Варианты конечных автоматов, соответствующих бесконечным итерационным деревьям морфизмов. Часть I

Б. Ф. Мельников

Аннотация—**Совсем кратко предмет настоящей статьи можно сформулировать следующим образом: в рассмотренных ранее бесконечных итерационных деревьях морфизмов мы объединяем эквивалентные вершины – фактически получая детерминированный конечный автомат; после этого мы исследуем некоторые свойства полученного автомата. Кроме того, в статье описана возможная связь рассматриваемых нами автоматов с задачами, возникающими в других областях теории формальных языков.**

Более подробно. Мы работаем с различными вариантами конечных автоматов, каждый из которых соответствует некоторому бесконечному итерационному дереву рассматриваемого морфизма. При этом построенные для различных морфизмов автоматы описывают основные свойства заданных морфизмов. Кроме того, в каждом случае (т. е. для каждого варианта автомата) возникает следующая «обратная задача»: описать морфизм (либо просто указать пару языков), для которого получается некоторый заданный автомат.

Среди вариантов автоматов, соответствующих бесконечному итерационному дереву морфизма для заданной упорядоченной пары конечных языков, мы сначала определяем т. н. первичный автомат. Он детерминированный, определён на множествах слов – и каждое из этих множеств является подмножеством множества префиксов второго из заданных языков. Далее мы рассматриваем соответствующие ему несколько вариантов специальных недетерминированных автоматов, фактически описывающих построение итерационного дерева морфизма. После этого мы вводим совершенно иной объект – т. н. упрощённый первичный автомат, который также описывает построение итерационного дерева морфизма, но который определён не на множествах слов, а на словах. Несмотря на существенную разницу с автоматами, построенными на множествах слов, все построения для конкретных примеров языков могут быть выполнены с помощью той же самой компьютерной программы.

Далее мы рассматриваем особенности, появляющиеся при применении алгоритмов, формирующих конечные автоматы, к парам совпадающих языков. В заключении мы кратко формулируем направления дальнейшей работы, связанные с вопросами, рассмотренными в настоящей статье.

В настоящей части I мы рассматриваем только детерминированные автоматы.

Ключевые слова—**формальные языки, итерации языков, морфизмы, бинарные отношения, бесконечные деревья, алгоритмы.**

I. ВВЕДЕНИЕ

В настоящей статье мы продолжаем тематику статей [1], [2]. *Сильно упрощая*, её предмет можно сформулировать следующим образом:

- в рассматриваемых ранее бесконечных итерационных деревьях морфизмов мы объединяем эквивалентные вершины (состояния) – фактически получая детерминированный конечный автомат;
- после чего мы исследуем некоторые свойства этого автомата.

Однако, повторим, это сильное упрощение.

Правильнее, конечно, говорить, что мы работаем с различными вариантами конечных автоматов, каждый из которых соответствует некоторому бесконечному итерационному дереву рассматриваемого морфизма. При этом построенные для различных морфизмов автоматы описывают основные свойства заданных морфизмов. Кроме того, в каждом случае (т. е. для каждого варианта автомата) возникает также следующая «обратная задача»: описать морфизм (либо просто указать пару языков), для которого (которых) получается некоторый заданный автомат [1].

В статье описана возможная связь рассматриваемого нами материала с задачами, возникающими в других областях теории формальных языков. Но, конечно, основная часть статьи посвящена описанию нескольких вариантов конечных автоматов, необходимых для представления итерационных деревьев морфизмов. Мы приводим определения, а также кратко описываем примеры, соответствующие двум из приведённых вариантов автоматов и продолжающие примеры из [1], [2]; в статье – продолжении настоящей мы предполагаем рассмотреть большее число примеров, а также кратко описать компьютерные программы, необходимые для их формирования.

Структура статьи такова. В разделе II («мотивация») описаны несколько задач, связанных с рассматриваемой нами; по-видимому, настоящая статья должна помочь их будущему решению. В разделе III описаны краткие сведения и обозначения, связанные с такими предметными областями наших предыдущих публикаций: недетерминированными конечными автоматами, теорией формальных языков и алгебраическими вопросами теории полугрупп.

В разделе IV определён автомат основного вида, соответствующий бесконечному итерационному дереву морфизма для заданной упорядоченной пары конечных языков; мы будем называть этот автомат первичным.

Статья получена 30 апреля 2021 г.

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

[1] В настоящей статье мы ни одной из таких «обратных задач» не рассматриваем – однако некоторые результаты статьи должны быть полезны для решения этих задач в будущем.

Он определён на множествах слов – и каждое из этих множеств является подмножеством множества префиксов второго из заданных языков. Соответствующие ему несколько вариантов недетерминированных конечных автоматов кратко перечислены далее, в разделе V – с которого начинается часть II.

В разделе VI описан совершенно иной объект – автомат, опредлённый не на множествах слов, а на словах. В связи с этим он является недетерминированным – но, по-видимому, является «более простым» для любой пары заданных языков. Несмотря на существенную разницу с автоматами, построенными на множествах слов, все построения для конкретных примеров языков могут быть выполнены с помощью той же самой компьютерной программы. В разделе VII описан автомат, также недетерминированный, являющийся более удобной модификацией предыдущего – мы будем называть его упрощённым первичным автоматом.

В разделе VIII кратко рассмотрены особенности, появляющиеся при применении алгоритмов, формирующих конечные автоматы, к парам совпадающих языков. В разделе IX мы рассматриваем важное свойство определённых нами автоматов – определяем специальную алгебраическую систему, свойства которую предполагаем более подробно рассмотреть в последующих публикациях.

А в заключении (раздел X) мы кратко формулируем и другие направления дальнейшей работы – частично уже приведённые ранее в «мотивации» и связанные с вопросами, рассмотренными в настоящей статье.

Заканчивая введение, приведём такое важное замечание ко всей статье (к обеим её частям): в ней мы постоянно будем ссылаться на [1], [2] – однако *не будем пользоваться самими результатами* тех статей.

II. «МОТИВАЦИЯ»

В этом разделе описаны несколько задач, связанных с рассматриваемой нами; по-видимому, настоящая статья должна помочь их будущему решению.

Сначала повторим (с небольшими добавлениями) один абзац из [1]. Само отношение $\widetilde{\infty}$ и некоторые его свойства мы впервые рассматривали в [3] – но ещё до того, в [4], был рассмотрен один из частных случаев этого отношения (этот случай логично называть «префиксным»). Кроме того, в нескольких наших последующих публикациях было рассмотрено применение этого частного случая при решении (также частных случаев) проблемы эквивалентности однозначных скобочных грамматик – что полезно для систем автоматизации построения компиляторов.

Среди этих задач упомянем только одну – описание для класса контекстно-свободных языков собственного подкласса с разрешимой проблемой эквивалентности, [5]. При этом такое наше описание не связано с применением магазинных автоматов и их подклассов ([6], [7], [8]) – а базируется только на ограничениях, накладываемых на контекстно-свободные грамматики [2]. И там же, в [5], а также в нескольких последующих статьях, приведены описания грамматических структур языков программирования – причём эти грамматические структуры удовлетворяют описанным ограничениям.

В качестве следующего «пункта мотивации» приведём такой [3]. Некоторые алгоритмы работы с недетерминированными конечными автоматами являются полиномиальными (относительно размера исходной задачи) – как правило, те из них, которые не требуют построения эквивалентного канонического автомата [4]. Осуществлённое в настоящей статье построение конечных автоматов для бесконечных итерационных деревьев (т. е. автоматов, фактически описывающих эти деревья) даст возможность формулировать «на языке конечных автоматов» алгоритмы построения инверсных морфизмов – причём таких, что соответствующие им морфизмы, применённые к некоторым расширенным максимальным префиксным кодам [5], дают любой исходный (конечный) язык. Следовательно, эквивалентность таких языков [6] удастся проверять за такое же время. Мы собираемся в одной из следующих публикаций привести возможный *полиномиальный* алгоритм обработки конечных автоматов специального вида (ниже – т. н. упрощённых первичных недетерминированных автоматов), и на основе этого алгоритма описать полиномиальный же алгоритм построения оптимального инверсного морфизма.

Такой алгоритм может быть применён в следующей задаче. Как известно ([9, проблема ТАЯ-1]), алгоритм проверки неэквивалентности недетерминированных конечных автоматов относится к классу NP-полных задач [7]. В этой задаче речь идёт о двух *произвольных* недетерминированных конечных автоматах – и в случае получения доказательства NP-полноты той же самой задачи для специального подкласса класса НКА [8] мы получим *эквивалентную переформулировку проблемы P=NP*. А именно, это равенство верно в том случае, когда не существует пары (язык, слово), для которой сформулированное нами ранее достаточное условие эквивалентности языков не является необходимым – т. е. выполнено следующее:

- заданное слово не принадлежит итерации выбранного подмножества потенциальных корней заданного языка [9];
- однако его добавление в язык не даст неэквивалентного языка (т. е. – с нашей точки зрения – «ничего не меняет» для итераций).

Повторим, что это действительно «пункт мотивации» – описывающий ещё и возможное направление дальнейших работ.

[2] Отметим по этому поводу, что *ни в одной* из известных автору монографий (не будем приводить конкретные ссылки) нет *строгого* доказательства эквивалентности КС-грамматик и МП-автоматов (как двух разных формализмов, описывающих КС-языки): обычно доказательство даётся только «в одну сторону». В связи с этим формально описание работы с МП-автоматами и КС-грамматиками должно даваться по отдельности – что и делается в наших статьях.

[3] Мы его в немного другом изложении приводили в нескольких наших предыдущих работах.

[4] И, следовательно, функций разметки состояний φ^{in} и φ^{out} и бинарного отношения #.

[5] Согласно описанным далее обозначениям – элементам множества языков $\mathrm{mp}^{+}(A)$.

[6] Термин употреблён верно – хотя обычно говорится «эквивалентность» для элементов некоторого формализма (автоматов и т. п.), но «равенство» для некоторых двух языков. Здесь мы имеем в виду, также согласно описанным далее обозначениям, выполнение условия $\Leftrightarrow$.

[7] См. там же ещё и проблему ТАЯ-9. В русском издании – страницы 339–344. В английском первоисточнике – проблемы AL-1 и AL-9.

[8] А именно – для т. н. semi-flower automata, упоминавшихся в наших предыдущих публикациях.

[9] Хотя, конечно, является префиксом некоторых слов этой итерации.

Про остальные возможные «пункты мотивации» скажем очень кратко. Конечно же, имеется связь задач, рассматриваемых в настоящей статье, с задачей извлечения корня из языка. Как уже было упомянуто в [2], [10], аналогий между этими двумя задачами несколько; самой очевидной является такая. В обоих случаях рассматриваются множества *потенциальных корней* n-й степени [10], обычно обозначаемые нами

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

именно из такого множества в обеих задачах – причём в разных их постановках – формируются языки-ответы.

Для использованного нами в предыдущих статьях термина «множество хвостов» не может не возникнуть аналогия с другим формализмом, описывающим множества суффиксов (либо множества префиксов) – т. н. суффиксным деревом (а также суффиксным лесом) [11]. Понятно, что исследование представлений всех префиксов / суффиксов («множество хвостов» по [2]) сходно с рассматриваемыми нами вопросами.

III. ПРЕДВАРИТЕЛЬНЫЕ СВЕ́ДЕНИЯ И ОСНОВНЫЕ ОБОЗНАЧЕНИЯ

Для настоящей статьи необходимы краткие све́дения и обозначения, связанные со следующими тремя предметными областями наших предыдущих публикаций:

- недетерминированными конечными автоматами;
- теорией формальных языков;
- алгебраическими вопросами теории отношений и теории полугрупп.

Во всех трёх случаях в наших предыдущих работах были введены соответствующие термины и обозначения.

Про автоматы такие предварительные све́дения (причём очень подробные) были приведены в трёх разделах недавно опубликованных статей [1], [2] – поэтому здесь эту информацию мы повторять не будем. Отметим только несколько нечасто применяемых обозначений, не указанных в [1], [2].

Дугу $\delta(q_1, a) \ni q_2$ будем иногда записывать в виде

$$q_1 \xrightarrow{a} q_2\,,$$

причём при необходимости название конкретной функции переходов приводится под стрелкой. Аналогично – для слова u, прочитав которое автомат может перейти из состояния q_1 в состояние q_2:

$$q_1 \xrightarrow{u} q_2\,.$$

Последняя запись может иметь и некоторые другие варианты: так, если опущено левое состояние, т. е. записано

$$\xrightarrow{u} q_2\,,$$

то это означает, что прочтя слово u автомат может оказаться в состоянии q_2.

Обозначение $\mathcal{L}(K)$ – это язык, задаваемый автоматом K. Для некоторого автомата

$$K = (Q, \Sigma, \delta, S, F)$$

входной язык состояния $q \in Q$ – это язык автомата

$$K = (Q, \Sigma, \delta, S, \{q\});$$

этот язык будем обозначать записью $\mathcal{L}^{in}_K(q)$.

Далее приведём необходимые нам предварительные све́дения о двух оставшихся предметных областях – и заранее отметим, что некоторыми из этих обозначений мы в [1], [2] уже пользовались.

В настоящей статье мы будем рассматривать слова и языки только над конечными алфавитами. «Главный» алфавит, над которым они рассматриваются, – Σ; обычно его буквы – a, b, c и т. д. (т. е. из начала латинского алфавита), при необходимости с индексами. «Вспомогательный» алфавит – Δ; обычно его буквы – 0, 1, 2 и т. д. (причём иногда без 0), а в формулах обычно d с индексами.

Слово над заданным алфпавитом – некоторая конечная последовательность его букв; чаще всего слова обозначаются последними буквами латинского алфавита, u, v и др. Мы будем также употреблять бесконечные последовательности букв (по-другому – бесконечные слова, либо ω-слова), они обычно обозначаются первыми строчными буквами греческого алфавита – α, β и др.

Всё множество слов над алфавитом Σ обозначаем Σ^*; подмножества этого множества – языки над алфавитом Σ, возможно, бесконечные. Основные операции над языками – конкатенация и итерация, подробности см. в [11]. Всё множество ω-слов над алфавитом Σ обозначаем Σ^ω; подмножества этого множества – ω-языки. Конкретные языки (как конечные, так и бесконечные) обычно будем обозначать несколькими первыми заглавными буквами латинского алфавита (A, B и т. д.), а конкретные ω-языки – тоже, но специальным шрифтом ($\mathcal{A}$, $\mathcal{B}$ и т. д.).

Далее не все используемые нами обозначения стандартные (хотя и не противоречат стандартным) – поэтому оформляем их в виде формальных определений.

Определение 1: Для заданного слова $u \in \Sigma^*$:

- язык $\mathrm{pref}(u)$ определяется как множество префиксов [12] (включая несобственный) слова u;
- язык $\mathrm{opref}(u)$ определяется как множество собственных префиксов слова u;
- язык $\mathrm{suff}(u)$ определяется как множество суффиксов (включая несобственный) слова u;
- язык $\mathrm{osuff}(u)$ определяется как множество собственных суффиксов слова u.

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

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

аналогично для $\mathrm{opref}(A)$, $\mathrm{suff}(A)$ и $\mathrm{osuff}(A)$. □

Определение 2: Морфизм (использование этого термина также согласовано с [11] [13]) – это отображение

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

[10] При этом для задачи извлечения корня n задано, а в случае нашей задачи n – любое возможное.

[11] По мнению автора, *хороших* публикаций в монографической литературе об этих объектах до сих пор нет (хотя впервые суффиксные деревья были описаны почти 50 лет назад) – поэтому приведём только ссылки на удачные сайты по соответствующей тематике: `https://habr.com/ru/post/258121/`, а также несколько сайтов на ресурсе `https://neerc.ifmo.ru/wiki/`.

[12] Формулы в этом определении писать не будем.

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

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

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

Нам будут очень важны морфизмы, соответствующие конечным языкам (над Σ^*, образ каждой буквы соответствует некоторому слову рассматриваемого языка).

Определение 3: 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 него – морфизм

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

задающийся следующим образом [14]:

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

Если A не обладает свойством префикса [15], то определённый здесь морфизм также будем называть *непрефиксным*. $\square$

Ранее в нескольких статьях (в [1] мы цитировали некоторые из них – в первую очередь речь идёт о [3], [4], [5], [10], [12]) мы рассматривали следующие важные бинарные отношения на (бесконечных) языках.

- Сначала для языков A и B мы рассматривали отношение $\overset{\subseteq}{\infty}$ (пишем $A \overset{\subseteq}{\infty} B$); оно выполняется в том и только том случае, когда

  $$(\forall u \in A)\ (\exists v \in B)\ (u \in \operatorname{opref}(v)).$$

  (Приведённое определение для необходимых нам бесконечных языков даст тот же самый ответ, если в нём убрать слово «собственных», т. е. заменить множество $\operatorname{opref}(v)$ на $\operatorname{pref}(v)$, – но в связи со сказанным далее это совершенно непринципиально.)
- А когда вдобавок к этому выполнено и «зеркальное» условие $B \overset{\subseteq}{\infty} A$, мы считаем, что для языков A и B выполняется отношение эквивалентности $A \overset{\sim}{\infty} B$.

Однако во всех работах (как в процитированных выше, так и в нескольких непроцитированных) нам для применения этих бинарных отношений были интересны только такие (бесконечные) языки, которые *являются итерациями* некоторых конечных языков (понятно, кроме итераций языков $\varnothing$ и $\{e\}$). В связи с этим обычно удобнее рассматривать бинарные отношения не для «проитерированных» языков, а для языков итерируемых. Для таких языков добавим ещё один важный факт, доказанный ранее в [3].

- Выполнение отношения $A^* \overset{\sim}{\infty} B^*$ для конечных языков A и B равносильно выполнению равенства $A^\omega = B^\omega$.

Для удобства работы с подобными объектами (более удачной записи формул и т. п.) введём новые обозначения в виде следующего определения – причём специально отметим ещё раз, что рассматриваемые в нём языки конечные, а формула отличается от приведённой выше.

Определение 4: Если для конечных языков A и B выполнено условие

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

будем писать $A \trianglelefteqslant B$ (либо $B \trianglerighteqslant A$).

Если одновременно выполнены условия $A \trianglelefteqslant B$ и $A \trianglerighteqslant B$, будем писать $A \trianglelefteqslant\!\trianglerighteqslant B$.

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 $B \triangleright A$) [16]. $\square$

Определение 5: Для рассматриваемого алфавита Δ множество максимальных префиксных кодов над Δ [13, стр. 135, 144] будем обозначать $\operatorname{mp}(\Delta)$. [17]

Множество конечных языков над алфавитом Δ, каждый из которых в качестве подмножества (возможно, несобственного) содержит некоторый максимальный префиксный код над Δ, будем обозначать $\operatorname{mp}^+(\Delta)$ – т. е. формально

$$\operatorname{mp}^+(\Delta) = \left\{\, A_\Delta \subseteq \Sigma^* \,\middle|\, (\exists B_\Delta \subseteq A_\Delta)\,(B_\Delta \in \operatorname{mp}(\Delta)) \,\right\}.$$

$\square$

Определение 6: Для рассматриваемых алфавита Δ и языка $A \subseteq \Sigma^*$ следующее множество языков над алфавитом Σ будем обозначать $\operatorname{mp}(A)$:

$$\operatorname{mp}(A) = \left\{\, B \in \Sigma^* \,\middle|\, (\exists A_\Delta \in \operatorname{mp}(\Delta))\,(B = h_A(A_\Delta)) \,\right\}.$$

Аналогично,

$$\operatorname{mp}^+(A) = \left\{\, B \in \Sigma^* \,\middle|\, (\exists A_\Delta \in \operatorname{mp}^+(\Delta))\,(B = h_A(A_\Delta)) \,\right\}.$$

$\square$

Очевидно, что для некоторых конечных языков $A, B \subseteq \Sigma^*$ условие

$$(\exists D \subseteq \Sigma^*)\,(A, B \in \operatorname{mp}^+(D)) \tag{1}$$

является достаточным для выполнения условия эквивалентности $A \trianglelefteqslant\!\trianglerighteqslant B$. По-видимому, верно и обратное, т. е.

[14] 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).

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

[16] Также можно очевидным образом объяснить обозначение $A \triangleleft\!\triangleright B$ (соответствующее ему условие выполнено, например, для пары $A = \{a\}$, $B = \{b\}$) – однако вряд ли это обозначение будет когда-либо востребовано.

[17] Возможный несложный *недетерминированный* алгоритм построения *любого* такого максимального префиксного кода – т. е. любого элемента множества $\operatorname{mp}(\Delta)$ – следующий.

- (База.) Максимальным префиксным кодом является сам язык Δ. (Заметим, что в некоторых случаях удобнее в качестве базы рассматривать язык $\{e\}$ – но в нашей ситуации этого делать нельзя.)
- (Шаг.) Если известно, что $A_\Delta \in \operatorname{mp}(\Delta)$, и при этом $u_\Delta \in A_\Delta$, то считаем, что $(A_\Delta \setminus \{u_\Delta\}) \cup \{u_\Delta\}\Delta \in \operatorname{mp}(\Delta)$.

(1) является также и *необходимым* условием для выполнения этой эквивалентности – в этом заключался основной результат одной из наших предыдущих статей, [3, Th. 1]. Однако в настоящей статье *мы этим результатом* (т. е. не только достаточностью, но и необходимостью выполнения условия (1) для эквивалентности $A \Leftrightarrows B$) *не пользуемся*.

Продолжим рассматривать несложные факты об отношении $\Leftrightarrows$. Очевидно, что оно разбивает все конечные языки (т. е. все элементы множества $\mathcal{P}(\Sigma^*)$, являющиеся конечными множествами [18]) на бесконечное число классов эквивалентности, [14, Гл. I]. Кроме того, очевидно, что при выполнении условия $A \Leftrightarrows B$ всегда также выполнено условие

$$(A \cup B) \Leftrightarrows B.$$

Таким образом, согласно [15, Гл. V], множество конечных языков, находящихся в отношении $\Leftrightarrows$ с некоторым заданным конечным языком A, образует *полурешётку* с операцией объединения. Однако подробнее алгебраические вопросы, связанные с введёнными понятиями, мы рассматривать не будем [19] – приведём только следующие замечания.

- Решётка (с дополнительной операцией пересечения), конечно же, при этом не образуется. Тривиальным подтверждающим примером является такой:

$$A = \{\, a, ba, bb \,\}, \quad B = \{\, aa, ab, b \,\}.$$

- Однако всегда (т. е. для любого конечного языка A) можно выделить некоторое специальное подмножество из всего множества эквивалентных ему языков [20], на котором рассматриваемая алгебраическая структура образует и другую полурешётку – с операцией пересечения, см. некоторые подробности в [3]. Таким образом, образуется *решётка*.
- Выше несколько раз было отмечено, что речь идёт о конечных языках – но иногда в таком же качестве можно рассматривать и бесконечные языки. См. некоторые подробности в [16].

Следующее обозначение связано с обозначениями, уже использованными в [1], [2] – в частности, с [1, рис. 2, 3], а также с функциями, описанными на [1, рис. 6, 7]. В таком виде, как мы записываем это обозначение на основе следующего определения, оно, по-видимому, более удобно для настоящей статьи.

Определение 7: Для пары непустых конечных языков $A, B \subseteq \Sigma^*$, таких что $A \not\ni \varepsilon$ и $B \not\ni \varepsilon$, язык

$$\underset{A-B}{\Phi}(u) \subseteq \Sigma^*$$

строится для некоторого слова $u \in \Sigma^*$ по следующему алгоритму – в нём используется вспомогательный язык D' и формируется язык-ответ D.

1) Для начала работы алгоритма полагаем
$$D' = \{u\}A, \quad D = \varnothing.$$
2) Если $D' = \varnothing$, то выход из алгоритма с ответом D.
3) Выбираем некоторое слово $v \in D'$, исключая его из D'.
4) Для выбранного v рассматриваем все варианты его представления в виде $v = uv'$, где $u \in B^+$,
$$D +\!: v'$$
(т. е. включаем каждое такое v' в язык D).
5) Если
$$(\exists w \in B)\,(v \in \operatorname{opref}(w)),$$
то $D +\!: v$ (т. е. включаем v в язык D) [21].
6) Переходим на п. 2.

(Согласно приведённому алгоритму, про итоговый «выходной» язык $D = \underset{A-B}{\Phi}(u)$ можно утверждать, что $D \subseteq \operatorname{opref}(B)$. [22]) □

Приведённое определение фактически формулирует алгоритм построения функции F из [1] – мы также называли значения этой функции «множествами хвостов». Заметим также, что на шаге 4 мы могли бы заменить условие $u \in B^+$ на $u \in B$ – но в этом случае нам бы пришлось иногда рассматривать «лишние» слова [23].

Определение 8: Для языка $C \subseteq \Sigma^*$ определяем

$$\underset{A-B}{\Phi}(C) = \bigcup_{u \in C} \underset{A-B}{\Phi}(u).$$

(Аналогично предыдущему определению, про итоговый язык можно утверждать, что $\underset{A-B}{\Phi}(C) \subseteq \operatorname{opref}(B)$.) □

IV. ПЕРВИЧНЫЙ (PRIMARY) АВТОМАТ ДЛЯ БЕСКОНЕЧНЫХ ИТЕРАЦИОННЫХ ДЕРЕВЬЕВ

Теперь, после введения обозначений, мы можем описать основной автомат, соответствующий бесконечному итерационному дереву морфизма для заданной упорядоченной пары конечных языков; назовём его *первичным (primary) детерминированным конечным автоматом для бесконечных итерационных деревьев морфизмов* (коротко – просто *первичным автоматом*). В предыдущих статьях [1], [2] задающие его языки обозначались A и B в формулах – либо `lA` и `lB` в соответствующих им программах. Алфавит Δ_A далее имеет тот же смысл, что и вспомогательный алфавит для заданного языка A выше

[18] Здесь и далее для некоторого множества X запись $\mathcal{P}(X)$ означает множество всех подмножеств множества X.

[19] Они просто не относятся к тематике журнала, поскольку вряд ли могут быть использованы в математических моделях вычислений.

[20] Про алгоритм построения этого подмножества для некоторого заданного конечного языка $A \subseteq \Sigma^*$ будет информация в заключении. Там же будет сказано о связи этого подмножества с нашими дальнейшими предполагаемыми публикациями.

В примере из предыдущего пункта таким подмножеством является $\{\, a, b \,\}$ – но нам интересны не такие тривиальные примеры, а различные варианты *непрефиксных* языков, а особенно – непрефиксные морфизмы непрефиксных языков.

[21] Иначе про v «забываем».

[22] Специально отметим, что условие $D \subseteq \operatorname{osuff}(B)$ может не выполняться. Однако для матриала, приведённого в настоящей статье, этот факт вряд ли интересен.

[23] Это в дальнейшем изложении повлияло бы на описание конечных автоматов и соответствующих им алгоритмов, использующих язык

$$\underset{A-B}{\Phi}(u) :$$

они стали бы «более громоздкими», но их содержательный смысл не изменился бы.

в настоящей статье – т. е., иными словами, мы фактически рассматриваем алфавит Δ_A и соответствующий ему морфизм h_A, но не рассматриваем Δ_B и h_B.[24]

Определение 9: (Детерминированный) конечный автомат $\overline{\text{PRI}}\,(A, B)$ для заданных конечных языков $A, B \subseteq \Sigma^*$ определяется следующим образом:

$$\overline{\text{PRI}}\,(A, B) = (\,\overline{Q}_B, \Delta_A, \delta_{A-B}, \{\{e\}\}, \{\varnothing\}\,),$$

где:

- $\Delta_A = \{\,0,\ 1,\ \ldots,\ n{-}1\,\}$, здесь n – число слов языка A;[25]
- $\overline{Q}_B = \mathcal{P}(\text{opref}\,(B))$;
- для некоторых $q_1, q_2 \in \overline{Q}_B$ и $a \in \Delta_A$ полагаем

$$q_1 \xrightarrow[\delta_{A-B}]{a} q_2$$

тогда и только тогда, когда

$$\underset{h_A(a)-B}{\Phi}(q_1) = q_2\,.$$ [26]

□

Самое важное замечание к приведённому определению – о входных и выходных состояниях этого автомата. Мы определяем, что будет *по одному* входному и выходному состоянию, а все состояния мы определяем как некоторые подмножества – этим и вызваны обозначения $\{\{e\}\}$ и $\{\varnothing\}$.

Также отметим, что в автомате $\overline{\text{PRI}}\,(A, B)$ – без изменения определяемого им языка – можно удалить недостижимые состояния[27]. В приведённом определении мы эту возможность не отразили – однако для упрощения приводимых далее примеров будем этим пользоваться.

Определение 10: Получающийся после описанного удаления недостижимых состояний автомат будем обозначать

$$\text{PRI}\,(A, B) = (\,Q_B, \Delta_A, \delta_{A-B}, \{\{e\}\}, \{\varnothing\}\,),$$

где $Q_B \subseteq \mathcal{P}(\text{opref}\,(B))$. □

Будем автомат $\text{PRI}\,(A, B)$ также называть первичным – это не вызовет недоразумений. Множество его состояний, как видно из приведённой формулы, обозначаем Q_B, а вводить новые обозначения для остальных элементов пятёрки не нужно. Более того, никогда не будет являться ошибочным употребление $\{\{e\}\}$ и $\{\varnothing\}$ в качестве множеств входных и выходных состояний: так, если $\varnothing$ мы удалили как недостижимое состояние, то $\text{PRI}\,(A, B)$ задаёт пустой язык, и поэтому множество его выходных состояний может быть задано не только как $\varnothing$, но и как $\{\varnothing\}$.

В качестве примера продолжим рассмотрение пары языков из [1], [2]. Напомним, что проводимые построения выполнялись в этих статьях для такой пары:

$$A = \{aaa, aabba, abba, bb\}, \quad B = \{aaaa, abb, abba, bbb\}.$$

В результате работы алгоритма нами было построено итерационное дерево, повторённое ниже на рис. 1[28].

На основе этого дерева согласно приведённым выше определениям[29] мы получаем следующий морфизм h_A:

$$h_A(0) = aaa,\ h_A(1) = aabba,\ h_A(2) = abba,\ h_A(3) = bb.$$

А на его основе мы формируем следующий (детерминированный) первичный автомат $\text{PRI}\,(A, B)$:

Таб. 1. Первичный автомат для рассматриваемого примера.

			0	1	2	3
←	0	Ø	0	0	0	0
→	1	{ε}	2	0	3	4
	2	{aaa}	5	3	0	0
	3	{ε,a}	6	0	3	7
	4	{bb}	0	0	0	8
	5	{aa}	9	0	0	0
	6	{ε,aaa}	10	3	3	4
	7	{ε,abb,bb}	10	3	3	11
	8	{b}	0	0	0	1
	9	{a}	1	0	0	12
	10	{aa,aaa}	13	3	0	0
	11	{b,bb}	0	0	0	14
	12	{ε,abb}	10	3	3	4
	13	{a,aa}	3	0	0	12
	14	{ε,b}	2	0	3	15
	15	{ε,bb}	2	0	3	11

Отметим ещё раз, что получающиеся при построении автомата недостижимые состояния мы в таблице не приводим.

Приведём ещё некоторые комментарии. Все состояния обозначены не только подмножествами языка $\text{opref}\,(B)$, но и числами; чисел получается 16, и мы применяем номера от 0 до 15. Поэтому – в отличие от примеров из наших предыдущих статей, где число состояний практически никогда не превышало 9 – здесь, например, значение 13 означает именно состояние 13, а *не* два состояния (1 и 3).

Для порядка состояний мы всегда (здесь и далее) будем придерживаться следующих естественных соглашений:

- $\varnothing$ всегда будет иметь номер 0;

[24] Однако приведённые определения «не запрещают» применять к словам над алфавитом Δ_A не только морфизм h_A, но и морфизм h_B – мы это предполагаем делать в одной из следующих публикаций.

[25] Как и ранее, условно считаем слова этого языка упорядоченными – т. е. считаем, что мы *знаем* соответствие различных букв алфавита Δ_A и различных слов языка A.

[26] В качестве пояснения последней формулы отметим ещё раз, что q_1 – это некоторое подмножество множества $\text{opref}\,(B)$, т. е. подмножество языка Σ^*.

Кроме того отметим, что приведённое нами определение является удачным как для обычных недетерминированных автоматов, так и для детерминированных: в последнем случае при применении специального варианта функции переходов (возможной для всюду определённых автоматов – "total") мы можем считать результатом этой функции состояние – а не множество состояний, как обычно. Повторим, что в случае использования приведённого определения функции переходов разницы нет.

[27] Важно, что *бесполезные состояния мы удалять не будем*: в случае их удаления мы в некоторых случаях получаем «вместо автомата» пустое множество. А пустой язык в наших моделях будет – а именно, в тех ситуациях, когда мы удалим из автомата (как недостижимое) единственное выходное состояние $\varnothing$.

[28] При этом исправлена имевшаяся в предыдущей статье опечатка – соответствующее ей состояние отмечено жирным контуром.

Автор выражает благодарность М. Э. Абрамяну (Южный федеральный университет), указавшему на эту неточность.

[29] Или просто отождествляя вершины дерева с «красными» пометками с вершинами с равными им «чёрными» пометками.

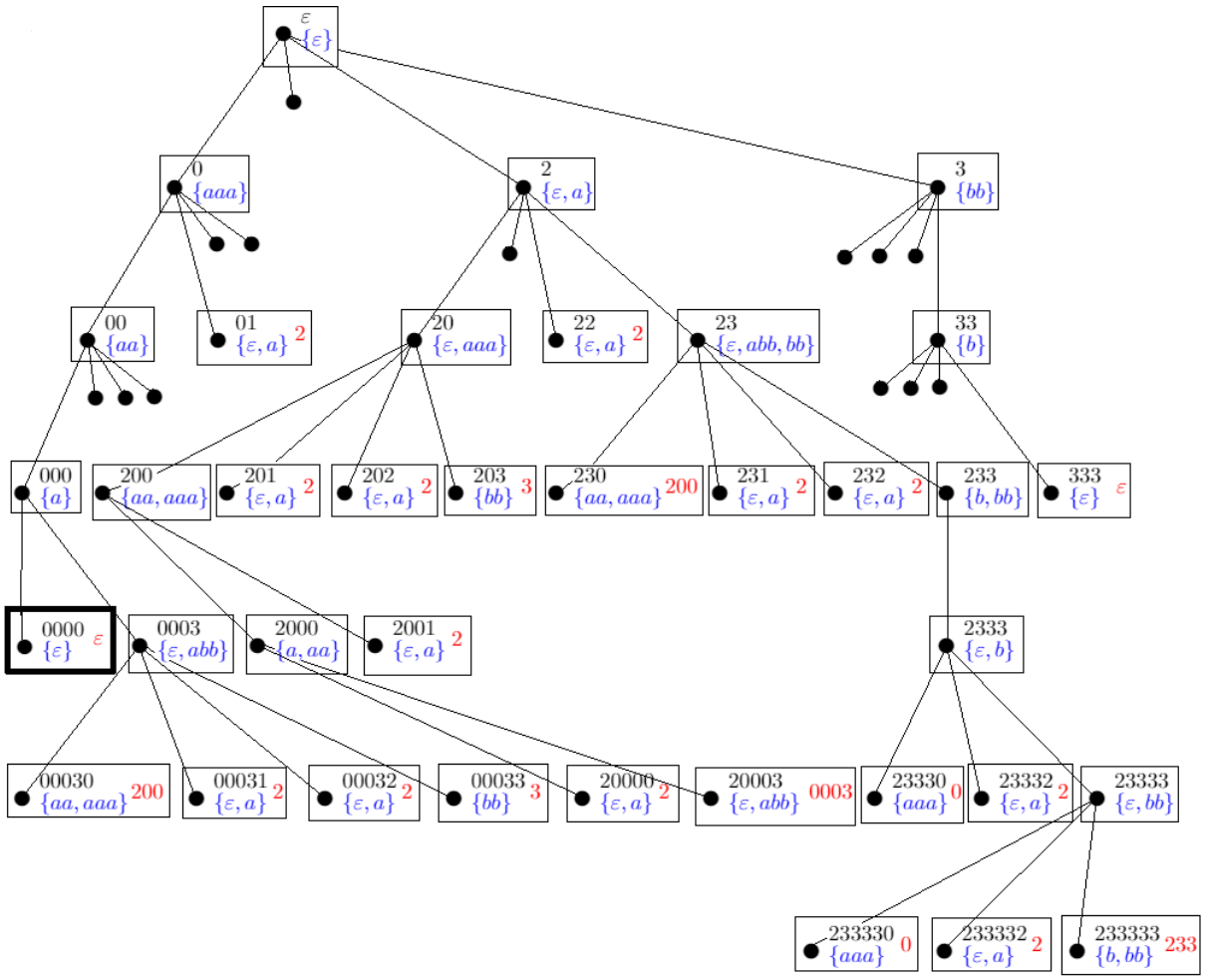

Рис. 1. Итерационное дерево для рассматриваемого примера. Подробности см. в тексте статьи и в [2].

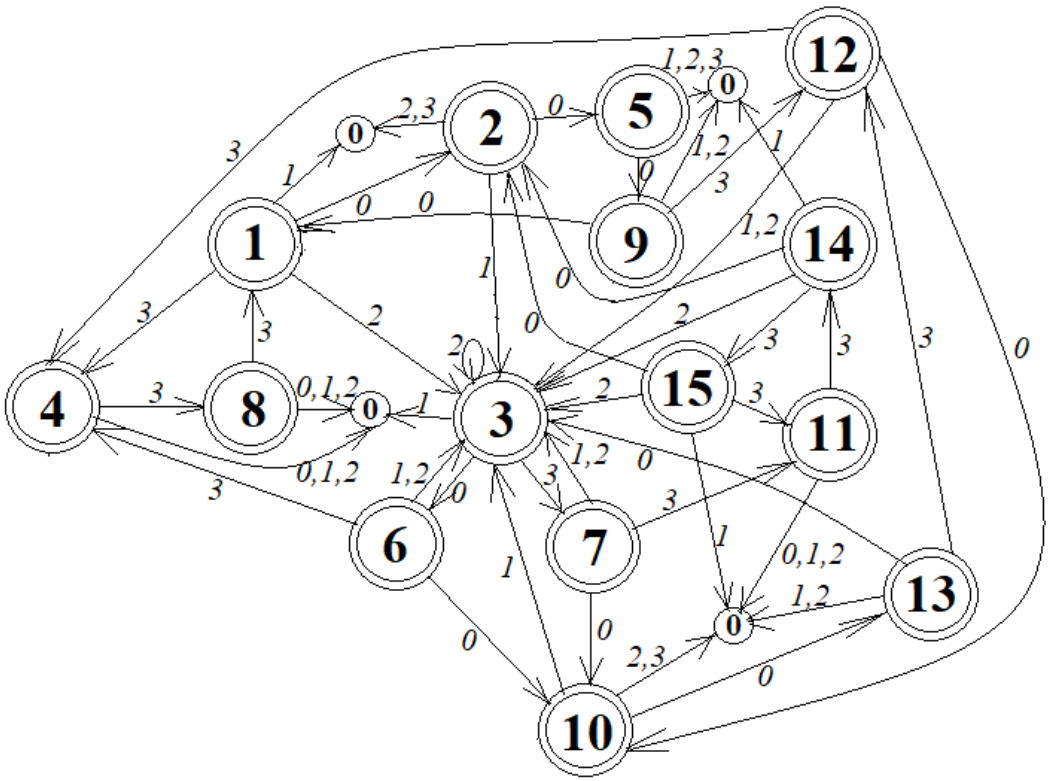

Рис. 2. Первичный автомат для рассматриваемого примера. (Вход – состояние 1, один выход – состояние 0.)

- $\{e\}$ всегда будет иметь номер 1;
- остальные состояния будут «получать номера» в порядке их появления в таблице.

И, по-видимому, невозможно «перепутать» применение значений от 0 до 3 для двух целей: для букв алфавита Δ_A и для номеров состояний; всё понятно из контекста.

На рис. 2 тот же самый автомат приведён в виде графа; Для отсутствия очень большого числа самопересечений дуг графа автомата мы нарисовали вершину 0 особым способом.

Теперь переформулируем основной результат статей [1], [2] в виде следующей теоремы; приведём для неё более короткое доказательство.

Теорема 1: Условие $A \trianglelefteqslant B$ выполнено тогда и только тогда, когда

$$\mathcal{L}(\mathrm{PRI}\,(A,B)) = \varnothing. \tag{2}$$

Доказательство. Сначала для любого выбранного состояния $q \in Q_B$ докажем по индукции, что для любого слова

$$u_\Delta \in \mathcal{L}^{in}_{\mathrm{PRI}\,(A,B)}(q)$$

(напомним, что это – слово над алфавитом Δ_A) выполнено следующее условие: $u \in q$ тогда и только тогда, когда

$$(\exists v \in B^*)\,(vu = h_A(u_\Delta)) \tag{3}$$

(также напомним, что согласно приведённой формуле u и v – слова над алфавитом Σ). Индукцию проведём по длине слова u_Δ.

База. Для $u_\Delta = \varepsilon$ всё очевидно [30]: слово u единственно (это тоже ε – но здесь как слово из Σ^*), и в качестве v также можно выбрать ε.

Шаг. Пусть для некоторого u_Δ выполнено условие

$$\xrightarrow[\delta_{A-B}]{u_\Delta} q$$

(т. е. читая его автомат попадает [31] в состояние q), причём

$$q \xrightarrow[\delta_{A-B}]{a_\Delta} q' \tag{4}$$

(где $a_\Delta \in \Delta$). Требуемое для шага индукции утверждение – т. е. что для любого слова

$$u'_\Delta \in \mathcal{L}^{in}_{\mathrm{PRI}\,(A,B)}(q')$$

включение $u' \in q'$ если и только если

$$(\exists v' \in B^*)\,(v'u' = h_A(u'_\Delta)),\ -$$

непосредственно выводится из следующих фактов:

- условия (4),
- представления u'_Δ в виде $u_\Delta a_\Delta$;
- предположения индукции (3)
- и определения функции переходов $\delta_{A-B}(q)$ как $\underset{h_A(a)-B}{\Phi}(q_1)$.

Теперь перейдём непосредственно к доказательству теоремы.

[30] Нам «не мешает» то, что в состояние $\{e\}$ (или, в других обозначениях, в состояние 1) могут быть переходы из других состояний; в приведённом выше примере такими другими состояниями являются 8 и 9.

[31] Здесь так правильнее, «попадает», а не «может попасть»: автомат детерминированный.

Достаточность утверждения теоремы. Пусть выполнено (2). Согласно построению автомата $\mathrm{PRI}\,(A,B)$, это возможно тогда и только тогда, когда состояние $\varnothing$ не входит в Q_B (т. е. является недостижимым в первоначально построенном автомате $\overline{\mathrm{PRI}}\,(A,B)$). А для всех остальных состояний (и, следовательно, для всех слов над алфавитом Δ_A) в условии (3) можно выбрать необходимое v, и, следовательно, $A \trianglelefteqslant B$.

Необходимость утверждения теоремы. Обратно, пусть $A \ntrianglelefteqslant B$. Тогда из каждого состояния Q_B есть переход для каждой буквы алфавита Δ_A – причём переход не в состояние $\varnothing$. Поэтому само состояние $\varnothing$ мы уже должны были удалить как недостижимое – т. е. (2). □

Как мы отмечали во введении, разделы V – X будут включены в часть II.

Список литературы

[1] 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.

[2] 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.

[3] 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.

[4] Мельников Б. *Некоторые следствия условия эквивалентности однозначных скобочных грамматик* // Вестник Московского университета, серия 15 («Вычислительная математика и кибернетика»). – 1991. – № 10. – С. 51–53.

[5] Дубасова О., Мельников Б. *Об одном расширении класса контекстно-свободных языков* // Программирование (РАН). – 1995. – № 6. – С. 46–58.

[6] Станевичене Л. *Об одном средстве исследования бесконтекстных языков* // Кибернетика. – 1989. – № 4. – С. 135–136.

[7] Мейтус В. *Разрешимость проблемы эквивалентности детерминированных магазинных автоматов* // Кибернетика и системный анализ. – 1992. – № 5. – С. 20–45.

[8] Sénizergues G. *L(A) = L(B)? decidability results from complete formal systems* // Theoretical Computer Science. – 2001. – Vol. 251. No. 1–2. – P. 1–166.

[9] Гэри М., Джонсон Д. *Вычислительные машины и труднорешаемые задачи*. – М., Мир. – 1982. – 416 с.

[10] 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.

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

[12] Melnikov B., Kashlakova E. *Some grammatical structures of programming languages as simple bracketed languages* // Informatica (Lithuanian Academy of Sciences). – 2000. – Vol. 11. No. 4. – P. 441–454.

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

[14] Скорняков Л. (ред.) *Общая алгебра. Том 1*. – М., Наука. – 1990. – 592 с.

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

[16] Алексеева А., Мельников Б. *Итерации конечных и бесконечных языков и недетерминированные конечные автоматы* // Вектор науки Тольяттинского государственного университета. – 2011. – № 3 (17). – С. 30–33.

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.

Variants of finite automata corresponding to infinite iterative morphism trees. Part I

Boris Melnikov

***Abstract*—Quite briefly, the subject of this paper can be formulated as follows: in the previously considered infinite iterative morphism trees, we combine equivalent states, obtaining in fact a deterministic finite automaton; after that, we investigate some properties of this automaton.**

In more detail, we work with various variants of finite automata, each of which corresponds to some infinite iterative tree of the morphism under consideration. In this case, the automata constructed for different morphisms describe the main properties of the given morphisms. In addition, in each case (i. e., for each variant of the automaton), the following "inverse problem" also arises: to describe a morphism (or simply specify a pair of languages) for which some given automaton is obtained. In addition, the paper describes the possible connection of the material under consideration with problems arising in other areas of the theory of formal languages.

Among the variants of automata corresponding to an infinite iterative morphism tree for a given ordered pair of finite languages, we first define the so-called primary automaton. It is deterministic, defined on sets of words, and each of these sets is a subset of the set of suffixes of the second of the given languages. Next, we consider several variants of nondeterministic automata corresponding to it. After that, we introduce a completely different object, i. e., a simplified primary automaton defined not on sets of words, but on words. Despite the significant difference with automata built on sets of words, all constructions for specific examples of languages can be performed using the same computer program.

Next, we consider the features that appear when applying algorithms that form finite automata to pairs of matching languages. At the end of the paper, we briefly formulate the directions for further work related to the issues discussed in it.

In this Part I, we consider deterministic automata only.

***Keywords*—formal languages, iterations of languages, morphisms, binary relations, infinite trees, algorithms.**

References

[1] 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. 2021. – Vol. 9. No. 4. – P. 1–11 (in Russian).

[2] 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. 2021. – Vol. 9. No. 5. – P. 1–11 (in Russian).

[3] 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.

[4] Melnikov B. *Some consequences of the equivalence condition of unambiguous bracketed grammars* // Bulletin of the Moscow University, Series 15 ("Computational Mathematics and Cybernetics"). – 1991. – No. 10. – P. 51–53 (in Russian).

[5] Dubasova O., Melnikov B. *On an extension of the context-free language class* // Programming (Russian Academy of Sciences). – 1995. – No. 6. – P. 46–58 (in Russian).

[6] Stanevicienė L. *On a research facility without contextual languages* // Cybernetics. – 1989. – No. 4. – P. 135–136 (in Russian).

[7] Meytus V. *The solvability of the equivalence problem of deterministic pushdown automata* // Cybernetics and systems analysis. – 1992. – No. 5. – P. 20–45 (in Russian).

[8] Sénizergues G. *L(A) = L(B)? decidability results from complete formal systems* // Theoretical Computer Science. – 2001. – Vol. 251. No. 1–2. – P. 1–166.

[9] Garey M., Johnson D. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. – San Francisco, Freeman and Company. – 1979. – 338 p.

[10] 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.

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

[12] Melnikov B., Kashlakova E. *Some grammatical structures of programming languages as simple bracketed languages* // Informatica (Lithuanian Academy of Sciences). – 2000. – Vol. 11. No. 4. – P. 441–454.

[13] Lallement G. *Semigroups and Combinatorial Applications*. – NJ, Wiley & Sons, Inc. – 1979. – 376 p.

[14] Skornyakov L. (Ed.) *General Algebra. Vol. 1*. – Moscow, Nauka. – 1990. – 592 p. (in Russian).

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

[16] 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).

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.