Полурешётки подмножеств потенциальных корней в задачах теории формальных языков. Часть I. Извлечение корня из языка

Б. Ф. Мельников

***Аннотация*—В статье мы рассматриваем все возможные подмножества множества потенциальных корней, образующие в некоторых ситуациях полурешётки – по пересечению и / или по объединению. Такие структуры возникают в двух похожих задачах теории формальных языков. Конкретно, для некоторого заданного конечного языка мы рассматриваем задачу извлечения корня заданной степени, а также задачу построения оптимального инверсного морфизма – где оптимальность можно определить, например, как длину максимального слова языка, являющегося инверсным морфическим образом.**

В обоих приведённых случаях необходимо построить множество так называемых потенциальных корней – т. е. таких слов рассматриваемого алфавита, для каждого из которых некоторая его степень входит в исходный язык. При этом важно отметить, что построение множества всех потенциальных корней может быть выполнено с помощью несложного полиномиального алгоритма. Экспоненциальные алгоритмы для обеих задач очевидны: надо просто перебрать все подмножества множества этих потенциальных корней, и среди этих подмножеств выбрать подходящее. Поэтому проблема состоит в описании возможных полиномиальных алгоритмов для этих задач.

Для обеих указанных задач вызывает интерес возможное существование двух полурешёток, имеющихся на предварительно построенном множестве подмножеств потенциальных корней.

Среди прочего, мы в статье приводим формулировку одной важной гипотезы теории формальных языков – при выполнении которой мы можем утверждать, что специальное подмножество множества языков, каждый из которых является инверсным морфическим образом заданного языка, образует не только полурешётку по объединению, но и полурешётку по пересечению (а, следовательно, решётку).

Первая часть статьи содержит мотивацию и определение основных понятий. После этого она включает материал, в большей степени посвящённый первой из рассматриваемых задач – задаче извлечения корня из языка.

***Ключевые слова*—формальные языки, итерации языков, морфизмы, инверсные морфизмы, бинарные отношения, полурешётки, алгоритмы.**

I. ВВЕДЕНИЕ

В настоящей статье мы продолжаем тематику наших предыдущих работ [1] (в первую очередь), а также [2], [3], [4], [5], [6], [7], [8]; кроме того, связанные вопросы приведены в [9], [10], [11]. Мы рассматриваем все возможные подмножества множества потенциальных корней (впервые определённые нами в [1]), образующие *в некоторых ситуациях* полурешётки – по пересечению и / или по объединению. Такие структуры возникают, например, в двух рассматриваемых далее схожих задачах [1] теории формальных языков.

Конкретно, для некоторого заданного конечного языка мы рассматриваем:

- задачу (задачи) извлечения корня заданной степени [2]
- и задачу (задачи) построения оптимального инверсного морфизма – где оптимальность можно определить, например, как *длину максимального слова языка, являющегося инверсным морфическим образом* [3,4].

Теперь – чуть подробнее про сами задачи. Для заданного языка $A \subseteq \Sigma^*$:

- извлечь корень n-й степени – это найти такой язык (такие языки) $B \subseteq \Sigma^*$, что

$$A = B^n;$$

- построить оптимальный инверсный морфизм – это найти такой язык (такие языки) $D \subseteq \Delta^*$ (здесь Δ – некоторый новый алфавит) и морфизм

$$h : \Delta \to \Sigma^*,$$

что:

  - D в качестве подмножества (возможно, несобственного) содержит некоторый максимальный префиксный код (над алфавитом Δ);
  - $A = h(D)$;
  - и, кроме того, язык D оптимален согласно некоторому особо формулируемому критерию –

[1] Точнее – группах задач; подробнее см. далее.

[2] Либо максимально возможной степени. Разницы практически нет: в конкретном частном случае задачи максимально возможная степень ограничена длиной максимального слова заданного языка.

[3] В предыдущих публикациях мы уже отмечали, что эту оптимальность можно определять и другими способами, различными – но при этом «естественными»; некоторые из них описаны далее в разделе II (а также в разделе XI, часть II). Отметим, что можно доказать, что рассматриваемые нами задачи – в случае различных определений этой оптимальности – получаются практически эквивалентными.

[4] Можно специально создать некоторую аналогию для сноски, относящейся к предыдущему пункту (к извлечению корня) – т. е. описать *шаг*, строящий «более хороший» инверсный морфизм При этом последовательность таких шагов также даст оптимальный инверсный морфизм – что можно доказать описанием *на множестве таких инверсных морфизмов* специальной полурешётки. Дугами на схемах таких полурешёток являются вышеупомянутые шаги.

Однако отметим, что такая полурешётка *не будет* полурешёткой подмножеств потенциальных корней – и поэтому её вряд ли сто́ит подробно рассматривать в настоящей статье. Однако в одной из следующих публикаций мы предполагаем вернуться к построению таких полурешёток и описанию их свойств.

Статья получена 31 января 2022 г.

Борис Феликсович Мельников, Университет МГУ – ППИ в Шэньчжэне (bf-melnikov@yandex.ru).

среди всех языков, удовлетворяющим обоим предыдущим условиям.

В обоих приведённых случаях желательно (по-видимому – даже необходимо) построить множество так называемых потенциальных корней – т. е. таких слов рассматриваемого алфавита, для каждого из которых некоторая его степень (причём в случае первой рассматриваемой нами задачи – именно n-я степень) входит в исходный язык A. При этом важно отметить, что *построение множества всех потенциальных корней может быть выполнено с помощью несложного полиномиального алгоритма* [5].

Экспоненциальные алгоритмы для обеих групп задач очевидны: надо просто перебрать все подмножества множества этих потенциальных корней, и среди этих подмножеств выбрать подходящее (подходящие). Поэтому проблема состоит в описании возможных *полиномиальных* алгоритмов для этих задач.

Мы уже упоминали, что рассматриваем не две задачи, а *две группы задач*, поскольку в обоих случаях имеется несколько вариантов для требуемого в задаче ответа:

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

Однако несложно понять, что все эти варианты очень близки между собой – и поэтому мы далее:

- либо не будем уточнять конкретную задачу,
- либо будем иметь в виду «построение любого» (корня из языка либо инверсного морфизма) [7].

И, как следует из заголовка настоящей статьи, для её темы особенно важны такие вопросы. Для обеих указанных задач (групп задач) вызывает интерес возможное существование двух полурешёток [12], имеющихся на предварительно построенном множестве подмножеств потенциальных корней: полурешётки по пересечению и полурешётки по объединению. Эти вопросы мы и будем рассматривать далее в первую очередь.

Также из заголовка настоящей статьи следует, что она разбита на *две* части; кроме того, как уже понятно по тексту введения, мы в статье рассматриваем *две* задачи. Однако вряд ли возможно «разделить» материал статьи так, чтобы каждая из частей «соответствовала бы своей задаче» [8]: в задачах имеется очень много пересекающихся подзадач, которые мы и рассматриваем в настоящей статье.

[5] Здесь действительно имеется тривиальный полиномиальный алгоритм: мы для всех слов исходного языка A (пусть u) просто проверяем все непустые префиксы этого слова (пусть v) – проверяем на предмет того, совпадает ли нужная степень слова v с исходным u.

[6] При этом очень важно отметить *связь* вариантов *метрики* с вариантами *частичного порядка* – в нашем случае на множестве языков. Более того, можно даже указать не менее трёх «уровней понимания»:

- интуитивно эта связь, по-видимому, понятна;
- немного подробнее – см. далее в разделе XI (часть II);
- а ещё подробнее, с «чисто математическими» объяснениями, комментариями и дальнейшими ссылками, можно почитать, например, на `https://math.stackexchange.com/questions/2405468/is-there-a-name-for-the-partial-order-of-metrics-defined-by-induced-topology-sub/2405536`.

[7] Т. е., по-видимому, самую простую (в сравнении с другими) задачу.

[8] «Частично» мы это всё-таки сделали.

Структура статьи такова. Разделы II и III – необходимые для настоящей статьи предварительные све́дения; в отличие от нашего обычного варианта изложения ([1], [6] и мн. др.), мы в настоящей статье размещаем такие разделы до мотивации, т. е. сразу после введения. Конкретно, в разделе II мы приводим основные используемые в статье понятия – связанные, в первую очередь, с морфизмами и итерациями языков. В разделе III мы приводим информацию, связанную с максимальными префиксными кодами – в том числе вводим очень важное для дальнейшего изложения определение т. н. расширенных максимальных префиксных кодов.

После этого, в разделе IV, посвящённом именно мотивации, мы приводим возможные варианты применения двух рассматриваемых в статье задач – в более сложных задачах теории формальных языков; среди этих более сложных задач, конечно же, наиболее важной является задача специальной *переформулировки* условия равенства $\mathrm{P} = \mathrm{NP}$.

Разделы V и VI практически полностью посвящены первой из рассматриваемых задач – задаче извлечения корня из языка. А именно, в разделе V приводится недетерминированный полиномиальный алгоритм решения этой задачи. В разделе VI приведён ответ на один из основных вопросов настоящей статьи – об *отсутствии* в общем случае для первой задачи *обеих полурешёток*, т. е. полурешёток по пересечению и по объединению. Примеры, в основном, взяты из заключения статьи автора [7] – в настоящей статье мы приводим эти же примеры с дополнительными комментариями.

Разделы VII – XII помещены в часть II настоящей статьи – с подзаголовком «Построение инверсного морфизма»; эти разделы посвящены в большей степени второй задаче.

Часть начинается разделом VII – в котором мы формулируем несколько задач про последовательности возможных удалений потенциальных корней; отметим, что несмотря на то, что эти последовательности были определены в части I, почти все эти проблемы нельзя назвать относящимися только к задаче извлечения корня: те же самые проблемы (практически без изменений условия) можно сформулировать и для второй задачи – задачи построения (оптимального) инверсного морфизма. Также важно, что при числе потенциальных корней, равном M, число подобных последовательностей равно $M!$ – и поэтому нам *вряд ли интересны переборные варианты* решения таких задач.

В разделе VIII рассматривается несложный пример, который также можно отнести одновременно к обеим рассматриваемым в статье задачам. Показывается, что множество потенциальных корней может включать в себя «лишние» – т. е. те, которые *не* были заданы при определении исходного языка с помощью некоторого морфизма. Также в разделе кратко описывается связь множества потенциальных корней с определёнными и кратко исследованными в [6], [7] автоматами PRI и NSPRI.

В разделе IX рассматривается один из возможных алгоритмов, определяющих принадлежность некоторого заданного языка множеству морфизмов расширенных максимальных префиксных кодов; при этом сам язык-морфизм также заранее задан. Отметим, что предлагаемый

в разделе алгоритм, конечно же, не является решением второй задачи: ведь оба рассматриваемых языка заранее заданы. А приводимые в разделе X примеры (новый и, в первую очередь, продолжение рассмотренного ранее в разделе в разделе VIII) не только поясняют алгоритм раздела IX, но и показать возможные варианты применения этого алгоритма.

В разделе XI мы определяем два «естественных» частичных линейных порядка на множестве языков. При этом, конечно, возможно определение ещё нескольких (также «естественных») частичных линейных порядков – мы выбрали такие два, которые, во-первых, легко применимы в рассматриваемых нами задачах, и, во-вторых, строятся на разных принципах.

В разделе XII мы рассматриваем недетерминированный полиномиальный алгоритм решения второй задачи – задачи построения оптимального инверсного морфизма. В отличие от недетерминированного же алгоритма из раздела V (часть I), в рассматриваемой здесь задаче задан только один язык, и требуется найти второй язык над тем же алфавитом, такой что первый является *некоторым* морфизмом расширенного максимального префиксного кода – причём сам морфизм определяется этим выбираемым алгоритмом вторым языком.

Разделы XIII – XVII помещены в часть III настоящей статьи – с подзаголовком «Условие существования решётки»; эти разделы также посвящены второй задаче – в которой при формулируемом нами условии имеется специальная решётка на подмножестве множества потенциальных корней по пересечению и объединению.

Часть начинается разделами XIII, XIV и XV – в которых мы рассматриваем одну важную гипотезу теории формальных языков. Немного упрощая, эту гипотезу можно сформулировать следующим образом: очевидное достаточное условие выполнения отношения $\trianglelefteqslant$ является необходимым и достаточным. Конкретно, в разделе XIII приводятся конкретные «претензии» к нашей старой статье [2]. В разделе XIV более подробно сформулировано само это утверждение – в виде гипотезы. При её выполнении мы можем утверждать, что специальное подмножество множества языков, каждый из которых является инверсным морфическим образом заданного языка, образует не только полурешётку по объединению, но и полурешётку по пересечению – а, следовательно, решётку [9]. А в разделе XV приведённая гипотеза сформулирована в терминах бесконечных итерационных деревьев – рассматривавшихся в наших предыдущих публикациях ([6], [7] и др.).

Выполнение факта существования полурешётки по пересечению мы показываем в разделе XVI. А раздел XVII – заключение, мы в нём приводим информацию о проблемах для дальнейшего исследования.

II. ПРЕДВАРИТЕЛЬНЫЕ СВЕ́ДЕНИЯ: ОСНОВНЫЕ ПОНЯТИЯ, МОРФИЗМЫ И ИТЕРАЦИИ ЯЗЫКОВ

Сначала повторим несколько определений из [6] – эти определения очень важны для настоящей статьи. При этом оставляем и комментарии, приведённые в [6].

[9] Точнее – если множество подмножеств потенциальных корней обозначить Ω, то решётка образуется на специальном подмножестве множества Ω.

Для заданного слова $u \in \Sigma^*$:

- язык $\mathrm{pref}(u)$ определяется как множество префиксов слова u (включая несобственный, само u);
- язык $\mathrm{opref}(u)$ определяется как множество собственных префиксов слова u;
- язык $\mathrm{suff}(u)$ определяется как множество суффиксов слова u (включая несобственный, само u);
- язык $\mathrm{osuff}(u)$ определяется как множество собственных суффиксов слова u.

Для заданного языка $A \subseteq \Sigma^*$

$$\mathrm{pref}(A) = \bigcup_{u \in A} \mathrm{pref}(u);$$

аналогичным образом определяются $\mathrm{opref}(A)$, $\mathrm{suff}(A)$ и $\mathrm{osuff}(A)$.

Морфизм (использование этого термина согласовано с [13] [10]) – это отображение

$$h : \Delta^* \to \Sigma^*,$$

для которого:

- для каждой буквы $d \in \Delta$ её образ $h(d) \in \Sigma^*$ задаётся;
- а для каждого слова $d_1 d_2 \dots d_n \in \Delta^*$ полагаем

$$h(d_1 d_2 \dots d_n) = h(d_1)\, h(d_2) \dots h(d_n).$$

Нам будут очень важны морфизмы, соответствующие конечным языкам (над Σ^*, причём образ каждой буквы соответствует некоторому слову рассматриваемого языка). Рассмотрим для таких морфизмов следующие обозначения.

Для некоторого языка

$$A \in \Sigma^*, \quad A = \{u_1, u_2, \dots u_n\}$$

мы рассматриваем алфавит

$$\Delta_A = \{d_1, d_2, \dots d_n\}$$

(если это не вызывает неоднозначностей, пишем обычно просто Δ), а для последнего – морфизм

$$h_A : \Delta_A^* \to \Sigma^*,$$

задающийся следующим образом [11]:

$$h_A(d_1) = u_1\,,\ h_A(d_2) = u_2\,,\ \dots\,,\ h_A(d_n) = u_n\,.$$

Если A не обладает свойством префикса [12], то определённый здесь морфизм также будем называть *непрефиксным*.

Для дальнейшего очень важной является задача построения *инверсного* морфизма [13] – т. е. следующая за-

[10] В одной из наших предыдущих публикаций мы отмечали, что в алгебре «морфизм» – обычно более общее понятие. Однако мы будем его использовать только «по Саломаа».

[11] Выше мы рассматривали язык A как множество – а *не* как упорядченное множество. Однако неточностей во вводимых обозначениях нет, поскольку мы в приведённой выше формуле фактически «перенумеровали» слова языка A. А чтобы «сверхформально» удовлетворить всем определениям, мы можем просто рассматривать любой из таких морфизмов (т. е. морфизм, соответствующий любому порядку слов языка A).

[12] Т. е. $(\exists u, v \in A)\,(u \in \mathrm{opref}(v))$.

[13] Отметим, что её можно рассматривать как частный случай задачи о регулярности языка, являющегося инверсным морфизмом некоторого другого регулярного языка; мы эту задачу («задачу Пина») рассматривали в предыдущих публикациях.

дача. Для заданных конечного языка A, морфизма h_A и некоторого слова $u \in \Sigma^*$ рассматриваем *язык* [14]

$$h_A^{-1}(u) = \{\, u_\Delta \in \Delta^* \mid h_A(u_\Delta) = u \,\};$$

специально отметим, что определяемое понятие является *множеством*.

Ту же самую конструкцию можно рассматривать и для некоторого языка (пусть B, вместо слова u) – причём нас будут интересовать только конечные языки:

$$h_A^{-1}(B) = \bigcup_{u \in B} h_A^{-1}(u).$$

Для *заданного* языка B будем рассматривать все возможные языки A, а для них – соответствующие инверсные морфизмы $h_A^{-1}(B)$, определённые выше. Среди всех таких морфизмов нас будет интересовать *оптимальный* (либо *оптимальные*) – и подробные определения такой оптимальности будут приведены в дальнейших разделах статьи.

Теперь перейдём к строгому определению так называемых потенциальных корней: как уже было отмечено, в обоих приведённых случаях (т. е. для обеих рассматриваемых групп задач) желательно [15] построить множество таких корней. Обычно для некоторых заданных языка $A \subseteq \Sigma^*$ и натурального n множество это множество нами обозначается следующим образом:

$$\overline{\sqrt[n]{A}} = \{\, u \in \Sigma^* \mid u^n \in A \,\}.$$

При этом для задачи построения инверсного морфизма n – любое подходящее натуральное число (а не заранее заданное), и мы иногда пишем

$$\overline{\sqrt[*]{A}} = \{\, u \in \Sigma^* \mid (\exists n \in \mathbb{N})\,(u^n \in A) \,\};$$

специально отметим, что равенство $n = 1$ допускается. Также важно отметить, что *построение множества всех потенциальных корней может быть выполнено с помощью несложного полиномиального алгоритма* – это просто следует из приведённых формальных определений.

В заключение раздела рассмотрим применяемую нами связь конечных языков и недетерминированных конечных автоматов [16]. А именно, для произвольного конечного языка

$$A = \{u_1, u_2, \ldots, u_n\},$$
$$\text{где} \quad u_i = a_{i,1} \ldots a_{i,l_i} \quad \text{для} \quad i = 1, \ldots, n,$$
$$\text{причём все} \quad l_i \geqslant 1,$$

мы определяем недетерминированный конечный автомат

$$\mathcal{K}(A) = (Q, \Sigma, \delta, \{s\}, Q),$$

где алфавит Σ включает все используемые в языке A буквы,

$$Q = \{s\} \quad \cup \bigcup_{i \leqslant n,\ j < l_i} q_{i,j},$$

а функция переходов δ определена следующим образом (мы рассматриваем приведённые далее определения для каждого из значений $i = 1, \ldots, n$).

- Если $|u_i| = 1$, то $\delta(s, a_{i,1}) \ni s$.
- Иначе:
  - $\delta(s, a_{i,1}) \ni q_{i,1}$;
  - $\delta(q_{i,j-1}, a_{i,j}) \ni q_{i,j}$ для каждого $j \leqslant l_i - 1$;
  - $\delta(q_{i,l_i-1}, a_{i,l_i}) \ni s$,

а других переходов функция δ не содержит. Отметим, что подобные автоматы нами рассматривались не только в [3], но и, например, в [14] [17].

Очевидно, что такой автомат определяет (бесконечный) язык $\mathrm{pref}(A^*)$. Поэтому, рассматривая описанное в наших последних статьях [4], [5], [6], [7] бинарное отношение $\Leftrightarrows$, мы можем говорить, что ответить на вопрос о выполнении условия $A \Leftrightarrows B$ можно, проверяя «студенческим способом» эквивалентность недетерминированных конечных автоматов $\mathcal{K}(A)$ и $\mathcal{K}(B)$ [18].

Повторим определение бинарного отношения $\Leftrightarrows$ и связанных с ним понятий. Если для конечных языков A и B выполнено условие

$$(\forall u \in A^*)\ (\exists v \in B^*)\ (u \in \mathrm{opref}(v)),$$

будем писать $A \trianglelefteqslant B$ (либо $B \trianglerighteqslant A$) [19]. Если одновременно выполнены условия $A \trianglelefteqslant B$ и $A \trianglerighteqslant B$, будем писать $A \Leftrightarrows B$. Наоборот, в случае, когда уже известно, что условие $A \trianglerighteqslant B$ не выполнено, мы в случае выполнения условия $A \trianglelefteqslant B$ можем также писать $A \triangleleft B$ (либо $B \triangleright A$).

III. ПРЕДВАРИТЕЛЬНЫЕ СВЕ́ДЕНИЯ: МАКСИМАЛЬНЫЕ ПРЕФИКСНЫЕ КОДЫ

По поводу приведённых далее терминов, связанных с кодами, в литературе «единства нет» [20]. Мы будем пользоваться терминологией, близкой к классическому студенческому учебнику [15] – но, однако, не совсем её повторяющую; при этом некоторые термины и связанные с ними факты взяты из [16].

Кодом мы будем называть как ранее определённый морфизм, так и соответствующий ему конечный язык. При этом в случае инъективности этого морфизма [21] будем, согласно [15], называть код (как множество, язык) – а также кодирование (как процесс) – *однозначным*. («Студенческий» алгоритм определения однозначности декодирования приведён там же, в [15]; см. также [17].)

[14] Состоящий, вообще говоря, не из одного, а из нескольких слов – а, возможно, являющийся пустым языком.

[15] Как мы уже отмечали, по-видимому – даже необходимо.

[16] Способ задания конечных автоматов совпадает с применяемым в наших предыдущих публикациях, процитирваннных выше.

[17] Также мы предполагаем существенно подробнее исследовать такие автоматы – т. н. «лепестковые» – в одной из ближайших публикаций.

[18] К сожалению, при такой проверке мы используем экспоненциальный алгоритм. Поэтому не будет большим преувеличением сказать, что цель *всех* наших публикаций по этой тематике – получить в её задачах полиномиальные алгоритмы всюду, где это возможно.

[19] Отметим, что «в качестве следствия» можно получить, что ни один из этих языков не равен ни $\varnothing$, ни $\{e\}$. Поэтому необязательно постулировать выполнение таких неравенств в формулировках как этого, так и следующих определений.

[20] По-видимому, в связи с этим статья Википедии «Алфавитное кодирование» на момент написания настоящей статьи «предлагается к удалению» – при том, что терминология текущей версии статьи «Код» (из той же Википедии) не выдерживает никакой критики …

[21] Ещё раз вернёмся к Википедии, процитируем текст из статьи «Мономорфизм»:

> Не в каждой категории можно говорить о том, что морфизму соответствует какая-то функция на множествах, однако это так в конкретных категориях. В любой такой категории «инъективный» морфизм будет мономорфизмом.

Как видно из приведённой цитаты, «чистые алгебраисты» взяли в кавычки название «инъективный морфизм» (причину этого автор настоящей статьи объяснить не может) – однако в нашем случае (даже по «их» терминологии) мы действительно можем применять этот термин: ведь здесь рассматривается именно *такая* «конкретная категория».

Для алфавита Σ, состоящего из n букв, *кодовый индекс* $ci(u)$ слова u определим как $n^{-|u|}$. *Кодовый индекс* $ci(A)$ языка

$$A = \{u_1, u_2, \dots u_n\}$$

определяется как сумма кодовых индексов всех его слов:

$$ci(A) = \sum_{i=1}^{n} ci(u_i).$$

Примеры. Для алфавита из 2 букв $\Sigma = \{a, b\}$ и слова $u = aba$ получаем $ci(u) = 2^{-3} = \frac{1}{8}$. Для языка

$$A = \{aa, ab, bbb\}$$

над тем же алфавитом получаем

$$ci(A) = 2^{-2} + 2^{-2} + 2^{-3} = \frac{1}{4} + \frac{1}{4} + \frac{1}{8} = \frac{5}{8}.$$

Для языка

$$B = \{aa, ab, b\}$$

получаем

$$ci(B) = 2^{-2} + 2^{-2} + 2^{-1} = \frac{1}{4} + \frac{1}{4} + \frac{1}{2} = 1.$$

Заметим, что в рассмотренных примерах A и B являются однозначными кодами. Вообще, можно доказать, что неравенство $ci(A) \leqslant 1$ является *необходимым* условием того, что A – однозначный код [22].

В случае конечных алфавитов и конечных языков эквивалентными являются следующие понятия:

- *максимальности* однозначного кода (а именно – равенства $ci(A) = 1$)
- и его *полноты* (т. е. невозможности добавления в язык A ещё хотя бы одного слова без нарушения условия однозначности декодирования) –

см. [16]. Поэтому такие коды можно называть *максимальными*. Среди них для нас представляют особый интерес *максимальные префиксные* (однозначные) коды – т. е. такие максимальные однозначные коды, которые обладают свойством префикса [23]. В приведённых примерах таковым является язык B.

Всё множество максимальных префиксных кодов над алфавитом Δ (как множество языков) в наших работах обозначается $mp(\Delta)$. А множество языков, каждый из которых содержит некоторый максимальный префиксный код в качестве подмножества (возможно, несобственного), обозначается $mp^+(\Delta)$; таким образом,

$$mp(\Delta) \subset mp^+(\Delta).$$

Мы будем называть каждый из таких языков *расширенным максимальным префиксным кодом*.

[22] Но, конечно, не является достаточным условием; тривиальный пример – язык $\{a, aa\}$, имеющий, в случае алфавита из 2 букв, кодовый индекс $\frac{1}{2} + \frac{1}{4} = \frac{3}{4}$.

[23] Специально отметим, что эти 3 определения:

- однозначности декодирования,
- префиксности
- и максимальности / полноты –

независимы одно от другого; а все вместе они и определяют максимальный префиксный код.

При этом *некоторую* «избыточность» такое определение всё-таки несёт: например, из префиксности всегда следует однозначность декодирования. Можно легко привести и другие варианты возникающей здесь «избыточности» – однако попытка от неё «избавиться», рассматривая название «максимальный префиксный язык» (и применяя соответствующие определения), вряд ли является удачной.

Вернёмся к рассмотрению «основного» алфавита, т. е. Σ, а также морфизмов вида

$$h_A : \Delta_A^* \to \Sigma^*.$$

Пусть у нас есть некоторый язык

$$A_\Delta \in mp(\Delta);$$

тогда будем считать, что выполнено условие

$$h_A(A_\Delta) \in mp(A)$$

(мы таким образом определяем множество языков $mp(A)$ над рассматриваемым алфавитом Σ). Аналогично, для некоторого языка

$$A_\Delta \in mp^+(\Delta)$$

будем считать, что

$$h_A(A_\Delta) \in mp^+(A).$$

Таким образом:

- $mp^+(\Delta)$ – это множество языков, каждый из которых содержит в качестве подмножества некоторый максимальный префиксный код над алфавитом Δ;
- $mp^+(A)$ – это множество языков, каждый из которых является A-морфизмом некоторого языка из множества $mp^+(A)$; т. е. каждый из таких языков – специальный морфизм некоторого расширенного максимального префиксного кода.

Приведём *недетерминированный бесконечный* алгоритм построения всех языков множества $mp(\Delta)$.

Алгоритм III.1: построение всех языков множества $mp(\Delta)$.

Вход: один из языков A, принадлежащий множеству $mp(\Delta)$ (где $A \neq \varnothing$, $A \neq \{e\}$).

Выход: другие языки множества $mp(\Delta)$.

Метод.

Шаг 1. Добавить язык A в ответ.

Шаг 2 (недетерминированный). Выбор произвольного слова $u \in A$.

Шаг 3. Вызов алгоритма III.1 со входом

$$(A \setminus \{u\}) \cup \{u\} \cdot \Delta.$$

Конец описания алгоритма.

Для работы описанного алгоритма необходим его однократный вызов (т. е. работа т. н. алгоритма-«вызывалки») с параметром Δ; сам параметр при этом воспринимается не как алфавит (множество букв), а как множество однобуквенных слов.

Отметим, что небольшие модификации этого алгоритма позволяют формулировать алгоритмы – также недетерминированные бесконечные – построения всех языков множеств $mp^+(\Delta)$, $mp(A)$ и $mp^+(A)$ для любого $A \subseteq \Sigma^*$.

Но, по-видимому, ещё более важным является алгоритм определения принадлежности некоторого языка множеству $mp^+(A)$ – для заданного конкретного языка A. Как уже было отмечено во введении, мы в части II (раздел XI) рассмотрим один из возможных алгоритмов ответа на этот вопрос.

А «с алгебраической точки зрения» мы, рассматривая языки, принадлежащие множествам $mp(A)$ и $mp^+(A)$,

получаем для каждого из этих случаев специальные подмоноиды глобального надмоноида свободного моноида – см. [18][24].

В заключение раздела отметим связь оптимальности инверсного морфизма с максимальными префиксными кодами – эта связь станет понятна на основе материала раздела XI (часть II), в котором будут рассмотрены возможные «естественные» частичные порядки на множестве языков. Более того, в связи с возможным использованием нескольких различных «естественных» порядков оптимальность инверсного морфизма можно также определять различными способами – но для рассматриваемых нами вопросов это в настоящее время неактуально.

IV. МОТИВАЦИЯ

В этом разделе мы приводим возможные варианты применения двух рассматриваемых в статье задач – в более сложных задачах теории формальных языков[25]; среди этих более сложных задач, конечно же, наиболее важной является задача переформулировки условия равенства $\mathrm{P} = \mathrm{NP}$.

Для такой переформулировки важно отметить следующее. *Автор в настоящее время считает недоказанным* один из опубликованных результатов статьи [2][26] – подробнее, с указанием конкретного замечания, мы рассмотрим этот момент в разделах XIII, XIV и XV (часть III). Поэтому мы в настоящей статье *только в качестве гипотезы* рассматриваем следующее утверждение: приведённое в [2] достаточное условие выполнения бинарного отношения $A \Subset B$ является одновременно необходимым.

Приведём план решения задачи переформулировки условия равенства $\mathrm{P} = \mathrm{NP}$ – являющийся фактически существенным изменением плана, приведённого в [3][27].

План доказательства возможности переформулировки условия равенства $\mathrm{P} = \mathrm{NP}$.

1. Рассматриваем два произвольных конечных языка A и B над одним и тем же конечным алфавитом Σ. Для них рассматривается задача проверки условия $A \Subset B$. (Понятно, что алгоритм такой проверки будет полиномиальным тогда и только тогда, когда полиномиальным будет алгоритм проверки условия $A \Subset B$ – также для двух произвольных конечных языков A и B.)

2. С одной стороны, имеется *полиномиальный* алгоритм построения конечного автомата, проверяющего условие $A \Subset B$, – см. [8][28,29]. Такая проверка даёт положительный результат тогда и только тогда, когда построенный автомат определяет полный язык (т. е. язык Σ^*) – однако за полиномиальное время это условие проверить невозможно. Поэтому *если* для любого конечного автомата K (конечно, удовлетворяющего специальным ограничениям) существует пара языков A и B, для которых условие $A \Subset B$ проверяется этим автоматом K, – то полиномиального алгоритма проверки выполнения условия $A \Subset B$ не существует.

3. Но, с другой стороны, в случае выполнения условия из статьи [2][30], существует полиномиальный алгоритм построения оптимального инверсного морфизма – что даст возможность описать *полиномиальный* же алгоритм проверки выполнения условия $A \Subset B$.

4. Таким образом – в случае завершения приведённого плана – мы сможем утверждать, что для выполнения равенства $\mathrm{P} = \mathrm{NP}$ необходимо и достаточно выполнения гипотезы (Ѫ). В первую очередь для этого нужно получить решение одной из «обратных» задач – задачи описания конкретных конечных языков A и B, для которых конечный автомат $\mathrm{PRI}(A, B)$ совпадает с некоторым *заранее заданным* конечным автоматом.

Конец формулировки плана доказательства.

Оставшаяся «небольшая часть мотивации» также связана с приведённым планом. Для любых связанных алгоритмов – причём как рассматриваемых в настоящей статье, так и сформулированных для будущего исполнения – представляет интерес их полиномиальные аналоги. А для первой из рассматриваемых двух задач (задачи извлечения корня из языка) перед созданием полиномиального алгоритма её решения необходимо провести исследование возникающих структур – не являющихся, вообще говоря, полурешётками; однако эти структуры могут быть рассмотрены как специальный вариант «объединения» нескольких полурешёток. В связи с этим и возникает несколько схожих задач (некоторые из них были упомянуты в [5]), связанных с извлечением корня из языка: например – построение любого корня и построение корня, содержащего минимально возможное число слов.

Также отметим, что подобные структуры возникают, например, в задачах минимизации недетерминированных конечных автоматов и дизъюнктивных нормальных форм. При этом построение любого корня – в качестве аналога в задаче минимизации ДНФ имеет построение (какой-нибудь) тупиковой ДНФ для заданной функции, а аналог построения некоторого «минимального» корня – построение некоторой ДНФ, содержащей минимальное число плоскостей[31].

[24] Название этой статьи как раз и говорит о том, что в ней исследуются подобные подмоноиды. При этом именно интересующие нас подмоноиды – т. е. те, которые возникают на основе множеств $\mathrm{mp}(A)$ и $\mathrm{mp}^+(A)$ – в этой статье не рассматривались.

Заодно отметим, что применяемое нередко для последнего объекта (глобального надмоноида свободного моноида) сокращённое название «супермоноид» (“supermonoid” в английской литературе) не очень удачно – однако детальное обсуждение этого термина выходит за рамки настоящей статьи.

[25] Некоторые из этих задач уже упомянуты во введении.

[26] А именно, употребляя терминологию настоящей статьи, – про *необходимое* условие выполнения бинарного отношения $A \Subset B$.

[27] В нескольких наших последних публикациях были приведены решения вспомогательных задач, необходимых для этого плана. И, как мы уже отмечали, в разделе XIII (часть III) будет приведена «критика» одной из теорем, на которых основан этот возможный план. В связи с этим приведённый здесь план отличается от плана, приведённого в той публикации, – однако, повторим, он может рассматриваться как его дальнейшая модификация.

[28] Этот автомат нами обозначался $\mathrm{PRI}(A, B)$, а его недетерминированный аналог – $\mathrm{NSPRI}(A, B)$.

[29] Конечно же, этот алгоритм построения такого «вспомогательного» конечного автомата не надо путать с алгоритмом проверки условия $A \Subset B$.

[30] Далее, в разделе XIV (часть III) настоящей статьи, это условие названо гипотезой (Ѫ).

[31] Термин «минимальная ДНФ» мы стараемся не употреблять: в литературе он используется в разных смыслах.

V. ЗАДАЧА ИЗВЛЕЧЕНИЯ КОРНЯ: НЕДЕТЕРМИНИРОВАННЫЙ ПОЛИНОМИАЛЬНЫЙ АЛГОРИТМ

Итак, сначала мы рассмотрим задачу извлечения корня из языка[32]. Как мы уже сказали, в этой задаче заданы язык A над некоторым конечным алфавитом Σ и число $n \geqslant 2$ (в случае $n = 1$ задача тривиальна), и требуется найти язык B над тем же алфавитом Σ, такой что $B^n = A$. В этом разделе приводится недетерминированный полиномиальный алгоритм решения этой задачи.

Алгоритм V.1: извлечение корня из языка.
Вход: язык $A \subseteq \Sigma^*$, число $n \geqslant 1$.
Выход: язык $B \subseteq \Sigma^*$, такой что $B^n = A$.
Метод.
Шаг 1. Построить язык C – множество потенциальных корней n-й степени из A:

$$C = \overline{\sqrt[n]{A}};$$

пусть $|C| = M$ (число таких корней равно M). Зафиксировать номер для каждого из элементов языка C.
Шаг 2. Выбрать очередное подмножество D множества C из общего числа $2^M - 1$ непустых подмножеств его элементов. Если таких множеств больше не существует, то выход из алгоритма.
Шаг 3. Если $D^n = A$, то добавление множества D в ответ. При этом если необходимо получить только один вариант ответа, то выход из алгоритма.
Шаг 4. Переход на шаг 2.
Конец описания алгоритма.

Отметим, что приведённый алгоритм является недетерминированным полиномиальным (относительно «размера задачи» – т. е. суммы длин всех слов исходного языка A) – это очевидно.

VI. ЗАДАЧА ИЗВЛЕЧЕНИЯ КОРНЯ: ОТСУТСТВИЕ ОБЕИХ ПОЛУРЕШЁТОК

Далее мы всюду будем пытаться строить полурешётки на множестве подмножеств потенциальных корней. По поводу *отсутствия* таких полурешёток в общем случае – см. примеры в заключении статьи [7]; здесь мы приведём эти же примеры с дополнительными комментариями. Во всех таких примерах мы будем рассматривать алфавит, состоящий из единственной буквы (например, a), а конечные языки задавать в виде ве́ктора, дли́ны слов в котором записаны справа налево начиная с длины́ 0. Будем записывать языки в виде равенства такому вектору; например, запись $A = 1011$ фактически определяет язык $A = \{ \varepsilon, a, a^3 \}$.

Отсутствие полурешётки относительно пересечения подтверждается достаточно простым примером (степень равна 2):

$$A = 110111, \quad B = 111011;$$

здесь $A^2 = B^2$, потенциальные корни – 111111 (то есть все элементы этого множества), и при этом

$$(A \cap B)^2 \neq A^2.$$

Примеры, подтверждающие отсутствие полурешётки относительно объединения, сложнее; «минимальный» из примеров, найденных компьютерной программой, такой: степень также равна 2,

$$A = 1101000111, \quad B = 1101001011;$$

здесь $A^2 = B^2$, потенциальные корни – 1111001111, при этом

$$(A \cup B)^2 \neq A^2.$$

В связи с наличием этих примеров (иными словами – в связи с отсутствием полурешёток) наиболее интересной[33] является задача построения *хотя бы одного* корня из заданного языка. А одной из вспомогательных для неё задач может быть следующая – сформулируем её подробно.

Для вышеуказанной основной задачи – задачи построения хотя бы одного корня – конечно же, надо рассматривать ситуацию, когда нужная степень[34] языка, состоящего из всех потенциальных корней, включает в себя исходное множество слов как *собственное* подмножество; такой пример и был приведён в [7]. Назовём любое такое подмножество потенциальных корней[35] *«бóльшим»* (под)множеством[36]. В такой терминологии, понятно, существуют и *меньшие* (под)множества потенциальных корней (тривиальный пример такого меньшего подмножества – пустое).

Постепенно удаляя *по одному* потенциальному корня из максимально возможного большего множества, мы в конце концов получаем меньшее множество. Для дальнейшего важно отметить, что для общего числа потенциальных корней M число таких возможных *последовательностей удаления* потенциальных корней равно $M!$.

Как уже было сказано, оставшийся материал статьи будет приведён в частях II и III.

Список литературы

[1] Melnikov B., Korabelshchikova S., Dolgov V. *On the task of extracting the root from the language* // International Journal of Open Information Technologies. – 2019. – Vol. 7. No. 3. – P. 1–6.
[2] Melnikov B. *The equality condition for infinite catenations of two sets of finite words* // International Journal of Foundation of Computer Science. – 1993. – Vol. 4. No. 3. – P. 267–274.
[3] Алексеева А., Мельников Б. *Итерации конечных и бесконечных языков и недетерминированные конечные автоматы* // Вектор науки Тольяттинского государственного университета. – 2011. – № 3 (17). – С. 30–33.
[4] Мельников Б., Мельникова А. *Бесконечные деревья в алгоритме проверки условия эквивалентности итераций конечных языков. Часть I* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 4. – P. 1–11.
[5] Мельников Б., Мельникова А. *Бесконечные деревья в алгоритме проверки условия эквивалентности итераций конечных языков. Часть II* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 5. – P. 1–11.
[6] Мельников Б. *Варианты конечных автоматов, соответствующих бесконечным итерационным деревьям морфизмов. Часть I* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 7. – P. 5–13.

[32] Она сложнее второй задачи? Или, наоборот, легче? По-видимому, ответ на этот вопрос зависит от отношения отвечающего к гипотезам, названным нами (Ⅱ) и (Ⅲ) и формулируемым далее в разделе XIV (часть III).

[33] Из задач, которые упомянуты во введении.

[34] В частном случае – во всех рассматриваемых нами примерах – квадрат.

[35] Не обязательно включающим все потенциальные корни. Обязательно – включающим в себя исходное множество слов как собственное подмножество.

[36] Ниже пишем без кавычек, также про «меньшие» множества.

[7] Мельников Б. *Варианты конечных автоматов, соответствующих бесконечным итерационным деревьям морфизмов. Часть II* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 10. – P. 1–8.

[8] Мельников Б., Мельникова А. *Полиномиальный алгоритм построения конечного автомата для проверки равенства бесконечных итераций двух конечных языков* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 11. – P. 1–10.

[9] Melnikov B., Melnikova A. *Some properties of the basis finite automaton* // The Korean Journal of Computational and Applied Mathematics (Journal of Applied Mathematics and Computing). – 2002. – Vol. 9. No. 1. – P. 135–150.

[10] Melnikov B., Sciarini-Guryanova N. *Possible edges of a finite automaton defining a given regular language* // The Korean Journal of Computational and Applied Mathematics (Journal of Applied Mathematics and Computing). – 2002. – Vol. 9. No. 2. – P. 475–485.

[11] Мельников Б., Сайфуллина М. *О некоторых алгоритмах эквивалентного преобразования недетерминированных конечных автоматов* // Известия высших учебных заведений. Математика. – 2009. – № 4. – С. 67–72.

[12] Скорняков Л. (ред.) *Общая алгебра. Том 2*. – М., Наука. – 1991. – 480 с.

[13] Саломаа А. *Жемчужины теории формальных языков*. – М., Мир. – 1986. – 159 с.

[14] Корабельщикова С., Мельников Б. *Итерации языков и максимальные префиксные коды* // Вестник Воронежского государственного университета. Серия: Физика. Математика. – 2015. – № 2. – С. 106–120.

[15] Яблонский С. *Введение в дискретную математику. Учебное пособие для вузов*. – М., Высшая школа. – 2001. – 384 с.

[16] Лаллеман Ж. *Полугруппы и комбинаторные приложения*. – М., Мир. – 1985. – 440 с.

[17] Новиков Ф. *Дискретная математика для программистов*. – СПб, Питер. – 2009. – 304 с.

[18] Мельников Б. *Описание специальных подмоноидов глобального надмоноида свободного моноида* // Известия высших учебных заведений. Математика. – 2004. – № 3. – С. 46–56.

Борис Феликсович МЕЛЬНИКОВ,
профессор Университета МГУ – ППИ в Шэньчжэне
(http://szmsubit.ru/),
email: bf-melnikov@yandex.ru,
mathnet.ru: personid=27967,
elibrary.ru: authorid=15715,
scopus.com: authorId=55954040300,
ORCID: orcidID=0000-0002-6765-6800.

Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part I. Extracting the root from the language

Boris Melnikov

***Abstract*—In the paper, we consider all possible subsets of the set of potential roots forming in some situations semi-lattices, by intersection and / or by union. Such structures arise in two similar problems in the theory of formal languages. Specifically, for some given finite language, we consider the problem of extracting the root of a given degree and the problem of constructing an optimal inverse morphism, where optimality can be defined, for example, as the length of the maximum word of a language that is an inverse morphic image.**

In both of the above cases, it is necessary to construct the set of so-called potential roots, i.e., such words of the alphabet in question, for each of which some degree of it is included in the source language. It is important to note that the construction of the set of all potential roots can be performed using a simple polynomial algorithm. Exponential algorithms for both problems are obvious: we just need to sort through all subsets of the set of these potential roots, and choose the appropriate one among these subsets. Therefore, the problem is to describe possible polynomial algorithms for these problems.

For both of these problems, the possible existence of two semi-lattices available on a pre-constructed set of subsets of potential roots is of interest.

Among other things, in the paper we present the formulation of one important hypothesis of the theory of formal languages, in which we can assert that a special subset of a set of languages, each of which is an inverse morphic image of a given language, forms not only a half-lattice by union, but also a half-lattice by intersection (and, therefore, a lattice).

The first part of the paper contains motivation, definition of basic concepts, after which includes the material, more dedicated the first of the tasks under consideration, i.e., the task of extracting a root from a language.

***Keywords*—formal languages, iterations of languages, morphisms, inverse morphisms, binary relations, semi-lattices, algorithms.**

References

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

[2] Melnikov B. *The equality condition for infinite catenations of two sets of finite words* // International Journal of Foundation of Computer Science. – 1993. – Vol. 4. No. 3. – P. 267–274.

[3] Alekseeva A., Melnikov B. *Iterations of finite and infinite languages and nondeterministic finite automata* // Vector of Science of Togliatti State University. – 2011. – No. 3 (17). – P. 30–33 (in Russian).

[4] Melnikov B., Melnikova A. *Infinite trees in the algorithm for checking the equivalence condition of iterations of finite languages. Part I* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 4. – P. 1–11 (in Russian).

[5] Melnikov B., Melnikova A. *Infinite trees in the algorithm for checking the equivalence condition of iterations of finite languages. Part II* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 5. – P. 1–11 (in Russian).

[6] Melnikov B. *Variants of finite automata, corresponding to infinite iterative morphism trees. Part I* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 7. – P. 5–13 (in Russian).

[7] Melnikov B. *Variants of finite automata, corresponding to infinite iterative morphism trees. Part II* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 10. – P. 1–8 (in Russian).

[8] Melnikov B., Melnikova A. *Polynomial algorithm of construction a finite automaton for checking equality infinite iterations of two finite languages* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 11. – P. 1–10 (in Russian).

[9] Melnikov B., Melnikova A. *Some properties of the basis finite automaton* // The Korean Journal of Computational and Applied Mathematics (Journal of Applied Mathematics and Computing). – 2002. – Vol. 9. No. 1. – P. 135–150.

[10] Melnikov B., Sciarini-Guryanova N. *Possible edges of a finite automaton defining a given regular language* // The Korean Journal of Computational and Applied Mathematics (Journal of Applied Mathematics and Computing). – 2002. – Vol. 9. No. 2. – P. 475–485.

[11] Melnikov B., Sayfullina M. *On some algorithms of equivalent transformation of nondeterministic finite automata* // News of higher educational institutions. Mathematics. – 2009. – No. 4. – P. 67–72 (in Russian).

[12] Skornyakov L. (Ed.) *General Algebra. Vol. 2*. – Moscow, Nauka. – 1991. – 480 p. (in Russian).

[13] Salomaa A. *Jewels of Formal Language Theory*. Computer Science Press, Rockville, Maryland. – 1981. – 144 p.

[14] Korabelshchikova S., Melnikov B. *Iterations of languages and maximum prefix codes* // Bulletin of the Voronezh State University. Series: Physics. Mathematics. – 2015. – No. 2. – P. 106–120 (in Russian).

[15] Yablonskiy S. *Introduction to Discrete Mathematics. Study Guide for Universities*. – Moscow, Vysshaya Shkola. – 2001. – 384 p. (in Russian).

[16] Lallement G. textitSemigroups and Combinatorial Applications. – NY, John Wiley & Sons. – 1979. – 376 p.

[17] Novikov F. *Discrete mathematics for programmers*. – Saint Petersburg, Piter. – 2009. – 304 p. (in Russian).

[18] Melnikov B. *Description of special submonoids of the global supermonoid of the free monoid* // News of higher educational institutions. Mathematics. – 2004. – No. 3. – P. 46–56 (in Russian).

Boris MELNIKOV,
Professor of Shenzhen MSU – BIT University, China
(http://szmsubit.ru/),
email: bf-melnikov@yandex.ru,
mathnet.ru: personid=27967,
elibrary.ru: authorid=15715,
scopus.com: authorId=55954040300,
ORCID: orcidID=0000-0002-6765-6800.