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: некоторые приемы разработки схем нормальных алгорифмов Маркова[[1]](#footnote-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 рассматриваются некоторые приемы разработки схем нормальных алгорифмов для решения вычислительных задач. На доступном уровне строгости рассматривается доказательство некоторых теорем, необходимых для решения конкретных задач. 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, методические проблемы.*

**Введение**

Данная статья является продолжением статьи [1].

В ней рассматриваются некоторые приемы разработки схем нормальных алгорифмов Маркова [2, 3], а также рассматриваются те теоретические результаты, понимание которых необходимо для усвоения конкретных примеров. Так, доказывается на доступном для студентов-программистов уровне строгости теорема композиции. Формулируются и другие теоремы сочетания, а именно теорема объединения и теорема повторения.

Затем рассматриваются примеры. Во-первых, это примеры на применение теорем сочетания, а во-вторых, примеры, обсуждаемые на практических (семинарских) занятиях, анализируя которые, можно овладеть некими навыками разработки схем нормальных алгорифмов.

Овладение такими навыками важно ввиду востребованности теории нормальных алгорифмов в современном программировании. Речь идет, в частности, о работах над разными версиями языка РЕФАЛ [4, 5, 6], в основе которого лежит именно модель нормального алгорифма.

Следует отметить также, что исследования в этой области продолжаются. Это относится как к чисто научным публикациям [7, 8], так и к методическим [9].

Материал данной 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 [10], в котором для понимания теории алгоритмов особенно важны элементы теории формальных языков.

Поскольку методика изложения теории нормальных алгорифмов подробно рассмотрена в статье [11], то в данной статье мы основное внимание уделяем именно технике разработки схем нормальных алгорифмов.

В статье используется аббревиатура НА (нормальный алгорифм), а также мы часто пишем просто «алгорифм» (вместо термина «нормальный алгорифм»).

**Методология и результаты исследования**

Обсуждаемая в статье методика основана на теории нормальных алгорифмов, как она разработана А.А. Марковым и его школой, и которая в окончательном виде представлена в книге [12]. К этому следует добавить исследования в области языков программирования, которые основаны на модели нормальных алгорифмов.

Использованные в работе методы - методы построения доказательств (в том числе упрощенных, как в книге [13]), принятые в теории нормальных алгорифмов, а также методы решения конкретных задач путем построения схем нормальных алгорифмов. В основных чертах методика изложения теории нормальных алгорифмов для аудитории студентов-программистов представлен в работе [14].

Как было указано выше, результаты представлены в двух частях. В первой рассматриваются теоретические результаты, необходимые для дальнейшего изложения, а во второй приводятся конкретные примеры построения схем нормальных алгорифмов.

*Теоремы сочетания*

Начнем с некоторых важных теоретических результатов.

Рассмотрим на «инженерном» уровне строгости доказательство теоремы композиции. Напомним формулировку:

**Теорема 1** (*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 .

**Доказательство**. Вполне строгое доказательство теоремы композиции мы не приводим (техника строгих доказательств в теории нормальных алгорифмов вообще выходит за пределы вопросов, обсуждаемых в этой работе). Рассмотрим построение схемы алгорифма  «на уровне идеи»

Эта схема приведена ниже (см. сл. стр.).

Она, по построению, является схемой нормального алгорифма в алфавите , где  - алфавит, составленный из «букв-двойников» букв алфавита , т.е. каждой букве сопоставляется буква  - «двойник» буквы  так, что соответствие является взаимно однозначным и пересечение  пусто. Буквы  и  считаются не принадлежащими алфавиту .

Через  обозначен набор формул подстановок, образованный из схемы замыкания  алгорифма  заменой всех «точек» (обозначающих заключительную формулу) буквой : тем самым каждая заключительная формула вида  в схеме  заменяется простой формулой .

Через  обозначен набор формул подстановок, полученный из схемы замыкания  алгорифма  путем замены в ней каждой буквы алфавита ее «двойником», всех «точек» буквами  и после этого замены каждой формулы вида  (т.е. формулы с пустой левой частью) формулой . Заметим, что слово  для преобразованной заключительной формулы исходного алгорифма (с пустой левой частью) начинается буквой .

Перед тем, как разбирать процесс работы алгорифма  с исходным словом, полезно обратить внимание учащихся на главный принцип при построении схем нормальных алгорифмов: *любая вышележащая формула в схеме «перехватывает» управление у всех нижележащих*. Работа алгорифма  с исходным словом может быть описана следующим образом.



Поскольку в исходном слове  нет вхождений букв  и , а также вхождений букв-«двойников», то ни одна из формул строк (1) – (8) не применима к .

Таким образом, к слову  будет применима первая формула набора , т.е. первая формула схемы алгорифма , если она простая, и формула вида , если соответствующая формула в схеме алгорифма  является заключительной.

Пусть алгорифм  применим к слову . Тогда, поскольку сам алгорифм  заменен его замыканием, а набор формул  построен по схеме замыкания алгорифма , то на последнем шаге работы алгорифма  со словом  будет применена заключительная формула; но тогда на соответствующем шаге работы алгорифма  со словом  будет применена формула вида , и, таким образом, если , то , где .

Итак, посредством применения формул (9) к исходному слову  алгорифм  «эмулирует» работу алгорифма , причем если определено слово  и только в этом случае, то на некотором шаге работы  с  будет получено слово , где .

Появление в слове, выводимом из  по схеме алгорифма , буквы  «переключает» схему  на применение формул строки (1), которые «передвигают» букву  в начало слова, т.е. имеет место выводимость .

Нетрудно также видеть, что неприменимость НА  к слову  влечет и неприменимость НА  к этому слову.

Как только  оказывается первой буквой слова, формулы (1) становятся неприменимыми, и начинают «работать» формулы строк (2) и (3), в результате чего слово  будет преобразовано в слово  - «двойник»  в алфавите . Т.е. при  выполняется . Пустое слово, разумеется, совпадает со своим двойником.

После этого могут быть применены только формулы набора , посредством применения которых «эмулируется» работа алгорифма  со словом , а именно, если , то , откуда следует, что  влечет  и, соответственно, . Если же , то (ввиду перехода к замыканию ) на последнем шаге будет применена заключительная формула схемы , что означает в процессе работы  с  применение на соответствующем шаге формулы вида , где  - слова в алфавите . Это значит, что если , то , где .

Появление буквы  на очередном шаге работы алгорифма  означает «переключение» его схемы на применение формул строки (4), которые «перегоняют»  влево до тех пор, пока оно не окажется перед первой буквой слова  и после буквы , неизменно стоящей на первом месте во всех словах выводимых из  по схеме алгорифма . Таким образом, имеет место выводимость .

После этого посредством применения формул строк (5) и (6) происходит «превращение» букв-«двойников» в буквы основного алфавита , т.е. из слова  выводится слово . В результате формулы строк (1) – (6) становятся неприменимыми, и применяется заключительная формула (7), т.е. в итоге получаем: .

Следовательно, если хотя бы одно из слов  или  не определено, т.е.  или , то , а если  и , то , и условное равенство  доказано.

Нам будет удобно дальше обозначать НА  как , то есть .

Разумеется, можно определить композицию произвольного числа нормальных алгорифмов, полагая . Более того, мы можем определить неотрицательную целую степень НА следующим образом: , где  - тождественный НА, задаваемый схемой  (см. [1]).

Приведем формулировки еще двух теорем сочетания: а именно теорем объединения и повторения.

Сочетание, называемое объединением, определяется таким предписанием:

1. к исходному слову  применить алгорифмы  и (независимо друг от друга);
2. если оба слова  и  определены, то результатом считается слово 

**Теорема 2** (теорема объединения). Каковы бы ни были нормальные алгорифмы  в алфавите  и  в алфавите , может быть построен нормальный алгорифм  над алфавитом  такой, что для любого слова  выполняется условное равенство .

Теорема объединения доказывается уже с учетом доказанной теоремы о композиции, т.е. доказательство состоит в том, что объединение сводят к композиции. Это доказательство мы не приводим (даже «на уровне идеи»; см. [15] и [16]).

**Замечание**. Объединение алгорифмов иногда называют ***параллельной композицией***. Существуют также разные варианты определения этой комбинации нормальных алгорифмов. Один из них таков: алгорифмы  и (независимо друг от друга) применяются к словам  и  соответственно, а для алгорифма  выполняется условное равенство (для любых слов ) ,

где «звездочка» (\*) – буква, не принадлежащая объединению алфавитов  
 и .

Нормальный алгорифм , построенный согласно теореме об объединении, обозначают .

Повторение НА может быть содержательно определено такой псевдопрограммной конструкцией с использованием нормальных алгорифмов  и :

**while**  **do**  **end,**

т.е. если слово  определено и пусто, то к слову  надлежит применить алгорифм , если же слово  не есть пустое, то результатом всего рассматриваемого алгоритма считается само слово . Далее, если слово  определено, и , то к слову  надлежит применить снова алгорифм ; если же , то общим результатом является слово  и т. д.

Заданный таком образом алгоритм называют ***повторением нормального алгорифма , управляемым нормальным алгорифмом*** , или, короче, -***повторением нормального алгорифма*** .

Можно доказать, что и повторение нормального алгорифма «программируется» в виде нормального алгорифма.

**Теорема 3** (*о повторении нормального алгорифма*). Каковы бы ни были нормальные алгорифмы  в алфавите  и  в алфавите , может быть построен нормальный алгорифм  над алфавитом  такой, что для любого слова , что имеет место равенство  тогда и только тогда, когда существует последовательность слов  такая, что ,  и , или  и .

Нормальный алгорифм  будем обозначать .

Условие повторения цикла можно изменить на противоположное. Такое повторение будем обозначать .

*Примеры*

Примеры в этом разделе делятся на две части. Сначала рассматриваются некоторые примеры решения задач с использованием теорем сочетания. Затем приводятся примеры, обсуждаемые на семинарских занятиях со студентами, где демонстрируются важные приемы, используемые при разработке схем нормальных алгорифмов.

Рассмотрим решение задач на построение некоторых алгорифмов с использованием теорем сочетания.

**Пример 1**. *Проекцирующие алгорифмы*. Построим семейство алгорифмов , над алфавитом  так, что для любых слов  в алфавите  выполняется



Построим алгорифм  со схемой



Нетрудно видеть, что .

Теперь построим алгорифм  со схемой



Понятно, что  .

Тогда . Обозначая кратную композицию алгорифма  с самим собой через  и полагая при этом, что , получим . В частности, 

Перед тем, как рассматривать следующий пример на теоремы сочетания, построим НА , который инвертирует входное слово в заданном произвольно алфавите , то есть, если слово , то (при этом, разумеется, инверсия, или обращение, пустого слова есть пустое слово).

**Пример 2**. *Инвертирующий нормальный алгорифм*.

Схема алгорифма  приведена ниже:



В этой схеме буквы  и  не принадлежат алфавиту .

Пусть , есть непустое слово в алфавите .

В схеме алгорифма  к слову  применима только самая нижняя формула. После ее применения, при условии, что , будет применима одна из формул верхней строки. Эти формулы применяются до тех пор, пока слово  не окажется в конце текущего слова процесса работы:



Затем буква  «превратится» в букву  (вторая сверху строка схемы) и получится слово  . К нему опять-таки применима только самая нижняя формула; после ее применения будем иметь (допуская, что ) :



Второе вхождение буквы  исчезает благодаря применению третьей сверху формулы (точнее, конкретной формулы, определяемой этой строкой).

Так будет происходить перенос в конец слова всех по очереди букв исходного слова до тех пор, пока не получится слово .

Далее:

 .

Приведенный разбор процесса работы нельзя считать строгим доказательством, но можно вполне строго доказать, что алгорифм  перерабатывает слово , длина которого больше 1, в его инверсию .

Для пустого и однобуквенного слова соответственно имеем:



и

.

Итак, мы можем утверждать, что .

**Пример 3.** *Алгорифм распознавания равенства слов*. Построим алгорифм  над алфавитом  такой, что для любых двух слов  имеет место 

Требуемый нормальный алгорифм можно построить, используя теоремы объединения и композиции, а именно

,

где  - алгорифм, задаваемый схемой:

.

Очевидно, что для любых двух слов выполняется 

**Пример 4**. *Алгорифм определения «центра» слова*.

Построим алгорифм , который любое слово  в алфавите  переводит в слово , где  при четной длине слова , и  в противном случае; .

Это можно реализовать, определив алгорифм следующим образом:

, где схемы входящих в указанное сочетание алгорифмов, выглядят так:









Алгорифм  вводит правый «челнок» , который при каждом повторении сдвигается влево на букву (при первом повторении это делает «челнок» , тут же превращающийся в).

Алгорифм  вводит левый «челнок» , при каждом повторении сдвигающийся на букву вправо. Верхняя формула служит для «блокировки» работы этой схемы в том случае, когда вхождение  уже возникло.

Как только «челноки»  и  встречаются, цикл прекращается (условие выхода из цикла проверяет алгорифм ).

Алгорифм  заменяет вхождение  на «доллар». Поскольку правый «челнок» опережает левый на шаг, то при нечетной длине входного слова правая часть разделенного «долларом» слова будет длиннее левой на одну букву.

Вторая формула в схеме алгорифма предназначена для корректной работы алгорифма  с пустым входным словом. В этом случае алгорифм вырабатывает пустое слово, цикл не выполняется, а алгорифм  вырабатывает в виде результата значок доллара, что и нужно (обе половинки пустого слова тоже, очевидно, пустые).

Теперь разберем несколько примеров, на которых покажем определенные приемы построения схем нормальных алгорифмов, решающих предъявленные содержательные задачи, или вычисляющие заданные вербальные функции.

**Пример 5**. Написать схему НА, аннулирующего всякое непустое слово в заданном произвольном алфавите , содержащееО ровно два вхождения заданного произвольного непустого слова  . При этом, если входное слово пустое ( ), то результатом должен быть символ @, а если входное слово непусто и не удовлетворяет условию, то результатом должно быть оно само (т. е. алгорифм должен вычислять тождественную функцию).

Схема:



Во всей схеме  - параметр, обозначающий произвольную букву алфавита .

Если входное слово пустое, то первой подходящей формулой будет формула (11), «печатается» «собака», и процесс заканчивается. Если входное слово не пустое, и не содержит ни одного вхождения слова , то сразу применяется формула, соответствующая строке (10), процесс заканчивается, и результатом будет то же входное слово. Если входное слово V содержит по крайней мере одно вхождение слова , то на первом шаге будет применена формула (9), появится «решетка», после чего управление «перехватит» формула строки (7) (но если слово  однобуквенное, то может оказаться подходящей и формула (6)). Заметим, что в общем случае различные вхождения слова  могут пересекаться, и мы позволяем «решетке» аккуратно «перепрыгнуть» только через первую букву слова u. «Решетка» играет роль своего рода челнока и сканирует входное слово в поисках 2-го вхождения слова u. Если его нет, «решетка» добегает до конца входного слова и исчезает (формула (8)). Если же второе вхождение есть, то в определенный момент управление будет передано формуле (6), и второй челнок (двойная «решетка») отправляется искать третье вхождение слова  (строка (2)). Если оно не обнаруживается, то, добежав до конца слова, двойная «решетка» превращается в «доллар», который стирает входное слово и исчезает сам (строки (4) и (5)). Условие выполнено, входное слово аннулировано. В противном случае челнок ## исчезает, и входное слово остается как было (формула (1)).

Если сделать специальный акцент на методике, то как раз на примере данной схемы полезно показать студентам, как, исходя из смысла задачи, писать схему нормального алгорифма. И тогда окажется, что процесс записи схемы может частично идти снизу вверх.

Мы прежде всего отрабатываем пустое слово на входе и сразу в самом низу схемы пишем формулу (11). Далее рассматривается вариант наличия по крайней мере одного вхождения интересующего нас слова. Тогда над формулой (11) пишем формулу (9), оставляя место для позднейшей вставки строки (10). Дальше объясняем всё, как написано выше и в схеме пишем сперва строку (7) - выше, а потом (8) - ниже. Строка (10) появляется в процессе разработки схемы в реальном времени в последний момент. Можно задать учащимся вопрос: чего не хватает в нашей схеме? После недолгого размышления становится понятно, что не отработано непустое слово на входе, в котором нет ни одного вхождения слова .

Студентам можно предложить в качестве несложного упражнения модифицировать схему так, чтобы НА аннулировал слова, в которых не менее трех вхождений заданного непустого слова.

**Пример 6**. Написать схему НА, распознающего (в том же смысле, как и в предыдущей задаче) такие слова в алфавите V, которые содержат хотя бы одно вхождение двух, отличных друг от друга( но возможно, что одно из них входит в другое) непустых слов.

В написанной ниже схеме параметр  пробегает алфавит V (т. е. ), буквы  ,  - слова, вхождения которых требуется найти.



Некоторая тонкость состоит в том, что первый челнок (#) выставляется сразу перед всем входным словом. Это сделано для того, чтобы не упустить вложенного вхождения.

Примеры процессов работы:

1) 

В третьем слове вывода пробелами специально выделено первое вхождение.

Заметим, что НА находит **первое** вхождение каждого слова.

2) 

Можно заметить, что расположение формул в 5-й и 6-й строках снизу друг относительно друга не существенно.

**Пример 7**. Написать схему НА, вычисляющего следующую функцию:

 ,

где входное слово x есть слово в произвольно заданном алфавите V, а  .



Если во входном слове нет требуемых вхождений, то применяется самая нижняя формула, и перед словом возникает  (для пустого входного слова это будет результат). Иначе после первой буквы первого вхождения слова  выскакивает первый челнок ($), который запускается для поиска 2-го вхождения. Если его нет, «доллар» заменяется на 0, который по формулам самой верхней строки идет до упора влево и через решетку приставляется к входному слову слева (2-я строка сверху). При обнаружении 2-го вхождения «доллар» запускает аналогично второй челнок (амперсанд), который ищет 3-е вхождение (его можно заменить и двойным «долларом»). Если 3-е вхождение есть, амперсанд превращается в 0 и точно так же, как и выше идет к началу входного слова и присоединяется к нему через «решетку» слева. При отсутствии 3-го вхождения амперсанд, добежав до конца входного слова, превращается в 1, которая так же, как и 0, бежит к левому краю и слева через решетку присоединяется к входному слову.

Рассмотрим теперь некоторые арифметические нормальные алгорифмы.

**Пример 8**. *Сложение и умножение конструктивных натуральных чисел*.

Схема сложения совсем простая:



Пара аргументов  перерабатывается в сумму .

Схема умножения значительно сложнее.



Нужно показать, что .

Рассмотрим следующие случаи.

1) 

 .

2) 



Итак, если хотя бы один аргумент равен нулю, то произведение равно нулю.

3) 



Доказано. Пример (с некоторыми изменениями) взят из книги [3].

**Пример 9**. *Усеченное вычитание*



Легко показать, что 

**Пример 10**. *Распознавание нестрогого неравенства*.



Также нетрудно проверить, что  .

**Заключение**

В статье рассмотрены некоторые приемы разработки схем нормальных алгорифмов, решающих конкретные вычислительные задача. Овладение такими приемами важно для студентов, обучающихся по направлениям, связанным с разработкой программных технологий.

Элемент новизны методики состоит отчасти в том, что в некоторых примерах разбирается сам процесс конструирования схемы, исходя из существа задачи. Даются достаточно подробные пояснения к принципам работы каждой схемы.

Основная цель статьи, вместе с рассмотрением техники разработки схем нормальных алгорифмов, состояла и в изложении доступного для понимания студентов-нематематиков доказательства теоремы композиции и формулировке некоторых важных результатов, понимание которых необходимо при решении задач.

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. Белоусов А.И. О методике изложения некоторых разделов теории алгоритмов: проблема применимости для нормальных алгорифмов Маркова // Modern European Researches. – 2019. - №5. – С. 17-36.

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

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

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. Shidik G.F., Pulungan R. Application of Markov’s Normal Algorithms // Advanced Science Letters.- 2015. – vol. 21(10).- P. 3270-3274.

8. Пруцков А.В. Наиболее общая модификация нормальных алгоритмов Маркова // Cloud of Science. – 2018. - №1. - P. 74-85.

9. Игошин В.И. О значении теории алгоритмов для современного профессионального образования и методики ее преподавания //Профессиональное образование в современном мире.- 2019.- №2. – С. 2224-2241.

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

11. Белоусов А.И. Указ. соч.

12. Марков А.А., Нагорный Н.М. Указ. соч.

13. Кушнер Б.А. Указ. соч.

14. Белоусов А.И. Указ. соч.

15. Марков А.А., Нагорный Н.М. Указ. соч.

16. Белоусов А.И. К проблеме распараллеливания алгоритмов // Программирование. – 1978. - №5. – С. 53-61.

*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 method of presentation of some sections of algorithm theory: some techniques of normal Markov's algorithms schemes working out**

***Abstract***. *The urgency of the considered methodical problem is caused by that in curricula of students-programmers the essential place takes the theory of algorithms, and it is required to develop a technique of strict and at the same time accessible to the students who are not specializing actually in mathematics, statements of bases of this theory.*

*The paper considers some approaches of working out of schemes of normal algorithms for solving computational tasks. The proofs of some theorems necessary for solving concrete tasks are considered at an accessible level of severity. The content of the paper will be useful for students of programming specialties as well as for teachers of courses of mathematical logic and theory of algorithms, courses of algorithm design.*

***Keywords****: algorithm, normal Markov's algorithm, methodical problems*

1. Опубликовано в Modern European Researches. – 2020. - №2(Т.1). – С. 38-49. [↑](#footnote-ref-1)