хованский А.В. Индивидуальная случайная последовательность и генераторы случайных чисел // Математическое моделирование. – 2009. - №7. – С. 93-105.

*Alexey Belousov*

*Candidate of Physical-Mathematical Sciences, Associate Professor,*

*Bauman Moscow State Technical University, Moscow*

[al\_belous@bk.ru](mailto:al_belous@bk.ru)

**On the methodology for the presentation of some sections of the theory of algorithms: the applicability problem for normal Markov algorithms**

***Abstract.*** *The relevance of the methodological problem under consideration is due to the fact that the theory of algorithms occupies a significant place in the curriculum of student programmers, and it requires the development of a rigorous and at the same time methodology for students who do not specialize in mathematics proper presentation of the foundations of this theory.*

*The article proposes a methodology for presenting the key provisions of the theory of algorithms using the example of normal Markov algorithms. The algorithmic unsolvability of the applicability problem (or the stop problem) for normal algorithms is proved in detail. The main results necessary for understanding the main theorem are outlined. The content of the article will be useful to students of programming specialties, as well as teachers of courses in mathematical logic and theory of algorithms, courses in designing algorithms.*

***Keywords:*** *algorithm, normal Markov algorithm, algorithmically unsolvable problems.*

1. Modern European Researches, 2019, 5, 17-36. [↑](#endnote-ref-1)