Лепестковые конечные автоматы: основные определения, примеры и их связь с полными автоматами. Часть II

Б. Ф. Мельников

Аннотация—**В статье рассматриваются (недетерминированные) конечные автоматы специального вида; в литературе на русском языке мы не нашли публикаций, определяющих название для таких автоматов, – поэтому предлагаем для них новое название, «лепестковые автоматы». (В единственной найденной публикации в англоязычной литературе употреблён термин “semi-flower automata” – по-видимому, не очень удачный.)**

Исследование подобных автоматов очень важно для нескольких тем (тематик), рассмотренных в наших предыдущих публикациях. Прежде всего это связь с алгоритмами решения задач, относящихся к различным вариантам исследования равенства бесконечных итераций двух конечных языков, – в частности, к описанию алгоритмов для решения этих задач в виде специального вида недетерминированных конечных автоматов (т. н. автоматов PRI и NSPRI). Близки к этой теме и другие задачи, которые в принципе также могут быть решены путём перебора всех подмножеств множества потенциальных корней слов некоторого заданного языка (например, задача извлечения корня некоторой степени из языка) – но очень важная вспомогательная цель заключается в том, чтобы решить эти задачи с помощью алгоритмов, менее сложных, чем экспоненциальные.

В качестве ещё двух возможных тематик применения лепестковых автоматов приведём такие. Возможно применение полученных результатов во вспомогательных алгоритмах, необходимых для более общих алгоритмов вершинной и дуговой минимизации недетерминированных конечных автоматов. Кроме того, исследование лепестковых автоматов можно связать с исследованием само́й таблицы бинарного отношения # – поскольку варианты таких таблиц разбивают всё множество регулярных языков на специальные классы эквивалентности.

Возможность применения лепестковых автоматов в последней тематике вытекает из основного результата настоящей статьи – а именно, из приведённого в ней описания алгоритма построения такого лепесткового автомата, таблица бинарного отношения # которого совпадает с заданной таблицей – возможно, с некоторым единственным специально добавленным правым столбцом.

В предлагаемой второй части настоящей статьи мы непосредственно переходим к работе с лепестковыми автоматами. Мы рассматриваем вопросы, связанные с алгоритмом получения соответствующего лепесткового автомата на основе заданной таблицы #.

Ключевые слова—**формальные языки, итерации языков, морфизмы, бинарные отношения, лепестковые автоматы, полные автоматы.**

Предлагаемая часть II настоящей статьи является продолжением части I, т. е. [1]. Мы продолжаем нумерацию разделов, формул, рисунков и таблиц, а нумерация ссылок на использованную литературу – новая. Кроме того, также аналогично части I, несколько таблиц помещены в конец статьи – об этом мы каждый раз будем специально упоминать.

IV. ЕЩЁ НЕМНОГО О МОТИВАЦИИ

В этой части мы непосредственно переходим к работе с лепестковыми автоматами (ранее, в части I, фактически было только их определение). К темам, связанным с мотивацией к исследованию лепестковых автоматов, добавим следующие три – которые связаны именно с автоматами такого вида.

Во-первых, в [2] описывались алгоритмы [1], связанные с объединением букв алфавита, над которым определён рассматриваемый конечный автомат – и, в частности, области применения таких алгоритмов. Подобное объединение – в качестве вспомогательного алгоритма – будет рассматриваться и в настоящей статье далее.

Во-вторых, несложно убедиться, что рассматриваемые здесь лепестковые автоматы играют очень важную роль при описании класса языков, имеющих звёздную высоту, равную 1, – см. в [3] описание соответствующей теории. При этом для наших целей гораздо интереснее немного иная задача, «более конкретная»: описание алгоритма проверки, допускает ли заданный каким-либо образом регулярный язык не просто представления в виде регулярного выражения со звёздной высотой 1, но и конкретно в виде A^* – для некоторого (конечного [2]) языка A, а также – в случае положительного ответа – алгоритм построения такого языка A. По-видимому, для ответа на последние вопросы возможны более простые алгоритмы, чем те, которые могут быть получены на основе [3] [3]. Отметим также, что связанные вопросы мы обсуждали в [4], [5].

В-третьих [4], как мы уже отмечали, лепестковые автоматы можно считать автоматами вида $\mathcal{K}(A)$ (для произвольных конечных языков $A \subseteq \Sigma^*$) – см. [6], [7], [8]. При

Статья получена 14 июня 2022 г.

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

[1] В том числе такие, которые *не являются* алгоритмами эквивалентных преобразований.

[2] Для описания бесконечных регулярных языков нужно по крайней мере одно употребление звезды Клини – т. е. итоговая звёздная высота регулярного выражения будет не менее 2; такие языки и формализмы для их описания для тематики настоящей статьи, по-видимому, интереса не представляют.

[3] У автора настоящей статьи пока нет описания подобных более простых алгоритмов.

[4] Автор выражает благодарность Н. И. Крайнюкову (Университет МГУ – ППИ в Шэньчжэне) – за указание этого варианта связи с другими областями теории формальных языков.

этом мы фактически рассматриваем специальные *классы* регулярных языков (подмоноиды глобального надмоноида свободного моноида, см. по этому поводу также [9]) причём в качестве операций в них применяем как обычные морфизмы, так и инверсные морфизмы.

При этом интересной является возможность описания *любого* регулярного языка в виде комбинации двух морфизмов и двух инверсных морфизмов, последовательно применённых к «достаточно простому» регулярному языку, который может быть задан регулярным выражением a^*b – см. [10][5]. Поэтому, поскольку мы также применяем инверсные морфизмы (прежде всего в [6], [7], [8]), представляется интересной потенциальная возможность сведéния к рассмотрению автоматов вида $\mathcal{K}(A)$ произвольного конечного автомата (произвольного регулярного языка); эту возможность мы тоже постараемся получить в будущих публикациях – на основе материала статьи [10].

Также к сказанному в части I (во введении) добавим следующее. Рассматриваемые в настоящей статье лепестковые автоматы представляют собой один из инструментов для проверки эквивалентности $A \Leftrightarrows B$ для некоторых конечных языков $A, B \subseteq \Sigma^*$ – см. [6], [7], [8], [11], [12], а также статьи, процитированные в них. С другой стороны, все регулярные языки разбиваются определённым в [13] (см. также [2]) отношением эквивалентности R на бесконечное число подклассов, в каждом из которых у всех языков совпадает таблица отношения #. При этом в каждом из получающихся подклассов имеется некоторый «основной» язык[6] – а именно, язык соответствующего полного автомата, [13]. Поэтому многие из задач исследования эквивалентности $A \Leftrightarrows B$ могут быть сведены к работе с сответствующими полными автоматами – и мы собираемся вернуться к этой тематике в следующих публикациях[7].

[5] Ранее в одной из публикаций мы приводили сноску про терминологию – «разницу с точки зрения алгебраистов» между морфизмом и гомоморфизмом. В текущей версии Википедии (июнь 2022-го) такой «разницы» практически нет: «морфизм (в теории категорий) – это сохраняющее структуру отображение одной математической структуры в другую того же типа», а «гомоморфизм – это отображение между двумя алгебраическими структурами одного типа, которое сохраняет операции этих структур». Возможно, на основе этого морфизм можно назвать «более общим» объектом.

Мы, как и ранее, в качестве «морфизма по Саломаа» рассматриваем *только* отображения вида

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

где для каждой буквы $a \in \Delta$ значение $h(a) \in \Sigma^*$ задаётся, и при этом всегда

$$h(a_1 a_2 \dots a_n) = h(a_1)h(a_2)\dots h(a_n).$$

Поэтому инверсный морфизм h^{-1} нужно рассматривать как функцию со множеством значений из $\mathcal{P}(\Delta^*)$:

$$h^{-1} : \Sigma^* \to \mathcal{P}(\Delta^*).$$

И, конечно, некоторые из конкретных значений таких функций могут быть равны $\varnothing$ – что и применяется при конструировании конкретных 2 морфизмов и 2 инверсных морфизмов в процитированной статье.

[6] Конечно же, это название *пока* условное.

[7] Вот одна из подобных задач. Как следует из материала настоящей статьи, любому конечному языку можно *различными способами* поставить в соответствие некоторый лепестковый автомат, а последнему – таблицу #. Можно ли при этом указать такое *конкретное* соответствие (здесь мы имеем в виду *алгоритм выбора* такого соответствия лепесткового автомата языку), что любые конечные языки A и B, находящиеся в отношении эквивалентности $A \Leftrightarrows B$, «получают» одну и ту же таблицу #?

(Ещё отметим следующий факт. В рассматривавшихся нами полных автоматах мы для большинства действий с ними – т. е. для вспомогательных алгоритмов преобразования таких атоматов – не конкретизировали некотор*ого* входного и некотор*ых* выходных состояний[8]. Мы утверждаем, что любая последовательность состояний этого полного автомата определяет – согласно рассматриваемым здесь алгоритмам – необходимый нам лепестковый автомат. При этом входное – «главное» – состояние обязательно является наибольшим в этой последовательности, рассматриваемой в настоящей статье как линейный порядок $>$.)

V. ПОДРОБНЫЙ ПРИМЕР ПОЛУЧЕНИЯ ЛЕПЕСТКОВОГО АВТОМАТА – ПРОДОЛЖЕНИЕ РАССМОТРЕНИЯ

В этом разделе мы продолжаем подробное рассмотрение примера, начатого в части I: мы показываем возможность провести (неэквивалентные) преобразования, приводящие в итоге к (детерминированному) автомату – также имеющему заданную таблицу бинарного отношения # и плюс к этому являющемуся лепестковым.

Как мы уже отмечали во введении, при рассмотрении примера (в разделе III и в настоящем разделе V) применена сквозная нумерация подразделов – поэтому здесь самый первый подраздел называется D.

D. Второе неэквивалентное упрощение

В качестве возможного выхода из ситуации, описанной в конце предыдущего подраздела, мы удалим «все красные состояния» (кроме, конечно, того, которое было обозначено особым образом, т. е. Ъ). Однако это удобнее сделать не для последнего, а для предпоследнего варианта, т. е. не для таб. 10, а для таб. 9. Специально отметим, что, в отличие от подавляющего большинства рассматриваемых в наших публикациях примеров, здесь мы начинаем построение группы связанных автоматов с одного из автоматов для *зеркального* языка – а именно с того из них, который мы обычно называем каноническим к зеркальному. Этим и вызвано обозначение исходного автомата $(K'')^R$, см. таб. 11.

(Таб. 11 приведена в конце текста статьи.)

В этой таблице предполагается, что «исходный автомат» был обозначен K'', причём для упрощения обозначений мы пользуемся тем, что оба этих автомата – K'' и $(K'')^R$ – канонические[9,10,11]. При этом пока не обращаем внимания на самый правый столбец (находящийся «за пределами» этого автомата).

[8] Как обычно при рассмотрении детерминированных автоматов – полный автомат является одним из таковых. Ещё раз отметим, что – как следует из материала части I и процитированных в ней статей – практически все построения, связанные с полными автоматами, могут быть выполнены и *без фиксации* входного и выходных состояний.

[9] Как обычно в подобных ситуациях – т. е. при рассмотрении *новых* языков – это означает, что они являются каноническими для *своих* языков. При этом в данном случае они по построению определяют языки, зеркальные один к другому.

[10] Напомним, что во всех публикациях, в том числе здесь, мы рассматриваем канонические автоматы *без* (единственного возможного) «дохлого» состояния (dead state).

[11] Специально повторим, что таб. 11 была получена из таб. 9 путём удаления из неё всех (кроме одного) «красных состояний» – т. е. состояний, не имеющих аналогов в автомате $(K^{\#})^R$.

Итак, мы считаем построенный автомат $(\mathsf{K}'')^{\mathsf{R}}$ *зеркальным к искомому*. Однако, несмотря на полученный автомат-ответ, нам всё-таки нужно ответить ещё и на такой вопрос: *является ли построенный автомат искомым?* [12] Или, иными словами: совпадает ли таблица построенного автомата [13] с исходной таблицей отношения # (таб. 1) – конечно, с одним «новым добавленным» столбцом? Получаем ли мы такую таблицу, которая подходит под *шаблон*, приведённый в следующей таб. 12?

Таб. 12. Таблица шаблона для искомого бинарного отношения #.

	X	Y	Z	U	Ъ
A		#	#		?
B	#		#		?
C	#	#	#	#	?
D	#	#	#		?

При этом понятно, что *в общем случае* незаполненный новый столбец в принципе может совпасть с одним из столбцов заданного отношения (таб. 1); в этом случае:

- в процессе канонизации нового зеркального автомата два столбца-состояния объединяются,
- столбцов становится на 1 меньше (т. е. построенная таблица полностью совпадёт с исходной),
- а этот «совпадающий» столбец описывает «главное состояние» зеркального лепесткового автомата, т. е., другими словами, это «главное состояние» имеется априори.

(Также повторим, что в наших построениях обычно образуется новый столбец – но не новая строка. Это потому, что мы «в качестве главного состояния» выбрали некоторое *уже существующее*, причём это выбранное состояние является состоянием «прямого» автомата – а не зеркального.)

Вернёмся к ранее поставленному вопросу. Тематика настоящего раздела не предполагает доказательства подобного факта в общем случае [14], это будет сделано в разделе VI – поэтому покажем требуемый факт путём подробного рассмотрения полученного выше автомата. Это рассмотрение мы проведём в следующем подразделе.

E. Подробное рассмотрение полученного автомата

Итак, в этом подразделе мы рассматриваем автомат, приведённый в таб. 11. Проведём с ним «обычные преобразования» ([14], [15], [16], [17] и мн. др.).

В правом столбце (находящемся «за пределами» этого автомата, таб. 11) мы заранее записали «подогнанные под ответ» обозначения его состояний (т. е. обозначения, похожие на обозначения состояний автомата исходного) – при рассмотрении его состояний как агрегатные, т. е. с будущим их переименованием с целью получения автомата детерминированного. Однако отметим, что логику работы мы этим не нарушаем, т. е. подобная «подгонка под ответ» вряд ли упрощает общий алгоритм работы рассматриваемой нами задачи: ведь соответствие состояний двух детерминированных автоматов легко проверяется за линейное время.

Итак, воспользовавшись этим правым столбцом, мы переписываем рассматриваемый автомат в виде таб. 13;

(Таб. 13 приведена в конце текста статьи.)

в ней новое состояние Ъ продолжаем обозначать красным цветом.

Далее строим автомат, являющийся зеркальным к последнему. В нашей стандартной нотации его надо обозначить

$$\left(\widetilde{(\mathsf{K}'')^{\mathsf{R}}}\right)^{\mathsf{R}},$$

он приведён в таб. 14 [15].

(Таб. 14 приведена в конце текста статьи.)

После этого мы делаем обычную детерминизацию последнего автомата – получаем автомат, приведённый в таб. 15;

(Таб. 15 приведена в конце текста статьи.)

согласно нашим договорённостям, обозначение для последнего автомата может быть таким:

$$\left(\widetilde{\widetilde{(\mathsf{K}'')^{\mathsf{R}}}}\right)^{\mathsf{R}}. \tag{4}$$

(Понятно, что это просто автомат $\widetilde{\mathsf{L}^{\mathsf{R}}}$ в случае языка $\mathsf{L} = \mathcal{L}(\mathsf{K}'')$ – однако формула (4) важна тем, что она показывает *процесс построения* автомата.)

Здесь, как обычно в настоящей статье, во «внешнем» столбце показаны обозначения состояний автомата для его переписывания в обычном детерминированном виде. Как и ранее, мы «подогнали под ответ» обозначения состояний A, B, C и D – но при этом возникают два вопроса, о которых далее.

Первый вопрос: почему мы *здесь* начинаем с состояния ЪYZ – при том, что стартового (либо «центрального») состояния в начале работы такого алгоритма маркировки агрегатных состояний мы не знаем? Ответ такой: в данном случае мы могли бы начать с любого приведённого «во внутренних клетках» таблицы 14 агрегатного состояния, и результат бы не изменился. Но, конечно, мы опять «подогнали под ответ» – и при этом опять такую «подгонку» вряд ли можно считать излишним упрощением проделываемых построений [16].

Второй вопрос: почему мы 3 состояния из 4 обозначили, добавив штрихи ($'$)? Здесь ответ следующий. «Желаемый» (для получения в результате построений) автомат был бы таким – таб. 16;

(Таб. 16 приведена в конце текста статьи.)

такой автомат получается *из исходного автомата* $\mathsf{K}^{\#}$ *путём удаления из него всех букв, которые мы и удалили* выше в процессе описанных в статье построений [17].

[12] Точнее – является ли он зеркальным к искомому?

[13] «Прямого», т. е. *не* того зеркального, который нами получен.

[14] Более того, мы ещё строго *не сформулировали алгоритм* построения автомата по таблице # – поэтому мы пока просто не можем рассматривать подобные доказательства.

[15] В этой таблице красный цвет уже вряд ли даст полезную информацию – и мы возвращаемся к «монохромной» записи.

[16] Как и в предыдущих ситуациях, «правильное» переобозначение полученных состояний может быть проведено за линейное время.

[17] Напомним, что удаляемое множество букв алфавита, первоначально состоявшего из 16 букв

$$\mathsf{X{-}A}, \ldots, \mathsf{X{-}D}, \ldots, \quad \ldots, \mathsf{U{-}A}, \ldots, \mathsf{U{-}D}$$

(Немного точнее. Этот «желаемый» автомат лепестковым не является. Но очевидно, что на его основе легко получить автомат лепестковый – путём удаления нескольких переходов.)

Однако реально после переобозначений состояний автомата (4) получается другой автомат – обозначим его K‴, таб. 17.

(Таб. 17 приведена в конце текста статьи.)

Такой автомат может быть получен и другим способом: аналогично предыдущему, он получается *из автомата* K′ (вместо $K^{\#}$ в прошлом случае) *путём удаления из него тех же самых букв*.

То есть мы получили «не самый желаемый» автомат – однако он удовлетворяет сформулированным выше требованиям. Действительно, таблица бинарного отношения # полученного автомата K‴, приведённая ниже (таб. 18), подходит под шаблон, который мы привели выше в таб. 12.

Таб. 18. Таблица бинарного отношения # полученного автомата.

#	X	Y	Z	U	Ъ
A	—	#	#	—	#
B'	#	—	#	—	—
C'	#	#	#	#	—
D'	#	#	#	—	—

#	X	Y	Z	U	Ъ
A	0	1	1	0	1
B'	1	0	1	0	0
C'	1	1	1	1	0
D'	1	1	1	0	0

Возникает последний вопрос [18]. Как проще всего проверить правильность приведённых построений? Конечно, можно сказать, что всё нам необходимое видно и на основе правого столбца таб. 18. Однако для нас более интересен другой способ: можно подробно рассмотреть более простой автомат (ниже K^{IV}) – автомат, полученный объединением двух букв последнего автомата. Мы приводим этот автомат, а также канонический к зеркальному к нему – они оба могут быть получены из двух предыдущих (таб. 17 и таб. 13) путём объединения двух букв:

$$a_{Y}^{B} \quad \text{и} \quad a_{Z}^{B} \quad —$$

или, в упрощённых обозначениях, применяемых в таблицах, Y—B и Z—B. Новая «объединённая» буква в приведённых далее таблицах обозначена YZ—B. Отметим, что аналогичные алгоритмы объединения двух букв – а также области применения таких алгоримов – рассматривались в [2].

Итак, для наших построений (основанных на исходной таблице #) оптимальной для рассмотрения является такая пара автоматов:

- во-первых, K^{IV} (таб. 19);
- и во-вторых, K_{IV} (таб. 20) [19].

Таб. 19. Автомат K^{IV}.

K^IV	X-A	Y-A	YZ-B	Y-C	Z-A	U-A	U-D
A	—	A	B	C	A	—	—
B	A	—	—	—	A	—	—
C	A	A	—	—	A	A	D
D	A	A	—	—	A	—	—

Таб. 20. Автомат K_{IV}.

K_IV	X-A	Y-A	YZ-B	Y-C	Z-A	U-A	U-D
X	—	—	Ъ	Ъ	—	—	U
Ъ	X	Y	—	—	Z	U	—
U	—	—	—	Ъ	—	—	—
Y	X	Y	—	Ъ	Z	U	U
Z	X	Y	Ъ	Ъ	Z	U	U

Важно также отметить следующее. Если мы будем рассматривать полный автомат для таблицы таб. 18 (понятно, что он отличается от полного автомата, рассматриваемого нами начиная с раздела III, только наличием 5 новых букв-столбцов с единственным переходом в каждом из них), то с помощью приведённого в [13, Prop. 4] *недетерминированного* алгоритма *неэквивалентного* преобразования автомата можно получить, среди прочих, автомат K^{IV}. Этот факт следует из совпадения отношений # для указанных автоматов.

Таким образом, мы получили искомый автомат бинарного отношения для заданного #. То, что этот автомат является лепестковым, следует из проделанных выше построений – использующих выбранный в части I линейный порядок состояний исходного автомата

$$A > B > C > D.$$

В заключение раздела ещё раз специально отметим, что при описанных выше (неэквивалентных) преобразованиях автомата в таблице отношения #, вообще говоря, образуется новый столбец. Но новая строка при таких преобразованиях образоваться не может, поскольку в качестве «главного» состояния мы выбрали некоторое *уже существующее*. Однако и возможное появление при описанных преобразованиях «одинаковых столбцов» не противоречит тому факту, что получающийся автомат (и соответствующий ему регулярный язык) имеют заранее заданную таблицу отношения #.

В частности – не противоречит изложенному материалу появление их «в одном общем квадрате» таблицы переходов автомата, получаемого на основе автомата полного. При этом понятно, что именно здесь названо «общим квадратом»: например, для рассматриваемого примера это все переходы, соответствующие буквам X—A, X—B, X—C и X—D [20]. *Термин «квадрат» мы будем применять и далее*.

зависело от выбираемого порядка состояний строимого лепесткового автомата. (В этой сноске мы, конечно же, использовали такие обозначения букв, которые применяются в таблицах настоящей статьи; они несколько отличаются от обозначений большинства наших предыдущих статей.)

[18] Последний, связанный с рассматриваемым примером.

[19] Отметим, что обозначения состояний в двух последних таблицах – снова «монохромные» и «без штрихов»: нам ни красный цвет, ни штрихи больше не нужны.

[20] Согласно способу построения полного автомата, число таких букв с одинаковым первым элементом – например, как в этом примере, X – совпадает с числом состояний рассматриваемого полного автомата. Поэтому действительно «квадрат», а не «прямоугольник».

VI. ВОЗМОЖНЫЙ АЛГОРИТМ ПОСТРОЕНИЯ ПОДХОДЯЩЕГО ЛЕПЕСТКОВОГО АВТОМАТА – ПО ЗАДАННЫМ ТАБЛИЦЕ ОТНОШЕНИЯ # И ЛИНЕЙНОМУ ПОРЯДКУ НА СОСТОЯНИЯХ

В этом разделе мы сформулируем в виде сформулируем алгоритма изложенное в разделах III и V. Отметим ещё раз, что строимый таким алгоритмом конечный автомат будет детерминированным.

Напомним, что в части I для бинарного отношения [21]

$$\# \subseteq Q_\pi \times Q_\rho$$

введён термин «корректное»; в терминах табличного задания # это означает, что выполнены все перечисленные далее условия:

- в таблице отсутствуют пустые строки [22];
- в таблице отсутствуют пустые столбцы;
- в таблице отсутствуют совпадающие строки;
- в таблице отсутствуют совпадающие столбцы.

Как мы уже отмечали, доказательства того, что именно такие (и только такие) бинарные отношения # возможны для произвольных регулярных языков, см. в [16], [18], [19] и др.

Алгоритм VI.1. *Построение лепесткового автомата.*
Вход:

- корректное бинарное отношение

$$\# \subseteq Q_\pi \times Q_\rho;$$

- линейный порядок, обозначаемый $>$, на множестве Q_π.

Не ограничивая общности будем считать, что для некоторого $n \geqslant 2$ элементы первого множества, образующего бинарное отношение, таковы:

$$Q_\pi = \{A_1,\ A_2,\ \ldots,\ A_n\},$$

и при этом линейный порядок следующий:

$$A_1 > A_2 > \cdots > A_n. \qquad (5)$$

Выход: Лепестковый конечный автомат, «главной» вершиной которого является A_1, удовлетворяющий порядку (5). При этом таблица отношения # языка автомата-ответа соответствует одному из следующих двух вариантов:

- либо получается на основе исходной таблицы # приписыванием ровно одного столбца [23];
- либо совпадает с этой таблицей [24].

Метод.

Шаг 1. На сонове заданного бинарного отношения # строим соответствующий полный автомат.

Шаг 2. Для каждого состояния $Y \in Q_\rho$ [25] рассмотрим соответствующий ему квадрат [26]. В нём для каждой пары $B, C \in Q_\pi$, такой что $A_1 > B > C$, при наличии перехода

$$C \xrightarrow{a_{Y}^{B}} B \quad -$$

его удаляем [27].

Шаг 3. В получившемся автомате возможно наличие неиспользованных букв [28] – удаляем их их рассматриваемого алфавита.

Шаг 4. Объединяем все возможные пары букв согласно [2] – в случае эквивалентности букв этой пары, в [2] обозначенной записью $\overset{K}{\|}$, где K – рассматриваемый автомат.

Шаг 5. Для A_1, рассматриваемого как стартовое состояние, удаляем все недостижимые состояния [29].

Получающийся автомат с единственным выходным состоянием A_1 является искомым.

Конец описания алгоритма.

Далее для построенного автомата будем использовать обозначение, более сложное, чем приведённое ранее: обозначая линейный порядок (5) символом σ, будем обычно писать

$$K^{\#}_{\sigma}$$

(вместо применявшихся в рассмотренных примерах более простых записей – K', K'' и т. п.); как видно, такое обозначение отражает как исходную таблицу отношения

- и автоматом $(\widetilde{L^R})^R$;
- и полным автоматом (complete automaton), для которого мы в предыдущих публикациях ещё не ввели обозначения;
- и автоматом-ответом (автоматом результирующим).

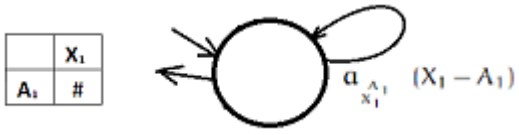

Рис 3. Таблица (без номера) отношения # и соответствующий ей лепестковый автомат; оба объекта – для вырожденного случая (1 состояние).

При этом во всех указанных ситуациях нужны специальные обозначения единственного состояния (на рисунке его обозначение опущено) и/или буквы (приведено два возможных варианта её обозначения, применяемых в статье).

[21] Мы *заранее* считаем, что бинарное отношение задано на паре множеств, обозначенных так же, как и множества состояний двух детерминированных автоматов. Однако, конечно, в алгоритме задаётся только сама таблица этого бинарного отношения.

[22] То есть – при рассмотрении в качестве элементов таблицы значений 0 и 1 – отсутствуют строки, состоящие только из элементов 0.

[23] Произвольного, но не пустого и не совпадающего ни с одним из уже имеющихся в ней столбцов.

[24] Сразу отметим, что не представляет проблемы и вырожденный случай, когда n=1; совпадение двух таблиц при этом имеется. В этой ситуации возможна единственная таблица отношения # и соответствующий ей автомат (оба объекта приведены далее в этой же сноске, на рис. 3).

Изображённый на рисунке простейший автомат при этом можно считать:

- и автоматом $\widetilde{L}$ (можно также считать, что именно приведённым автоматом задаётся язык, обозначенный здесь как L);
- и автоматом $\widetilde{L^R}$;

[25] Состояние канонического автомата для зеркального языка здесь специально обозначено Y, а не X (как чаще в наших предыдущих статьях): буквой X желательно всегда обозначать единственное *стартовое* состояние такого автомата – каковым рассматриваемое нами состояние, вообще говоря, не является.

[26] См. выше про употребление в настоящей статье термина «квадрат».

[27] Из описания шага 2 можно понять, что наличие ровно 2 элементов в множестве Q_π также представляет собой вырожденный случай – поскольку при этом ни для какого квадрата ни одной нужной тройки вида $A_1 > B > C$ выбрать нельзя. Однако этот факт не влияет на описание алгоритма и последующее доказательство его корректности.

Отметим ещё, что вырожденность этого случая (ровно 2 элемента в множестве Q_π) можно было показать ещё при формулировке описания алгоритма (его входа и выхода) – там же, где мы показали вырожденность случая с 1 состоянием.

[28] Иными словами – столбцов, оказавшихся «пустыми» после описанных на шаге двух удалений.

[29] Из материала статьи можно заключить, что бесполезных состояний при описанном построении не будет. Впрочем, можно сформулировать ещё и требование об удалении бесполезных состояний.

#, так и использованный при построении лепесткового автомата линейный порядок.

VII. ДОКАЗАТЕЛЬСТВО КОРРЕКТНОСТИ АЛГОРИТМА

Будем рассматривать автомат $K^{\#}_{\sigma}$, построенный в предыдущем разделе. Здесь мы покажем, что такой автомат является искомым; при этом то, что он является лепестковым, следует из алгоритма его построения [30] – поэтому докажем, что он является каноническим [31] и, кроме того, что таблица отношения # определяемого им языка удовлетворяет требованиям, сформулированным в алгоритме VI.1.

Сразу отметим, что автомат $K^{\#}_{\sigma}$ – детерминированный: это следует из того, что детерминированным является применявшийся при его построении автомат $K^{\#}$, в который при его модификации в алгоритме VI.1 новые переходы не добавлялись (только удалялись).

Во-вторых – покажем, что автомат $K^{\#}_{\sigma}$ является каноническим. Поскольку мы уже знаем, что он детерминированный – то для того, чтобы он удовлетворял определению канонического автомата, достаточно отсутствие в нём эквивалентных состояний; а последнее будет следовать из отсутствия в нём одинаковых строк [32]. Рассмотрим некоторый квадрат его таблицы #. Согласно определению автомата $K^{\#}$, все строки этого квадрата могут быть одного из следующих двух видов.

1. Либо все элементы строки являются $\varnothing$ [33, 34].
2. Либо такая строка представляет собой последовательность

$$(A_1, A_2, \ldots, A_n),$$

соответствующую буквам

$$a_{Y^{A_1}}, a_{Y^{A_2}}, \ldots, a_{Y^{A_n}},$$

рассматриваемым в том же самом порядке.

Из условия выполнения 2-го случая сразу следует несовпадение рассматриваемых строк: из рассматриваемых двух строк мы удаляем разные подмножества переходов, см. таб. 21 [35] – в ней красный цвет используется так же, как и ранее, в подробно рассмотренном примере построения автомата $K^{\#}_{\sigma}$. А 1-й случай *одновременно для всех квадратов* выполняться не может: если бы он выполнялся для всех квадратов, то в автомате $K^{\#}$ соответствующие две строки были бы эквивалентными [36] – что невозможно (противоречит его определению).

Таб. 21. Пример одного из квадратов бинарного отношения # и процесс построения автомата $K^{\#}_{\sigma}$ для этого квадрата.

K^#_σ		Y-A1	Y-A2	Y-A3	Y-A4	Y-A5	
A1	...	A1	A2	A3	A4	A5	...
A2	...	—	—	—	—	—	...
A3	...	A1	A2	A3	A4	A5	...
A4	...	A1	A2	A3	A4	A5	...
A5	...	—	—	—	—	—	...

(Сказанное в предыдущем абзаце можно дополнительно прокомментировать следующим образом. Даже, например, для самой последней строки – в приведённой таблице-примере это строка-состояние A_5 – в получаемом автомате $K^{\#}_{\sigma}$ не может появляться последовательность из всех элементов $\varnothing$, если её не было в исходном автомате $K^{\#}$. А хотя бы в одном квадрате автомата $K^{\#}$ такой последовательности не было – в противном случае, как мы уже отмечали, в таблице бинарного отношения # была бы пустая строка.)

Осталось доказать, что получившемуся автомату $K^{\#}_{\sigma}$ соответствует заданное бинарное отношение #. Действительно, согласно приведённым выше рассуждениям, каждая непустая часть строки (переходов некоторого состояния), соответствующая некоторому квадрату автомата $K^{\#}$ – остаётся непустой и в автомате $K^{\#}_{\sigma}$. А согласно алгоритму построения автомата $K^{\#}$ ([13] и др.), *любой* подобный переход – пусть это

$$A_i \xrightarrow{a_{Y^{A_j}}} A_j,$$

где по смыслу построений либо $j = 1$, либо $i < j$ – приведёт к появлению нужного состояния Y^{A_i} базисного автомата [37]; сказанное здесь верно для любого варианта $A_i \# Y$, выполняющегося в заданном бинарном отношении #.

Конец доказательства корректности алгоритма.

VIII. ЗАКЛЮЧЕНИЕ

Связь рассматриваемых в настоящей статье лепестковых автоматов с тематикой других наших работ уже была отмечена – во введении, а также в ещё одном разделе, посвящённом непосредственно мотивации (разделы I и IV). При этом самой важной является, по-видимому, связь лепестковых автоматов с ω-автоматами, причём в первую очередь – с т. н. iterating ω-finite automata.

Напомним по этому поводу, что выше, в разделе IV, мы кратко говорили о связи лепестковых автоматов с

[30] Поскольку из любого состояния B могут существовать переходы только в меньшие, чем B, состояния, а также в «главное». (Отметим, что при этом про B даже не надо специально отмечать, что оно не является «главным».)

[31] Как обычно в таких ситуациях, правильнее говорить «каноническим для определяемого *им* языка».

[32] Здесь необходимы такие два замечания про *общий вариант* алгоритма проверки эквивалентности двух состояний.

Во-первых, при проверке такой эквивалентности мы должны рассматривать ещё и выходы автомата (т. е. среди двух эквивалентных состояний не может быть в точности одного выходного). Однако в нашем случае выходным мы считаем единственное состояние, «главное» – т. е. дополнительных алгоритмических сложностей здесь не возникает.

Во-вторых – также для общего случая, причём даже при отсутствии вышеуказанных «проблем» с выходными состояниями – совпадение строк *не является необходимым условием* эквивалентности состояний. Однако при этом должны быть переходы по какой-либо букве в два различных состояния, про которые из предыдущих построений уже известна их эквивалентность; но, как несложно убедиться, для автоматов вида $K^{\#}_{\sigma}$ такое невозможно.

[33] В настоящей статье и в предыдущих наших публикациях в клетках таблиц такие переходы обычно обозначаются прочерком (тире).

[34] Напомним также, что детерминированные автоматы мы всегда рассматриваем без возможного «дохлого состояния» (“dead state”) N. А при наличии такого состояния переходы $\varnothing$ соответствовали бы переходам в такое N.

[35] Запись доказательства «в виде формул» приводить, по-видимому, незачем.

[36] Более того, такой вариант приводил бы к некорректности соответствующего отношения # – наличия в нём пустых строк.

[37] В наших предыдущих публикациях такое состояние иногда обозначается непосредственно записью $A_i \# Y$. При этом смысл такой записи – т. е. что именно имеется в виду, элемент бинарного отношения или состояние базисного автомата – всегда понятен из контекста.

возможным построением «хорошего» регулярного выражения для рассматриваемого языка – т. е. регулярного выражения со звёздной высотой, равной 1 ([3] и др.). Понятно, что при естественном перенесении определения звёздной высоты «обычного»[38] конечного автомата на случай ω-автоматов – именно у автоматов, названных в [20] [39] термином “iterating ω-finite automata”, будет именно такое значение звёздной высоты.

Кроме того, в [20] фактически была доказана возможность перенесения алгоритма детерминизации обычных конечных автоматов на случай ω-автоматов – причём для «сильно связанной версии»[40]. Важно добавить, что такая «сильно связанная версия» исследовалась для обычных (не ω-) автоматов в других процитированных публикациях [21], [22]. Кроме того, дополнительное требование о «сильной связанности» не влияет на алгоритмы проверки звёздной высоты регулярного языка: такие алгоритмы могут быть применены к каждой сильно связанной компоненте графа переходов автомата по отдельности.

Понятно, что аналогично случаю обычных конечных автоматов, для ω-случая также интересны такие автоматы, которые могут рассматриваться как автоматы-инварианты; в классическом случае кроме чаще всего употребляемого канонического автомата таковыми (т. е. автоматами-инвариантами заданного регулярного языка) также являются:

- универсальный автомат [16], [23],
- базисный автомат [18], [24],
- полный автомат [13][41].

Однако, несмотря на то, что фактически в части I статьи [20] канонический ω-автомат *был* определён[42] – *формального* определения этого объекта там не было: было определение ω-автомата $\mathcal{P}(\mathcal{K})$, но, понятно, это «не совсем» требуемый автомат. Поэтому в будущей части II статьи об ω-автоматах (которая должна продолжить часть I, т. е. [20]) мы предполагаем привести алгоритм проверки эквивалентности двух сильно связанных ω-автоматов.

Список литературы

[1] Мельников Б. *Лепестковые конечные автоматы: основные определения, примеры и их связь с полными автоматами. Часть I* // International Journal of Open Information Technologies. – 2022. – Vol. 10. No. 9. – P. 1–11.

[2] Мельников Б., Долгов В. *Упрощённые регулярные языки и специальное отношение эквивалентности на классе регулярных языков. Часть I* // International Journal of Open Information Technologies. – 2022. – Vol. 10. No. 9. – P. 12–20.

[3] Hashiguchi K. *Regular languages of star height one* // Information and Control. – 1982. – Vol. 53. No. 3. – P. 199–210.

[38] Далее будем писать без кавычек.

[39] Часть II этой статьи планируется к публикации в ближайшее время, см. некоторые подробности далее.

[40] Однако нужно специально отметить следующее. Из части I можно заключить, что вряд ли возможен «самый простой» перенос на случай ω-автоматов классических результатов про обычные конечные автоматы: нужны специальные дополнительные определения и построения.

[41] В предыдущих публикациях мы не раз отмечали, что полный автомат (the complete automaton) является инвариантом – но *не является полным инвариантом* (the complete invariant) заданного регулярного языка. Причина этой возможной терминологической сложности (полный автомат не является полным инвариантом) объяснена в упомянутой статье про полный автомат – а также, вероятно, понятна на основе материала настоящей статьи.

[42] Важно добавить, что, аналогично сказанному в предыдущем абзаце, нужно рассматривать сильно связанные ω-автоматы и соответствующие им ω-языки. (Далее это по умолчанию, и подобных специальных оговорок мы делать не будем.)

[4] Melnikov B. *The star-height of a finite automaton and some related questions* // International Journal of Open Information Technologies. – 2018. – Vol. 6. No. 7. – P. 1–5.

[5] Melnikov B. *Some more on the equivalent transformation of nondeterministic finite automata. Part III. The “adding” algorithm* // International Journal of Open Information Technologies. – 2019. – Vol. 7. No. 12. – P. 1–4.

[6] Мельников Б. *Полурешётки подмножеств потенциальных корней в задачах теории формальных языков. Часть I. Извлечение корня из языка* // International Journal of Open Information Technologies. – 2022. – Vol. 10. No. 4. – P. 1–9.

[7] Мельников Б. *Полурешётки подмножеств потенциальных корней в задачах теории формальных языков. Часть II. Построение инверсного морфизма* // International Journal of Open Information Technologies. – 2022. – Vol. 10. No. 5. – P. 1–8.

[8] Мельников Б. *Полурешётки подмножеств потенциальных корней в задачах теории формальных языков. Часть III. Условие существования решётки* // International Journal of Open Information Technologies. – 2022. – Vol. 10. No. 7. – P. 1–9.

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

[10] Čulík II K., Fich F., Salomaa A. *A homotorphic characterization of regular languages* // Discrete Applied Mathematics – 1982. – Vol. 4. No. 2. – P. 149–152.

[11] 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.

[12] 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.

[13] Melnikov B. *The complete finite automaton* // International Journal of Open Information Technologies. – 2017. – Vol. 5. No. 10. – P. 9–17.

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

[15] Melnikov B. *Once more on the edge-minimization of nondeterministic finite automata and the connected problems* // Fundamenta Informaticae. – 2010. – Vol. 104. No. 3. – P. 267–283.

[16] Долгов В., Мельников Б. *Построение универсального конечного автомата. II. Примеры работы алгоритмов* // Вестник Воронежского государственного университета. Серия: Физика. Математика. – 2014. – № 4. – С. 78–85.

[17] Мельников Б. *Регулярные языки и недетерминированные конечные автоматы (монография)*. – М., Изд-во Российского государственного социального ун-та. – 2018. – 179 с. – ISBN 978-5-7139-1355-7.

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

[19] Melnikov B., Dolgov V. *Some more algorithms for Conway's universal automaton* // Acta universitatis sapientiae informatica. – 2014. – Vol. 6. No. 1. – P. 5–20.

[20] Melnikov B., Melnikova A. *Some more on ω-finite automata and ω-regular languages. Part I: The main definitions and properties* // International Journal of Open Information Technologies. – 2020. – Vol. 8. No. 8. – P. 1–7.

[21] Мельников Б. *Об одном подклассе класса регулярных языков («сильно связанные» языки): определения и соответствующие канонические автоматы* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 3. – P. 1–10.

[22] Мельников Б. *Автоматы – полные инварианты для сильно связанных регулярных языков* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 6. – P. 1–10.

[23] Lombardy S., Sakarovitch J. *The universal automaton* // Logic and Automata, Amsterdam University Press. – 2008. – P. 457–504.

[24] Melnikov B., Melnikova A. *A new algorithm of constructing the basis finite automaton* // Informatica (Lithuanian Academy of Sciences Ed.). – 2002. – Vol. 13. No. 3. – P. 299–310.

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.

Таб. 11. Автомат $(K'')^R$.

(K'')^R	X-A	Y-A	Y-B	Y-C	Z-A	Z-B	U-A	U-D	
BCD	—	—	A	A	—	A	—	C	X
A	BCD	ACD	—	—	ABCD	—	C	—	Ъ
C	—	—	—	A	—	—	—	—	U
ACD	BCD	ACD	—	A	ABCD	—	C	C	Y
ABCD	BCD	ACD	A	A	ABCD	A	C	C	Z

Таб. 13. Автомат $\widetilde{(K'')^R}$.

^R-det	X-A	Y-A	Y-B	Y-C	Z-A	Z-B	U-A	U-D
X	—	—	Ъ	Ъ	—	Ъ	—	U
Ъ	X	Y	—	—	Z	—	U	—
U	—	—	—	Ъ	—	—	—	—
Y	X	Y	—	Ъ	Z	—	U	U
Z	X	Y	Ъ	Ъ	Z	Ъ	U	U

Таб. 14. Автомат $\left(\widetilde{(K'')^R}\right)^R$.

^R-det-^R	X-A	Y-A	Y-B	Y-C	Z-A	Z-B	U-A	U-D
X	ЪYZ	—	—	—	—	—	—	—
Y	—	ЪYZ	—	—	—	—	—	—
Z	—	—	—	—	ЪYZ	—	—	—
U	—	—	—	—	—	—	ЪYZ	XYZ
Ъ	—	—	XZ	XYZU	—	XZ	—	—

Таб. 15. Автомат $\left(\widetilde{\widetilde{(K'')^R}}\right)^R$.

(...)-kan	X-A	Y-A	Y-B	Y-C	Z-A	Z-B	U-A	U-D	
ЪYZ	—	ЪYZ	XZ	XYZU	ЪYZ	XZ	—	—	A
XZ	ЪYZ	—	—	—	ЪYZ	—	—	—	B'
XYZU	ЪYZ	ЪYZ	—	—	ЪYZ	—	ЪYZ	XYZ	C'
XYZ	ЪYZ	ЪYZ	—	—	ЪYZ	—	—	—	D'

Таб. 16. «Желательный» для получения в результате построений автомат.

	X-A	Y-A	Y-B	Y-C	Z-A	Z-B	U-A	U-D
A	—	A	B	C	A	B	—	—
B	A	—	—	—	A	B	—	—
C	A	A	B	C	A	B	A	D
D	A	A	B	C	A	B	—	—

Таб. 17. Построенный автомат K''', полученный переобозначением состояний автомата $\left(\widetilde{\widetilde{(K'')^R}}\right)^R$.

K'''	X-A	Y-A	Y-B	Y-C	Z-A	Z-B	U-A	U-D
A	—	A	B'	C'	A	B'	—	—
B'	A	—	—	—	A	—	—	—
C'	A	A	—	—	A	—	A	D'
D'	A	A	—	—	A	—	—	—

Petal (semi-flower) finite automata: basic definitions, examples and their relation to complete automata. Part II

Boris Melnikov

***Abstract*—This paper discusses (non-deterministic) finite automata of a special kind. We have not found any publications in the literature in Russian that define the title for such automata, so we propose a new name for them, i.e. "petal automata". (In the only publication found in the English-language literature, the term "semi-flower automata" is used, apparently it is not very successful.)**

The study of such automata is very important for several topics discussed in our previous publications. First of all, it is a connection with algorithms for solving problems, related to various variants of the study of the equality of infinite iterations of two finite languages: in particular, to the description of algorithms for solving these problems in the form of a special type of nondeterministic finite automata (i.e., PRI and NSPRI automata). Other problems are also close to this topic, which, in principle, can also be solved by considering all subsets of the set of potential word roots of a given language: for example, it is the problem of extracting the root of some power for the given language. But a very important auxiliary purpose is to solve these problems using algorithms, less complex than exponential ones.

Two more possible topics of the applications of petal automata are as follows. It is possible to apply the obtained results in auxiliary algorithms necessary for more general algorithms of state- and edge-minimization of nondeterministic finite automata. In addition, the study of petal automata can be associated with the study of the binary relation table #, since variants of such tables divide the whole set of regular languages into special equivalence classes.

The possibility of using petal automata in the last topic follows from the main result of this paper, namely, from the considered description of the algorithm for constructing such a petal automaton, the table of binary relation # of which is the same as the given table (perhaps, with the only specially added right column).

In the proposed second part of the paper, we directly go to to work with petal automata. We consider issues related to the algorithm for obtaining the corresponding petal automaton based on a given table #.

***Keywords*—formal languages, iterations of languages, morphisms, binary relations, semi-flower automata, complete automata.**

References

[1] Melnikov B. *Petal (semi-flower) finite automata: basic definitions, examples and their relation to complete automata. Part I* // International Journal of Open Information Technologies. – 2022. – Vol. 10. No. 9. – P. 1–11. (in Russian).

[2] Melnikov B., Dolgov V. *Simplified regular languages and a special equivalence relation on the class of regular languages. Part I* // International Journal of Open Information Technologies. – 2022. – Vol. 10. No. 9. – P. 12–20. (in Russian).

[3] Hashiguchi K. *Regular languages of star height one* // Information and Control. – 1982. – Vol. 53. No. 3. – P. 199–210.

[4] Melnikov B. *The star-height of a finite automaton and some related questions* // International Journal of Open Information Technologies. – 2018. – Vol. 6. No. 7. – P. 1–5.

[5] Melnikov B. *Some more on the equivalent transformation of nondeterministic finite automata. Part III. The "adding" algorithm* // International Journal of Open Information Technologies. – 2019. – Vol. 7. No. 12. – P. 1–4.

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

[7] Melnikov B. *Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part II. Constructing an inverse morphism* // International Journal of Open Information Technologies. – 2022. – Vol. 10. No. 5. – P. 1–8. (in Russian).

[8] Melnikov B. *Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part III. The condition for the existence of a lattice* // International Journal of Open Information Technologies. – 2022. – Vol. 10. No. 7. – P. 1–9. (in Russian).

[9] 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).

[10] Čulík II K., Fich F., Salomaa A. *A homomorphic characterization of regular languages* // Discrete Applied Mathematics – 1982. – Vol. 4. No. 2. – P. 149–152.

[11] 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).

[12] 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).

[13] Melnikov B. *The complete finite automaton* // International Journal of Open Information Technologies. – 2017. – Vol. 5. No. 10. – P. 9–17.

[14] 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).

[15] Melnikov B. *Once more on the edge-minimization of nondeterministic finite automata and the connected problems* // Fundamenta Informaticae. – 2010. – Vol. 104. No. 3. – P. 267–283.

[16] Dolgov V., Melnikov B. *Constructing universal finite automaton. II. Examples of algorithms* // Bulletin of the Voronezh State University. Series: Physics. Mathematics. – 2014. – No. 4. – P. 78–85. (in Russian).

[17] Melnikov B. *Regular languages and nondeterministic finite automata (monograph)*. – Moscow, Russian Social State University Ed. – 2018. – 179 p. – ISBN 978-5-7139-1355-7. (in Russian).

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

[19] Melnikov B., Dolgov V. *Some more algorithms for Conway's universal automaton* // Acta universitatis sapientiae informatica. – 2014. – Vol. 6. No. 1. – P. 5–20.

[20] Melnikov B., Melnikova A. *Some more on ω-finite automata and ω-regular languages. Part I: The main definitions and properties* // International Journal of Open Information Technologies. – 2020. – Vol. 8. No. 8. – P. 1–7.

[21] Melnikov B. *About one subclass of the class of regular languages ("strongly connected" languages): definitions and corresponding canonical automata* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 3. – P. 1–10. (in Russian).

[22] Melnikov B. *Automata – complete invariants for strongly connected regular languages* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 6. – P. 1–10. (in Russian).

[23] Lombardy S., Sakarovitch J. *The universal automaton* // Logic and Automata, Amsterdam University Press. – 2008. – P. 457–504.
[24] Melnikov B., Melnikova A. *A new algorithm of constructing the basis finite automaton* // Informatica (Lithuanian Academy of Sciences Ed.). – 2002. – Vol. 13. No. 3. – P. 299–310.

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.