УДК 519.713, 004.021
DOI: 10.25559/SITITO.17.202101.728

Оригинальная статья

Алгоритмы преобразования конечных автоматов, соответствующих бесконечным итерационным деревьям

М. Э. Абрамян[1*], Б. Ф. Мельников[2]

[1] ФГАОУ ВО «Южный федеральный университет», г. Ростов-на-Дону, Российская Федерация
344006, Российская Федерация, г. Ростов-на-Дону, ул. Б. Садовая, д. 105/42
[*] mabr@sfedu.ru

[2] Совместный университет МГУ – ППИ, г. Шэньчжэнь, Китайская Народная Республика
517182, Китайская Народная Республика, провинция Гуандун, г. Шэньчжэнь, р-н Лунган, Даюнь-синьчэн, ул. Гоцзидасюеюань, д. 1

Аннотация

В настоящей статье мы работаем с несколькими различными вариантами конечных автоматов, каждый из которых соответствует бесконечному итерационному дереву, построенному для некоторого заданного морфизма. При этом каждый из построенных для некоторого морфизма автоматов описывает основные свойства этого морфизма. Кроме того, в каждом случае (т. е. для каждого варианта автомата) возникает также следующая «обратная задача»: необходимо описать морфизм (либо просто указать пару языков), для которого получается такой заданный автомат. Мы приводим компьютерную программу построения одного из таких автоматов — т. н. автомата PRI. После этого мы рассматриваем подробный пример автомата PRI для пары несовпадающих языков. Продолжая рассматривать этот пример, мы с последним автоматом выполняем обычные преобразования, описанные и неоднократно применённые в наших предыдущих публикациях, а именно — детерминизацию и канонизацию зеркального автомата для возможного применения полученных результатов в алгоритме минимизации недетерминированных автоматов. В рассматриваемой нами ситуации таким минимальным автоматом является другой строимый на основе заданного дерева морфизма автомат — недетерминированный, т. н. автомат NSPRI# — и равенство этих автоматов (что влечёт эквивалентность PRI и NSPRI#) мы также показываем в статье на примере. На основе автомата NSPRI# с помощью тривиального (но неэквивалентного) преобразования строится недетерминированный автомат NSPRI — подробное исследование которого предполагается в дальнейших публикациях. Интерес представляют также примеры автоматов PRI и NSPRI# для пар совпадающих языков — мы также приводим один такой пример в настоящей статье.

Ключевые слова: алгоритмы, формальные языки, итерации языков, бинарные отношения, бесконечные деревья.

Авторы заявляют об отсутствии конфликта интересов.

Для цитирования: Абрамян, М. Э. Алгоритмы преобразования конечных автоматов, соответствующих бесконечным итерационным деревьям / М. Э. Абрамян, Б. Ф. Мельников. — DOI 10.25559/SITITO.17.202101.728 // Современные информационные технологии и ИТ-образование. — 2021. — Т. 17, № 1. — С. 13-23.

Original article

Algorithms for Converting Finite Automata Corresponding to Infinite Iterative Trees

M. E. Abramyan[a*], B. F. Melnikov[b]
[a] Southern Federal University, Rostov-on-Don, Russian Federation
105/42 Bolshaya Sadovaya St., Rostov-on-Don 344006, Russian Federation
[*] mabr@sfedu.ru
[b] Shenzhen MSU – BIT University, Shenzhen, People's Republic of China
1 Guoji-daxueyuan St., Dayunxincheng, Longgang District, Shenzhen 517182, Guangdong Province, People's Republic of China

Abstract
In this paper, we work with some different variants of finite automata, each of which corresponds to an infinite iterative tree constructed for some given morphism. At the same time, each of the automata constructed for a given morphism describes the main properties of this morphism. Besides, in each case (i.e., for each variant of the automaton), the following "inverse problem" also arises: to describe the morphism (or simply specify a pair of languages) for which such a given automaton is obtained. We present a computer program for constructing one such automaton, so-called PRI automaton. After that, we consider a detailed example of a PRI automaton for a pair of different languages. Continuing to consider this example, we use the last automaton to perform usual transformations described and repeatedly applied in our previous publications, i.e., the determination and canonization of the mirror automaton for possible application of the results obtained in the algorithm for minimizing nondeterministic automata. In the considered situation, such a minimal automaton is another automaton constructed on the basis of a given morphism tree, a nondeterministic automaton, the so-called NSPRI# automaton, and we also show the equality of these automata (which implies the equivalence of PRI and NSPRI#) in the paper by an example. Based on the NSPRI# automaton, a non-deterministic NSPRI automaton is constructed using a trivial (but non-equivalent) transformation; a detailed study of this automaton is expected in future publications. Examples of PRI and NSPRI# automata for pairs of matching languages are also of interest, we also give one such example in this paper.

Keywords: algorithms, formal languages, iterations of languages, binary relations, infinite trees.

The authors declare no conflict of interest.

For citation: Abramyan M.E., Melnikov B.F. Algorithms for Converting Finite Automata Corresponding to Infinite Iterative Trees. *Sovremennye informacionnye tehnologii i IT-obrazovanie* = Modern Information Technologies and IT-Education. 2021; 17(1):13-23. DOI: https://doi.org/10.25559/SITITO.17.202101.728

Введение

Настоящая работа продолжает исследование алгоритмов эквивалентных преобразований недетерминированных конечных автоматов и связанных с этими алгоритмами вспомогательных структур данных. Среди них мы имеем в виду в первую очередь структуры данных для представления бесконечных итерационных деревьев морфизмов [1, 2]. В них мы объединяем эквивалентные вершины — фактически получая некоторый детерминированный конечный автомат; мы также определяем эквивалентный ему недетерминированный автомат. Таким образом, каждому итерационному дереву морфизма мы ставим в соответствие два конечных автомата, описывающих эти морфизмы. В настоящей статье мы на приведённых примерах показываем некоторые важные для дальнейших работ свойства таких автоматов.

Структура статьи такова. Раздел 2 — предварительные сведения, в нём мы рассматриваем общие обозначения, связанные с недетерминированными конечными автоматами. В разделе 3 рассматриваются автоматы, соответствующие бесконечным итерационным деревьям морфизмов: в предыдущих статьях не требовалось формального строгого определения таких конструкций, они впервые приведены в настоящей статье. Конкретно, для некоторых двух заданных конечных языков *A* и *B* мы определяем соответствующие им бесконечные итерационные деревья, а также построенные для них автоматы, обозначенные PRI(*A*,*B*) и NSPRI$^{\#}$(*A*,*B*).

В разделе 4 приведена программа построения автомата PRI на языке C#. Основная цель того, что мы привели текст этой программы, состоит в том, чтобы привести альтернативное описание определений этого автомата — т. е., иными словами, описание алгоритма построения его функции переходов.

В разделе 5 подробно рассмотрен результат работы этой программы — т. е. фактически пример автомата PRI для двух несовпадающих языков. С этим автоматом проделаны стандартные действия, рассмотренные в наших предыдущих публикациях ([3, 4, 5] и мн. др.): построение, детерминизация и канонизация автомата для зеркального языка, а также построение бинарного отношения #.

В разделе 6 приведено описание построения автомата NSPRI$^{\#}$ для той же самой пары языков — эквивалентного автомату PRI. Мы на примере показываем эту эквивалентность также путём детерминизации автомата. В разделе 7 приведён пример автомата PRI, построенного для пары совпадающих языков.

Раздел 8 — заключение. В нём мы описываем возможные направления дальнейшей работы, связанные с рассмотренными в настоящей статье вопросами.

Предварительные сведения

Используемые в настоящей статье обозначения уже были ранее приведены в нескольких публикациях, в частности в [3, 4, 5, 6], — однако мы дадим их ещё раз: мы их приводим именно в том объёме, который нужен для настоящей статьи.

Пусть

$K = (Q, \Sigma, \delta, S, F)$ — (1)

недетерминированный конечный автомат Медведева (Рабина – Скотта), задающий регулярный язык

$L = \mathsf{L}(K)$.

Здесь Q — множество состояний, $S \subseteq Q$ и $F \subseteq Q$ — соответственно множества входных и выходных состояний.

Специально отметим, что функция переходов δ автомата (1) будет далее рассматриваться как

$\delta : Q \times \Sigma \to \mathsf{P}(Q)$

(где запись $\mathsf{P}(Q)$ означает множество подмножеств множества Q), а не в виде

$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathsf{P}(Q)$;

таким образом, мы будем рассматривать автоматы без ε-переходов.

Некоторую дугу $\delta(q,a)$ (точнее, $r \in \delta(q,a)$) мы иногда будем записывать в виде $q \xrightarrow{a,\delta} r$

или, если это не вызовет неоднозначности, в виде $q \xrightarrow{a} r$.

Зеркальный автомат для автомата, заданного с помощью (1), определяется как

$(Q, \Sigma, \delta^R, F, S)$,

где

$q' \xrightarrow{a,\delta^R} q''$ если и только если $q'' \xrightarrow{a,\delta} q'$.

Мы будем обозначать его записью K^R; заметим, что K^R определяет (регулярный) язык L^R.

Далее в этом разделе будем считать, что некоторый регулярный язык L задан; для него будем использовать следующие обозначения.

Его канонический автомат[1] будем обозначать записью $\tilde{L}$; специально подчеркнём, что это — автомат, а не язык.

Элементы «пятёрок», задающих автоматы $\tilde{L}$ и L^R, таковы:

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

Более того, мы не будем рассматривать язык $L = \varnothing$, поэтому оба канонических автомата должны иметь по (одному) стартовому состоянию.

Определим бинарное отношение # и функции разметки состояний ϕ^{in} и ϕ^{out}; некоторые подробности см. в [3, 5].

Отношение

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

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

$(\exists uv \in L)\,(u \in \mathsf{L}^{in}_{\tilde{L}}(A), v^R \in \mathsf{L}^{in}_{\widetilde{L^R}}(X))$.

Отметим, что это определение *неконструктивное*; его экви-

[1] В предыдущих публикациях мы отмечали, что канонические автоматы всегда рассматриваются без возможного «дохлого» состояния («dead state»). Отметим ещё, что вследствие этого мы, конечно, не требуем, чтобы канонический автомат был всюду определённым («total»).

валентный конструктивный аналог (т. е. определение-*алгоритм*) приведён в [7, 8].
(Прямая) функция разметки состояний $\phi_K^{in} : Q \to \mathsf{P}(Q_\pi)$

определяется следующим образом:
$\tilde{q} \in \phi_K^{in}(q)$ если и только если $\mathsf{L}_K^{in}(q) \cap \mathsf{L}_{\tilde{L}}^{in}(\tilde{q}) \neq \varnothing$.

А обратная функция разметки состояний
$\phi_K^{out} : Q \to \mathsf{P}(Q_\rho)$

определяется таким же образом, но для другой пары автоматов, K^R и L^R вместо K и $\tilde{L}$.
Некоторые другие применяемые нами обозначения, связанные с недетерминированными конечными автоматами, можно при необходимости найти в уже процитированных статьях [3, 4, 5, 6], а также, например, в [9, 10].
На основе определённого выше бинарного отношения # формируется множество *псевдогридов* (см. [7] и др.): а именно, каждый из них представляет собой пару (P, R) (где $P \subseteq Q_\pi$ и $R \subseteq Q_\rho$), такую что для каждой пары состояний $p \in P$ и $r \in R$ выполнено условие $p \# r$.
В произвольном НКА для заданного регулярного языка L каждое состояние образует некоторый псевдогрид. Более того, *необходимым* условием того, что автомат действительно определяет заданный язык L, является «покрытие» всеми псевдогридами[2] всех элементов бинарного отношения #, см. подробнее [9, 10, 19].
А если для некоторого псевдогрида (P, R) мы не можем расширить ни множество P, ни множество R – без того, чтобы не нарушилось определение псевдогрида, то такой псевдогрид мы будем называть *гридом*.
(Алгоритм построения множества (псевдо)гридов кажется простым — однако даже у этого алгоритма сложность более чем полиномиальна. Это следует хотя бы из (достижимой) верхней оценки числа гридов — которую можно получить, например, на основе [11]: эта оценка для бинарного отношения, в котором мощности обоих множеств $Q_{\neq}$ и d совпадают и равны n, равна $M(n)$ — n-му числу Дедекинда, [12, 13] и др.)

Бесконечные итерационные деревья и соответствующие автоматы

В предыдущих статьях не требовалось формального строгого определения бесконечные итерационного дерева — приведём такое определение в настоящей статье.
Определение 1. Для заданного конечного алфавита $\varnothing$ рассмотрим бесконечное дерево, в котором для каждой вершины (пусть v) и для каждой буквы алфавита $\varnothing$ (пусть d) имеется ровно одна дуга, ведущая из v к некоторой дочерней вершине и помеченная d. (Такую вершину обозначим $v\bar{d}$.)
При этом все вершины принадлежат одному из нескольких классов эквивалентности[3], и для каждой пары вершин, входящих в один класс эквивалентности (пусть v_1 и v_2), и для каждой буквы $d \in \Delta$ вершины $v_1\bar{d}$ и $v_2\bar{d}$ принадлежат одному и тому же классу эквивалентности.
Как несложно убедиться, рассматривавшиеся в [1, 2] конструкции этому определению удовлетворяют. На рисунках, приведённых в этих статьях, вершины дерева помечены не только различными вариантами обозначений классов эквивалентности, но и пометками путей, который ведёт из корня к соответствующей вершине (т. е. словами из $\varnothing^*$), — однако это не противоречит приведённому нами определению.
Определение 2. (Детерминированный) конечный автомат $\overline{\mathrm{PRI}}(A, B)$ для заданных конечных языков $A, B \subseteq \Sigma^*$ определяется следующим образом:
$\overline{\mathrm{PRI}}(A, B) = (\bar{Q}_B, \Delta_A, \delta_{A-B}, \{\{\varepsilon\}\}, \{\varnothing\})$,

где:

- $\Delta_A = \{0, 1, \ldots, n-1\}$, здесь n — число слов языка A;
- $\bar{Q}_B = \mathsf{P}(\mathrm{opref}(B))$;
- для некоторых $q_1, q_2 \in \bar{Q}_B$ и $a \in \Delta_A$ полагаем

$q_1 \xrightarrow{a, \delta_{A-B}} q_2$ тогда и только тогда, когда $F(q_1) = q_2$,

где F определяется согласно [2; стр. 6] [4].

(Детерминированный) конечный автомат $\mathrm{PRI}(A, B)$ определяется как $\overline{\mathrm{PRI}}(A, B)$ с удалёнными недостижимыми состояниями.
Важные примеры к этому и следующему определению будут рассмотрены в дальнейших разделах настоящей статьи.
Определение 3. (Недетерминированный) конечный автомат $\mathrm{NSPRI}^{\#}(A, B)$ для заданных конечных языков $A, B \subseteq \Sigma^*$ определяется следующим образом:
$NSPRI^{\#}(A, B) = (Q_B^{\#}, \Delta_A, \delta_{A-B}^{\#}, \{\varepsilon\}, \varnothing)$,

где:

- последний элемент пятёрки (множество выходных состояний) — это *состояние*, обозначенное нами Ø (а не пустое множество выходных состояний);
- $Q_B^{\#} = \mathrm{opref}(B) \cup \{\varnothing\}$;
- для некоторых $q_1, q_2 \in Q_B^{\#}$ и $a \in \Delta_A$ полагаем

$q_1 \xrightarrow{a, \delta_{A-B}^{\#}} q_2$

тогда и только тогда, когда $q_2 \in F(q_1)$,

где F, как и выше, определяется согласно [2; стр. 6].
Будем таким же образом (т. е. $\mathrm{NSPRI}^{\#}(A, B)$) обозначать тот же самый автомат с уже удалёнными недостижимыми состояниями[5].

[2] Здесь псевдогриды рассматриваются как множества своих элементов; аналогично — для всего бинарного отношения #.

[3] Мы рассматриваем только те модели, в которых таких классов эквивалентности — конечное число.

[4] Здесь определение непротиворечиво: запись $F(q_1)=q_2$ «неявно включает в себя» рассматриваемую букву a алфавита Δ_A. (Аналогично — в определении 3.)

[5] По-видимому — в отличие от пары $\overline{\mathrm{PRI}}(A, B)$ и $\mathrm{PRI}(A, B)$ — подобные одинаковые обозначения не должны вызвать разночтения: из контекста будет понятно, какой именно автомат имеется в виду.

Программа построения автомата PRI

В этом разделе мы привели программу построения автомата PRI. Программа выполнена на языке C#. Основная цель того, что мы привели текст этой программы, состоит в том, чтобы привести альтернативное описание определений этого автомата — т. е. фактически описание алгоритма построения его функции переходов.

```
static void GenPRI(string name, string nameout) {
var a = File.ReadAllLines(name);
var sout = "";
var d = new Dictionary<string, int>();
var comm = new List<string>();
var amax = 0;
comm.Add("{}");
d["null"] = 0;
foreach (var e in Regex.Split(a[2], @"\* ")) {
var k = e.Split(':');
if (k.Length == 1) break;
if (!k[1].Contains('|')) continue;
if (k[1] == "|") k[1] = "||";
k[1] = "{"
+ k[1].Replace("||", "|@|").Trim('|', ' ').
Replace("|", ",")
+ "}";
comm.Add(k[1]);
d[k[0]] = comm.Count - 1;
if (k[0].Length == 1) amax = Math.Max(amax, k[0][0] - 48);
}
foreach (var e in a[3].Split(' ')) {
var k = e.Split(':');
if (k.Length == 1) break;
if (d.ContainsKey(k[1])) d[k[0]] = d[k[1]];
}
for (int j = 0; j <= amax; j++) sout += " " + j;
sout += "\n";
foreach (var k in d.Keys.Take(comm.Count)) {
if (d[k] == 0) sout += "<== ";
if (d[k] == 1) sout += "==> ";
sout += $"{d[k]} ";
for (int i = 0; i <= amax; i++) {
var kk = k + i;
if (d.ContainsKey(kk)) sout += d[kk] + " ";
else sout += "0 ";
}
sout += $"% {comm[d[k]]}\n";
}
File.WriteAllText(nameout, sout);
}
```

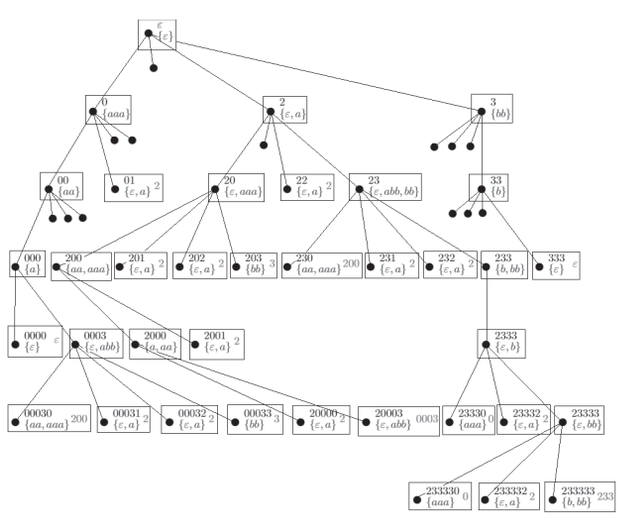
Приведём комментарии к программе — которую, как мы уже отмечали, саму можно рассматривать как «большой комментарий» ко всему алгоритму. Входным является файл с именем name, описывающий бесконечное итерационное дерево морфизмов для заданной пары конечных языков. Алгоритм построения такого дерева и реализующая его программа на языке C++ описаны в [1, 2]. Приведём пример входного файла для пары языков

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

взятый из [1] (результат работы приведённой в той статье программы построения бесконечного итерационного дерева морфизма):

```
aaa|aabba|abba|bb|
aaaa|abb|abba|bbb|
:|* 0:aaa|* 1:* 2:a||* 3:bb|* 00:aa|* 20:aaa||* 23:abb|bb||* 33:b|*
000:a|* 200:aaa|aa|* 233:bb|b|* 0003:abb||* 2000:aa|a|* 2333:b||*
23333:bb||* #
22:2 21:1 03:1 02:1 01:2 32:1 31:1 30:1 003:1 002:1 001:1 203:3
202:2 201:2 232:2 231:2 333: 332:1 331:1 330:1 0002:1 0001:1
0000: 2003:1 2002:1 2001:2 230:200 2332:1 2331:1 2330:1
00033:3 00032:2 00031:2 00030:200 20003:0003 20002:1
20001:1 20000:2 23332:2 23331:1 23330:0 233333:233 233332:2
233331:1 233330:0 #
```

Здесь строки 1–2 содержат исходные языки, строки 3–5 (в файле это одна строка, оканчивающаяся символом #) — различные «множества хвостов», строки 6–10 (тоже одна строка, оканчивающаяся символом #) — множества пар слов над алфавитом $\varnothing_A$ с равными в каждой паре значениями функции P, см. [1, 2]. Ниже приведено дерево морфизмов, соответствующее данному примеру[6].



При построении автомата PRI используются две основные структуры: словарь (Dictionary) d и список (List) comm. В списке comm сохраняются обозначения состояний автомата PRI, словарь d содержит в качестве ключей слова языка $\varnothing_A^*$ (указываемые между пробелом и двоеточием), а в качестве значений — состояния автомата PRI, указываемые между двоеточием и символом * (точнее, индексы этих состояний в списке comm). В переменной amax хранится максимальное значение из $\varnothing_A$ (при реализации функции GenPRI, как и программы из [1, 2], предполагается, что $\varnothing_A$ содержит не более 10 элементов —

[6] Исправлена замеченная в [15] опечатка — а именно, переход из вершины, помеченной 000.

цифр от 0 до 9). Формирование списка comm, словаря d и значения amax выполняется в первом цикле foreach; при этом условные обозначения для «набора хвостов» преобразуются в более наглядные строковые выражения, в которых в качестве ε применяется специальный символ @.

Второй цикл foreach дополняет словарь d новыми элементами, учитывая информацию о словах с равными значениями функции *P* (эти элементы в дальнейшем используются при определении функции переходов автомата PRI).

Затем строится строковое представление автомата PRI, начиная со вспомогательной строки, содержащей все символы из $\varnothing_A$ (цикл for, в котором перебираются все цифры от 0 до amax). Прочие строки формируются в третьем цикле foreach; их количество и порядок определяется содержимым списка comm; каждая строка соответствует некоторому состоянию автомата PRI. Для двух первых состояний дополнительно указываются метки, определяющие первое состояние (соответствующее пустому множеству) как выходное, а второе (соответствующее множеству $\{\varepsilon\}$) как входное. Во вложенном цикле for определяются переходы данного состояния для различных символов $\varnothing_A$; при этом используются дополнительные элементы словаря d, добавленные к нему при выполнении второго цикла foreach. Заметим, что оператор var kk = k + i формирует очередное слово над $\varnothing_A$, добавляя к строке k строковое представление цифры i.

Полученное строковое представление автомата PRI сохраняется в файле с именем nameout. Далее приведён пример полученного представления — после применения к нему дополнительных операций форматирования:

```
      0   1   2   3
<== 0   0   0   0   0    % {}
==> 1   2   0   3   4    % {@}
      2   5   3   0   0    % {aaa}
      3   6   0   3   7    % {a,@}
      4   0   0   0   8    % {bb}
      5   9   0   0   0    % {aa}
      6   10  3   3   4    % {aaa,@}
      7   10  3   3   11   % {abb,bb,@}
      8   0   0   0   1    % {b}
      9   1   0   0   12   % {a}
      10  13  3   0   0    % {aaa,aa}
      11  0   0   0   14   % {bb,b}
      12  10  3   3   4    % {abb,@}
      13  3   0   0   12   % {aa,a}
      14  2   0   3   15   % {b,@}
      15  2   0   3   11   % {bb,@}
```

Подробный пример автомата PRI для несовпадающих языков

Будем продолжать рассматривать пример из [1, 2]. Для рассматривавшейся в этих статьях и в предыдущем разделе пары языков мы получили автомат $\mathrm{PRI}(A,B)$, который подробнее запишем в виде следующей таблицы:

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

(По-видимому, обозначения понятны, однако небольшие комментарии приведём. В первом столбце — пометки для входных и выходных состояний; состояния 0 и 1 обозначаем специально, естественным образом, — а все остальные состояния нумеруем в порядке их появления. Во третьем столбце — сами состояния, согласно определению автомата $\mathrm{PRI}(A,B)$; краткие обозначения этих состояний, которые мы будем использовать далее — во втором столбце. По столбцам самой таблицы — буквы алфавита $\varnothing$.)

Зеркальный автомат для построенного нами автомата $\mathrm{PRI}(A,B)$ таков:

		0	1	2	3
→	0	0 4 8 11	0 1 3 4 5 8 9 11 13 14 15	0 2 4 5 8 9 10 11 13	0 2 5 10
←	1	9			8
	2	1 14 15			
	3	13	2 6 7 10 12	1 3 6 7 12 14 15	
	4				1 6 12
	5	2			
	6	3			
	7				3
	8				4
	9	5			
	10	6 7 12			
	11				7 15
	12				9 13
	13	10			
	14				11
	15				14

(Мы не употребляем фигурные скобки для множеств состояний, а сами элементы этих множеств приводим через пробелы. И, конечно, пустым ячейкам таблицы соответствуют пустые множества.)

Проведём для последнего автомата детерминизацию и канонизацию[7]:

[7] При этом промежуточные построения мы опускаем — в таблице приведены только итоговые состояния. Эти состояния мы обозначили так: X (так во всех наших предыдущих публикациях обозначалось входное состояние канонического автомата для зеркального языка), а далее Y01, ..., Y13. (В наших предыдущих публикациях последующие состояния автомата, канонического для зеркального языка, всегда обозначались несколькими последними заглавными

		0	1	2	3	
→	X	Y01	Y02	Y03	Y04	0
	Y01	Y01	Y02	Y03	Y05	0 4 8 11
←	Y02	Y03	Y06	Y06	Y07	0 1 3 4 5 8 9 11 13 14 15
	Y03	Y07	Y02	Y03	Y05	0 2 4 5 8 9 10 11 13
	Y04	Y08	Y02	Y03	Y04	0 2 5 10
←	Y05	Y09	Y02	Y03	Y10	0 1 2 4 5 6 7 10 12 15
←	Y06	Y06	Y06	Y06	Y06	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
←	Y07	Y09	Y02	Y03	Y06	0 1 2 4 5 6 7 8 10 11 12 14 15
←	Y08	Y11	Y02	Y03	Y06	0 1 2 4 6 7 8 11 12 14 15
←	Y09	Y02	Y06	Y06	Y06	0 1 2 3 4 6 7 8 9 11 12 14 15
←	Y10	Y06	Y06	Y06	Y03	0 1 2 3 5 6 8 9 10 12 13 14
←	Y11	Y12	Y06	Y06	Y07	0 1 3 4 8 9 11 14 15
	Y12	Y13	Y02	Y03	Y05	0 4 5 8 9 11 13
	Y13	Y08	Y02	Y03	Y05	0 2 4 5 8 10 11

В последнем столбце, за пределами самой таблицы, обозначены соответствующие состояния исходного автомата PRI(A, B).[8]
На основе последней таблицы несложно формируется бинарное отношение #:

	X	Y01	Y02	Y03	Y04	Y05	Y06	Y07	Y08	Y09	Y10	Y11	Y12	Y13
0	#	#	#	#	#	#	#	#	#	#	#	#	#	#
1			#			#	#	#	#	#	#	#		
2				#	#	#	#	#	#	#	#			#
3			#				#			#	#	#		
4		#	#	#		#	#	#	#	#		#	#	#
5			#	#	#	#	#	#			#		#	#
6						#	#	#	#	#	#			
7						#	#	#	#	#				
8		#	#	#			#	#	#	#	#	#	#	#
9			#	#			#			#	#	#	#	
10				#	#	#	#	#			#			#
11		#	#	#			#	#	#	#		#	#	#
12						#	#	#	#	#	#			
13			#	#			#			#	#		#	
14			#				#	#	#	#	#	#		
15			#			#	#	#	#			#		

Во многих наших предыдущих статьях ([4, 14, 15, 16] и др.) мы рассматривали задачу минимизации недетерминированных конечных автоматов. Эта задача важна и для задачи рассматриваемой в настоящей статье — что можно видеть на основе приведённого в следующем разделе примера. Но, конечно, мы не можем привести список всех получающихся в процессе минимизации гридов: даже 8-е число Дедекинда *M(8)* записывается 23 десятичными знаками ([11, 13, 24] и др.) — а у нас размерность бинарного отношения # равна 15×14.

Продолжение примера: автомат NSPRI#

Будем продолжать рассматривать ту же самую пару языков
A = {*aaa, aabba, abba, bb*}, *B* = {*aaaa, abb, abba, bbb*}.
Также на основе приведённых выше в разделе 3 определений мы получаем следующий автомат NSPRI#:

Важно отметить, что номера его состояний не связаны с номерами состояний приведённого выше автомата PRI. Кроме того, специально отметим отсутствие переходов из состояния 0 (в отличие от автомата PRI — это возможно, поскольку автомат недетерминированный): смысла эти переходы не несут, приведённое выше определение выполняется, алгоритм *применения* автомата NSPRI# эти переходы также не меняют — однако их присутствие изменило бы смысл дальнейших преобразований, связанных с автоматом NSPRI#.

			0	1	2	3
←	0	Ø				
→	1	ε	2	0	1 4	7
	2	aaa	3	1 4	0	0
	3	aa	4	0	0	0
	4	a	1	0	0	1 6
*	5	ab	0	0	0	8
	6	abb	2 3	1 4	1 4	7
	7	bb	2	0	0	8
	8	b	0	0	0	1

Итак, приводим тот же самый автомат без недостижимого состояния:

			0	1	2	3
←	0	Ø				
→	1	ε	2	0	1 4	7
	2	aaa	3	1 4	0	0
	3	aa	4	0	0	0
	4	a	1	0	0	1 6
	6	abb	2 3	1 4	1 4	7
	7	bb	2	0	0	8
	8	b	0	0	0	1

Далее приводим таблицу, отражающую процесс детерминизации последнего автомата; её можно рассматривать как иллюстрацию к эквивалентности двух рассмотренных автоматов.

			0	1	2	3	
→	1	ε	2	0	14	7	1
	2	aaa	3	14	0	0	2
←	0	Ø	0	0	0	0	0
	14	ε,a	12	0	14	167	3
	7	bb	2	0	0	8	4
	3	aa	4	0	0	0	5
	12	ε,aaa	23	14	14	7	6
	167	ε,abb,bb	23	14	14	78	7
	8	b	0	0	0	1	8
	4	a	1	0	0	16	9
	23	aa,aaa	34	14	0	0	10
	78	b,bb	2	0	0	18	11
	16	ε,abb	23	14	14	7	12
	34	a,aa	14	0	0	16	13
	18	ε,b	2	0	14	17	14
	17	ε,bb	2	0	14	78	15

К таблице необходимы следующие комментарии.

- Как обычно, множества состояний заносятся в строки таблицы «в процессе своего появления».
- Обозначения состояний (как в пометках строк, так и в клетках) приведены «в формате множеств» (так, 014 означает {0,1,4} — соответствующие состояния автомата NSPRI#); такой вариант обозначений более удобен (он использовался в некоторых наших предыдущих публикациях, в частности, [5]) — поскольку в исходном автомата NSPRI# менее 10 состояний.

буквами латинского алфавита — но таковых сейчас недостаточно.)

[8] Отметим, что на основе этого столбца можно легко восстановить «пропущенные» (т. е. не описанные в настоящей статье) шаги процесса канонизации автомата.

• Наоборот, в последнем столбце приведены двузначные числа: однако здесь это не множества, а обычные номера соответствующих состояний рассмотренного выше детерминированного автомата. После такого переобозначения состояний становится очевидной эквивалентность этих двух автоматов (фактически — просто их равенство).

Важно отметить, что автомат, полученный нами с помощью алгоритма (программы) построения автоматов NSPRI# (PRI), «практически совпадает» с минимальным автоматом для такого регулярного языка. Мы предполагаем вернуться к подобным вопросам в следующих публикациях.

Пример автомата PRI для пары совпадающих языков

В этом разделе мы рассмотрим пример автомата PRI(*A*,*A*) — т.е. будем производить построения для пары совпадающих языков. На первый взгляд может показаться, что подобные построения лишены смысла: ведь одна из задач, которую мы решаем при рассмотрении конечных автоматов, соответствующих бесконечным итерационным деревьям морфизмов, — это проверка равенства бесконечных итераций двух заданных конечных языков (как ω-языков), а в нашем случае такие бесконечные итерации заведомо совпадают. Однако рассмотрение пары совпадающих языков может быть полезно в качестве подзадачи для другой задачи, связанной со сформулированной здесь, а именно — с задачей построения оптимального инверсного морфизма для некоторого заданного конечного языка. Для этой задачи бывает необходимо сравнить бесконечные итерации двух практически совпадающих языков — а именно, $A \cup \{u\}$ и A для некоторых языка A и не принадлежащего ему слова u; а для такого сравнения нужны конструкции, получаемые в настоящей статье (и, в частности, в этом разделе).

Итак, применим ещё раз взятый из [1] алгоритм — уже для новых входных данных, а именно — для языков

$A = B = \{ab, abaab, abaaba, abaababaababa\}$.

Отметим, что согласно [17], каждый из этих языков представляет собой морфизм расширенного максимального префиксного кода. А согласно [18], достаточным условием равенства бесконечных итераций двух *разных* конечных языков является то, что они оба являются одним и тем же морфизмом двух разных расширенных максимальных префиксных кодов.

Результат работы приведённой в [1] программы построения бесконечного итерационного дерева морфизма таков:

```
ab|abaab|abaaba|abaababaababa|
ab|abaab|abaaba|abaababaababa|
:|* 0:ab||* 1:abaab||* 2:abaaba|a||* 3:abaababa||aba|a|* 10:abaabab|ab||* 11:abaababaab|abaab||* 12:abaababaaba|abaaba|a||*
30:abaababaab|ab|abaab||* 31:abaab||ab|* 32:abaaba|a||aba|* 110:abaababaababab|abaabab|ab||* 112:abaababaaba|abaaba|aba|a||* #
00:0 01:1 22:2 21:1 20:0 02:2 33:3 23:3 13:3 03:3 103:3 102:2 101:1 100:0 113:3 123:3 122:2 121:1 120:0 303:3 301:30 111:30 313:3
312:12 311:11 310:10 323:3 322:32 321:31 320:31 1103:3 1102:2 1101:1 1100:0 300:110 1123:3 1122:32 1121:31 1120:31 302:112 #
```

Далее — получающийся детерминированный конечный автомат, соответствующий итерационному дереву морфизма (комментарии те же самые, что и в разделе 4):

```
      0   1   2   3
<== 0   0   0   0   0    % {}
==> 1   2   3   4   5    % {@}
  2   2   3   4   5    % {ab,@}
  3   6   7   8   5    % {abaab,@}
  4   2   3   4   5    % {abaaba,a,@}
  5   9   10  11  5    % {abaababa,@,aba,a}
  6   2   3   4   5    % {abaabab,ab,@}
  7   12  9   13  5    % {abaababaab,abaab,@}
  8   2   3   4   5    % {abaababaaba,abaaba,a,@}
  9   12  9   13  5    % {abaababaab,ab,abaab,@}
  10  6   7   8   5    % {abaab,@,ab}
  11  10  10  11  5    % {abaaba,a,@,aba}
  12  2   3   4   5    % {abaababaabab,abaabab,ab,@}
  13  10  10  11  5    % {abaababaaba,abaaba,aba,a,@}
```

Понятно, что автомат «даёт положительный ответ» — т. е. выполняется эквивалентность заданных языков; поэтому при детерминизации зеркального автомата получается тривиальный автомат:

```
     0  1  2  3
==> 0  0  0  0  0
```

Другие примеры бесконечных итерационных деревьев и соответствующих им конечных автоматов, построенные для эквивалентных (но, однако, не равных) языков, мы рассмотрим в следующих публикациях.

Заключение

В этом разделе мы описываем возможные направления дальнейшей работы, связанные с рассмотренными в настоящей статье вопросами.

По-видимому, самая важная задача связана с возможной неполиномиальностью всего алгоритма построения автомата NSPRI*(A,B)* (а, следовательно, и других рассмотренных нами автоматов). Однако отметим, что из возможного положительного ответа на этот вопрос ещё не будет следовать NPполнота основной рассматриваемой нами задачи — а именно, вопрос об эквивалентности в бесконечности некоторых заданных языков A и B: возможно, для этого имеются какие-нибудь принципиально отличающиеся алгоритмы.

А самая ближайшая задача (конечно, значительно более простая) — это описание полиномиального алгоритма построение автомата NSPRI.

По-видимому, основной результат настоящей статьи — подтверждение эквивалентности автоматов PRI и NSPRI# на рассматриваемых примерах. Строгое доказательство этого факта мы приведём в одной из дальнейших публикаций. Такая эквивалентность даёт возможность *на практике* быстро проверять

равенство бесконечных итераций конечных языков — несмотря на то, что имеющийся у нас в настоящее время алгоритм проверки такого равенства с формальной точки зрения является экспоненциальным.

Список использованных источников

[1] Мельников, Б. Ф. Бесконечные деревья в алгоритме проверки условия эквивалентности итераций конечных языков. Часть I / Б. Ф. Мельников, А. А. Мельникова // International Journal of Open Information Technologies. — 2021. — Т. 9, № 4. — С. 1-11. — URL: https://elibrary.ru/item.asp?id=45595955 (дата обращения: 19.02.2021). — Рез. англ.

[2] Мельников, Б. Ф. Бесконечные деревья в алгоритме проверки условия эквивалентности итераций конечных языков. Часть II / Б. Ф. Мельников, А. А. Мельникова // International Journal of Open Information Technologies. — 2021. — Т. 9, № 5. — С. 1-11. — URL: https://elibrary.ru/item.asp?id=45671876 (дата обращения: 19.02.2021). — Рез. англ.

[3] Melnikov, B. Some properties of the basis finite automaton / B. Melnikov, A. Melnikova. — DOI 10.1007/BF03012345 // Korean Journal of Computational & Applied Mathematics. — 2002. — Vol. 9, issue 1. — Pp. 135-150.

[4] Melnikov, B. The state minimization problem for nondeterministic finite automata: the parallel implementation of the truncated branch and bound method / B. Melnikov, A. Tsyganov // Proceedings of the International Symposium on Parallel Architectures, Algorithms and Programming (PAAP-2012). — Taipei, Taiwan, 2012. — Pp. 194-201.

[5] Melnikov, B. Some more algorithms for Conway's universal automaton / B. Melnikov, V. Dolgov. — DOI 10.2478/ausi-2014-0015 // Acta Universitatis Sapientiae, Informatica. — 2014. — Vol. 6, no. 1. — Pp. 5-20.

[6] Melnikov, B. The complete finite automaton / B. Melnikov // International Journal of Open Information Technologies. — 2017. — Vol. 5, no. 10. — Pp. 9-17. — URL: https://elibrary.ru/item.asp?id=30101608 (дата обращения: 19.02.2021).

[7] Долгов, В. Н. Построение универсального конечного автомата. I. От теории к практическим алгоритмам / В. Н. Долгов, Б. Ф. Мельников // Вестник Воронежского государственного университета. Серия: Физика. Математика. — 2013. — № 2. — С. 173-181. — URL: https://elibrary.ru/item.asp?id=20267924 (дата обращения: 19.02.2021). — Рез. англ.

[8] Долгов, В. Н. Построение универсального конечного автомата. II. Примеры работы алгоритмов / В. Н. Долгов, Б. Ф. Мельников // Вестник Воронежского государственного университета. Серия: Физика. Математика. — 2014. — № 1. — С. 78-85. — URL: https://elibrary.ru/item.asp?id=21445963 (дата обращения: 19.02.2021). — Рез. англ.

[9] Melnikov, B. F. Some more on the finite automata / B. F. Melnikov, A. A. Vakhitova. — DOI 10.1007/BF03008877 // Korean Journal of Computational & Applied Mathematics. — 1998. — Vol. 5, issue 3. — Pp. 495-505.

[10] Melnikov, B. F. Possible edges of a finite automaton defining a given regular language / B. F. Melnikov, N. V. Sciarini-Guryanova. — DOI 10.1007/BF03021555 // Korean Journal of Computational & Applied Mathematics. — 2002. — Vol. 9, issue 2. — Pp. 475-485.

[11] Lombardy, S. The universal automaton / S. Lombardy, J. Sakarovitch // Logic and Automata: History and Perspectives. Texts in Logic and Games; J. Flum, E. Grädel, T. Wilke (Eds.). — Vol. 2. — Amsterdam University Press, 2007. — Pp. 457-504.

[12] Dedekind, R. Über Zerlegungen von Zahlen durch ihre größten gemeinsamen Teiler / R. Dedekind // Gesammelte Werke. — Vol. 2. — Braunschweig, 1897. — Pp. 103-148.

[13] Коршунов, А. Д. О числе монотонных булевых функций / А. Д. Коршунов // Проблемы кибернетики. — 1981. — Т. 38. — С. 5-108.

[14] Абрамян, М. Э. О применении некоторых эвристик при исследовании задачи вершинной минимизации недетерминированных конечных автоматов методом ветвей и границ. Часть 1 / М. Э. Абрамян, Б. Ф. Мельников // International Journal of Open Information Technologies. — 2020. — Т. 8, № 9. — С. 1-7. — URL: https://elibrary.ru/item.asp?id=43925422 (дата обращения: 19.02.2021). — Рез. англ.

[15] Абрамян, М. Э. О применении некоторых эвристик при исследовании задачи вершинной минимизации недетерминированных конечных автоматов методом ветвей и границ. Часть 2 / М. Э. Абрамян, Б. Ф. Мельников // International Journal of Open Information Technologies. — 2020. — Т. 8, № 10. — С. 1-9. — URL: https://elibrary.ru/item.asp?id=44106795 (дата обращения: 19.02.2021). — Рез. англ.

[16] Абрамян, М. Э. Исследование задачи вершинной минимизации недетерминированных конечных автоматов с помощью метода ветвей и границ / М. Э. Абрамян, Б. Ф. Мельников // Cloud of Science. — 2020. — Т. 7, № 2. — С. 297-319. — URL: https://elibrary.ru/item.asp?id=42708816 (дата обращения: 19.02.2021). — Рез. англ.

[17] Алексеева, А. Г. Итерации конечных и бесконечных языков и недетерминированные конечные автоматы / А. Г. Алексеева, Б. Ф. Мельников // Вектор науки Тольяттинского государственного университета. — 2011. — № 3(17) — С. 30-33. — URL: https://elibrary.ru/item.asp?id=18066263 (дата обращения: 19.02.2021). — Рез. англ.

[18] Melnikov, B. F. The equality condition for infinite catenations of two sets of finite words / B. F. Melnikov. — DOI 10.1142/S0129054193000171 // International Journal of Foundations of Computer Science. — 1993. — Vol. 4, issue 3. — Pp. 267-274.

[19] Melnikov, B. F. A new algorithm of the state-minimization for the nondeterministic finite automata / B. F. Melnikov. — DOI 10.1007/BF03014374 // Korean Journal of Computational & Applied Mathematics. — 1999. — Vol. 6, issue 2. — Pp. 277-290.

[20] Perrin, D. Finite Automata / D. Perrin. — DOI 10.1016/B978-0-444-88074-1.50006-8 // Formal Models and Sematics. Handbook of Theoretical Computer Science; J. van Leeuwen (Ed.). — Vol. B. — Elsevier Science. 1990. — Pp. 3-57.

[21] Gruber, H. Provably Shorter Regular Expressions From Finite Automata / H. Gruber, M. Holzer. — DOI 10.1142/S0129054113500330 // International Journal of Foundations of Computer Science. — 2013. — Vol. 24, issue 8. — Pp. 1255-1279.

[22] Fernau, H. Algorithms for learning regular expressions from positive data / H. Fernau. — DOI 10.1016/j.ic.2008.12.008 // Information and Computation. — 2009. — Vol. 207. — Pp. 521-541.

[23] Cormen, T. H. Introduction to Algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein C. — 3rd edition. — MIT Press, 2009.

[24] Meng, J. Regarding covering as a collection of binary relations / J. Meng, T. Lin. — DOI 10.1109/GRC.2014.6982833 // 2014 IEEE International Conference on Granular Computing (GrC). — Noboribetsu, Japan, 2014. — Pp. 191-195.

[25] Carayol, A. Automata on Infinite Trees with Equality and Disequality Constraints Between Siblings / A. Carayol, C. Loding, O. Serre // 2016 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). — New York, NY, 2016. — Pp. 1-10.

Поступила 19.02.2021; одобрена после рецензирования 27.03.2021; принята к публикации 05.04.2021.

Об авторах:

Абрамян Михаил Эдуардович, доцент Института математики, механики и компьютерных наук им. И. И. Воровича, ФГАОУ ВО «Южный федеральный университет» (344006, Российская Федерация, г. Ростов-на-Дону, ул. Б. Садовая, д. 105/42), кандидат физико-математических наук, доцент, **ORCID:** http://orcid.org/0000-0002-2802-6144, mabr@sfedu.ru

Мельников Борис Феликсович, профессор факультета вычислительной математики и кибернетики, Совместный университет МГУ – ППИ (517182, Китайская Народная Республика, провинция Гуандун, г. Шэньчжэнь, р-н Лунган, Даюньсиньчэн, ул. Гоцзидасюеюань, д. 1), доктор физико-математических наук, профессор, **ORCID:** http://orcid.org/0000-0002-6765-6800, bormel@mail.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.

References

[1] Melnikov B., Melnikova A. Infinite trees in the algorithm for checking the equivalence condition of iterations of finite languages. Part I. *International Journal of Open Information Technologies*. 2021; 9(4):1-11. Available at: https://elibrary.ru/item.asp?id=45595955 (accessed 19.02.2021). (In Russ., abstract in Eng.)

[2] Melnikov B., Melnikova A. Infinite trees in the algorithm for checking the equivalence condition of iterations of finite languages. Part II. *International Journal of Open Information Technologies*. 2021; 9(5):1-11. Available at: https://elibrary.ru/item.asp?id=45671876 (accessed 19.02.2021). (In Russ., abstract in Eng.)

[3] Melnikov B., Melnikova A. Some properties of the basis finite automaton. *Korean Journal of Computational & Applied Mathematics*. 2002; 9(1):135-150. (In Eng.) DOI: https://doi.org/10.1007/BF03012345

[4] Melnikov B., Tsyganov A. The state minimization problem for nondeterministic finite automata: the parallel implementation of the truncated branch and bound method. *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Programming (PAAP-2012).* Taipei, Taiwan; 2012. p. 194-201. (In Eng.)

[5] Melnikov B., Dolgov V. Some more algorithms for Conway's universal automaton. *Acta Universitatis Sapientiae, Informatica*. 2014; 6(1):5-20. (In Eng.) DOI: https://doi.org/10.2478/ausi-2014-0015

[6] Melnikov B. The complete finite automaton. *International Journal of Open Information Technologies*. 2017; 5(10):9-17. Available at: https://elibrary.ru/item.asp?id=30101608 (accessed 19.02.2021). (In Eng.)

[7] Dolgov V., Melnikov B. Construction of universal finite automaton. I. From theory to the practical algorithm. *Proceedings of Voronezh State University. Series: Physics. Mathematics*. 2013; (2):173-181. Available at: https://elibrary.ru/item.asp?id=20267924 (accessed 19.02.2021). (In Russ., abstract in Eng.)

[8] Dolgov V., Melnikov B. Construction of universal finite automaton. II. Examples of algorithms functioning. *Proceedings of Voronezh State University. Series: Physics. Mathematics*. 2014; (1):78-85. Available at: https://elibrary.ru/item.asp?id=21445963 (accessed 19.02.2021). (In Russ., abstract in Eng.)

[9] Melnikov B.F., Vakhitova A.A. Some more on the finite automata. *Korean Journal of Computational & Applied Mathematics.* 1998; 5(3):495-505. (In Eng.) DOI: https://doi.org/10.1007/BF03008877

[10] Melnikov B.F., Sciarini-Guryanova N.V. Possible edges of a finite automaton defining a given regular language. *Korean Journal of Computational & Applied Mathematics.* 2002; 9(2):475-485. (In Eng.) DOI: https://doi.org/10.1007/BF03021555

[11] Lombardy S., Sakarovitch J. The universal automaton. In: J. Flum, E. Grädel, T. Wilke (Eds.) Logic and Automata: History and Perspectives. Texts in Logic and Games, vol. 2. Amsterdam University Press; 2007. p. 457-504. (In Eng.)

[12] Dedekind R. Über Zerlegungen von Zahlen durch ihre größten gemeinsamen Teiler. Gesammelte Werke, vol. 2. Braunschweig; 1897. p. 103-148. (In German)

[13] Korshunov A. On the number of monotone Boolean functions. *Problemy kibernetiki* = Problems of Cybernetics. 1981; 38:5-108. (In Russ.)

[14] Abramyan M.E., Melnikov B.F. On the application of some heuristics in the study of the problem of state minimization of nondeterministic finite automata by the branch and bound method. Part 1. *International Journal of Open Information Technologies*. 2020; 8(9):1-7. Available at: https://elibrary.ru/item.asp?id=43925422 (accessed 19.02.2021). (In Russ., abstract in Eng.)

[15] Abramyan M.E., Melnikov B.F. On the application of some heuristics in the study of the problem of state minimization of nondeterministic finite automata by the branch and

bound method. Part 2. *International Journal of Open Information Technologies*. 2020; 8(10):1-9. Available at: https://elibrary.ru/item.asp?id=44106795 (accessed 19.02.2021). (In Russ., abstract in Eng.)

[16] Abramyan M.E., Melnikov B.F. Investigation of the problem of state minimization of nondeterministic finite automata using the branch and bound method. *Cloud of Science*. 2020; 7(2):297-319. Available at: https://elibrary.ru/item.asp?id=42708816 (accessed 19.02.2021). (In Russ., abstract in Eng.)

[17] Alekseyeva A.G., Melnikov B.F. Iterations of finite and infinite languages and nondeterministic finite automata. *Science Vector of Togliatti State University*. 2011; (3):30-33. Available at: https://elibrary.ru/item.asp?id=18066263 (accessed 19.02.2021). (In Russ., abstract in Eng.)

[18] Melnikov B.F. The equality condition for infinite catenations of two sets of finite words. *International Journal of Foundations of Computer Science*. 1993; 4(3):267-274. (In Eng.) DOI: https://doi.org/10.1142/S0129054193000171

[19] Melnikov B.F. A new algorithm of the state-minimization for the nondeterministic finite automata. *Korean Journal of Computational & Applied Mathematics*. 1999; 6(2):277-290. (In Eng.) DOI: https://doi.org/10.1007/BF03014374

[20] Perrin D. Finite Automata. In: J. van Leeuwen (Ed.) Formal Models and Sematics. Handbook of Theoretical Computer Science, vol. B. Elsevier Science; 1990. p. 3-57. (In Eng.) DOI: https://doi.org/10.1016/B978-0-444-88074-1.50006-8

[21] Gruber H., Holzer M. Provably Shorter Regular Expressions From Finite Automata. *International Journal of Foundations of Computer Science*. 2013; 24(8):1255-1279. (In Eng.) DOI: https://doi.org/10.1142/S0129054113500330

[22] Fernau H. Algorithms for learning regular expressions from positive data. *Information and Computation*. 2009; 207:521-541. (In Eng.) DOI: https://doi.org/10.1016/j.ic.2008.12.008

[23] Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms. 3rd edition. MIT Press; 2009. (In Eng.)

[24] Meng J., Lin T. Regarding covering as a collection of binary relations. In: *2014 IEEE International Conference on Granular Computing (GrC)*. Noboribetsu, Japan; 2014. p. 191-195. (In Eng.) DOI: https://doi.org/10.1109/GRC.2014.6982833

[25] Carayol A., Loding C., Serre O. Automata on Infinite Trees with Equality and Disequality Constraints Between Siblings. In: *2016 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS).* New York, NY; 2016. p. 1-10. (In Eng.)

Submitted 19.02.2021; approved after reviewing 27.03.2021; accepted for publication 05.04.2021.

About the authors:

Mikhail E. Abramyan, Associate Professor of the Institute of Mathematics, Mechanics, and Computer Science named after of I.I. Vorovich, Southern Federal University (105/42 Bolshaya Sadovaya St., Rostov-on-Don 344006, Russian Federation), Ph.D. (Phys.-Math.), Associate Professor, **ORCID:** http://orcid.org/0000-0002-2802-6144, mabr@sfedu.ru

Boris F. Melnikov, Professor of the Faculty of Computational Mathematics and Cybernetics, Shenzhen MSU – BIT University (1 Guoji-daxueyuan St., Dayunxincheng, Longgang District, Shenzhen 517182, Guangdong Province, People's Republic of China), Dr.Sci. (Phys.-Math.), Professor, **ORCID:** http://orcid.org/0000-0002-6765-6800, bormel@mail.ru

All authors have read and approved the final manuscript.