Упрощённые регулярные языки и специальное отношение эквивалентности на классе регулярных языков. Часть I

Б. Ф. Мельников, В. Н. Долгов

Аннотация—**Рассматриваемое в статье отношение эквивалентности S на классе регулярных языков необходимо для более полного исследования ранее определённого в наших статьях отношения R. Кроме этого, мотивация к рассмотрению отношения S заключается в необходимости его применения для исследования т. н. лепестковых автоматов – которое также было начато в одной из наших предыдущих статей. Конкретно, для получения лепесткового автомата необходим такой алгоритм неэквивалентного преобразования автомата, как объединение двух букв рассматриваемого алфавита, – и с этим преобразованием связаны оба упомянутых отношения эквивалентности на классе регулярных языков, R и S. В свою очередь лепестковые автоматы возникают при описании нескольких различных алгоритмов проверки бинарного отношения эквивалентности в бесконечности, также рассмотренного в нескольких наших предыдущих статьях, в частности – при использовании в этих алгоритмах конечных автоматов PRI и NSPRI.**

Итак, в настоящей статье мы определяем S – специальное бинарное отношение на множестве регулярных языков, показываем выполнение всех свойств отношения эквивалентности – и поэтому отношение S разделяет весь класс регулярных языков на непересекающиеся подклассы. Вследствие этого большинство рассматриваемых в теории языков задач можно решать только для одного представителя каждого такого подкласса – причём обычно желательно рассматривать так называемый упрощённый язык, он один в каждом подклассе.

Понятие упрощённого языка основано на соединении т. н. параллельных букв, и такому упрощённому языку соответствует специально вводимый нами упрощённый конечный автомат. Всё сказанное позволяет работать лишь с конечным числом рассматриваемых регулярных языков, при условии, что соответствующие им конечные автоматы имеют априори ограниченное число состояний.

Оба эти отношения эквивалентности, S и R, сохраняют отношение #, определяемое для конкретного регулярного языка, – и, следовательно, позволяют использовать многие доказанные ранее свойства регулярных языков и недетерминированных конечных автоматов с любым из автоматов его класса эквивалентности и соответствующим ему языком.

В представляемой части I мы приводим определение отношения S, доказываем его самые простые свойства и рассматриваем несложный пример.

Ключевые слова—**регулярные языки, недетерминированные конечные автоматы, бинарные отношения, отношение эквивалентности, упрощение автоматов, алгоритмы.**

Статья получена 17 мая 2022 г.

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

Василий Николаевич Долгов, Ульяновский государственный технический университет (terenga74@mail.ru).

I. ВВЕДЕНИЕ И МОТИВАЦИЯ

Рассматриваемое в настоящей статье отношение эквивалентности на классе регулярных языков – «менее ёмкое», чем ранее определённое нами (но ещё мало исследованное) отношение R, см. про последнее отношение в [1], [2]. Бинарное отношение, рассматриваемое далее, будем называть отношением S – но, возможно, *в будущем мы введём более содержательные обозначения* для обоих отношений [1]. Можно сказать немного иными словами: после определения в [1] отношения R мы так и не начали его серьёзного исследования – поэтому мы начинаем такое исследование в настоящей статье, рассматривая отношение S.

Про бинарное отношение S уже была публикация – [3]; однако:

- во-первых, та публикация малодоступна;
- во-вторых, мы публикуем этот материал на русском языке;
- в-третьих, мы исправляем замеченные опечатки;
- и в-четвёртых, в настоящей статье мы приводим много важных добавлений; в частности, эти добавления связаны с «мотивацией» (необходимостью подробного рассмотрения этой темы) – но и не только с ней; при этом важно отметить, что бóльшая часть мотивации – совсем новая, т. е. она «возникла» уже после выхода статьи [3] (вследствие материала, рассмотренного нами в [2], [4], [5] и других недавних работах).

Перейдём непосредственно к объяснению того, *почему* важны вопросы, связанные с исследованием свойств отношения S (а также «более ёмкого» отношения R). При этом мы изложим сказанное именно из [2] – с существенными изменениями и добавлениями. Основной предмет той статьи заключается в получении т. н. лепесткового автомата [2], обладающего *любой заданной* таблицей отношения # [3], причём автомата, соответствующего к тому же *любому заданному* порядку состояний. В качестве искомого мы строим автомат, практически повторяя в процессе построения (недетерминированный) алгоритм, описанный в [1, Sect. VII], в частности, в [1, Prop. IV]. При этом в качестве одного из примитивов

[1] Впрочем, сто́ит отметить, что сами названия R и S в статьях практически не используются.

[2] По умолчанию – т. е. если не сказано иного – мы имеем в виду недетерминированные конечные автоматы.

[3] Нужно, конечно, добавить про специальные ограничения на эту таблицу (это отношения). Такие ограничения были сформулированы в процитированных публикациях.

мы применяем объединение двух (или нескольких) букв алфавита, над которым задан автомат, в одну букву. Этот вспомогательный алгоритм очень сильно связан с материалом настоящей статьи – в частности, про упрощённые конечные автоматы и упрощённые регулярные языки (раздел V и далее).

На основе всего упомянутого можно увидеть и связь с тематикой других наших работ, [6], [7], [8][4]: ведь лепестковые автоматы (или употребляемые в этих статьях автоматы $\mathcal{K}(A)$) возникают именно при описании нескольких различных алгоритмов проверки рассматриваемого в этих статьях бинарного отношения $\Diamond$, определённого на множестве конечных языков[5], – или, по терминологии [9], на образующих элементах конечных подмоноидов глобального надмоноида свободного моноида[6].

Немного менее очевидной кажется связь с алгоритмами вершинной и дуговой минимизации недетерминированных конечных автоматов – [10], [11] и мн. др. *Очень неформально* эту связь можно объяснить так: причина существования исследованных в [12] автомата Ватерлоо и подобных автоматов заключается в том, что при построении покрывающего автомата ([13] и др.) в автомат-ответ попадают *не все* циклы, соответствующие циклам базисного автомата (см. [14][7]), – и, следовательно, могут «теряться» входы и / или выходы автомата универсального.

Итак, в настоящей статье определяется специальное бинарное отношение на множестве регулярных языков, причём оно обладает всеми свойствами отношения эквивалентности – и поэтому разделяет весь класс регулярных языков на непересекающиеся классы. Вследствие этого для большинства наших целей[8] можно рассматривать только по одному представителю от каждого такого класса; при этом обычно желательно рассматривать так называемый *упрощённый язык* – он один в каждом классе. Понятие упрощённого языка основано на соединении т. н. *параллельных букв* – а такому упрощённому языку соответствует вводимый нами *упрощённый конечный автомат*[9].

Таким образом, отношение эквивалентности S позволяет *ограничить число рассматриваемых регулярных языков* – то есть рассматривать *конечное число* соответствующих конечных автоматов, причём с заранее фиксированным числом состояний. Кроме того (что особенно важно), это отношение эквивалентности сохраняет отношение #, рассмотренное в наших предыдущих работах[10], – и, следовательно, позволяет использовать практически всё ранее доказанное об этом отношении с любым из автоматов того же класса эквивалентности. Например, на основе полученных результатов мы можем применять различные алгоритмы эквивалентных преобразований недетерминированных конечных автоматов не для заданных автоматов, а для их упрощённых аналогов, – мы имеем в виду такие алгоритмы, как:

- построение эквивалентного автомата с минимально возможным числом состояний [11], [15], [16];
- построение эквивалентного автомата с минимально возможным числом дуг [10], [17];
- построение эквивалентного универсального автомата [13], [16], [18];
- построение эквивалентного базисного автомата [14], [19], [20] …

Таким образом мы получаем некоторые новые объекты – которые более приемлемы с точки зрения большинства требуемых характеристик, например, получаем автоматы с меньшим количеством вершин, дуг и т. п. Вообще, упрощённые автоматы для соответствующего языка являются каноническими[11]. Упрощённый язык имеет такое же бинарное отношение # – и это позволяет нам использовать всю предыдущую теорию, в первую очередь изложенную в [1], [21], [22]; при этом особо отметим [23][12]. Из всего сказанного следует, что обычно мы *можем рассматривать упрощённые автоматы и соответствующие им упрощённые языки* – вместо заданных регулярных языков.

(Некоторые другие «пункты мотивации» для рассмотрения бинарного отношения S приведены далее по тексту статьи – в сносках и в заключении.)

Перейдём к содержанию статьи по разделам. В разделе II мы кратко повторяем наиболее важные обозначения, постоянно применемые нами в работах по тематике «конечные автоматы», а также подробно описываем такие обозначения, которые в этих работах встречались редко или вообще не встречались.

Основную идею разделов III, IV и V неформально можно изложить следующим образом: для заданного регулярного языка на множестве букв алфавита Σ удаётся определить зависящее от этого языка (либо от заданного конечного автомата) отношение «параллельности букв» –

[4] А также – работ, процитированных в трёх частях этой статьи.

[5] А возможный «выход» в языки бесконечные, также рассмотренный в одной из наших предыдущих публикаций, вряд ли представляет большой интерес: он, по-видимому, совсем не связан с рассматриваемыми нами алгоритмами.

[6] Термин «супермоноид» мы стараемся не употреблять: он иногда вызывает возражения (отечественных) алгебраистов. Однако при этом сто́ит отметить, что в англоязычной литературе в таком смысле авторы видели *только* термин “supermonoid”, причём обычно в одно слово.

[7] Особо отметим, что в этой статье сделана попытка *классификации* циклов базисного автомата. Впоследствие теория, связанная с этой классификацией, уже была применена в некоторых наших публикациях – однако эта тематика ещё далеко не завершена.

[8] То есть практически для всех приведённых в наших предыдущих публикациях задач, а также для задач, предполагаемых для будущего решения.

[9] И на множестве языков, находящихся между собой в отношении S, возникает *полурешётка* с операцией объединения букв – конечно, это объединение возможно только для параллельных букв. Также полурешётка возникает и для аналогичных операций при рассмотрении отношения R – причём для последнего отношения возможны и более сложные операции, а не только объединение букв. Подробнее такие полурешётки мы предполагаем рассмотреть в одной из будущих публикаций.

(Отметим, что слово «полурешётка» входит в название нескольких наших статей – в том числе процитированных выше. Однако во всех этих статьях мы рассматривали *совершенно иные предметные области*: например, мы рассматривали полурешётку не на автоматах, а на подмножествах множества т. н. потенциальных корней.)

[10] Его определение кратко повторяется далее.

[11] Для заданного регулярного языка L мы обычно рассматриваем *два* канонических автомата – не только для L, но и для зеркального к нему языка L^R. Сами эти автоматы (элементы их пятёрок) обозначаются

$$\widetilde{L} = (Q_\pi, \Sigma, \delta_\pi, \{s_\pi\}, F_\pi) \quad \text{и} \quad \widetilde{L^R} = (Q_\rho, \Sigma, \delta_\rho, \{s_\rho\}, F_\rho).$$

Напомним также, что во всех публикациях, в том числе в настоящей, мы, если специально не сказано иного, рассматриваем канонические автоматы *без* (единственного возможного) «дохлого» состояния (dead state).

[12] В качестве одной из ближайших публикаций предполагается статья о связи рассматриваемого здесь отношения S с проблемой звёздной высоты регулярного языка.

после чего такие параллельные буквы могут быть объеденены в одну, причём без потери каких-либо важных свойств заданного исходного языка. Для этого в разделе III вводится само определение параллельных букв – причём сначала с точки зрения регулярных языков, а потом с точки зрения конечных автоматов. Далее, в разделе IV, рассматриваются простейшие свойства этого отношения, а также приводятся соответствующие примеры.

Начиная с раздела V идёт часть II настоящей статьи. В самóм разделе V мы продолжаем рассматривать свойства отношения параллельности букв – применительно к детерминированным и, в частности, каноническим автоматам. На их основе в разделе вводится понятие упрощённого автомата – и уже для него формулируется одно из его свойств.

Раздел VI озаглавлен «Применение отношения параллельности букв в задачах теории формальных языков». При этом мы имеем в виду, что доказываемые в этом разделе теоремы формулируют такие важные свойства регулярных языков, как вид их отношения #, а также вид функций φ^{in} и φ^{out} для некоторого соответствующего рассматриваемому языку недетерминированного конечного автомата.

Таким образом, до раздела VI включительно мы рассматриваем связь отношения параллельности букв с отношением # для рассматриваемого языка – т. е. фактически эквалентность в «широком» смысле, по отношению #, иными словами – отношение R на регулярных языках. А в разделе VII мы начинаем рассматривать непосредственно эквалентность в «узком» смысле – т. е. собственно отношение S. Также обобщая весь материал статьи, можно сказать, что на его основе (прежде всего – именно на основе материла раздела VII) можно получить алгоритм построения автомата, определяющего заданный язык и не имеющего параллельных букв. Однако, конечно, такой алгоритм желательно оформить специальным образом – и в разделе VIII мы приводим строгое, притом конструктивное, определение подобного автомата (определение-алгоритм).

Раздел IX – заключение. В нём мы кратко формулируем потенциальные ближайшие публикации по основным возможным направлениям работы, связанным с рассматриваемой в статье тематикой.

II. ОСНОВНЫЕ ОБОЗНАЧЕНИЯ

Перейдём к используемым в статье определениям и обозначениям. Мы кратко повторяем наиболее важные обозначения, постоянно применемые нами в работах по тематике «конечные автоматы», а также подробно описываем такие обозначения, которые в этих работах встречались редко или вообще не встречались.

Итак, способ задания конечных автоматов и основные связанные с ними понятия совпадают с применяемыми в наших предыдущих публикациях – подробное их описание приведено, например, в [1], [4]. При этом ε-переходы (т. е. ситуации вида $\delta(q, a) \ni \varepsilon$) мы здесь для некоторого упрощения допускать не будем: как отмечено далее, основным предметом статьи является т. н. параллельность букв – а пустое слово ε буквой не является.

Ещё отметим, что во всех случаях мы *иногда* будем заменять «обычную версию» функции переходов δ на другой вариант подобной функции, обозначаемый γ ([24] и др.), – причём, конечно, *всегда с теми же самыми нижними индексами*. В общем случае функция γ имеет вид

$$\gamma : Q \times Q \to \mathcal{P}(\Sigma) \quad [13],$$

($\mathcal{P}$ – множество подмножеств); и при этом

$$\gamma(q, q') \ni a \quad \overset{\text{Опр.}}{\Longleftrightarrow} \quad \delta(q, a) \ni q'.$$

Далее приведём обозначение, в наших предыдущих работах практически не использовавшееся. Записью $\hat{K}$ мы будем обозначать детерминированный конечный автомат, полученный из K обычной процедурой детерминизации [14] – [24], [25] и мн. др.

После детерминизации обычно применяется процедура канонизации (объединение в одно эквивалентных состояний) – и аналогично языкам, мы будем обозначать канонический автомат, эквивалентный двум последним автоматам, записью $\widetilde{K}$. Пятёрку элементов автомата $\widetilde{K}$ будем обозначать так:

- либо совершенно аналогично пятёрке автомата $\widetilde{L}$, т. е.
$$\widetilde{K} = (Q_\pi, \Sigma, \delta_\pi, \{s_\pi\}, F_\pi)$$
(обычно мы будем так делать, когда нужно подчеркнуть свойства *языка*),
- либо добавляя знак $\widetilde{\ }$ над обычными обозначениями множеств состояний, функции переходов, входов и выходов (когда нужно подчеркнуть свойства *автомата*).

Далее специально повторим лишь способ задания функций φ^{in} и φ^{out} – посольку их можно вводить разными (эквивалентными) способами.

Перед определениями этих функций нужно ввести бинарное отношенине

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

[13] А «в ещё более общем случае», т. е. когда мы ε-переходы допускаем, функция γ имеет вид

$$\gamma : Q \times Q \to \mathcal{P}(\Sigma \cup \{e\}).$$

[14] Здесь возникает следующий вопрос. Если говорить про *канонические* автоматы – то ещё с 1950-х годов известно, что для заданного регулярного языка два построенных любыми алгоритмами канонических (минимальных детерминированных) автомата эквивалентны не только с точки зрения теории языков, но и с точки зрения теории графов – т. е. попросту равны *с точностью до переобозначения состояний*.

Однако: верно ли то же самое для детерминированных автоматов – не обязательно канонических, но обязательно построенных по некоторому конкретному алгоритму? (При этом, конечно, сам этот алгоритм мы «вольны» описать сами.) Несложно показать, что такой факт *тоже верен*, а конкретный алгоритм может быть таким: рассмотреть *все* подмножества множества состояний исходного недетерминированного автомата (часто в этом контексте называемые агрегатными состояниями), далее обычным образом обобщить на них функцию переходов исходного автомата, после чего в полученном детерминированном автомате удалить все бесполезные и недостижимые состояния.

(Заметим, что кратко описанный в предыдущем абзаце алгоритм формулируется, по-видимому, наиболее простым способом – однако, конечно, *на практике* он не является быстрым: во всех примерах из предыдущих публикаций мы использовали немного иные варианты. Возможное совпадение / несовпадение автоматов, построенных какими-либо другими алгоритмами – алгоритмами, зависящими от порядка рассмотрения вершин и / или букв, – мы в настоящей статье обсуждать не будем.)

Таким образом, мы действительно вправе считать, что вводимый и используемый далее автомат $\hat{K}$ определён *строго и однозначно*.

оно определяется для пар состояний автоматов $\widetilde{L}$ и $\widetilde{L^R}$ следующим образом: условие $A\#X$ выполнено тогда и только тогда, когда

$$\left(\exists uv \in L\right)\left(u \in \mathcal{L}^{in}_{\widetilde{L}}(A) \;\;\&\;\; v^R \in \mathcal{L}^{in}_{\widetilde{L^R}}(X)\right),$$

или, в несколько более удобной записи,

$$\left(\exists u, v \in \Sigma^*\right)\left(\xrightarrow[\delta_\pi]{u} A \;\;\&\;\; X \xrightarrow[\delta^R_\rho]{v} \;\;\&\;\; uv \in L\right)$$

(здесь δ^R_ρ – функция переходов автомата $(\widetilde{L^R})^R$, ранее мы не вводили для неё стандартного обозначения[15]). Заметим, что это определение неконструктивное; однако, например, в [24] можно найти эквивалентный конструктивный вариант (иными словами – определение-алгоритм).

Алгоритм объединения двух или нескольких состояний в одно[16] – в том числе иногда объединения неэквивалентных состояний, для которых почему-либо такой процесс нужно осуществить, – мы в некоторых наших предыдущих публикациях обозначали $\mathcal{J}$; см., например, [24], [26] и др.[17] Таким же образом обозначался и автомат, получаемый в результате применения такого алгоритма. Например, для автомата K, имеющего состояния q и q', полученный в результате их объединения автомат может быть обозначен $\mathcal{J}^{q,q'}(K)$; аналогично – при объединении в одно нескольких состояний.

Теперь перейдём непосредственно к функциям разметки состояний φ^{in} и φ^{out} [18]. Первая задаётся в виде

$$\varphi^{in}_K : Q \to \mathcal{P}(Q_\pi)$$

(здесь K – заданный автомат, Q – множество его состояний, остальные обозначения выше уже пояснялись), и при этом условие

$$\varphi^{in}_K(q) \ni A$$

выполнено тогда и только тогда, когда

$$\left(\exists u \in \Sigma^*\right)\left(\xrightarrow[\delta]{u} q \;\;\&\;\; \xrightarrow[\delta_\pi]{u} A\right)$$

(δ – функция переходов автомата K). Вторая функция разметки задаётся в виде

$$\varphi^{out}_K : Q \to \mathcal{P}(Q_\rho)$$

и определяется аналогичным образом для второго канонического автомата, $\widetilde{L^R}$.

(Заметим, что при определении функции φ^{out}_K множество Q обязательно то же самое, что и при определении функции φ^{in}_K. Этот факт позволяет альтернативным способом определить бинарное отношение # – причём *не до*, а уже *после* определения функций φ^{in}_K и φ^{out}_K, на основе этих определений; часто такой альтернативный способ более удобен. А именно, *для рассматриваемого* автомата K мы определяем выполнение отношения $B\#Y$ тогда и только тогда, когда для некоторого состояния q автомата K одновременно выполнены условия

$$\varphi^{in}_K(q) \ni B \quad \text{и} \quad \varphi^{out}_K(q) \ni Y.$$

Далее можно показать, что в предыдущем предложении слова «для рассматриваемого» можно *эквивалентно заменить* на «для любого, определяющего рассматриваемый язык».)

Считаем, что для некоторых двух регулярных языков выполняется бинарное отношение R тогда и только тогда, когдат таблицы бинарных отношений # у них совпадают[19].

III. ОТНОШЕНИЕ ПАРАЛЛЕЛЬНОСТИ БУКВ ДЛЯ АВТОМАТА И РЕГУЛЯРНОГО ЯЗЫКА: ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Неформально основная идея этого и двух следующих разделов может быть изложена таким образом: для заданного над алфавитом Σ регулярного языка мы на множестве букв Σ определяем отношение «параллельности» (являющееся отношением эквивалентности) – после чего каждое получающееся подмножество параллельных букв может быть объеденено в одну букву, причём без потери каких-либо важных свойств заданного регулярного языка. Конкретно, в настоящем разделе приводятся определения таких параллельных букв – сначала с точки зрения регулярных языков, а потом с точки зрения конечных автоматов.

Определение 1: Пусть $L \subseteq \Sigma^*$ – заданный регулярный язык. Пусть также $a, b \in \Sigma$, и для *некоторого* конечного автомата

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

[15] При этом сам автомат $(\widetilde{L^R})^R$ в наших публикациях иногда употребляется; одно из ранее применявшихся его обозначений – <L>. Но мы, по-видимому, ни разу не говорили, что для рассматриваемого регулярного языка L базисный автомат $\mathcal{BA}(L)$ можно определять как *декартово произведение* $\widetilde{L}$ и такого автомата $(\widetilde{L^R})^R$.

Важно отметить, что здесь даже нет необходимости специальным образом вводить это понятие, т. е. можно использовать «обычную версию» декартового произведения: «лишние» состояния *определяемого таким образом* базисного автомата должны оказаться бесполезными и / или недостижимыми, а дуга между двумя состояниями имеется в наличии в том и только том случае, когда такая дуга имеется и при обычном определении.

[16] Конечно, его не надо путать с объединением в одну двух или нескольких букв в одну.

Сто́ит также отметить, что при объединении состояний нам практически всегда нужны *эквивалентные* преобразования рассматриваемых автоматов.

[17] Понятно, что нужны строгие определения – однако, во-первых, сам процесс объединения очевиден, во-вторых, такие строгие определения можно найти в упомянутых публикациях, а в-третьих, для настоящей статьи достаточно только рассматриваемых далее примеров.

[18] Также приводим более удобные определения – но также неконструктивные. Их конструктивные аналоги тоже можно найти в упомянутых публикациях.

[19] При этом можно рассматривать и *важные модификации* определённого таким образом бинарного отношения R.

- Во-первых, можно считать две таблицы бинарных отношений # эквивалентными, если для одной из них можно указать две перестановки (перестановку строк и перестановку столбцов), переводящие эту таблицу во вторую. (Здесь мы применили несколько неформальное описание – однако, понятно, его несложно сделать вполне строгим.) При этом, например, возникает задача *пересчёта* подобных попарно неэквивалентных таблиц (вариантов отношения #).
  Иными словами – здесь речь идёт о проверке изоморфизма двудольных неориентированных графов.
- Во-вторых (и это независимо от предыдущего пункта), при таких рассмотрениях можно фиксировать первую строку и первый столбец – что по смыслу означает фиксацию стартовых состояний автоматов $\widetilde{L}$ и $\widetilde{L^R}$. Следовательно, однозначно определяются и множества финальных состояний обоих этих автоматов.
  (Здесь также возможны различные *модели*, связанные с двудольными неориентированными графами.)

Мы ещё не рассматривали такие варианты бинарных отношений в опубликованных статьях.

такого что $\mathcal{L}(K) = L$, выполнено условие[20]

$$(\forall q \in Q)\ (\forall a \in \Sigma)\ (\delta(q,a) = \delta(q,b)).$$

Тогда мы будем называть буквы a и b *параллельными над этим языком* L, и записывать этот факт следующим образом:

$$a \overset{L}{\parallel} b.$$

Теперь приведём ещё два определения, *эквивалентных* определению 1. Однако подробно доказывать эту эквивалентность мы не будем, поскольку одно из этих новых определений в настоящей статье не понадобится, а второе очевидно.

Определение 2: Буквы a и b называются *параллельными над заданным языком* L, если существует (хотя бы одно) определяющее L регулярное выражение R_0, такое что[21]

$$R_0 = R_0((a+b), c_1, \ldots, c_k),$$
$$\text{где}\ \ c_1, c_2, \ldots, c_k \in \Sigma \setminus \{a,b\}.$$

Определение 3: Буквы a и b называются *параллельными над заданным языком* L, если существует конечный автомат

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

такой что $\mathcal{L}(K) = L$ и при этом

$$(\forall q_1, q_2 \in Q) \quad \big(\gamma(q_1,q_2) \ni a \iff \gamma(q_1,q_2) \ni b\big).$$

(Напомним, что замечание про функции, обозначаемые в наших публикациях γ с индексами, было приведено во введении.)

А следующее определение 4 можно рассматривать как некоторое упрощение определения 1: здесь мы тоже определяем параллельность – но не для языка, а для конечного автомата[22].

Определение 4: Пусть заданы автомат

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

буквы $a, b \in \Sigma$, и пусть также

$$(\forall q \in Q)\ (\delta(q,a) = \delta(q,b)).$$

Тогда буквы a и b будем называть *параллельными над автоматом* K. Будем записывать этот факт следующим образом:

$$a \overset{K}{\parallel} b.$$

IV. ОТНОШЕНИЕ ПАРАЛЛЕЛЬНОСТИ БУКВ ДЛЯ АВТОМАТА И РЕГУЛЯРНОГО ЯЗЫКА: ПРОСТЕЙШИЕ СВОЙСТВА И ПРИМЕРЫ

В этом разделе рассматриваются простейшие свойства отношения параллельности букв, а также приводятся соответствующие примеры.

Утверждение 1: Если $a \overset{K}{\parallel} b$, то $a \overset{\hat{K}}{\parallel} b$ и $a \overset{\widetilde{K}}{\parallel} b$.

Доказательство.

$$\big(\forall Q_1 \in \mathcal{P}(Q)\big)\ \big(\forall a, b \in \Sigma\big)$$

$$\Big(\hat{\delta}(Q_1, a) = \bigcup_{q \in Q_1} \delta(q,a) \overset{\text{Опр.4}}{=} \bigcup_{q \in Q_1} \delta(q,b) = \hat{\delta}(Q_1, b)\Big).$$

Таким образом, при обычном преобразовании произвольного автомата в детерминированный сохраняется параллельность букв, что в общем виде может быть записано как

$$a \overset{K}{\parallel} b \Rightarrow a \overset{\hat{K}}{\parallel} b.$$

А далее, уже в процессе приведения автомата к каноническому виду, мы лишь объединяем некоторые его состояния (эквивалентные), то есть

$$\widetilde{\gamma}(r, q_1) = \hat{\gamma}(r, q_1) \cup \hat{\gamma}(r, q_2),$$

где множества $\hat{\gamma}(r, q_1)$ и $\hat{\gamma}(r, q_2)$ могут содержать буквы a и b только одновременно. То же самое остаётся верным и для $\widetilde{\gamma}(r, q_1)$ (вместо $\hat{\gamma}(r, q_1)$).

Рассуждения, аналогичные приведённым в предыдущей части доказательства, могут быть применены и для «зеркального варианта», т. е. когда

$$\widetilde{\gamma}(q_1, r) = \hat{\gamma}(q_1, r) \cup \hat{\gamma}(q_2, r).$$

(Приведём ещё и такое замечание – важное как для рассмотренного доказательства в частности, так и для практических алгоритмов теории регулярных языков вообще. Принято считать, что алгоритм поиска эквивалентных состояний детерминированного автомата очень прост; однако в монографической литературе – ни в «старой» [25], [27], [28], [29], ни в «новой» [30] и др., – конкретного *полного*, и при этом *удачного*, описания подобного алгоритма найти не удаётся[23]. Конечно, имеется описание алгоритма Хопкрофта [31] – но последний

[20] Конечно же, ниже в равенстве $\delta(q,a) = \delta(q,b)$ имеется в виду совпадение *множеств*.

[21] Мы применяем запись регулярного выражения «как в физике» – т. е. как некоторую функцию (здесь R_0), зависящую от своих аргументов. Аргументами при этом являются либо буквы рассматриваемого алфавита Σ, либо другие регулярные выражения (обычно «простые», в нашем случае – $a + b$).

Ранее авторы встречали применение такого способа возможной «функциональной» записи регулярных выражений в периодической (но не в монографической) литературе – однако конкретной ссылкой мы в настоящее время не располагаем.

[22] Конечно же, обычно в теории формальных языков применяется «прямой порядок» – а не «обратный», как здесь: практически всегда сначала определяется какое-то свойство для формализма – а уже потом для определяемых таким формализмом объектов. Например, однозначность / неоднозначность в контекстно-свободном случае: мы сначала определяем эти понятия для КС-грамматик (менее распространён вариант их определения для магазинных автоматов) – а уже потом для класса КС-языков.

Как один из примеров сказанного здесь отметим, что, несмотря на такой «обратный порядок», определённое выше и определяемое здесь понятия находятся в том же соответствии, что и понятия регулярного выражения и регулярного языка. А именно, если две буквы параллельны на регулярном некотором языке, то этот язык всегда может быть определён, среди прочих, и таким автоматом, для которого эта параллельность не выполняется.

[23] На «студенческих» (образовательных) ресурсах можно найти, например, такое описание алгоритма (конкретные ссылки не приводим, очень похожий текст имеется на нескольких сайтах, первоисточник найти сложно):

> … если мы предположим, что начальные состояния автоматов эквивалентны, то мы можем получить и другие пары эквивалентных состояний. Если в одну из таких пар попадёт заключительное состояние вместе с незаключительным, то начальные состояния автоматов неэквивалентны.

очень сложен для реализации [24]. Мы же ориентируемся на *наше* описание всего процесса канонизации, приведённое в монографии [24], а до того – в статье [17] [25]. Можно показать, что всё сказанное в настоящем доказательстве этому описанию соответствует.) □

Утверждение 2: Если $a \overset{K}{\parallel} b$ и $b \overset{K}{\parallel} c$, то $a \overset{K}{\parallel} c$.

Доказательство.

$$(\forall q \in Q)\,\big(\delta(q,a) \overset{\text{Опр.4}}{=} \delta(q,b) \overset{\text{Опр.4}}{=} \delta(q,c)\big),$$

т. е. $a \overset{K}{\parallel} c$. □

Утверждение 3: Пусть задан регулярный язык L, а для него $a \overset{L}{\parallel} b$ и $b \overset{L}{\parallel} c$. Тогда $a \overset{L}{\parallel} c$.

Доказательство. Из определения 1 вытекает, что

$$\big(\exists K_1,\ \mathcal{L}(K_1) = L\big)\,\big(a \overset{K_1}{\parallel} b\big),$$

а также

$$\big(\exists K_2,\ \mathcal{L}(K_2) = L\big)\,\big(b \overset{K_2}{\parallel} c\big).$$

Поскольку $\widetilde{K_1} = \widetilde{K_2} = \widetilde{K}$ [26], то, согласно утверждению 1,

$$a \overset{K_1}{\parallel} b \Rightarrow a \overset{\widetilde{K_1}}{\parallel} b \Rightarrow a \overset{\widetilde{K}}{\parallel} b$$
$$\text{и}\quad b \overset{K_2}{\parallel} c \Rightarrow b \overset{\widetilde{K_2}}{\parallel} c \Rightarrow b \overset{\widetilde{K}}{\parallel} c\,.$$

Отсюда, согласно утверждению 2, $a \overset{\widetilde{K}}{\parallel} c$, и, следовательно, по определению 1 мы получаем, что $a \overset{L}{\parallel} c$. □

Всё правильно, приведена лаконичная формулировка, которая «в разы понятнее», чем алгоритм Хопкрофта (про него далее)… однако на практике такой алгоритм очень неэффективен; например, посчитаем (estimate) число его операций для случая, когда исходный недетерминированный автомат содержит 6 состояний. Детерминированный аналог такого автомата содержит до $2^6{-}1 = 63$ «агрегатных» состояний, и количество пар состояний равно $63 \cdot 62 / 2 = 1953$. А ведь на этих парах в программе надо организовывать бинарные отношения, определяющие переходы автомата, – и число таких бинарных отношений равно числу букв алфавита; т. е. в нашем примере при стандартной реализации бинарного отношения его таблица будет содержать 1953^2 элементов (более 3.8 миллионов). А после этого нужно получать транзитивные замыкания всех таких бинарных отношений…

[24] Достаточно удачно описание этого алгоритма изложено на Викиконспектах, `https://neerc.ifmo.ru/wiki/` – но, повторим, для реализации этот алгоритм очень сложен. И, несмотря на то, что полученное в примере из предыдущей сноски значение 63 (число состояний соответствующего детерминированного автомата) не нужно возводить в четвёртую степень (поскольку формально сложность алгоритма Хопкрофта $\sim N \cdot \log N$) – однако, по-видимому, реальное временно́е улучшение возникает для значений размерности, существенно бóльших, чем рассматриваемое значение 6 (число состояний исходного недетерминированного автомата).

Впрочем, подробных вычислительных экспериментов мы ещё не проводили – однако некоторые уже сделанные варианты реализации алгоритмов подтверждают сказанную выше информацию об их эффективности.

[25] Сама эта статья опубликована в т. н. «высокорейтинговом» журнале – что, по-видимому, является «косвенным аргументом» отсутствия в литературе описаний *практических* алгоритмов приведения автомата к каноническому виду.

Итак, желательны алгоритмы, эквивалентные упомянутым в двух предыдущих сносках, – однако такие алгоритмы, которые *на практике* выполняются существенно быстрее. По мнению авторов, таковым является наша интерпретация подобного алгоритма из процитированных публикаций.

Ещё отметим, что на основе приведённых в настоящей статье утверждений также удаётся запрограммировать такие быстрые *практические* алгоритмы.

[26] С точностью до переобозначения состояний – поскольку канонический автомат является полным инвариантом регулярного языка.

Итак, для заданного регулярного языка L отношение $a \overset{L}{\parallel} b$ является рефлексивным, симметричным (по определению) и транзитивным; поэтому оно является *отношением эквивалентности*, а множество букв заданного алфавита Σ может быть разбито на подмножества (классы эквивалентности) – которые также зависят от заданного языка L.

Пример 1: Над алфавитом $\Sigma = \{a, b\}$ рассмотрим регулярный язык, определяемый регулярным выражением

$$a^* + b^* + (a + b)^*.$$

Он может быть задан автоматом K, приведённым на рис. 1:

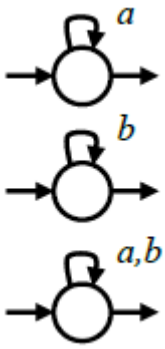

Рис. 1. Заданный недетерминированный автомат K.

(В настоящей статье мы иногда будем опускать обозначения состояний – в тех случаях, когда эти обозначения не нужны.)

На основе K мы получаем эквивалентный детерминированный автомат $\hat{K}$, рис. 2:

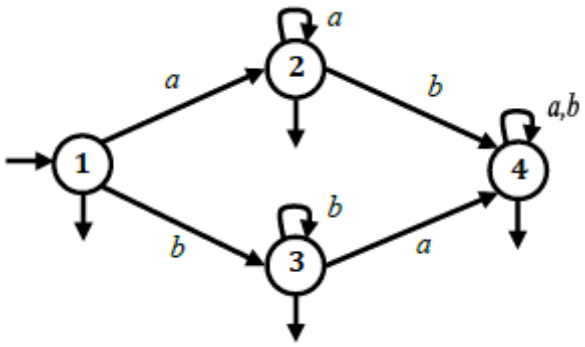

Рис 2. Эквивалентный детерминированный автомат $\hat{K}$.

Все состояния последнего автомата эквивалентны, поэтому после их объединения мы получаем (рис. 3) канонический автомат

$$\widetilde{K} = \mathcal{J}^{1,2,3,4}(\hat{K})\ :$$

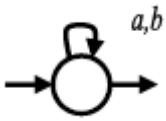

Рис. 3. Эквивалентный канонический автомат $\widetilde{K}$.

Обобщим материал этого раздела следующим образом. Пусть некоторый регулярный язык L задаётся каноническим автоматом $\hat{K}$, у которого для некоторых букв $a, b \in \Sigma$ выполнено условие $a \overset{\hat{K}}{\parallel} b$. Тогда согласно введённым выше определениям выполнено также условие $a \overset{L}{\parallel} b$ [27]. Таким образом мы получаем разделение букв алфавита на классы эквивалентности – причём это разделение связано не с тем, как мы определяем язык (с помощью регулярного выражения, детерминированного конечного автомата, недетерминированного автомата и т. п.), а является неотъемлемым свойством самого языка[28]. (В рассмотренном примере разделение алфавита Σ образует единственный класс эквивалентности $\{a, b\}$.)

Как уже было отмечено, разделы V – IX будут приведены в части II настоящей статьи.

Список литературы

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

[2] 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.

[3] Melnikov B., Dolgov V., Melnikova E. *An equivalence relation on the class of regular languages* // Communications in Computer and Information Science. – 2020. – Vol. 1140. – P. 93–107.

[4] 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.

[5] 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.

[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] Melnikov B., Melnikova A. *Edge-minimization of non-deterministic finite automata* // The Korean Journal of Computational and Applied Mathematics (Journal of Applied Mathematics and Computing). – 2001. – Vol. 8. No. 3. – P. 469–479.

[11] Melnikov B., Tsyganov A. *The state minimization problem for nondeterministic finite automata: the parallel implementation of the truncated branch and bound method* // In: Proceedings – 5th International Symposium on Parallel Architectures, Algorithms and Programming, PAAP–2012. – 2012. – P. 194–201.

[12] Долгов В., Мельников Б. *Об алгоритмах автоматического построения Ватерлоо-подобных конечных автоматов на основе полных автоматов* // Эвристические алгоритмы и распределенные вычисления. – 2014. – Т. 1. № 4. – С. 24–45.

[27] Отметим такие два обстоятельства. Во-первых, этот факт вряд ли можно считать очевидным для исходного регулярного выражения. Во-вторых, мы показали его выполнение таким образом (указанием соответствующих алгоритмов), который даёт возможность построения соответствующих компьютерных программ.

[28] Аналогично отношению # – отношению на состояниях двух автоматов, каждый из которых является полным инвариантом заданного регулярного языка.

[13] Долгов В., Мельников Б. *Построение универсального конечного автомата. II. Примеры работы алгоритмов* // Вестник Воронежского государственного университета. Серия: Физика. Математика. – 2014. – № 1. – С. 78–85.

[14] Долгов В., Мельников Б., Мельникова А. *Циклы графа переходов базисного конечного автомата и связанные вопросы* // Вестник Воронежского государственного университета. Серия: Физика. Математика. – 2016. – № 4. – С. 95–111.

[15] Kameda T., Weiner P. *On the state minimization of nondeterministic finite automata* // IEEE Transactions on Computers. – 1970. – Vol. C-19. No. 7. – P. 617–627.

[16] Polák L. *Minimizations of NFA using the universal automaton* // International Journal of Foundation of Computer Sciences. – 2005. – Vol. 16. No. 5. – P. 999–1010.

[17] Melnikov B. *Once more on the edge-minimization of nondeterministic finite automata and the connected problems* // Fundamenta Informaticae. – 2010. – Vol. 104. No.37. – P. 267–283.

[18] Lombardy S., Sakarovitch J. *The Universal Automaton* // Logic and Automata, Texts in Logic and Games, Amsterdam Univ. Press. – 2008. – Vol. 2, P. 457–504.

[19] 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.

[20] 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.

[21] Melnikov B., Vakhitova A. *Some more on the finite automata* // The Korean Journal of Computational and Applied Mathematics (Journal of Applied Mathematics and Computing). – 1998. – Vol. 5. No. 3. – P. 495–505.

[22] Melnikov B., Dolgov V. *Some more algorithms for Conway's universal automaton* // Acta Universitatis Sapientiae, Informatica. – 2014. – Vol. 6. No. 1. – P. 5–20.

[23] 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.

[24] 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.

[25] Брауэр В. *Введение в теорию конечных автоматов*. – М., Радио и связь. – 1987. – 390 с.

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

[27] Гинзбург С. *Математическая теория контекстно-свободных языков*. – М., Мир. – 1970. – 326 с.

[28] Ахо А., Ульман Дж. *Теория синтаксического анализа, перевода и компиляции. Т. 1*. – М., Мир. – 1978. – 613 с.

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

[30] Pin J.-E. *Mathematical Foundations of Automata Theory*. – Berlin, Springer-Verlag. – 2012. – 310 p.

[31] Хопкрофт Дж., Мотвани Р., Ульман Дж. *Введение в теорию автоматов, языков и вычислений*. – М., Вильямс, 2002. – 528 с.

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.

Василий Николаевич ДОЛГОВ,
заместитель директора Тереньгульского лицея
при Ульяновском государственном
техническом университете (https://ulstu.ru/),
email: terenga74@mail.ru,
elibrary.ru: authorid=700000,
ORCID: orcidID=0000-0001-5925-541X.

Simplified regular languages and a special equivalence relation on the class of regular languages. Part I

Boris Melnikov, Vasily Dolgov

***Abstract*—The equivalence relation S on the class of regular languages, considered in this paper, is necessary for a more complete study of the relation R previously defined in our articles. In addition, the motivation for considering the relation S is the need to apply it to the study of so-called petal (semi-flower) automata, which was also started in one of our previous papers. Specifically, in order to obtain a petal automaton, there is necessary to consider such an algorithm of the nonequivalent transformation of the automaton as the union of two letters of the considered alphabet, and both the mentioned equivalence relations on the class of regular languages, i.e. relations R and S, are associated with this transformation. In turn, petal automata arise when describing several different algorithms for checking the binary relation "equivalence at infinity", also discussed in some our previous papers, in particular, when using PRI and NSPRI finite automata in these algorithms.**

Thus, in this paper we define S, a special binary relation on a set of regular languages, show the fulfillment of all the properties of equivalence relations; therefore the relation S divides the whole class of regular languages into some disjoint subclasses. As a result, for most of the problems of the theory of formal languages, we can take only one representative of each such class; and it is usually desirable to consider the so-called simplified language, it is the only in each such subclass.

The concept of a simplified language is based on the combination of so-called parallel letters, and the simplified finite automaton specially introduced by us corresponds to such a simplified language. All this makes it possible to limit the number of regular languages under consideration to work with a finite number of corresponding finite automata; moreover, such automata have a pre-fixed number of states.

Both these equivalence relations, S and R, preserve the relation # defined for a particular regular language, and, therefore, allow to use many previously proven properties of regular languages and nondeterministic finite automata for arbitrary automaton of its equivalence class and the corresponding language.

In the presented part I, we give a definition of the relation S, then prove its simplest properties and consider a simple example.

***Keywords*—regular languages, nondeterministic finite automata, binary relations, equivalence relation, simplification of automata, algorithms.**

References

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

[2] 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).

[3] Melnikov B., Dolgov V., Melnikova E. *An equivalence relation on the class of regular languages* // Communications in Computer and Information Science. – 2020. – Vol. 1140. – P. 93–107.

[4] 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).

[5] 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).

[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).

[6] *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 a free monoid* // News of higher educational institutions. Mathematics. – 2004. – No. 3. – P. 46–56 (in Russian).

[10] Melnikov B., Melnikova A. *Edge-minimization of non-deterministic finite automata* // The Korean Journal of Computational and Applied Mathematics (Journal of Applied Mathematics and Computing). – 2001. – Vol. 8. No. 3. – P. 469–479.

[11] Melnikov B., Tsyganov A. *The state minimization problem for nondeterministic finite automata: the parallel implementation of the truncated branch and bound method* // In: Proceedings – 5th International Symposium on Parallel Architectures, Algorithms and Programming, PAAP–2012. – 2012. – P. 194–201.

[12] Dolgov V., Melnikov B. *On the automatic construction algorithms of Waterloo-like finite automata based on complete automata* // Heuristic algorithms and distributed computing. – 2014. – Vol. 1. No. 4. – P. 24–45 (in Russian).

[13] Dolgov V., Melnikov B. B. *Building a universal finite automaton. II. Examples of how algorithms work* // Bulletin of the Voronezh State University. Series: Physics. Mathematics. – 2014. – No. 1. – P. 78–85 (in Russian).

[14] Dolgov V., Melnikov B., Melnikova A. *Cycles of the transition graph of a basic finite automaton and related issues* // Bulletin of the Voronezh State University. Series: Physics. Mathematics. – 2016. – No. 4. – P. 95–111 (in Russian).

[15] Kameda T., Weiner P. *On the state minimization of nondeterministic finite automata* // IEEE Transactions on Computers. – 1970. – Vol. C-19. No. 7. – P. 617–627.

[16] Polák L. *Minimizations of NFA using the universal automaton* // International Journal of Foundation of Computer Sciences. – 2005. – Vol. 16. No. 5. – P. 999–1010.

[17] Melnikov B. *Once more on the edge-minimization of nondeterministic finite automata and the connected problems* // Fundamenta Informaticae. – 2010. – Vol. 104. No.37. – P. 267–283.

[18] Lombardy S., Sakarovitch J. *The Universal Automaton* // Logic and Automata, Texts in Logic and Games, Amsterdam Univ. Press. – 2008. – Vol. 2, P. 457–504.

[19] 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.

[20] 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.

[21] Melnikov B., Vakhitova A. *Some more on the finite automata* // The Korean Journal of Computational and Applied Mathematics (Journal of Applied Mathematics and Computing). – 1998. – Vol. 5. No. 3. – P. 495–505.

[22] Melnikov B., Dolgov V. *Some more algorithms for Conway's universal automaton* // Acta Universitatis Sapientiae, Informatica. – 2014. – Vol. 6. No. 1. – P. 5–20.

[23] 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.

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

[25] Brauer W. *Automatentheorie. Eine Einführung in die Theorie endlicher Automaten*. – Stuttgart, B. G. Teubner. – 1984. – 493 S (in German).

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

[27] Ginsburg S. *The mathematical theory of context-free languages*. – Columbus, McGraw-Hill. – 1966. – 232 p.

[28] Aho A., Ullman J. *The Theory of Parsing, Translation and Compiling. Vol. 1. Parsing*. NJ, Prentice Hall. – 1972. – 2051 p.

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

[30] Pin J.-E. *Mathematical Foundations of Automata Theory*. – Berlin, Springer-Verlag. – 2012. – 310 p.

[31] Hopcroft J., Motwani R., Ullman J. *Introduction to Theory of Automata, Formal Languages and Computation*. – Boston, Addison-Wesley, 2006. – 535 p.

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.

Vasily DOLGOV,
Vice-director of Terenga Lyceum
at Ulyanovsk State Technical University
(https://ulstu.ru/),
email: terenga74@mail.ru,
elibrary.ru: authorid=700000,
ORCID: orcidID=0000-0001-5925-541X.