УДК 519.713

НЕКОТОРЫЕ ВСПОМОГАТЕЛЬНЫЕ АЛГОРИТМЫ ДЛЯ ПОСТРОЕНИЯ ВАТЕРЛОО-ПОДОБНЫХ АВТОМАТОВ

А. В. Криволапова*, Е. А. Мельникова, Н. В. Софонова***

**Тольяттинский государственный университет*
***Тольяттинский филиал Самарского государственного университета*

Поступила в редакцию 26.09.2016 г.

Аннотация. Пример покрывающего конечного автомата, который не эквивалентен исходному минимизируемому автомату, называется Ватерлоо. В настоящее время получен алгоритм построения любого такого автомата (когда покрывающий не эквивалентен исходному) на основе последовательных объединений букв соответствующего полного автомата. В данной статье рассматриваются некоторые вспомогательные алгоритмы, необходимые для описания алгоритма, реализующего подобные последовательные объединения букв, а именно алгоритм проверки: является ли построенный по полному автомату автомат Ватерлоо-подобным.
Ключевые слова: недетерминированные конечные автоматы, алгоритмы минимизации, универсальный автомат, автомат Ватерлоо.
Annotation. Example names Waterloo, if it is the example of this covering a finite automaton, which is not equivalent to the original automaton to be minimized. Subsequently it has been received of any such algorithm for constructing the automaton (when the covering is not equivalent to the original one), based on sequence association of letters of the respective complete automaton. This article discusses some supporting algorithms needed to describe the algorithm that implements a similar sequence association of letters, this check algorithm: isn't automation, which construct of complete automation, waterloo like automation.
Keywords: nondeterministic finite automata, algorithms for minimization, universal automation, Waterloo automaton.

ВВЕДЕНИЕ

Как известно, для регулярного языка имеются разные инварианты, например, канонический, базисный [1] и универсальный [2–4] конечные автоматы и многие другие автоматы, структура которых зависит от постановки задачи и реализации алгоритмов ее решения. Выбор нужного инварианта автомата, на котором строится решение задачи [5], может вестись в двух направлениях:

1) минимизация числа внутренних состояний автоматов с целью сокращения формы записи и построения на минимальном количестве элементов памяти и

2) минимизация автоматов с целью получения оптимальных (по отношению определенных критериев и в первую очередь в смысле экономичности) решений в процессе их технической реализации [6].

Даже минимизация числа внутренних состояний автомата для случаев с большим количеством состояний является нерешенной проблемой и сегодня.

А в некоторых случаях эта задача (выбор минимального инварианта автомата) не только сложна, но и опасна возможностью допустить ошибку и получить совсем другой автомат, о котором и идет речь в данной статье.

Впервые пример такого неэквивалентного автомата, нашли Камеда и Вайнер в 1970 г. [7], назвав его Ватерлоо. Подобная конструкция в данной статье будет именоваться walibad (от выражения "Waterloo-like badness" – Ватерлоо-подобная проблема) – когда существует покрывающий автомат, не являющийся эквивалентным исходному автомату [8]. Об оценках таких ошибок и пути их разрешения один из

авторов писал в [9], где ко всему прочему говорится о результатах случайной генерации НКА, но при этом Ватерлоо-подобный автомат там не был сгенерирован ни разу.

Алгоритмы минимизации недетерминированных конечных автоматов [10] использующие свойства покрывающих подмножеств при работе с walibad'ом могут выдать результирующий автомат не эквивалентный изначальному минимизируемому автомату. Поэтому хотелось бы заранее (до применения алгоритмов минимизации) знать ответ на вопрос – является ли автомат, который мы хотим минимизировать, walibad'ом. При этом переборный алгоритм для поиска минимального Ватерлоо-подобного автомата вряд ли возможен (поскольку общее число таблиц, размерность которых меньше чем у самого автомата Ватерлоо – 8×10, составляет около 20000), поэтому очевидно, что для разработки алгоритма поиска минимального Ватерлоо-подобного конечного автомата должны применяться эвристические алгоритмы.

В 1993г. в [11] было доказано, что задача вершинной минимизации недетерминированных конечных автоматов является NP-трудной – и по этому для решения этой проблемы важной составляющей является описание эвристических алгоритмов, т. е. алгоритмов, дающих при приемлемом времени работы решение, близкое к оптимальному.

Ещё одно основное положение нашего подхода к задаче вершинной минимизации заключается в том, что вместо сложных структур мы работаем с таблицей соответствия состояний двух канонических автоматов – прямого и обратного или как его ещё называют зеркального (т.е. работаем с таблицей бинарного отношения #). Эта таблица эквивалентна упомянутым алгебраическим структурам – однако значительно удобнее для программной реализации.

Но для начала необходим ответ на другой вопрос: какие автоматы могут оказаться walibad'ом. На этот вопрос отвечает так называемое понятие прото-walibad [12] – покрывающее множество блоков, не совпадающее со всем множеством блоков в таблице бинарного отношения # (а также соответствующий ему покрывающий автомат).

Ранее нами был описан алгоритм получения Ватерлоо-подобного конечного автомата на основе ранее нами описанного полного конечного автомата (последний строится только на основе таблицы бинарного отношения #) [9].

Далее в случае задачи вершинной минимизации мы пытаемся удалить некоторые вершины, а в случае дуговой минимизации – некоторые дуги.

При этом проверка эквивалентности автомата, полученного после такого удаления, с исходным НКА, производится далеко не всегда (сложность алгоритма такой проверки экспоненциальна). Существуют эвристики, с помощью которых вычисляется оценка вероятности возможности удаления дуги (состояния) [13] – и, следовательно, вероятности того, что получаемый после удаления автомат эквивалентен исходному; на основе этой вероятности можно получить «желательность строгого осуществления» проверки эквивалентности, которая всегда и производится одним из параллельных процессов.

В случае если мы всё же получаем неэквивалентность полученного и исходного автоматов (при вызове вспомогательного алгоритма проверки эквивалентности – причём данный вспомогательный алгоритм всё же не требует полной процедуры канонизации автомата), то из списка подзадач, предназначаемых для решения незавершённым методом ветвей и границ, удаляются все те, для которых такая неэквивалентность возможна [14]. И, конечно, проверка эквивалентности осуществляется для любого текущего допустимого решения (т.е. решения, являющегося оптимальным на данный момент времени).

Всё это даёт надежду на то, что в ближайшем будущем нам удастся получить ответ на вопрос, действительно ли автомат Ватерлоо является минимальным Ватерлоо-подобным. При этом минимальность может быть определена разными способами, но, по-видимому, наиболее естественным является размер таблицы бинарного отношения # (для автомата Ватерлоо он составляет 80 элементов – возможных букв).

ВСПОМОГАТЕЛЬНЫЕ ПОСТРОЕНИЯ

Будем считать регулярный язык L заданным, и рассматривать необходимые построения для него из [1, 3, 4].

Так недетерминированный конечный автомат $K = \{Q, \Sigma, \delta, S, F\}$ будем рассматривать без ε-переходов и функция переходов будет: $\delta: Q \times \Sigma \to P(Q)$.

Канонические автоматы будут следующими:

$\widetilde{L} = (Q_\pi, \Sigma, \delta_\pi, \{s_\pi\}, F_\pi)$ и

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

Бинарное отношение $\# \subseteq Q_\pi \times Q_\rho$ определяется для пар состояний автоматов $\widetilde{L}$ и $\widetilde{L^R}$ следующим образом: $A \# X \Leftrightarrow (\exists uv \in L)$ $(u \in \mathcal{L}^{in}\widetilde{L}(A),\ v^R \in \mathcal{L}^{in}\widetilde{L^R}(X))$.

В процессе получения отношения # из любого недетерминированного конечного автомата нам приходится детерминировать зеркальный автомат и переименовывать в один символ множества состояний для удобства. Подробнее процесс получения отношения # см. в [15].

Базисный автомат $BA(L)$ для L (см. [1, 6]): $BA(L) = \{\widehat{Q}, \Sigma, \widehat{\delta}, \widehat{S}, \widehat{F}\}$.

Автомат $COM(L)$ (см. [3, 4]): $COM(L) = \{Q_B, \Sigma, \delta_Q, S_Q, F_Q\}$, где Q_B – множество блоков, $S_Q = \{B \in Q_B \mid s_\pi \in \alpha(B)\}$, $F_Q = \{B \in Q_B \mid s_\rho \in \beta(B)\}$, и два условия существования перехода $B_1 \xrightarrow[\delta_Q]{a} B_2$:

1) $(\forall p \in \alpha(B_1))\ \ (\delta_\pi(p, a) \in \alpha(B_2))$,

2) $(\forall r \in \beta(B_2))\ \ (\delta_\rho(r, a) \in \beta(B_1))$.

Покрывающий автомат $COM(L)$ – это автомат, состоящий из такой совокупности блоков, которой достаточно что бы покрыть все элементы в таблице отношения #. Усеченный COM автомат – создание подмножеств из текущего множества псевдоблоков. Из этих автоматов выбираются те, которые являются покрывающими. Построение же walibad'a в рассматриваемых алгоритмах ведется на основе таблицы бинарного отношения #, по которой строится полный автомат (см. [3,4]) $L_\# = \{Q_\pi, \Sigma_\#, \delta_\#, \{s_\pi\}, F_\pi\}$, где

- $F_\pi = \{f_\pi \in Q_\pi \mid f_\pi \# s_\rho\}$ – множество финальных состояний,
- $\Sigma_\# = \{a_{X^A} \mid A \in Q_\pi,\ X \in Q_\rho\}$ – алфавит,
- $\delta_\#(A; a_{X^B}) = \begin{cases} \{B\}, & A \# X \\ \varnothing, & \text{иначе} \end{cases}$ – функция переходов.

Пусть Σ' – некоторый алфавит. Рассмотрим функцию $c: \Sigma' \times \Sigma_\# \to P(Q_\pi)$.

Назовём c помечающей функцией из [9] автомата $L_\#$, если для любых $a \in \Sigma'$, $a_{X^B} \in \Sigma'$ и $A \in Q_\pi$ выполняется следующее условие: $A \in c(a, a_{X^B}) \Rightarrow A \# X$ (т. е. $\delta_\#(A, a_{X^B}) \neq \varnothing$).

Введём помечающую функцию на основе базисного автомата $c_\#$ [9]. Для функции $c_\#: \Sigma \times \Sigma_\# \to P(Q_\pi)$ полагаем $c_\#(a, a_{X^B}) \ni A$, если в базисном автомате существует переход $\begin{matrix} A \\ X \end{matrix} \xrightarrow{a} \begin{matrix} B \\ Y \end{matrix}$; иных значений эта функция не содержит.

В силу определения базисного автомата $c_\#(a, a_{X^B}) \ni A$ влечет

$$\delta_\pi(A, a) = B. \qquad (1)$$

Утверждение. *$c_\#$ – допустимо помеченные переходы.*

Доказательство. Пусть $A \in c(a, a_{X^B})$ и $A \in c(a, a_{Y^C})$, тогда $\delta_\pi(A, a) = B$ и $\delta_\pi(A, a) = C$ (см. (1)), то есть $B = C$.

Теорема 1. *Объединение на основе помеченных переходов $c_\#$ даёт канонический автомат.*

Доказательство. Следует из (1).

Теорема 2. Для каждой дуги автомата $COM(L)$ $B_1 \xrightarrow[\delta_Q]{a} B_2$ существует соответствующая дуга полного автомата $L_\#$.

Теорема 3. *Для каждой дуги базисного автомата BA $\begin{matrix} A_1 \\ X_1 \end{matrix} \xrightarrow[\hat{\delta}]{a} \begin{matrix} A_2 \\ X_2 \end{matrix}$ существует соответствующая дуга полного автомата $L_\#$ $A_1 \xrightarrow[\delta_\#]{a_{X_1 A_2}} A_2$.*

Доказательства данных теорем представлены в [12].

Получение исходного автомата по таблице бинарного отношения # происходит на основе базисного автомата $BA(L)$.

Утверждение. Таким образом, если для некоторой заданной таблицы бинарного отношения # существует автомат, для которого покрывающий автомат не эквивалентен ис-

ходному автомату, то этот исходный автомат может быть получен переборным способом на основе полного автомата $L_{\#}$ и некоторого допустимого объединения его столбцов (букв).

Докажем данное утверждение для самого автомата Ватерлоо.

АЛГОРИТМ ПРОВЕРКИ НА ВАТЕРЛОО-ПОДОБНОСТЬ

Приведем алгоритм проверки, является ли автомат Ватерлоо-подобным.

Вход: таблица бинарного отношения #, в нашем случае автомата Ватерлоо:

	X	**Y**	**Z**	**U**	**V**	**W**	**P**	**Q**	**R**	**S**
A		#						#		
B		#			#					
C				#	#					
D						#	#			
E	#					#	#			
F	#		#				#		#	
G			#						#	#
H			#							#

Если при обходе таблицы в ячейке встречаем «#», то на ее место в новом автомате вставляем столбцы с состояниями, содержащимися в левом столбце.

Например: попадая в ячейку на пересечении Y и A, в новом автомате получаем:

	Y-A	**Y-B**	**Y-C**	**Y-D**	**Y-E**	**Y-F**	**Y-G**	**Y-H**
A	A	B	C	D	E	F	G	H

1. Построение полного автомата $L_{\#}$ по таблице бинарного отношения #.

Таким образом, мы получим полный автомат из 10×80 состояний, в табл. 1 представлен фрагмент полученной таблицы.

2. Построение зеркального автомата $L_{\#}^{R}$.

Следующим шагом нашего алгоритма является построение зеркального автомата, который строится на основе полного. Обход начинается с прохода по строкам одного столбца полного автомата, запоминая состояние, в которое мы можем попасть и, запоминая состояние, из которого мы туда попадаем.

В нашем случае: мы идем по столбцу X-A и когда встречаем ячейки с буквой A, запоминаем сначала входное состояние E, затем F. После чего заполняем зеркальный автомат.

	X-A	**X-B**	**X-C**	**…**
A				
B				
C				
D				
E	A	B	C	
F	A	B	C	
G				
H				

$L_{\#}^{R}$	**X-A**	**X-B**	**X-C**	**…**
A	EF			
B		EF		
C			EF	
D				
E				
F				
G				
H				

Таблица 1

$L_{\#}$	**X-A**	**X-B**	**X-C**	**X-D**	**X-E**	**X-F**	**X-G**	**X-H**	**Y-A**	**Y-B**	**Y-C**	**Y-D**	**...**
A									A	B	C	D	
B									A	B	C	D	
C													
D													
E	A	B	C	D	E	F	G	H					
F	A	B	C	D	E	F	G	H					
G													
H													

Таблица 2

$L^R_{\#}$	**X-A**	**X-B**	**X-C**	**X-D**	**X-E**	**X-F**	**X-G**	**X-H**	**Y-A**	**Y-B**	**Y-C**	**Y-D**	**Y-E**	**Y-F**	**Y-G**	**Y-H**	**Z-A**
A	EF								AB								FGH
B		EF								AB							
C			EF								AB						

Зеркальный автомат также как и полный имеет размерность 10×80.

3. Детерминизация $L^R_{\#}$.

Далее на этом шаге происходит детерминизация зеркального автомата (табл. 2).

В цикле мы идем по строке зеркального автомата и запоминаем все встречающиеся нам состояния. Запомненные нами состояния становятся строками нового детерминированного автомата. После чего возвращаемся к зеркальному автомату и запоминаем, куда нас приводит каждое из состояний.

$\widetilde{L^R_{\#}}$	**X-A**	**X-B**	**X-C**	**X-D**
EF				
AB				
FGH				
C				
BC				
DE				
DEF				
A				
FG				
GH				

$L^R_{\#}$	**X-A**	**X-B**	**X-C**	**X-D**	**X-E**	**X-F**	**X-G**	**X-H**
A	EF							
B		EF						
C			EF					
D				EF				
E					**EF**			
F						**EF**		
G							EF	
H								EF

Записываем найденные состояния и переходы в детерминированный автомат:

$\widetilde{L^R_{\#}}$	**X-A**	**X-B**	**X-C**	**X-D**	**X-E**	**X-F**	**X-G**
EF					**EF**	**EF**	
AB							
FGH							
C							
BC							
DE							
DEF							
A							
FG							
GH							

Детерминированный автомат имеет размерность 14×80.

4. Переименованный автомат для $L^R_{\#}$.

Переобозначим полученный нами автомат.

$\widetilde{L^R_{\#}}$	**X-A**	**X-B**	**X-C**	**X-D**	**X-E**	**X-F**	**X-G**	**X-H**
X					X	X		
Y	X	X						
Z						X	X	X
U			X					
V		X	X					
W				X	X			
P				X	X	X		
Q	X							
R						X	X	
S							X	X

5. Канонический автомат для $L^R_{\#}$.

Автомат получился каноническим – все его состояния неэквивалентны по выходному языку.

Следующим этапом алгоритма является построение *COM* автомата.

6. Построение вспомогательных автоматов (автомата удовлетворяющего первому из условий существования дуги автомата

Таблица 3

Left, Right	**(1) A × YQ**	**(2) AB × Y**	**(3) B × YV**	**…**
(1) A × YQ				
(2) AB × Y				
(3) B × YV				
(4) BC × V				
(5) C × UV				
(6) DE × WP				
(7) E × XWP				
(8) EF × XP				
(9) F × XZPR				
(10) FG × ZR				
(11) G × ZRS				
(12) GH × ZS				
(13) FGH × Z				
(14) DEF × P				

Таблица 4

	.....	**Y-A**	**Y-B**	**Y-C**	**Y-D**	**Y-E**	**Y-F**	**Y-G**	**Y-H**	**.....**	**Q-A**	**Q-B**	**Q-C**	**Q-D**	**Q-E**	**....**
A	.....	A	B	C	D	E	F	G	H	.....	A	B	C	D	E	.....

$COM(L)$ и автомата удовлетворяющего второму из этих условий).

Для этого нам необходимо построить два автомата обозначим их как Left и Right. Логика построения у них одинаковая, разница заключается в том, что на вход алгоритма в первом случае подается полный автомат $L_{\#}$, а во втором – канонический $\widetilde{L}^{R}_{\#}$. Оба автомата являются таблицами 14×14 (табл. 3), где $A \times YQ$, $AB \times Y$, $B \times YV$, и т. д. – покрывающие блоки для таблицы бинарного отношения #. Для построения автомата Left мы смотрим на левые части покрывающих блоков и по полному автомату проверяем, как можно попасть в нужное нам состояние. Например, чтобы попасть из A в A мы смотрим в полном автомате на строчку для состояния A и запоминаем, куда мы можем попасть по букве A (табл. 4).

Получаем:

Left	**(1) A × YQ**	**(2) AB × Y**	**(3) B × YV**
(1) A × YQ	Y-A, Q-A		
(2) AB × Y			
(3) B × YV			

Таким же образом строим таблицу Right, только смотрим уже на правую часть покрывающих блоков ($YQ \times YQ$ и т.д.).

7. Построение усеченного COM автомата.

Далее мы усекаем COM автомат, оставляя только необходимые блоки: столбцы 1, 3, 5, 6, 8, 10, 12 и получаем усеченный COM автомат (табл. 5).

После чего выбираем общее в схожих ячейках обоих автоматов – это и будет COM автомат. Далее на вход метода подается содержимое соответствующих ячеек одной и второй таблицы, т.к. в строке могут находиться несколько букв–переходов, то мы разбиваем строку на части и сравниваем в цикле содержимое каждой из ячеек.

Таблица 5

COM	(1) **A × YQ**	(3) **B × YV**	(5) **C × UV**	(6) **DE × WP**	(8) **EF × XP**	(10) **FG × ZR**	(12) **GH × ZS**
(1) **A × YQ**	Y-A, Q-A	Y-B, Q-B	Y-C, Q-C	Y-D, Y-E, Q-D, Q-E	Y-E, Y-F, Q-E, Q-F	Q-F,Q-G,Y-F,Y-G	Q-G,Q-H, Y-G,Y-H
(3) **B × YV**	Y-A, V-A	Y-B, V-B	Y-C, V-C	Y-D, Y-E, V-D, V-E	Y-E, Y-F, V-E, V-F	V-F,V-G, Y-F,Y-G	V-G,V-H, Y-G,Y-H
(5) **C × UV**	U-A, V-A	U-B, V-B	U-C, V-C	U-D, U-E, V-D, V-E	U-E, U-F, V-E, V-F	U-F,U-G, V-F,V-G	U-G,U-H, V-G,V-H
(6) **DE × WP**	W-A, P-A	W-B, P-B	W-C, P-C	W-D, W-E, P-D, P-E	W-E, W-F, P-E, P-F	P-F,P-G, W-F,W-G	P-G,P-H, W-G,W-H
(8) **EF × XP**	X-A, P-A	X-B, P-B	X-C, P-C	X-D, X-E, P-D, P-E	X-E, X-F, P-E, P-F	P-F,P-G, X-F,X-G	P-G,P-H, X-G,X-H
(10) **FG × ZR**	Z-A, R-A	Z-B, R-B	Z-C, R-C	Z-D, Z-E, R-D, R-E	Z-E, Z-F, R-E, R-F	R-F,R-G, Z-F,Z-G	R-G,R-H, Z-G,Z-H
(12) **GH × ZS**	Z-A, S-A	Z-B, S-B	Z-C, S-C	Z-D, Z-E, S-D, S-E	Z-E, Z-F, S-E, S-F	S-F,S-G, Z-F,Z-G	S-G,S-H, Z-G,Z-H

Таблица 6

	Y-E	**Y-F**	**Y-G**	**Y-H**	**Z-A**	**Z-B**	**Z-C**	**Z-D**	**Z-E**	**Z-F**	**Z-G**	**Z-H**	**U-A**
A	E	F	G	H									
B	E	F	G	H									
C													A
D													
E													
F					A	B	C	D	E	F	G	H	
G					A	B	C	D	E	F	G	H	
H					A	B	C	D	E	F	G	H	

После формирования автомата на предыдущем шаге, мы из него путем слияния некоторых столбцов получаем новые автоматы и проводим проверки эквивалентности этих автоматов исходному. Перебор всевозможных перестановок для данного автомата дает нам поиск комбинаций столбцов.

Далее перебираются все возможные варианты слияния столбцов выбранных на предыдущем этапе. Данный алгоритм сравнивает состояния в ячейках таблицы для нового автомата с состояниями в ячейках таблицы усеченного COM автомата. Если в исходном автомате содержатся все значения из нового автомата, то полученный нами автомат не является Ватерлоо-подобным и поиск продолжается. Если в новом автомате в помеченной ячейке не содержится состояний из исходного автомата, это является допустимым, т.к. исходный автомат охватывает все возможные допустимые состояния, и их отсутствие в новом автомате не означает неэквивалентности.COM автомат содержит в себе состояния нового усеченного автомата, что тоже является признаком эквивалентности. Если же полученный усеченный автомат содержит какие-либо лишние состояния, то алгоритм дает ответ, что мы нашли Ватерлоо-подобный автомат.

ЭВРИСТИКА

Перебор всевозможных комбинаций столбцов полного автомата является довольно долгой и ресурсоемкой операцией. Поэтому для сокращения перебора были разра-

ботаны следующие эвристики. В алгоритме подсказывается какие столбцы полного автомата стоит обязательно взять, чтобы с большей вероятностью получить автомат Ватерлоо. Для этого в нашем случае используется сам Ватерлоо. На примере его таблицы это будет так (табл. 6).

То есть в новый сформированный автомат должны войти, по крайней мере, столбцы с выделенными состояниями. Что значительно сокращает время работы программы, т.к. не приходится перебирать все состояния автомата.

Так мы выбираем столбцы для последующего объединения или удаления и используем при этом сам автомат Ватерлоо.

ЗАКЛЮЧЕНИЕ

Предметом дальнейшей работы в данном направлении является запуск на параллельном кластере перечисления всех таблиц бинарного отношения # меньших или равных таблице Ватерлоо и попытка для каждой полученной таблицы перебрать всевозможные объединения и удаления букв полного автомата. Каждый текущий полученный усеченный автомат мы сравниваем с исходным автоматом и отвечаем на вопрос: является ли текущий автомат walibad'ом.

К настоящему моменту авторы не нашли примеров таблиц #, существенно отличных от таблицы автомата Ватерлоо, но при этом дающих Ватерлоо-подобные автоматы. (Мы не считаем таковыми автоматы, созданные искусственно – например, путём добавления строки ли столбца в таблицу бинарного отношения # автомата Ватерлоо.)

СПИСОК ЛИТЕРАТУРЫ

1. *Мельников Б. Ф.* Недетерминированные конечные автоматы / Б. Ф. Мельников // Тольятти : Изд-во ТГУ, 2009. – 160 с.

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

3. *Долгов В. Н.* Построение универсального конечного автомата. I. От теории к практическим алгоритмам / В. Н. Долгов, Б. Ф. Мельников // Вестник Воронеж. гос. ун-та. Сер. Физика. Математика. – 2013. – № 2. – С. 173–181.

4. *Долгов В. Н.* Построение универсального конечного автомата. II. Примеры работы алгоритмов / В. Н. Долгов, Б. Ф. Мельников // Вестник Воронеж. гос. ун-та. Сер. Физика. Математика. – 2014. – № 1. – С. 78–85.

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

6. *Brzozowski J.* Open problems about regular languages. Formal languages Theory: perspectives and open problems / J. Brzozowski. // New York: Academic Press, 1980. – P. 23–47.

7. *Kameda T.* On the state minimization of nondeterministic finite automata / T. Kameda, P. Weiner // IEEE Trans. on Comp. – 1970. – Vol. 19, № 7. – P. 617–627.

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

9. *Рогова О. А.* Репрезентативность случайно сгенерированных дискретных структур на примерах конечных автоматов и булевых функций / О. А. Рогова, Н. В. Софонова // Эвристические алгоритмы и распределенные вычисления. – 2014 – Т. 1, № 6. – С. 58–75.

10. *Melnikov B.* Once more about the state-minimization of the nondeterministic finite automata / B. Melnikov // The Korean Journal of Computational and Applied Mathematics. – 2000. – Vol. 7, № 3. – P. 655–662.

11. *Jiang T.* Minimal NFA problems are hard / T. Jiang, B. Ravikumar // SIAM J. Comput. – 1993. – Vol. 22, № 6. – P. 1117–1141.

12. *Долгов В. Н.* О дугах конечных автоматов для описания алгоритмов построения Ватерлоо-подобных автоматов / В. Н. Долгов, Н. В. Софонова // Стохастическая оптими-

зация в информатике. – 2015. – Т. 11, № 1. – С. 65–77.

13. *Мельников Б. Ф.* Мультиэвристический подход к задачам дискретной оптимизации // Кибернетика и Системный Анализ. Изд-во НАН Украины, 2006. – № 3. – С. 32–42.

14. *Мельников Б. Ф.* Некоторые эвристические алгоритмы в задаче вершинной минимизации недетерминированных конечных автоматов / Б. Ф. Мельников, Е. А. Мельникова // Стохастическая оптимизация в информатике. – 2013. – Т. 9, № 2. – С. 73–87.

15. *Софонова Н. В.* Эвристические алгоритмы генерации матриц бинарного отношения на состояниях канонических автоматов / Н. В. Софонова, В. А. Дудников // Стохастическая оптимизация в информатике. – 2015. – Т. 11, № 3 – С. 44–55.

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

Криволапова А. В. – бакалавр кафедры прикладной математики и информатики, Институт математики, физики и информационных технологий Тольяттинского государственного университета.
E-mail: emmilletta@gmail.com

Мельникова Е. А. – канд. физ.-мат. наук, доцент кафедры прикладной математики и информатики Тольяттинского филиала Самарского государственного университета
E-mail: ya.e.melnikova@yandex.ru

Софонова Н. В. – аспирант кафедры прикладной математики и информатики, Институт математики, физики и информационных технологий Тольяттинского государственного университета
E-mail: natashamilkova@yandex.ru

Krivolapova A. V. – student of department Applied Mathematics and Informatics, Institute of Mathematics, Physics and Information Technology, Togliatti State University.
E-mail: emmilletta@gmail.com

Melnikova E. A. – Candidate of Physical and Mathematical Sciences of department Applied Mathematics and Informatics, Togliatti Branch of Samara State University
E-mail: ya.e.melnikova@yandex.ru

Sofonova N. V. – post-graduate student of department Applied Mathematics and Informatics, Institute of Mathematics, Physics and Information Technology, Togliatti State University
E-mail: natashamilkova@yandex.ru