УДК 519.766, 519.1, 519.713.1, 519.713.2 DOI: 10.14529/cmse230X0Z

О несуществовании простого варианта полиномиального алгоритма извлечения корня из языка

© 2023 Б.Ф. Мельников[1]

[1] *Совместный университет МГУ – ППИ в Шэньчжэне*
(Китай, 518172, провинция Гуандун, г. Шэньчжэнь, район Лунган,
Даюньсиньчэн, улица Гоцзидасюеюань, дом 1)
E-mail: bormel@mail.ru

Поступила в редакцию: 07.10.2023

Для стандартной операции конкатенации слов, рассматриваемой как умножение, естественным образом определяется конкатенация языков, а на основе последней операции – степень языка и, при наличии, корень заданной степени. При описании алгоритмов построения языка, являющегося корнем степени M из заданного языка, большое значение имеют так называемые потенциальные корни: это такие слова (не языки), рассматриваемая M-я степень которых входит в заданный язык. Несложно показать, что все потенциальные корни для заданного языка строятся с помощью полиномиального алгоритма.

Эта задача, по-видимому, не упрощается при рассмотрении слов и языков над 1-буквенным алфавитом – что и делается в настоящей статье.

Табуированная пара потенциальных корней – это такая пара, конкатенация слов которой в язык не входит. В предыдущих публикациях на тему описания алгоритмов извлечения корней из языка возникала гипотеза, что полиномиальный алгоритм извлечения корня из языка может быть описан на основе рассмотрения только множества тубуированных пар – путём перебора специально описываемых подмножеств множества потенциальных корней. В настоящей статье показывается, что подобный алгоритм (называемый «простым») невозможен, т. е. если и существует полиномиальный алгоритм извлечения корня из языка, то он (алгоритм) должен использовать какую-то дополнительную информацию.

Ключевые слова: формальные языки, конкатенация языков, степень языка, корень из языка, полиномиальные алгоритмы.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Мельников Б.Ф. О несуществовании простого варианта полиномиального алгоритма извлечения корня из языка // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. X, № Y. С. Z1–Z2. DOI: 10.14529/cmse230X0Z.

Введение

Настоящую статью можно рассматривать как продолжение [1]. В ней (а также в более ранней статье [2]) мы рассматривали задачу извлечения корня из заданного языка.

При этом корень – как и в других областях алгебры – является обратной операцией к операции возведения в степень и, также (так же) как и в других областях алгебры, определяется, вообще говоря, неоднозначно. При этом, конечно, последняя операция (возведение в степень) стандартным образом определяется через операцию умножения, которой в теории формальных языков обычно называют конкатенацию. Какой-либо пример неоднозначности извлечения корня легко конструируется даже для случая 1-буквенного алфавита: квадрат языка

$$B = \{\varepsilon, a, a^2, a^3, a^4\}$$

равен

$$A = B^2 = \{\varepsilon, a, a^2, \ldots, a^7, a^8\},$$

и из языка A извлекается не только корень B, но и

$$B' = \{\varepsilon, a, a^3, a^4\}.$$

Повторим информацию, уже приведённую нами в [1], причём более подробно. Несмотря на простоту формулировки задачи, автор не смог найти в литературе (а также в Интернете) какого-либо её описания, в том числе даже её постановки. Но, несмотря на это, возникает впечатление, что весь материал [1] и настоящей статьи является очень лёгким, и вызывает недоумение тот факт, что в известных автору монографиях нет хотя бы формулировок подобных задач.

Вообще, в настоящей статье мы случай 1-буквенного алфавита и 2-й степени для корня будем рассматривать всё время: на основе материала статьи [1] создаётся впечатление, что получаемые при таком упрощении результаты на более сложные ситуации *обобщаются всегда*; но, конечно, строго доказать сформулированное в этом абзаце предположение вряд ли возможно.

Задачу извлечения корня можно рассматривать в нескольких вариантах (см. подробности далее) – но мы будем говорить про задачу построения *за полиномиальное время хотя бы одного* подходящего корня. При этом, конечно, всюду в настоящей статье полиномиальность понимается относительно размера входных данных – что в нашей ситуации можно считать суммарной длиной слов заданного языка (в примере выше – A).

Для формулировки результатов очень важно понятие «потенциальный корень» – и, в отличие от корня (являющегося языком), последний объект является словом: потенциальные корни нужной степени для заданного языка A – это те слова, у которых такая степень принадлежит языку A. Множество потенциальных корней строится за полиномиальное время [1]. При этом *экспоненциальные алгоритмы извлечения корня из языка описываются тривиальным образом*: надо просто перебрать все подмножества множества потенциальных корней. К последнему сто́ит добавить, что при наличии проверяемого множества потенциальных корней – в литературе алгоритм его выбора часто называется оракулом, [4–8] и др. – саму проверку того, является ли рассматриваемое подмножество корнем нужной степени, можно выполнить за полиномиальное время [2].

(Потенциальные корни используются и в другом классе задач – по-видимому, более сложном, чем задачи извлечения корня из языка. Это – задачи, которые условно можно назвать «раскодированием», или «построением инверсного морфизма», см. [3] и ссылки из этой работы. При этом, несмотря на бо́льшую сложность задач «раскодирования», описать полиномиальный алгоритм построения инверсного морфизма – при некотором специально сделанном предположении теории формальных языков, – в отличие от рассматриваемой здесь задачи извлечения корня из языка, уже удалось.)

Итак, мы считаем невозможным полный перебор всех подмножеств множества потенциальных корней (“brute-force algorithm”). Но основным результатом статьи [1] была теорема о том, что если мы знаем некоторый корень, то при выполнении специальных несложных условий мы можем добавить к этому корню (как к подмножеству) ещё один потенциальный

[1] Это не очень сложное утверждение – однако тривиальным его назвать нельзя: алгоритмы, решающие эту задачу «в лоб», полиномиальными не являются. Полинмиальность похожих алгоритмов была показана, например, в [3].

[2] В отличие от предыдущей проверки, полиномиальность этой проверки очевидна. Однако при этом надо осознавать, что степень полинома не может не зависеть от требуемой в задаче степени для извлечения корня.

корень (как элемент) [3]. Поэтому может возникнуть предположение, что можно осуществить следующую схему алгоритма:

- как-то описать все такие подмножества множества потенциальных корней, к которым заведомо нельзя добавить ни одного нового элемента (ещё одного потенциального корня);
- проверить все такие подмножества, не является ли какое-либо из них корнем, –

и всё это должно дать решение основной задачи (построение полиномиального алгоритма извлечения корня). Однако основной предмет настоящей статьи заключается в том, что

> *подобные простые действия не могут привести к желаемому описанию полиномиального алгоритма*,

поскольку существуют примеры (и мы их описываем), в которых число подмножеств, необходимых для подобной проверки, от размерности задачи зависит экспоненциально.

Важно отметить, что при этом мы вовсе

> *не утверждаем несуществование полиномиального алгоритма*

извлечения корня: мы пока говорим лишь о том, что не может сработать подобный простой способ, то есть если и существует полиномиальный алгоритм решения рассматриваемой задачи, то в нём должна использоваться более существенная информация, чем та, которая формулирует возможность добавления нового элемента (потенциального корня) в некоторое подходящее под ответ множество потенциальных корней.

Отметим ещё, что предмет настоящей статьи может быть охарактеризован и по-другому: в ней формулируются те утверждения, которые можно получить рассмотрением только таких пар потенциальных корней, оба элемента которых *не могут одновременно входить* в решение задачи.

Приведём содержание статьи по разделам. *Раздел 1* – предварительные све́дения и обозначния, частично стандартные, а частично введённые ранее в предыдущих публикациях. Специальные обозначения, связанные с гиперкубами, вынесены в отдельный *раздел 2*; там же приведены и некоторые примеры.

В разделе 3 приводится дополненное доказательство теоремы о возможном увеличении одной координаты имеющегося корня: в опубликованном в [1] варианте доказательства этой теоремы не были рассмотрены все возможные ситуации.

Продолжение рассмотрения гиперкуба – применительно к N-мерному гиперкубу потенциальных корней – приведено *в разделе 4*. В основном, в этом разделе рассматриваются множество простых путей в этом гиперкубе – между «максимальной» и «минимальной» его вершинами. Уточняется формулировка гипотезы о множестве т. н. важных потенциальных корней.

В разделе 5 показывается возможность применение оптимизационных задач на графах – для решения некоторых подзадач, связанных с задачей извлечения корня из языка; в частности, рассматривается применение задачи 2-SAT.

Основные результаты статьи приведены *в разделе 6*: показана возможность конструирование примеров на тему несуществования простого алгоритма извлечения корня степени 2 – причём примеров для сколь угодно большой (но заранее известной) размерности; как

[3] Заранее отметим, что далее в настоящей статье будет приведено более подробное доказательство этой теоремы.

и всюду в настоящей статье, простыми мы называем алгоритмы, основанные только на информации о табуированных потенциальных корнях.

В разделе 7 очень кратко рассмотрен частный случай для степени 3; при этом есть основания полагать, что на основе приведённого примера можно построить примеры для степени 3 сколь угодно большой (но, конечно, заранее известной) размерности – примеры, показывающие несуществование простого алгоритма извлечения корня 3-й степени.

В заключении повторены основные результаты статьи и приведены возможные направления дальнейших исследований по рассматриваемой тематике – описанию алгоритмов извлечения корня из заданного языка.

1. Предварительные све́дения и основные обозначения

Приведём основные обозначения – частично стандартные, частично введённые ранее в [1], а частично новые.

Смысл слова «умножение» для всех рассматриваемых объектов – обычный для теории формальных языков: это просто результат произведения (приписывания, конкатенации) двух слов (двух языков). Для иллюстрации рассмотрим такой пример для 1-буквенного алфавита – основного объекта настоящей статьи: если

$$C = \{\, a^i \mid i \in I \,\} \quad \text{и} \quad D = \{\, a^j \mid j \in J \,\}$$

(I и J – некоторые конечные подмножества целых неотрицательных чисел), то

$$C \cdot D = \{\, a^k \mid (\exists\, i \in I,\, j \in J)\,(k = i + j) \,\}.$$

Слово «умножение» будем в дальнейшем применять без кавычек: в нашем случае оно обладает не только обычными полугрупповыми свойствами (которые практически всегда необходимы при рассмотрении моделей теории формальных языков), но и, дополнительно к ним, коммутативностью.

Среди конечных подмножеств множества целых неотрицательных чисел, рассматривавшихся в предыдущем абзаце, особую роль будут играть множества вида

$$\{\, K, K+1, \ldots, L-1, L \,\} \quad \text{для некоторых} \quad K, L \in \mathbb{N}, \quad L \geqslant K. \tag{1}$$

Множество вида (1) будем обозначать записью $\overline{K, L}$.

На основе умножения (произведения) естественным образом определяется степень языка. Далее нам необходимо ввести описание операции извлечения корня – однако сначала зафиксируем несколько соглашений о терминах и о применяемых константах, которых мы всегда будем придерживаться при описании задач и рассматриваемых к ним примеров.

- Первое соглашение. Всюду далее, если не сказано иного, решается задача поиска языка X, являющегося корнем уравнения

$$X^2 = A, \tag{2}$$

  где язык A заранее задан. Отметим, что в [1] рассматривалось уравнение $X^M = A$ для различных $M \geqslant 2$, а в настоящей статье мы кратко рассмотрим случай $M = 3$ в разделе 7.

- При этом *потенциальный корень* – это некоторое слово $u \in \Sigma^*$, такое что $u^2 \in A$. В отличие от них (т. е. слов) – множество (язык) X будем называть корнем (без прилагательного «потенциальный»), если выполнено условие (2). Как мы уже отмечали, найти все потенциальные корни можно за полиномиальное время с помощью несложных алгоритмов.
- Второе соглашение: N – количество потенциальных корней, будем их нумеровать от 1 от N, здесь мы используем «обычную индексацию Паскаля».
- Третье соглашение. Если, как в приведённом во введении примере, «задача формируется» возведением во 2-ю степень некоторого языка (пусть это язык B; считаем что при решении получаем язык X), то представление этого языка (как множества, B или X) выполняем перечислением разрядов, от 0 до некоторого заданного n; здесь мы используем «индексацию Си-подобных алгоритмических языков».
- Таким образом, размерность каждого из языков B и X равна $n+1$ – и можно рассматривать только такие ситуации, когда и старший, и младший разряды равны 1. При этом $2{\cdot}n+1$ будет размерностью *исходного* языка A (т. е. для его задания нужно каким-то способом определить разряды от 0-го до $2n$-го, причём также можно считать, что старший и младший разряды равны 1).

Для того, что выше названо «третьим соглашением», мы во всех случаях (т. е. как для A, так и для $B\,/\,X$) будем применять 4 варианта обозначения множества слов:

- во-первых, «самый обычный»: например,

$$\{\, \varepsilon{=}a^0, a{=}a^1, aaa{=}a^3, aaaaaa{=}a^6 \,\}. \tag{3}$$

Остальные три варианта записи (см. далее) будут применяться как для подмножеств множества потенциальных корней [4], так и для нескольких элементов множества

$$\{\, \varepsilon, a, aa, \ldots a^{K-1}, a^K \,\}$$

(при этом обычно будет либо $K = n$, либо $K = 2n$). Приведём примеры – для языка (3):

- [0 1 3 6] – множество используемых степеней записываем в квадратных скобках в порядке возрастания;
- 1001011 – двоичное число, в котором разряды предыдущего списка отмечены 1; разряды двоичного числа нумеруем начиная с 0 справа налево, каждые 10 разрядов (при наличии) отделяем небольшим пробелом;
- 75 – число, принадлежащее $\mathbb{N}_0$, представляющее собой десятичную запись предыдущего пункта.

Как уже отмечалось, те же самые обозначения будем использовать, когда $K = N$, и при этом мы рассматриваем подмножества множества потенциальных корней; однако в этом случае разряды двоичного числа удобнее записывать слева направо и нумеровать начиная с 1. Например, пусть всего 5 элементов, а подмножество – $\{2,5\}$:

- [2 5], иногда просто 2 5, даже иногда 25 – неоднозначности это не вызовет; пустое множество обозначается обычно, т. е. $\varnothing$;
- 01001; пустым множеством здесь является последовательность нулей;
- 9; пустое множество – 0.

[4] Для зафиксированного множества этих корней, мы ранее договорились, что их число равно N. Будем считать, что также зафиксирована и их нумерация.

Отметим, что что недоразумений (неоднозначности прочтения) в статье возникнуть не должно.

2. Обозначения и примеры, связанные с гиперкубами

В нескольких приведённых выше варантах записи уже были использованы подмножества; при этом подмножества множества потенциальных корней (напомним, что мы предположили, что число их равно N) будем рассматривать как вершины N-мерного гиперкуба. Определим несколько связанных с подобным гиперкубом вспомогательных понятий.

- Определение максимальной вершины гиперкуба естественное – это $(1, 1, \ldots, 1, 1)$.
- Для дальнейшего – в случае рассмотрения задачи извлечения корня 2-й степени – часто будут использованы *пары* потенциальных корней; в частности, т. н. табуированные пары (о них подробности далее). Для примера – если:
  - мы принимаем «третье соглашение» (напомним, что в нём мы рассматриваем только случай 1-буквенного алфавита)
  - и при этом рассматриваем множество [0 1 2 3 6] (оно действительно будет подробно рассмотрено в примерах в дальнейшей части статьи),
  - а пара – $\{2, 6\}$ (конечно, порядок элементов пары несущественен),

  то возможны ещё и такие обозначения этой пары:
  - $\overbrace{2, 6}$ (это просто их по значениям); это то же самое, что и $\overbrace{6, 2}$;
  - $\overbrace{\#3, \#5}$ (это их по номерам, считаем номера начиная с 1).
- Для некоторой табуированной пары табуированная гиперплоскость – это такая N−2-мерная гиперплоскость N-мерного гиперкуба, для которой обе координаты элементов пары равны 1.
- Обобщением табуированной пары и табуированной гиперплоскости для степеней извлекаемого корня 3 более (мы продолжаем работать с 1-буквенным алфавитом) являются гиперплоскости соответствующих размерностей – но мы, однако, в настоящей статье подробно эти вопросы рассматривать не будем, это нужно только для примера, кратко рассматривающегося в разделе 7. Однако уже здесь сто́ит отметить, что в случае больши́х размерностей (пусть k) нужно рассматривать гиперплоскости не только размерности N−k, но и бо́льших размерностей: среди элементов подмножества множества потенциальных корней, необходимого для формирования одного элемента исходного языка, могут быть совпадающие потенциальные корни.
- Примеры обозначний для степени 3 –

  $$\overbrace{2, 5, 6} \quad \text{и} \quad \overbrace{2, 6}.$$

  В первом случае элементы, которых *не должно быть* в исходном языке – это

  [6 9 10 12 13 14 16 17 18],

  (поскольку каждый из элементов исходной табуированной тройки может входить в формируемую сумму более одного раза); а во втором случае –

  [6 10 14 18].

- Отметим, что максимальная вершина этого гиперкуба входит в пересечение любого (ненулевого) количества любых табуированных гиперплоскостей – независимо от

их размерностей (т. е. от размерности рассматриваемой задачи, иными словами – от степени извлекаемого корня).

- На рисунках будем изображать гиперкуб «в обычном виде» в тех случаях, когда он приводится полностью; иначе (в частности, при больших размерностях) будем необходимую его часть изображать как подграф графа булеана.
- Для некоторой точки гиперкуба $(b_1 b_2 \dots b_N)$ обозначение $(b_1 b_2 \dots b_N)_{+k}$ означает вершину, полученной из предыдущей путём замены k-й координаты на 1.

Перейдём к уточнению непосредственной постановки решаемой задачи. Как отмечалось в [1] (и ранее в [9]), рассматривая для некоторого заданного конечного языка задачу извлечения корня заданной степени [5], мы фактически имеем дело не с одной задачей, а с целой *группой задач*, – поскольку имеется несколько вариантов для требуемого ответа:

- найти *любой* (корень из языка);
- найти *все* [6];
- найти *минимальный* (по некоторой метрике),
- и т. п.

Ниже мы всегда будем иметь в виду задачу построения *любого* корня.

Повторим рисунок из предыдущей статьи (ранее он был опубликован как [1, рис. 9]) – с некоторыми новыми комментариями.

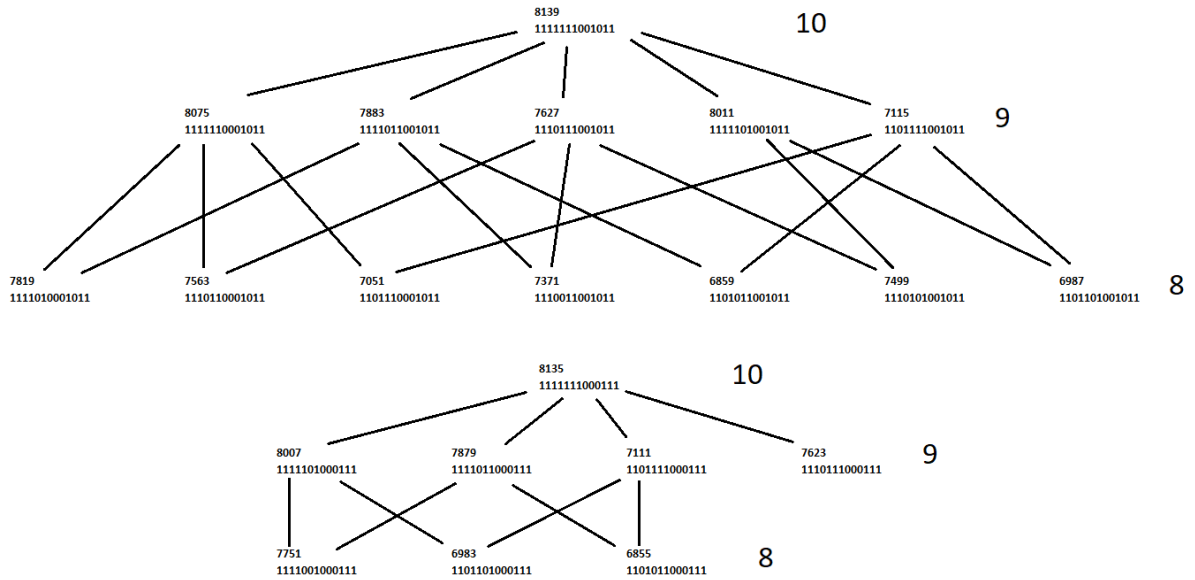

Рис. 1. Подграф графа булеана размерности 13.

Как было сказано в [1], на рисунке показан подграф графа булеана размерности 13 со всеми квадратными корнями квадрата слова

$$1101011000111 \quad [0\ 1\ 2\ 6\ 7\ 9\ 11\ 12] \quad (6855).$$

[5] Либо максимально возможной степени.

[6] По этому поводу повторим в новых обозначениях пример с несколькими возможными корнями: у языка [0 1 2 3 4 5 6 7 8] имеется не только очевидный корень [0 1 2 3 4], но и корень [0 1 3 4]. Понятно, что конструировать подобные примеры совсем несложно – гораздо сложнее обратная операция, ею мы в настоящей статье и занимаемся.

Справа от элементов показаны уровни булеана – количества потенциальных корней, входящих (вкачестве элементов) в рассматриваемый корень из языка. Сразу отметим, что всего существует $C_{13}^{8} = 1287$ вариантов подмножеств из 8 потенциальных корней, но лишь 10 из них являются квадратными корнями из заданного языка.

В заключение раздела отметим, что существуют примеры, когда (квадратные) корни есть – но ближайший к максимальной вершине корень находится от неё на сколь угодно большом (но заданном заранее) расстоянии. В [1] такой пример был приведён для расстояния 4 – но при этом был приведён с опечаткой, одна из форм записи рассматриваемого языка (квадрат которого и поступает на вход задачи) – это

$$[0\ 1\ 4\ 6\ 8]$$

(в [1] была опечатка: ошибочно было указано $[0\ 2\ 4\ 7\ 8]$, но при этом остальные возможные варианты записи рассматриваемого языка были приведены верно).

Более важно, что на основе такого примера для любого заданного $L \in \mathbb{N}$ можно построить пример языка, когда ближайший к максимальной вершине корень находится от неё на расстоянии L – в то время как общая размерность исходной задачи не превышает $2 \cdot L$. Эти примеры очень простые, поэтому не будем приводить их общего описания (т. е. описания для произвольного заданного L), а рассмотрим только минимальный пример для следующего значения, $L = 5$ (как и ранее в [1], минимальность понимается для исходного значения, взятого для возведения в квадрат – причём именно этот квадрат подаётся на вход рассматриваемой задачи). Итак,

- язык, взятый для формирования примера, – 1363; в других вариантах записи его же получается

$$[0\ 1\ 4\ 6\ 8\ 10] \quad \text{или} \quad 10101010011;$$

тогда:

- его квадрат (считаем именно его исходным языком для этой задачи) –

$$10101010111\,1111110111 \qquad (\text{т. е. } 1400823);$$

- потенциальные корни – $[0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10]$ (т. е. все возможные значения в заранее известном промежутке, здесь – от 0 от 10), в другой записи – 2047.

Перебором на компьютере можно убедиться, что на расстояниях от максимальной вершины от 0 до 4 корней нет; ближайший корень – $[0\ 1\ 4\ 6\ 8\ 10]$, находящийся на расстоянии 5.

Вообще, приведённый ваше способ даёт возможность конструировать такие примеры, в которых ближайший корень находится на любом заданном заранее расстоянии от максимальной вершины гиперкуба. При этом если расстояние должно быть равно L, то размерность примера (т. е. значение максимального потенциального корня) равна, как несложно убедиться, $2 \cdot L$.

3. О возможном увеличении координаты имеющегося корня

В этом разделе приводится дополненное доказательство теоремы о возможном увеличении одной координаты имеющегося корня. Вариант теоремы был опубликован в [1] – однако приведённое там доказательство нельзя назвать полным [7].

[7] Конкретно – с формальной точки зрения возможна такая ситуация, когда выбирается одно значение i_1 (i в обозначениях приведённого в [1] доказательства) – в то время как условие $\overbrace{k, i_2} \in \mathcal{T}(A)$ выполняется

Теорема 1. Пусть

$$B = (b_1 b_2 \dots b_N) \in 2^N \quad -$$

некоторый корень уравнения (2), т. е. $M = 2$. Пусть при этом

$$b_k = 0 \quad \text{для некоторого} \quad k \in \overline{1, N}.$$

Пусть также для каждого $i \in \overline{1, N} \setminus \{k\}$, такого что $b_i = 1$, выполнено условие

$$\widehat{k, i} \notin \mathcal{T}(A).$$

Тогда

$$B_{+k} = (b_1 b_2 \dots b_N)_{+k}$$

также является корнем уравнения (2).

Доказательство. Очевидно, что $B^2 \subseteq (B_{+k})^2$. Поэтому если бы было выполнено неравенство $(B_{+k})^2 \neq A$ то мы получали бы обязательное выполнение условия $(B_{+k})^2 \supset A$. Поэтому существовал бы по крайней мере один разряд $i \in \overline{1, N} \setminus \{k\}$, такой что:

- $b_i = 1$,
- и при этом при возведении в квадрат у произведения появлялся бы *новый* разряд, равный 1 («новый» – отсутствовавший в B, т. е. принимающий в B значение, равное 0).

Отметим, что мы не утверждаем единственность такого значения i, поэтому обозначим множество, состоящее из всех таких i, как I.

А поскольку мы добавили к B только один разряд, а именно k-й, то последний факт возможен только при условии

$$\left(\exists\, i \in I\right) \left(\widehat{k, i} \in \mathcal{T}(A)\right),$$

что противоречит условию теоремы. □

4. О путях в N-мерном гиперкубе потенциальных корней

В этом разделе рассматриваются вопросы, связанные с возможными путями в N-мерном гиперкубе потенциальных корней. Как и ранее, будем рассматривать 1-буквенный алфавит – хотя, как и многие вопросы настоящей статьи, материал этого раздела может быть обобщён на случай произвольного конечного алфавита. Отметим, что материал этого раздела связан с возможным описанием *эвристических* алгоритмов задачи извлечения корня из языка.

Итак, у нас есть N-мерный гиперкуб, каждая его точка (а их всего 2^N) – это некоторое подмножество множества потенциальных корней. Придадим *самим точкам* следующие значения:

- значение 1 – это корень (в примере на рис. 1 их всего 21);
- значение 2 устанавливаем в том случае, когда квадрат этого языка включает в себя требуемый (исходный) язык как *собственное подмножество*; отметим, что таковой (имеющей значение 2) должна являться *максимальная* точка гиперкуба, т. е. $[1^N] = (1, 1, \dots, 1)$ – в противном случае рассматриваемая задача неинтересна:

не для i_1, а для некоторого $i_2 \neq i_1$. Понятно, что конкретых примеров мы привести не можем, поскольку доказательство теоремы ведётся от противного, т. е. показывается невозможность подобных ситуаций.

– если это значение равно 1, то искомый корень известен;
– если это значение равно 0 (см. ниже), то и все значения вершин гиперкуба равны 0 и корней нет;
– значения, равного ∅ (также см. ниже это обозначние), у максимальной точки гиперкуба быть не может – это несложно доказывается а основе введённых определений.

- значение 0 устанавливаем в том случае, когда квадрат этого языка является *собственным подмножеством* требуемого (исходного) языка аналогично предыдущему пункту, таковой (имеющей значение 0) должна являться *минимальная* точка гиперкуба, т. е. $[0^N] = (0, 0, \ldots, 0)$;
- значение ∅ – оставшийся вариант: ни одно из рассматриваемых двух множеств (квадрат языка, соответствующего точке гиперкуба, а также исходный язык) не является подмножеством другого [8].

Будем рассматривать все простые пути по рёбрам гиперкуба из точки $[1^N]$ в точку $[0^N]$ [9]. При этом у любого такого пути:

- сначала несколько значений, установленных для вершин, равны 2;
- затем, *возможно*, равны либо 1 либо ∅ (одновременное включение в путь как 1, так и ∅, невозможно – этот факт можно несложно доказать аналогично сказанному выше;
- в конце пути несколько значений равны 0.

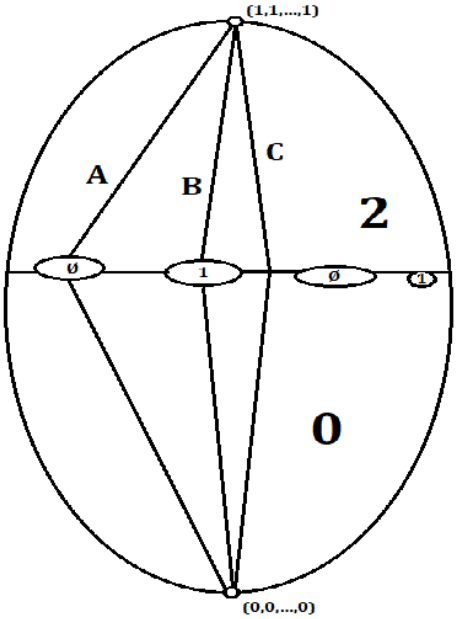

Рис. 2. Схема различных путей в графе булеана.

На приведённом рис. 2:

- при наличии вершин со значениями ∅ путь помечен A;
- при наличии вершин со значениями 1 путь помечен B;
- при отсутствии вершин как со значениями ∅, так и со значениями 1, путь помечен C.

[8] Вариант совпадения этих множеств уже был рассмотрен: в этом случае значение равно 1, то есть это один из возможных корней.

[9] Всего таких путей N! – поэтому вряд ли для какой-либо задачи представляют интерес варианты алгоритма, связанного с их полным перебором (brute force method).

Далее будем рассматривать только пути вида C (если таковые имеются). Переход из значения 2 в значение 0, как и любое другое единичное движение, осуществляется удалением из рассматриваемого множества какого-либо из имеющихся в нём потенциальных корней; такой потенциальный корень (для которого существует хоть один подобный путь по вершинам гиперкуба) назовём *важным*. Заметим, что задача, заключающаяся в поиске всех важных потенциальных корней, конечно, может быть решена простым переборным алгоритмом (brute force method) – но мы в настоящей статье формулируем задачу, заключающуюся в описании быстрых *эвристических* алгоритмов для неё.

В [1] была кратко сформулирована гипотеза, заключающаяся в том, что множество всех важных потенциальных корней (рассматриваемое как язык) – это один из корней из исходного языка. Автор располагает примером, опровергающим эту гипотезу, – но публикация примера была бы очень объёмна и поэтому вряд ли интересна для настоящей статьи [10]; действительно, как уже было отмечено, всего существует 1287 вариантов подмножеств из 8 потенциальных корней [11], но лишь 10 из них являются квадратными корнями из заданного языка.

5. О применении оптимизационных задач на графах

Рассматривавшийся выше в разделе 2 пример с 13 потенциальными корнями (в этом примере – все возможные значения в заранее известном промежутке от 0 от 12), по-видимому, малоинтересен:

- в нём имеется только одно невозможное значение возможной суммы потенциальных корней (а именно, значение 5);
- при этом, в отличие от материала следующего раздела, непонятно, как обобщить этот пример на произвольную размерность; отметим, что при возможном таком обобщении нам интерсны:
  - во-первых, существование решения (в нашем примере оно есть),
  - и, во-вторых, линейная зависимость числа табуированных пар от размерности задачи.

В связи с этим рассмотрим сначала тривиальные определения, потом – построения, связанные с применением задачи 2-SAT, и уже после этого более сложный пример.

Итак, для очевидным образом определяемого *графа табуированных пар*, в котором:

- множество вершин V является множеством потенциальных корней,
- а множество рёбер E соответствует всем табуированным парам.

Несложно убедиться, что такой граф является дополнением к графу, рассматривавшемуся в [1, разд. III], один из примеров приведён на [1, рис. 3].

Рассмотрим тривиальный пример графа табуированных пар – соответствующий языку, приведённому в разделе 2 [12], см. рис. 3.

Далее мы в N-мерном гиперкубе рассматриваем «нетабуированные» точки булеана – т. е. такие точки, в координаты которых не входит ни одна табуированная пара; сто́ит отметить, что для каждой табуированной пары размерность получаемой «табуированной

[10] Если совсем кратко – мы ссылаемся на пример, рассмотренный на рис. 1.

[11] Можно считать, что горизонтальная линия на рис. 2, прерываемая несколькими овалами – подмножествами со значениями 1 и ∅, это и есть множество вершин гиперкуба, каждая из которых имеет ровно 8 таких координат, каждая из которых равна 1.

[12] Важно отметить, что тривиальным пример является с точки зрения материала настоящего раздела – но, как уже понятно, не с точки зрения материала раздела 2.

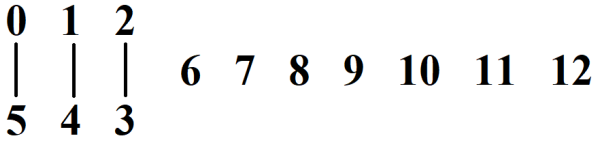

Рис. 3. Очень простой пример графа табуирванных пар.

гиперплоскости» равна $N-2$. На основе результатов теоремы 1 можно утверждать, что для поиска хоть какого-то решения исходной задачи можно искать «нетабуированные» точки булеана, лежащие ближе всего к максимальной вершине. Решение – если хотя бы одно имеется – обязательно будет среди тех «нетабуированных» точек, для которых не существует соседней «нетабуированной», лежащей при этом ближе к максимальной вершине.

Однако простой подсчёт даже при закреплении каких-то конкретных значений элементов табуированной пары даёт общее число элементов табуированной плоскости 2^{N-2}; интуитивно понятно – даже без проведения точных подсчётов возможного числа соседних элементов – что подобное знчение вряд ли может дать полиномиальный алгоритм. И, по-видимому, такой подсчёт вообще не может являться доказательством несуществованя простого (полиномиального) алгоритма, заключающегося именно в переборе точек, лежащих рядом (т. е. на расстоянии 1) с точками, соответствующими табуированным плоскостям. А следует такое несуществование из примеров, приведённых в разделе 6.

Дальнейшие построения связаны с формированием на основе графа табуированных пар задач дискретной оптимизации: задачи построения независимых множеств вершин графа (в том числе максимальных множеств), а также, в упрощённом случае, задачи 2-SAT – см. [7] и мн. др. При формировании соответствующих задач (2-SAT в первую очередь) отметим, что все входящие в получающуюся 2-КНФ переменные *не* содержат отрицаний. Поэтому возможный переборный алгоритм, сформированный на основе такой 2-SAT, представляет собой:

- во-первых, поиск всех максимальных независимых подмножеств множества вершин V;
- и, во-вторых, рассмотрение всех точек N-мерного гиперкуба, соответствующих уменьшению какой-либо координаты какой-либо вершины одного из этих подмножеств.

Отметим, немного забегая вперёд, что примеры следующего раздела показывают, что количество соответствующих вершин (необходимых для рассмотрения) может быть экспоненциально – относительно размера исходной задачи.

Рассмотрим более интересный пример. В качестве *исходного* языка выберем

$$[0\ 1\ 3\ 6\ 10];$$

квадрат этого языка (*входной* язык рассматриваемой задачи) –

$$[0\ 1\ 2\ 3\ 4\ 6\ 7\ 9\ 10\ 11\ 12\ 13\ 16\ 20],$$

множество потенциальных корней –

$$[0\ 1\ 2\ 3\ 5\ 6\ 8\ 10].$$

Несложно получаемое множество табуированных пар сразу изобразим на графе, см. рис. 4.

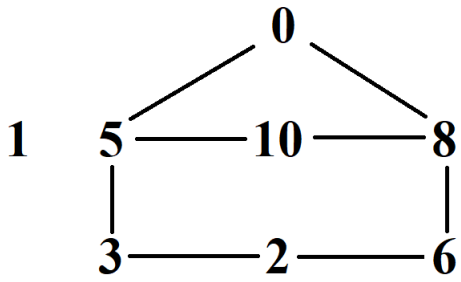

Рис. 4. Более сложный пример графа табуирванных пар.

Из предыдущего материала следует, что существуют две «максимальные» вершины N-мерного куба (т. е. такие вершины, увеличение любой равной 0 координаты которых приведёт к появлеию хотя бы одной табуированной пары); они являются решениям как задачи построения максимального независимого множества вершин графа, так и задачи 2-SAT. Записи этих вершин приведём подробные [13], обозначая координату, соответствующую птенциальному корню q, записью x_q. Итак, координаты двух вершин следующие:

$$x_0 = 1,\ x_1 = 1,\ x_2 = 0,\ x_3 = 1,\ x_5 = 0,\ x_6 = 1,\ x_8 = 0,\ x_{10} = 1$$
$$\text{и} \quad x_0 = 0,\ x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_5 = 1,\ x_6 = 0,\ x_8 = 1,\ x_{10} = 0.$$

Таким образом, в рассматриваемом примере для поиска «максимальных» корней достаточно проверить только 2 точки 8-мерного гиперкуба; первая из них в точности соответствует исходному языку, а вторая при проверке даёт ошибку: квадрат построенного языка не равен языку исходному.

Важно, что в рассмотренном примере мы специально не отмечали разницу между двумя оптимизационными задачами на графах, т. е. между 2-SAT и существенно более сложной задачей построения независимых множеств вершин графа. Причина такова. Основная цель получения *необходимых* условий для задачи извлечения корня заключается в том, чтобы *на примерах* показать невозможность краткого решения основной задачи, т. е. невозможность построения полиномиального алгоритма, использующего лишь информацию о табуированных парах потенциальных корней. А показать это можно и на очень простых графах – которые, однако, соответствуют экспоненциально большому числу вершин гиперкуба, которые необходимы для перебора; см. следующий раздел.

А в завершение этого раздела приведём очевидную *формальную* запись задачи 2-SAT – которая выше фактически была решена; для входящих в задачу переменных будем пользоваться теми же обозначениями.

$$(x_0 \,|\, x_5)\ \&\ (x_5 \,|\, x_3)\ \&\ (x_3 \,|\, x_2)\ \&\ (x_2 \,|\, x_6)\ \&\ (x_6 \,|\, x_8)\ \&\ (x_8 \,|\, x_0)\ \&\ (x_5 \,|\, x_{10})\ \&\ (x_{10} \,|\, x_8)\,.$$

Понятно, что реальные примеры решения задачи построения независимых множеств вершин графа существенно сложнее, чем приведённые в этом разделе. Однако нам нет необходимости рассматривать такие более сложные примеры: нам достаточно существование таких примеров, в которых при рассмотрении только решения 2-SAT требуется перебор вершин N-мерного гиперкуба, число которых от размерности задачи зависит экспоненциально; это сделано в следующем разделе.

[13] Выше в настоящей статье такой способ записи не использовался.

6. Примеры про несуществование простого алгоритма: сколь угодно большая размерность для степени 2

В этом разделе приведены основные результаты статьи: показана возможность конструирование примеров на тему несуществования простого алгоритма извлечения корня степени 2 – причём примеров для сколь угодно большой (но заранее известной) размерности; как и всюду в настоящей статье, простыми мы называем алгоритмы, основанные только на информации о табуированных потенциальных корнях.

Для осуществления описания подобных примеров возникают следующие вопросы. Первый, более простой (но, видимо, неочевидный) – про существование описания задачи, для которой построенное число «максимальных» точек от размерности этой задачи зависит *экспоненциально*. Второй, более сложный, – когда дополнительно к этому сформулированная таким образом задача *должна иметь решение* (т. е. для заданной задачи корень нужной степени должен извлекаться). Описанный в этом разделе пример (на самом деле – целое семейство примеров) даёт положительный ответ *сразу на оба* эти вопроса.

Итак, сначала рассмотрим следующий пример. Исходный язык такой:

$$[0\ 1\ 3\ 5\ 8\ 9\ 10],$$

его квадрат (заданный, входной язык) –

$$[0\ 1\ 2\ 3\ 4\ 5\ 6\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 20] \quad = \quad \overline{0,6} \cup \overline{8,20}$$

(т. е. все значения от 0 до 20, за исключением 7), язык потенциальных корней –

$$[0\ 1\ 2\ 3\ 4\ 5\ 6\ 9\ 8\ 9\ 10] \quad = \quad \overline{0,10}$$

(т. е. все возможные значения для рассматриваемой размерности входного языка, равной 20), в наших обозначениях $N = 11$. Формулировку задачи можно изобразить в виде таблицы, приведённой на рис. 5.

	0	1	2	3	4	5	6	7	8	9	10
0	1	1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	0	1	1	1	1
2	1	1	1	1	1	0	1	1	1	1	1
3	1	1	1	1	0	1	1	1	1	1	1
4	1	1	1	0	1	1	1	1	1	1	1
5	1	1	0	1	1	1	1	1	1	1	1
6	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1	1

Рис. 5. Возможная формулировка задачи для $M = 2$, $n = 10$.

А решение задачи – фактически приведённое выше в качестве исходного языка – можно изобразить в виде таблицы, приведённой на рис. 6; в ней (в её клетках) имеются все числа от 0 до 20, кроме 7.

	0	1	3	5	8	9	10
0	0	1	3	5	8	9	10
1	1	2	4	6	9	10	11
3	3	4	6	8	11	12	13
5	5	6	8	10	13	14	15
8	8	9	11	13	16	17	18
9	9	10	12	14	17	18	19
10	10	11	13	15	18	19	20

Рис. 6. Решение приведённой задачи для $M = 2$, $n = 10$.

В рассмотренном примере граф табуированных пар содержит рёбра

$$\{\overbrace{0,7},\ \overbrace{1,6},\ \overbrace{2,5},\ \overbrace{3,4}\},$$

которые удобно обозначать так же, как и сами пары. Итак, если размерность входной задачи считать равной 20 (что естественно), то число этих рёбер равно $4 = \frac{20}{4} - 1$. Поэтому число «максимальных» точек N-мерного гиперкуба (как уже было отмечено, в рассматриваемом примере $N = 11$), необходимых для проверки наличия корня, равно $2^4 = 16$: для 3 координат 11-мерного гиперкуба, не входящих в табуированные пары, нужно установить значения 1 (иначе будет нарушено требование максимальности для формируемой точки гиперкуба), а для остальных 4 пар координат (отметим, что каждая координата входит в единственную табуированную пару) мы должны рассмотреть по 2 возможности (пары значений (0,1) и (1,0)), которые можно устанавливать независимо от остальных значений пар.

Покажем, как для ранеее использованного обозначения n (в примере выше было $n = 10$) получать необходимые примеры *решаемых* задач при задании больших значений n. Для простоты будем считать n чётным, причём $n \geqslant 10$; пусть $n = 2q$. Входными языками примеров будут

$$\overline{0, n-4} \ \cup\ \overline{n-2, 2n}.$$

Для некоторого конкретного $n = 2q$ граф табуированных пар содержит рёбра

$$\{\overbrace{0, n-3},\ \overbrace{1, n-2},\ \ldots,\ \overbrace{q-3, q},\ \overbrace{q-2, q-1}\},$$

причём только их; всего таких рёбер $q - 3$. Поэтому в получаемых примерах для любого выбранного чётного $n \geqslant 10$ из-за независимости рассмотрения пар существует

$$2^{\frac{n}{2}-3} \tag{4}$$

вариантов значений точек $n+1$-мерного гиперкуба, необходимых для переборного поиска решения – в случае, когда такой поиск основан только на информации о табуированных парах.

7. Пример для степени 3

Совершенно аналогично [1, разд. VIII], частный случай для степени 3 мы рассмотрим очень кратко; однако есть основания полагать, что:

- во-первых, на основе приведённого далее примера можно построить примеры для степени 3 сколь угодно большой (но, конечно, заранее известной) размерности – примеры, показывающие несуществование простого алгоритма извлечения корня 3-й степени [14];
- во-вторых, примеры обобщаются и на произвольную степень $M > 3$.

Итак, пусть исходное множество степеней буквы таково:

$$[0\ 1\ 4\ 7\ 10\ 16\ 17\ 18]\,.$$

На его основе приведём разложения чисел от 0 до 54 в виде суммы трёх чисел, входящих в исходное множество (если вариантов несколько, то приводим только один):

$0 = 0+0+0$	$1 = 0+0+1$	$2 = 0+1+1$	$3 = 1+1+1$	$4 = 0+0+4$
$5 = 0+1+4$	$6 = 1+1+4$	$7 = 0+0+7$	$8 = 0+1+7$	$9 = 1+1+7$
$10 = 0+0+10$	$11 = 0+1+10$	$12 = 1+1+10$	13 ...	$14 = 0+4+10$
$15 = 1+4+10$	$16 = 0+0+16$	$17 = 0+0+17$	$18 = 0+0+18$	$19 = 0+1+18$
$20 = 0+10+10$	$21 = 1+10+10$	$22 = 0+4+18$	$23 = 1+4+18$	$24 = 4+10+10$
$25 = 0+7+18$	$26 = 1+7+18$	$27 = 0+10+17$	$28 = 0+10+18$	$29 = 1+10+18$
$30 = 10+10+10$	$31 = 7+7+17$	$32 = 7+7+18$	$33 = 0+16+17$	$34 = 0+17+17$
$35 = 0+17+18$	$36 = 10+10+16$	$37 = 10+10+17$	$38 = 10+10+18$	$39 = 1+10+18$
$40 = 4+18+18$	$41 = 7+17+17$	$42 = 7+17+18$	$43 = 7+18+18$	$44 = 10+17+17$
$45 = 10+17+18$	$46 = 10+18+18$	47 ...	$48 = 16+16+16$	$49 = 16+16+17$
$50 = 16+16+18$	$51 = 17+17+17$	$52 = 17+17+18$	$53 = 17+18+18$	$54 = 18+18+18$

Таким образом мы в рассмотренном примере получаем 2 «пропущенных» значения для *входного* языка – что даёт основания предполагать возможность описать на основе этого примера примеров сколь угодно большой размерности для этой же степени 3. Однако оценок, аналогичных (4), мы в настоящей статье делать не будем.

Заключение

Итак, в настоящей статье мы, по-видимому, закончили рассмотрение тех вопросов, связанных с извлечением корня 2-й степени из языка над 1-буквеным алфавитом, которые можно получить на основе одних лишь табуированых пар потенциальных корней. В частности, мы показали, что на основе информации лишь о потенциальных корнях – и рассмотрении специальных вершин гиперкуба потенциальных корней – нельзя построить полиномиальный алгоритм извлечения корня из языка. Однако мы не утверждаем несуществование такого алгоритма: возможно, такие алгоритмы и существуют, но они долждны получаться на основе более полной информации об исходной задаче.

В качестве одного из направлений развития темы, рассмотренной в настоящей статье, укажем связь между рассмотренными выше вершинами гиперкуба потенциальных корней (0, 1, 2 и $\varnothing$) – и практически такими же обозначениями, применявшимися в нескольких относительно недавних статьях Г. Г. Рябова [15] для задач, связанных с ДНФ и проблемами их минимизации; упомянем статьи [10–12]. На вершинах гиперкуба потенциальных корней, как и на т. н. кубантах в этих статьях, может быть построена специальная *алгебра*, которую

[14] Табуированные тройки – аналоги табуированных пар для степени 3 – также были кратко рассмотрены в заключении процитированной статьи.

[15] К сожалению, исследование этой тематики, по-видимому, не продолжает никто.

можно использовать в дальнейших построениях и задачах – как чисто теоретических, так и вычислительных.

Работа частично поддержана грантом научной программы китайских университетов "Higher Education Stability Support Program" (раздел "Shenzhen 2022 – Science, Technology and Innovation Commission of Shenzhen Municipality").

Литература

1. Мельников Б.Ф., Мельникова А.А. О задачах извлечения корня из заданного конечного языка // International Journal of Open Information Technologies. 2023. Vol. 11, No. 5. P. 1–14.
2. Melnikov B.F., Korabelshchikova S.Yu., Dolgov V.N. On the task of extracting the root from the language // International Journal of Open Information Technologies. 2019. Vol. 7, No. 3. P. 1–6.
3. Мельников Б.Ф. Полиномиальный алгоритм построения оптимального инверсного морфизма // International Journal of Open Information Technologies. 2023. Vol. 11, No. 6. P. 1–10.
4. Stockmeyer L.J. The polynomial-time hierarchy // Theoretical Computer Science. 1976. Vol. 3, No. 1. P. 1–22.
5. Chandra A.K., Kozen D., Stockmeyer L.J. Alternation // Journal of the ACM. 1981. Vol. 28, No. 1. P. 114–133.
6. Immerman N. TDescriptive Complexity. Berlin: Springer, 1999. 284 p.
7. Hromkovič J. Theoretical Computer Science: Introduction to Automata, Computability, Complexity, Algorithmics, Randomization, Communication, and Cryptography. Berlin: Springer, 2003. 323 p.
8. Hromkovič J. Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics. Berlin: Springer, 2004. 547 p.
9. Мельников Б.Ф. Полурешётки подмножеств потенциальных корней в задачах теории формальных языков. Часть I. Извлечение корня из языка // International Journal of Open Information Technologies. 2022. Vol. 10, No. 4. P. 1–9.
10. Рябов Г.Г. О четверичном кодировании кубических структур // Вычислительные методы и программирование. 2009. Т. 10. № 3. С. 340–347.
11. Рябов Г.Г. Хаусдорфова метрика на гранях N-мерного куба // Фундаментальная и прикладная математика. 2010. Т. 16. № 1. С. 151–155.
12. Рябов Г.Г. Полиморфизм символьных троичных матриц и генетическое пространство кратчайших K-путей в N-кубе // International Journal of Open Information Technologies. 2015. Vol. 3, No. 7. P. 1–11.

Мельников Борис Феликсович, д.ф.-м.н., профессор, факультет Вычислительной математики и кибернетики, Совместный университет МГУ–ППИ в Шэньчжэне (Шэньчжэнь, Китай)

DOI: 10.14529/cmse230X0Z

On the non-existence of a simple version of the polynomial algorithm for extracting the root from the language

© 2023 B.F. Melnikov[1]

[1] *Shenzhen MSU – BIT University (No. 1, International University Park Road, Dayun New Town, Longgang District, Shenzhen, Guangdong Province, 518172, China)*

E-mail: bormel@mail.ru

Received: 07.10.2023

For the usual operation of concatenation of words considered as multiplication, the concatenation of languages is obviously determined, and on the basis of the last operation, the degree of the language and the root of a given degree (if available) is determined. When describing algorithms for constructing a language that is a root of degree M from a given language, so called potential roots are of great importance: these are the words (not the languages) whose considered M-th degree is included in a given language. It is easily to show that all potential roots for a given language are constructed using a polynomial algorithm.

This task, apparently, is not simplified when considering words and languages over the 1-letter alphabet, which is done in this paper.

So called taboo pair of potential roots is a pair whose word concatenation is not included in the language. In previous publications on the topic of describing algorithms for extracting roots from a language, the hypothesis arose that a polynomial algorithm for extracting a root from a language can be described on the basis of considering the set of taboo pairs only, by iterating over specially described subsets of the set of potential roots. This paper shows, that such an algorithm (called "simple") is impossible, i.e., if there is a polynomial algorithm for extracting the root from the language, then this algorithm must use some additional information.

Keywords: formal languages, concatenation of languages, degree of the language, root of the language, polynomial algorithms.

FOR CITATION

Melnikov B.F. On the non-existence of a simple version of the polynomial algorithm for extracting the root from the language. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. X, no. Y. P. Z1–Z2. (in Russian) DOI: 10.14529/cmse230X0Z.

References

1. Melnikov B.F., Мельникова A.A. On the problems of extracting the root from a given finite language // International Journal of Open Information Technologies. 2023. Vol. 11, No. 5. P. 1–14. (In Russian.)

2. Melnikov B.F., Korabelshchikova S.Yu., Dolgov V.N. On the task of extracting the root from the language // International Journal of Open Information Technologies. 2019. Vol. 7, No. 3. P. 1–6.

3. Melnikov B.F. Полиномиальный алгоритм построения оптимального инверсного морфизма // International Journal of Open Information Technologies. 2023. Vol. 11, No. 6. P. 1–10. (In Russian.)

4. Stockmeyer L.J. The polynomial-time hierarchy // Theoretical Computer Science. 1976. Vol. 3, No. 1. P. 1–22.

5. Chandra A.K., Kozen D., Stockmeyer L.J. Alternation // Journal of the ACM. 1981. Vol. 28, No. 1. P. 114–133.

6. Immerman N. TDescriptive Complexity. Berlin: Springer, 1999. 284 p.

7. Hromkovič J. Theoretical Computer Science: Introduction to Automata, Computability, Complexity, Algorithmics, Randomization, Communication, and Cryptography. Berlin: Springer, 2003. 323 p.

8. Hromkovič J. Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics. Berlin: Springer, 2004. 547 p.

9. Melnikov B.F. Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part I. Extracting the root from the language // International Journal of Open Information Technologies. 2022. Vol. 10, No. 4. P. 1–9. (In Russian.)

10. Ryabov G.G. On the quaternary coding of cubic structures // Computational methods and programming. 2009. Vol. 10. No. 3. P. 340–347. (In Russian.)

11. Ryabov G.G. Hausdorff metric on the faces of an N-dimensional cube // Fundamental and Applied Mathematics. 2010. Vol. 16. No. 1. P. 151–155. (In Russian.)

12. Ryabov G.G. Polymorphism of symbolic ternary matrices and genetic space of shortest K-paths in an N-cube // International Journal of Open Information Technologies. 2015. Vol. 3, No. 7. P. 1–11. (In Russian.)