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

Б. Ф. Мельников

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

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

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

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

В настоящей части II мы рассматриваем в первую очередь недетерминированные автоматы. Также кратко рассмотрены особенности, появляющиеся при применении алгоритмов, формирующих конечные автоматы, к парам совпадающих языков. Кроме того, мы рассматриваем важное свойство введённых нами автоматов – определяем специальную алгебраическую систему, свойства которую предполагаем более подробно рассмотреть в последующих публикациях.

Статья получена 25 июня 2021 г.

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

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

В настоящей части II мы продолжаем нумерацию разделов, формул, рисунков, таблиц, определений и теорем. Нумерация ссылок на использованную литературу – новая.

V. О РАЗЛИЧНЫХ НЕДЕТЕРМИНИРОВАННЫХ АВТОМАТАХ, СООТВЕТСТВУЮЩИХ ПЕРВИЧНОМУ

В этом разделе очень кратко описано несколько вариантов недетерминированных конечных автоматов, соответствующих первичному; как уже было сказано во введении, бóльшие подробности и соответствующие примеры мы предполагаем рассмотреть в статье-продолжении.

Прежде всего отметим, что выше мы рассмотрели не один вид автоматов, а два: во втором случае мы удалили недостижимые состояния. После того, как мы доказали теорему 1 (часть I настоящей статьи), мы можем из автомата PRI удалять и недостижимые состояния – однако при этом надо осознавать, что автомат после подобных эквивалентных преобразований может стать пустым.

Следующий вариант эквивалентного автомата – это канонический автомат, соответствующий автомату PRI. Действительно, на приведённом ранее примере автомата (часть I, таб. 1 и рис. 2) имеются, например, эквивалентные состояния 6 и 12 – и в обычных алгоритмах канонизации автомата такие пары состояний обычно объединяют. Таким образом, можно определить канонический автомат, эквивалентный $\mathrm{PRI}\,(A, B)$ – причём имеющий практически все свойства, которые есть у автомата $\mathrm{PRI}\,(A, B)$. Подробные примеры приводить не будем: мы предполагаем рассмотреть их в статье-продолжении.

Существует и другая группа эквивалентных автоматов. А именно, наших предыдущих публикациях были рассмотрены многочисленные алгоритмы эквивалентного преобразования недетерминированных конечных автоматов; среди них упомянем лишь [1], [2], [3], [4], [5]. Основные из этих алгоритмов – варианты минимизации автоматов (в первую очередь это вершинная минимизация). Также в статье-продолжении мы предполагаем описать, *зачем* нужно рассматривать такие минимальные автоматы; но, в любом случае, все эквивалентные автоматы описывают *различные* варианты работы алгоритма проверки условия $A \trianglelefteqslant B$.

(Мы привели этот небольшой раздел в первую очередь для того, чтобы специально повторить следующее. Для

каждого варианта автомата очень интересна «обратная задача» – описать морфизм, либо просто указать пару языков, для которых получается некоторый заданный автомат. В другой предметной области примером такой задачи – точнее, примером её решения – является полный автомат, [6] (см. также [7], [8]): он строится на основе заданного бинарного отношения #, и, более того, с помощью нескольких вариантов преобразования из него можно получить любой конечный автомат для любого регулярного языка, которому соответствует некоторая заданная таблица бинарного отношения #.)

VI. ВСПОМОГАТЕЛЬНЫЙ НЕДЕТЕРМИНИРОВАННЫЙ АВТОМАТ

В этом разделе описан совершенно иной объект – хотя забегая вперёд отметим, что построение соответствующего ему конечного автомата может быть выполнено с помощью той же самой компьютерной программы – т. е. программы, практически совпадающей с приведёнными в наших предыдущих статьях. Поэтому, конечно, между описываемыми здесь и ранее автоматами имеется много общего. Однако автомат, рассматриваемый в настоящем разделе, определён не на множествах слов, а на словах; в связи с этим он, по-видимому, является «более простым» для любой пары рассматриваемых языков – однако, с нашей точки зрения, возможность его применения связана с более сложными построениями.

Продолжим рассматривать первичный автомат – точнее, его возможные преобразования. Мотивация для преобразований, описанных в этом и следующем разделах, такова. Выше мы рассматривали автоматы, состояния которых соответствовали всем возможным множествам префиксов языка B («множествам хвостов»)[1]. Однако эти возможные префиксы можно рассматривать и по отдельности. При этом наличие пути из вершины, соответствующей некоторому префиксу, в вершину 0 не означает отрицательный ответ алгоритма проверки условия $A \leqslant B$: ведь автомат недетерминированный, и поэтому для слова языка Δ_A, соответствующего этому пути, может существовать и какой-либо иной путь.

Итак, сначала автомат «со старым вариантом выхода».

Определение 11: (Недетерминированный) конечный автомат $\mathrm{NSPRI}^{\#}(A,B)$ для заданных конечных языков $A, B \subseteq \Sigma^*$ определяется следующим образом:

$$\mathrm{NSPRI}^{\#}(A,B) = (\, Q_B^{\#}, \Delta_A, \delta_{A-B}^{\#}, \{e\}, \varnothing\,),$$

где:

- $Q_B^{\#} = \mathrm{opref}\,(B) \cup \{\varnothing\}$;
- для некоторых $q_1, q_2 \in Q_B^{\#}$ и $d \in \Delta_A$ мы полагаем наличие перехода

$$q_1 \xrightarrow[\delta_{A-B}^{\#}]{d} q_2$$

тогда и только тогда, когда

$$\underset{\{h_A(d)\}-B}{\Phi}(q_1) \ni q_2\,.$$

□

[1] Неточности в терминологии нет: мы действительно называем «хвостами» некоторые префиксы слов языка B, а не их суффиксы. Всё это легко видеть, например, на основе описания шага 5 алгоритма, приведённого в определении 7 (часть I).

Некоторые комментарии к определению 11. В нём мы привели определение функции переходов именно таким образом – чтобы оно было похоже на аналогичную функцию определения 9 (часть I). При этом формула практически такая же: мы функцию

$$\underset{A-B}{\Phi}(q)$$

ранее определяли двумя способами (определения 7 и 8, часть I); поэтому формула может трактоваться как для состояния q, представляющего собой слово, так и для состояния q, представляющего собой язык.

Однако здесь (в отличие от определения 9, часть I), поскольку автомат недетерминированный, возможна и более короткая запись того же самого условия:

$$(\forall q \in Q_B^{\#},\, d \in \Delta_A)\,\big(\delta_{A-B}^{\#}(q,d) = \underset{\{h_A(d)\}-B}{\Phi}(q)\big).$$

При этом вводить «второе» состояние незачем.

Отметим также следующее. Мы пользуемся «практически одинаковой работой и с множеством, и его элементом» –

- не только в только что отмеченном случае, когда вместо аргумента функции как слóва (q в наших обозначениях) мы подставляем язык (множество слов, «множество хвостов»),
- но и когда вместо обычного языка A используется слово, т. е. в наших обозначениях –

$$\underset{\{h_A(a)\}-B}{\Phi}(q_1)\,;$$

поэтому, конечно, не является ошибкой более удобная запись, применённая в части I (а именно – $h_A(d)$ вместо $\{h_A(d)\}$ внизу слева под Ф).

Специально отметим, что для «особого» состояния $\varnothing$ переходов нет (что возможно, поскольку автомат недетерминированный) – это следует из приведённого определения 11.

Далее мы рассмотрим пример автомата $\mathrm{NSPRI}^{\#}(A,B)$, построенный для той же самой пары языков

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

которая рассматривалась в части I. Однако перед самим примером ещё раз отметим, что этот автомат может быть построен на основе *той же самой* компьютерной программы, с помощью которой мы в [9] получали дерево морфизма (а ранее в настоящей статье – автомат $\mathrm{PRI}\,(A,B)$). Таким же способом в настоящей статье можно получить и все остальные автоматы для аналогичных примеров.

Для возможности этих построений (с помощью той же самой компьютерной программы) мы снова пользуемся тем, что функция $\underset{A-B}{\Phi}(q)$ определена двумя способами: и для слова q, и для языка q.[2] В связи с этим добавлений в компьютерную программу, приведённую в [9], [10], вносить не нужно: достаточно вызывать нужным образом описанный нами ранее метод `ConcatSimple()`. Таким образом, для построения автомата NSPRI мы можем использовать следующий фрагмент программы (её основную часть см. в [9]):

[2] Мы в статье определяли и будем определять состояния автоматов именно такими двумя способами.

```
// задание исходного слова над Delta
// (как единственного слова языка tata):
TailsArr tata("");
cout << tata << endl;
// задание языка lA:
Language lA;
lA.AddWord("aaa");
lA.AddWord("aabba");
lA.AddWord("abba");
lA.AddWord("bb");
cout << lA << endl;
// задание языка lB:
Language lB;
lB.AddWord("aaaa");
lB.AddWord("abb");
lB.AddWord("abba");
lB.AddWord("bbb");
cout << lB << endl;
// собственно вычисления:
tata.ConcatSimple(lA,lB);
cout << tata << endl;
```

В рассматриваемом примере было приведено задание тех же самых языков, что и в части I. При этом мы в программе на отразили следующее: после получения её ответа надо продолжить работу, последовательно заменяя в первом операторе аргумент "" (пустую строку) на каждую из строк, *получающихся в результате выполнения этой программы* – и так далее[3]. Однако очевидно, что такой алгоритм не зацикливается (он конечен).

Итак, построенный для нашего примера автомат приведён в таблице 2 и далее на рис. 3:

Таб. 2. Автомат $\mathrm{NSPRI}^{\#}(A,B)$ для рассматриваемого примера.

			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

(Комментарии к таблице и рисунку – такие же, как к примеру автомата PRI из части I настоящей статьи.)

Специально отметим следующие отличия этого автомата от предыдущих.

- Автомат недетерминированный – поэтому в клетках таблицы стоят подмножества множества его состояний.

[3] Таким образом, мы *не стали строить автомат автоматически* (т. е. дописывать, «апгрейдить» приведённую ранее компьютерную программу): нам пришлось бы определять для такого автомата специальную структуру данных, а для неё – реализовывать соответствующие методы, и т. п. Для целей настоящей статьи такая детализация, по-видимому, является излишней.

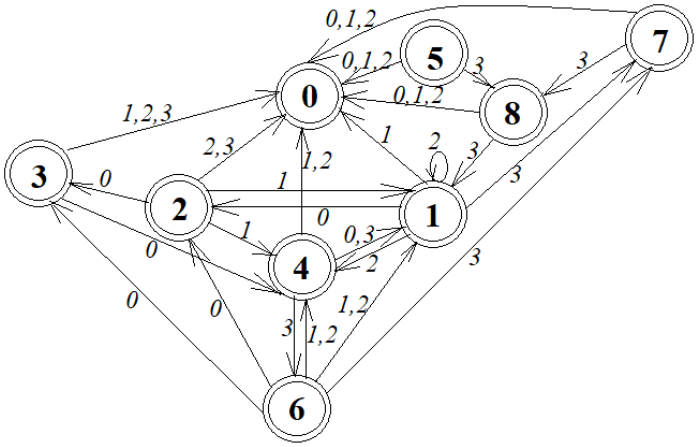

Рис. 3. Автомат $\mathrm{NSPRI}^{\#}(A,B)$ для рассматриваемого примера

- Символы { и } для обозначения множеств мы снова не употребляем – но элементы множеств пишем в обычном виде, через пробел.
- ∅ теперь представляет собой элемент, добавленный к множеству суффиксов[4].
- Рассмотрение состояний происходит не «в порядке появления» – а в порядке уменьшения длины префиксов слов языка B, причём сами эти слова рассматриваются в том порядке, в котором они входят в морфизм h_B.
- Мы рассматриваем *всё* множества собственных префиксов языка B.
- В связи с последним – в таких автоматах возможно появление недостижимых состояний[5]. В рассматриваемом нами примере таковым является состояние 5 (оно же ab), мы его отметили символом *. Как и в случае рассмотренных ранее детерминированных автоматов, такие состояния можно удалять без изменения смысла дальнейшего изложения.

Из всего вышесказанного следует такая теорема.

Теорема 2: Язык автомата $\mathrm{NSPRI}^{\#}(A,B)$ совпадает с языком автомата $\mathrm{PRI}(A,B)$.

При этом условие $A \not\trianglelefteqslant B$ *не выполняется* тогда и только тогда, когда

$$\mathcal{L}(\mathrm{NSPRI}^{\#}(A,B)) \neq \varnothing. \quad [6]$$

□

VII. УПРОЩЁННЫЙ ПЕРВИЧНЫЙ (SIMPLIFIED PRIMARY) НЕДЕТЕРМИНИРОВАННЫЙ АВТОМАТ

Однако (и это также следует из всего вышеизложенного) вряд ли имеет смысл рассматривать именно такой автомат: нам удобнее рассматривать не завершение работы в случае ответа «нет» алгоритма[7] – а сколь угодно

[4] Причём можно сказать «принудительно добавленный» – поскольку он имеет совсем иной смысл, не являясь словом языка Σ^*.

[5] Они у нас уже были, но пока применялись только в одном автомате, $\overline{\mathrm{PRI}}(A,B)$.

[6] То есть когда язык последнего автомата непуст.

И в очередной раз отметим, что ∅ – это и просто пустое множество (пустой язык), и обозначение состояний рассматриваемых автоматов. Ранее оно в подобных ситуациях символизировало пустое множество состояний – теперь же это обозначение применено для аналогии с предыдущими автоматами.

[7] В нашем случае автомат представляет собой алгоритм.

долгую («бесконечную») его работу, причём для любого слова языка $(\Delta_A)^*$, в случае ответа «да». В связи с этим изменим смысл выходов автомата[8] – что приводит к следующему определению.

Определение 12: (Недетерминированный) конечный автомат $\mathrm{NSPRI}(A,B)$ для заданных конечных языков $A, B \subseteq \Sigma^*$ определяется следующим образом:

$$\mathrm{NSPRI}(A,B) = (\,\mathrm{opref}(B), \Delta_A, \delta^{\#}_{A-B}, \{e\}, \mathrm{opref}(B)\,),$$

где функция переходов $\delta^{\#}_{A-B}$ совпадает с введённой в определении 11[9]. □

Смысл приведённого в определении автомата определяется следующей теоремой.

Теорема 3: Условие $A \trianglelefteqslant B$ выполняется тогда и только тогда, когда

$$\mathcal{L}(\mathrm{NSPRI}(A,B)) = (\Delta_A)^*.$$

□

То есть автомат – для того, чтобы алгоритм давал положительный ответ – должен определять полный язык над алфавитом Δ_A.

Для рассматриваемого нами примера пары языков автомат получается такой[10]:

Таб. 3. Автомат $\mathrm{NSPRI}(A,B)$ для рассматриваемого примера.

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

Конечно, интересен также пример автомата, дающего *положительный* ответ. Как мы знаем ([11] и др.), достаточным условием этого является существование некоторого конечного языка D, такого что $A, B \in \mathrm{mp}^+(D)$. Поэтому рассмотрим такой пример:

$$A = \{\, a, ba, bb, babab \,\}, \quad B = \{\, aa, ab, abb, b \,\};$$

при этом несложно показать, что $D = \{\, a, b \,\}$.

В результате работы алгоритма получается автомат, приведённый ниже в таб. 4. Понятно, что в построенном автомате также имеется недостижимое состояние (как и ранее, помеченное *) – однако основное свойство автомата видно и на основе приведённой таблицы: в таблице отсутствуют «пустые клетки».

[8] Та же самая операция (но для детерминированных автоматов) применялась «в студенческих курсах по формальным языкам» – в аналогичном случае, при доказательстве замкнутости класса регулярных языков относительно операции дополнения.

[9] Со следующим небольшим уточнением. Возможные её значения $\varnothing$, не являющиеся в случае автомата $\mathrm{NSPRI}(A,B)$ одним из состояний, воспринимаются «в обычном смысле», т. е. просто как пустые множества; это возможно, поскольку автомат недетерминированный.

[10] Понятно, что нумерацию состояний теперь удобно начинать с 1. Буквы алфавита Δ_A, как и ранее, нумеруем с 0. Состояние, являющееся одновременно входным и выходным, помечаем ↔.

Таб. 4. Автомат $\mathrm{NSPRI}(A,B)$ для примера с положительным ответом.

			0	1	2	3
↔	1	ε	2	2	1	1 4
←	2	a	1	2	1	1 4
*	3	b	2	2	1	1 4
←	4	ab	2	2	1	1 4

В качестве заключения раздела отметим следующее. К сожалению, определить «полноту» некоторого языка (т. е. его совпадение с языком $(\Delta_A)^*$), заданного, например, недетерминированным конечным автоматом, за полиномиальное время, вообще говоря, невозможно[11]: приведённое выше условие отсутствия «пустых клеток» является достаточным – но не необходимым и достаточным. Даже в случае однобуквенного алфавита – что для наших проблем неинтересно – подобная задача является NP-полной, см. формулировку в [12, упр. 10.14][12]. Однако на основании последнего сформулированного факта мы *ещё не можем утверждать, что полиномиального алгоритма проверки выполнения отношения* $\trianglelefteqslant$ *не существовует*: для такого утверждения необходимы дополнительные исследования.

VIII. ВАРИАНТЫ АВТОМАТОВ ДЛЯ ПАР СОВПАДАЮЩИХ ЯЗЫКОВ

Очень кратко рассмотрим особенности, появляющиеся при применении алгоритмов, формирующих конечные автоматы, к парам совпадающих языков. При этом важно отметить, что не очень интересно рассмотрение примеров, когда язык, формирующий пару, обладает свойством префикса. А наиболее интересно (опять же для случая совпадающих языков A и B) рассмотрение ситуаций, когда этот язык является непрефиксным морфизмом непрефиксного языка – т. е. когда для некоторого языка $A_\Delta \in \Delta$, причём такого, что

$$A_\Delta \in \mathrm{mp}^+(\Delta), \quad \text{но} \quad A_\Delta \notin \mathrm{mp}(\Delta),$$

выполнено условие $A = h(A_\Delta)$. (Отметим, что нижний индекс у h мы здесь опускаем: он не совпадает с A, и поэтому для рассматриваемого нами примера малоинтересен.)

Итак, пусть

$$D = \{\, 0, 10, 11, 10101 \,\}$$

(это фактически язык A из нашего предыдущего примера – но над другим алфавитом),

$$h(0) = ab, \;\; h(1) = aba,$$

тогда

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

а для автомата $\mathrm{PRI}(A,A)$ мы получаем следующие 12 языков-состояний (не считая состояния $\varnothing$):

$$\{\varepsilon, ab\},\ \{\varepsilon, abaab\},\ \{\varepsilon, a, abaaba\},\ \{\varepsilon, a, aba, abaababa\},$$
$$\{\varepsilon, ab, abaabab\},\ \{\varepsilon, abaab, abaababaab\},$$

[11] Автор выражает благодарность за консультации по этому вопросу независимому исследователю В. Н. Долгову, а также А. А. Рубцову (Московский физико-технический институт).

[12] Решение в книге, однако, не приводится – а помечено оно как **, что означает самую большую сложность.

$$\{\varepsilon, a, abaaba, abaababaaba\},\ \{\varepsilon, ab, abaab, abaababaab\},$$
$$\{\varepsilon, ab, abaab\},\ \{\varepsilon, a, aba, abaaba\},$$
$$\{\varepsilon, ab, abaabab, abaababaabab\},$$
$$\{\varepsilon, a, aba, abaaba, abaababaaba\}.$$

Состояния автомата приведены в порядке их появления при работе описанного выше алгоритма. А состояние $\{e\}$, конечно, в результате работы алгоритма построения автомата $\mathrm{PRI}\,(A, A)$ не получается, поскольку здесь заранее понятно, что алгоритм проверки отношения $\preccurlyeq$ (в старой нотации – отношения $\overset{\subset}{\infty}$ для проитерированных языков) должен дать положительный результат. (А ещё один признак этого – наличие элемента-слóва ε в любом состоянии, рассматриваемом как множество слов.)

Мы собираемся продолжить исследование таких автоматов в последующих публикациях.

IX. ОБ ОДНОМ ВАЖНОМ СВОЙСТВЕ РАССМАТРИВАЕМЫХ АВТОМАТОВ

Рассмотрим ещё раз автомат $\mathrm{PRI}\,(A, B)$ для некоторых языков $A, B \subseteq \Sigma^*$. [13] Согласно определению функции

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

для любой буквы $d \in \Delta_A$ и для любой пары состояний $q_1, q_2 \subseteq \mathcal{P}(\mathrm{opref}\,(B))$, таких что $q_1 \subseteq q_2$, выполнено следующее условие:

$$\underset{\{h_A(d)\}-B}{\Phi}(q_1) \subseteq \underset{\{h_A(d)\}-B}{\Phi}(q_2). \tag{5}$$

Будем использовать условие (5) в дальнейших построениях и опускать очевидные фигурные скобки в нижних индексах [14].

По определению, для операции объединения состояний [15] выполнены все условия операции, необходимой для задания полурешётки (см. [13, с. 192], а также близкую к нашей тематике монографию [14] [16]): идемпотентность, ассоциативность и коммутативность. Как всегда для полурешёток, мы можем рассматривать эту полурешётку как полугруппу (моноид), причём с такой единицей этого моноида: $\mathcal{P}(\mathrm{opref}\,(B))$.

Теперь определим операцию на том же самом множестве $\mathcal{P}(\mathrm{opref}\,(B))$ более сложным (но тоже «естественным») образом: для каждой буквы $d \in \Delta_A$ и для любой пары состояний $q_1, q_2 \subseteq \mathcal{P}(\mathrm{opref}\,(B))$ положим в качестве результата этой операции:

- $q_1 \cup q_2$, если

$$\underset{h_A(d)-B}{\Phi}(q_1 \cup q_2) = \underset{h_A(d)-B}{\Phi}(q_1) \cup \underset{h_A(d)-B}{\Phi}(q_2); \tag{6}$$

- неопределённость в противном случае.

Такую операцию обозначим $\cup_d$. Из предыдущего (из определения функции Φ и условия (5)) следует, что мы таким способом действительно определяем *алгебру*, т. е. для любых элементов $q_1, q_2 \in \mathcal{P}(\mathrm{opref}\,(B))$ и для любой буквы $d \in \Delta_A$ результат операции

$$q_1 \cup_d q_2$$

существует (и, конечно, принадлежит тому же самому множеству $\mathcal{P}(\mathrm{opref}\,(B))$) – т. е. неопределённость получаться никогда не может.

Более того, раз результат всегда существует, то операция $\cup_d$ представляет собой просто операцию объединения, т. е. $\cup$. Также из изложенного в этом разделе следует, что вместо (6) можно использовать и более сложное условие, т. е. *одновременное* выполнение этого же равенства для всех букв $d \in \Delta_A$. И, конечно, удобно считать, что *именно эта* операция (одновременное выполнение равенства (6) для всех букв $d \in \Delta_A$) и есть операция полурешётки. При этом решётки, вообще говоря, не образуется – потому что у автомата $\mathrm{PRI}\,(A, B)$, вообще говоря, имеются недостижимые состояния.

Это отсутствие решётки (наличие недостижимых состояний) является *ещё одним «косвенным аргументом» за то, что полиномиальные алгоритмы решения рассматриваемых нами задач отсутствуют* [17]. Действительно, если N – максимальная длина слова в алфавите B (для небольшого упрощения будем считать N чётным), то только у этого слова имеется N собственных префиксов, и мы можем построить автомат $\mathrm{PRI}\,(A, B)$, имеющий не менее

$$\mathrm{C}_N^{N/2} \tag{7}$$

состояний; «обработать» такое число состояний за полиномиальное относительно N время невозможно.

Немного подробнее. На «стандартных» картинках, изображающих все множества подмножеств как решётку, мы полурешётку могли бы выбрать как «середину плюс всю верхнюю часть»; тогда «нижняя часть» будет описывать недостижимые состояния автомата PRI. Такой выбор обусловлен тем, что, как следует из материала настоящего раздела, для получения ответа на вопрос о принятии автоматом полного языка $(\Delta_A)^*$ нам достаточно рассмотреть только все «нижние состояния» полурешётки – её т. н. *атомы*.

В качестве примера рассмотрим приведённый ниже рис. 4. На нём цифры 1, 2, 3 и 4 означают номера собственных префиксов некоторого слова языка B (включая ε). Запись 124 означает подмножество множества префиксов, содержащее элементы с номерами 1, 2 и 4 – аналогично для остальных подмножеств, включая $\varnothing$.

Понятно, что каждое подмножество соответствует некоторому состоянию автомата PRI. В нашем примере

[13] Заранее отметим, что все приведённые в этом разделе построения могут быть выполнены и для рассматриваемых в настоящей части II недетерминированных автоматов: просто для последних вместо состояний нужно рассматривать подмножества множества состояний. Однако в данном случае рассмотрение именно детерминированного автомата удобнее.

[14] Выше мы уже отмечали такую возможность. Кроме того, мы использовали подобную «бесскобочную» запись в части I.

[15] Напомним, что у автомата $\mathrm{PRI}\,(A, B)$ каждое из состояний является множеством (слов) – именно с этой точки зрения мы и рассматриваем операцию объединения состояний.

[16] Стр. 215 новой версии этой монографии, 2020 г. издания.

[17] А первый «косвенный аргумент» был приведён в конце раздела VII; он также выделен курсивом.

Важно также отметить, что приведённые здесь и далее аргументы (в том числе – пример конкретной полурешётки) *нельзя рассматривать как доказательство отсутствия полиномиальных алгоритмов* для решения задач, рассматриваемых в настоящей статье. Более того, мы не сможем утверждать этого, даже построив по некоторой полурешётке, содержащей «достаточно много» состояний, конкретный недетерминированный автомат – например, для числа состояний, равного рассматриваемому нами значению (7): ведь мы при этом не докажем, что для этого «достаточно большого» значения существует соответствующее ему бесконечное итерационное дерево морфизма (и, что ещё важнее, пара задающих его языков A и B).

Однако, повторим, «косвенными аргументами» оба приведённых нами обоснования, конечно, являются.

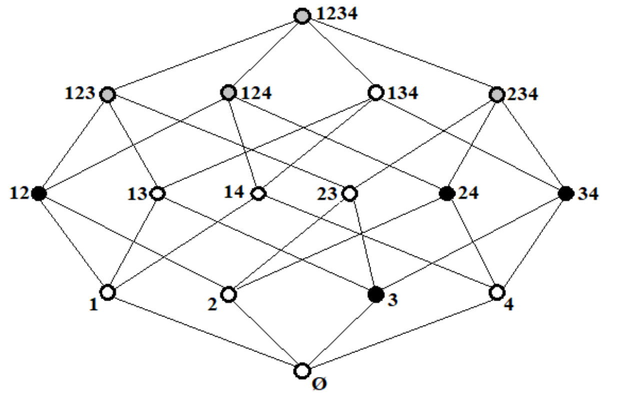

Рис. 4. Пример полурешётки на подмножествах множества $\mathrm{opref}(u)$; подробные комментарии см. в тексте раздела.

мы будем считать, что в автомате $\mathrm{PRI}(A, B)$ – для некоторых заданных языков A и B и некоторого соответствия 4 префиксов слова B числам от 1 до 4 – *автоматически* (при построении) получаются состояния, обозначенные чёрными кружкáми, т. е. $\{3\}$, $\{1,2\}$, $\{2,4\}$ и $\{3,4\}$.[18] Состояния, обозначенные серыми кружкáми (т. е. $\{1,2,3\}$, $\{1,2,4\}$, $\{2,3,4\}$ и $\{1,2,3,4\}$), можно не рассматривать (хотя они в автомат могут входить) – в связи со сформулированным выше наличием полурешётки по объединению состояний.

Возможность такого «нерассмотрения нескольких состояний» можно неформально объяснить следующим образом. Наша цель – ответить на вопрос, непуст ли выходной язык каждого из этих состояний[19] – а это, в свою очередь, нужно для получения ответа на вопрос об условии $A \trianglelefteq B$. А такие ответы можно получить и без рассмотрения «невыделенных» состояний: они, как мы предполагаем доказать в будущем, входить в автомат $\mathrm{PRI}(A, B)$ не должны. И, по-видимому, рассматривая все префиксы слóва длины 4, можно подобрать примеры, когда подобных «чёрных» состояний (самых важных для исследования свойств автомата PRI) будет $C_4^2 = 6$. Но, повторим, приведённые аргументы пока являются лишь неформальным объяснением.

X. ЗАКЛЮЧЕНИЕ

Кратко сформулируем направления дальнейшей работы – частично приведённые в пердыдущем тексте статьи, прежде всего в разделах II (часть I), VII и IX, – связанные с рассмотренными в настоящей статье вопросами.

В разделе IX работа с «чёрными» и «серыми» состояниями автомата $\mathrm{PRI}(A, B)$ была описана очень схематично. Более того, нам более удобен для работы не сам автомат $\mathrm{PRI}(A, B)$ а практически совпадающий с ним автомат, получающийся в процессе детерминизации и детерминизации автомата $\mathrm{NSPRI}(A, B)$ (язык которого, в отличие от автоматов $\mathrm{PRI}(A, B)$ и $\mathrm{NSPRI}^{\#}(A, B)$, сравнивается с «полным» языком $(\Delta_A)^*$); а такой автомат – с чисто формальной точки зрения – нами даже не был определён. Поэтому необходимо:

- строго определить сам этот автомат;
- также определить «чёрные» и «серые» состояния – как для него, так и для автомата $\mathrm{PRI}(A, B)$;
- а также описать алгоритмы их построения,

после чего подробно сформулировать и доказать свойства этих объектов – в частности, оценить время, необходимое для их построения.

По-видимому, *самая важная* задача связана с возможной неполиномиальностью всего алгоритма построения автомата $\mathrm{NSPRI}(A, B)$ (а, следовательно, и других автоматов, рассмотренных в обеих частях статьи). Однако отметим, что из возможного положительного ответа на этот вопрос не будет следовать NP-полнота основной рассматриваемой нами задачи – а именно, ответа на вопрос об эквивалентности в бесконечности некоторых заданных языков A и B: возможно, имеются какие-нибудь принципиально отличающиеся алгоритмы ответа на этот вопрос. Иными словами – пока мы не доказали, что нельзя проверить полиномиальность алгоритма проверки отношения $\trianglelefteq$ (а следовательно, и $\Leftrightarrow$) каким-либо другим способом.

А *самая ближайшая* задача (конечно, значительно более простая, чем предыдущая) – это описание полиномиального алгоритма построения автомата NSPRI. (Интуитивно понятно, что само построение этого автомата – без получения ответа на вопрос о принятии им «полного» языка $(\Delta_A)^*$ – можно осуществить за полиномиальное время.)

Более сложная задача, связанная с предыдущей, такова. Можно ли утверждать, что для тех недетерминированных автоматов, для которых построенный с помощью стандартной процедуры детерминированный аналог обладает описанным в разделе IX «полурешёточным» свойством, также невозможно получение ответа на вопрос о принятии ими «полного» языка за полиномиальное время? (Ведь, возможно, для *таких* автоматов более эффективные алгоритмы существуют.)

Следующая группа возможных задач такова. Для «особого» состояния $\varnothing$ мы не рассматриваем переходы из него – это просто следует из приведённого определения 11. Вопрос такой: может ли рассмотрение т. н. всюду определённого (total) варианта автомата[20] привести к изменению смысла? (То есть – может ли это привести к неверному ответу на основной вопрос, о результате выполнения отношения $\trianglelefteq$?) У автора примеров такого «изменения смысла» нет – хотя в нашем случае преобразование автомата не является эквивалентным, поскольку в недетерминированном случае для некоторого слова возможное попадание автомата в «дохлое» состояние *не* означает, что это слово не может быть префиксом некоторого другого слова языка преобразованного автомата.

Теперь перейдём к областям, «стоящим немного в стороне» от рассматриваемых нами вопросов – но, конечно, тоже связанных с ними и с материалом статьи [15]. В одной из ближайших публикаций мы собираемся

[18] Это и есть атомы для нашего примера.

[19] Или, что то же самое, совпадает ли выходной язык каждого из состояний автомата, полученного детерминизацией $\mathrm{NSPRI}(A, B)$, с полным языком $(\Delta_A)^*$.

[20] Имеется в виду результат следующего преобразования автомата – аналогичного подобному эквивалентному преобразованию в детерминированном случае, дающему там т. н. всюду определённый (total) детерминированный автомат: добавление всех «отсутствующих» переходов – в т. н. «дохлое» состояние (dead state), обычно специально для этого добавляемое. Такую же процедуру можно провести и для недетерминированного автомата – и в нашем случае в качестве «дохлого» состояния рассматривая состояние $\varnothing$.

рассмотреть вопросы, связанные с построением *и применением* множества потенциальных корней (они были упомянуты, например, в разделе 2 части I, а также в нескольких наших предыдущих публикациях, первой из которых, по-видимому, была [16]; см. также [17]) – в частности, про единственность построения этого множества. Смысл этих действий таков: ведь фактически все слова, которые рассматривались для построения автомата $\mathrm{PRI}\,(A, B)$ (т. е., немного упрощая, слова языков A и B), – это и есть такие потенциальные корни некоторых других, «исходных» языков. А для получения такого множества потенциальных корней мы, например, можем рассмотреть следующий алгоритм работы с некоторым заданным языком $A \subseteq \Sigma^*$.

Сначала определяем вспомогательный язык

$$D' = \left\{\, u \in \Sigma^* \;\middle|\; (\exists v \in A, \exists n \in \mathbb{N})\,(v = u^n) \,\right\};$$

отметим, что используя наши обозначения, применённые в предыдущих статьях, можно D' записать следующим образом:

$$D' = \bigcup_{u \in A,\, n \in \mathbb{N}} \overline{\sqrt[n]{\{u\}}} \quad \text{либо} \quad D' = \bigcup_{n \in \mathbb{N}} \overline{\sqrt[n]{A}}$$

(в этих формулах мы не делаем различий между объединением элементов и объединением множеств).

Далее мы можем построить такой *минимальный* [21] язык D, что

$$D \subseteq D' \;\;\&\;\; D^* \supseteq A.$$

Важно отметить, что несмотря на «декларативность» последнего определения, на его основе может быть легко определены и несколько вариантов *алгоритма* построения языка D, причём в результате применения любого из них D определяется *однозначно*.

А итоговое подмножество языков, находящихся в отношении $\diamondsuit$ с исходным языком A, является множеством $\mathrm{mp}^+(D)$.

То, что было описано выше про множество потенциальных корней – относится как к задаче, рассматриваемой в настоящей статье, так и к задаче извлечения корня из языка, [16] и др. Однако на этом аналогии, по-видимому, кончаются: в случае нашей задачи подходящие подмножества потенциальных корней образуют решётку относительно операций пересечения и объединения [22] – в то время как для задачи извлечения корня из языка ни одной из двух полурешёток не получается [23].

[21] Формально эта «минимальность» может быть определена несколькими естественными способами.

[22] Для объединения это очевидно – а выполнение условий полурешётки для операции пересечения мы предполагаем обсудить в статье-продолжении.

[23] Для описания подтверждающих примеров к последнему факту будем рассматривать алфавит, состоящий из одной буквы a, а конечные языки задавать в виде вектора, дли́ны слов в котором записаны справа налево начиная с длины́ 0; будем записывать языки в виде равенства, например, $A = 1011$ – это фактически язык $A = \{\, \varepsilon, a, a^3 \,\}$.

Отсутствие полурешётки относительно пересечения подтверждается очень простым примером: $A = 110111$, $B = 111011$, здесь $A^2 = B^2$, потенциальные корни – 111111, и при этом $(A \cap B)^2 \neq A^2$.

Примеры, подтверждающие отсутствие полурешётки относительно объединения, сложнее; «минимальный» из примеров, найденных компьютерной программой, такой: $A = 1101000111$, $B = 1101001011$, здесь $A^2 = B^2$, но $(A \cup B)^2 \neq A^2$.

В статье-продолжении мы предполагаем рассмотреть эти вопросы подробнее.

Кроме того, возможными направлениями дальнейшей работы являются несколько вариантов «обратной задачи», кратко описанные в части I (во введении).

Список литературы

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

[2] Мельников Б., Мельникова А. *Многоаспектная минимизация недетерминированных конечных автоматов (Часть I. Вспомогательные факты и алгоритмы)* // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. – 2011. – № 4 (20). – С. 59–69.

[3] Мельников Б., Мельникова А. *Многоаспектная минимизация недетерминированных конечных автоматов (Часть II. Основные алгоритмы)* // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. – 2012. – № 1 (21). – С. 31–43.

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

[5] Abramyan M., Melnikov B. *An approach to algorithmizing the problem of vertex minimization of nondeterministic automata. Part I. Problem statement and the brief description of the basis methods* // IOP Conference Series: Materials Science and Engineering. Krasnoyarsk Science and Technology City Hall of the Russian Union of Scientific and Engineering Associations. 2020. C. 52055.

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

[7] Melnikov B., Melnikova A. *An approach to the classification of the loops of finite automata. Part I: Long corresponding loops* // International Journal of Open Information Technologies. – 2018. – Vol. 6. No. 9. – P. 9–14.

[8] Melnikov B., Melnikova A. *An approach to the classification of the loops of finite automata. Part II: The classification of the states based on the loops* // International Journal of Open Information Technologies. – 2018. – Vol. 6. No. 11. – P. 1–6.

[9] Мельников Б., Мельникова А. *Бесконечные деревья в алгоритме проверки условия эквивалентности итераций конечных языков. Часть II* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 5. – P. 1–11.

[10] Мельников Б., Мельникова А. *Бесконечные деревья в алгоритме проверки условия эквивалентности итераций конечных языков. Часть I* // International Journal of Open Information Technologies. – 2021. – Vol. 9. No. 4. – P. 1–11.

[11] Melnikov B. *The equality condition for infinite catenations of two sets of finite words* // International Journal of Foundation of Computer Science. – 1993. – Vol. 4. No. 3. – P. 267–274.

[12] Ахо А., Хопкрофт Дж., Ульман Дж. *Построение и анализ вычислительных алгоритмов*. – М., Мир. – 1979. – 536 с.

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

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

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

[16] Melnikov B., Korabelshchikova S., Dolgov V. *On the task of extracting the root from the language* // International Journal of Open Information Technologies. – 2019. – Vol. 7. No. 3. – P. 1–6.

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

Борис Феликсович МЕЛЬНИКОВ,
профессор Университета МГУ – ППИ в Шэньчжэне
(http://szmsubit.ru/),
email: bf-melnikov@yandex.ru,
mathnet.ru: personid=27967,
elibrary.ru: authorid=15715,
scopus.com: authorId=55954040300,
ORCID: orcidID=0000-0002-6765-6800.

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

Boris Melnikov

***Abstract*—Quite briefly, the subject of this paper can be formulated as follows: in the previously considered infinite iterative morphism trees, we combine equivalent vertices, in fact, obtaining a deterministic finite automaton; after that, we investigate some properties of the resulting automaton. In addition, the article describes the possible connection of the automata we are considering with problems arising in other areas of the theory of formal languages.**

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

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

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

In this Part II, we consider primarily nondeterministic automata. The features that appear when applying algorithms that form finite automata to pairs of matching languages are also briefly considered. In addition, we consider an important property of the introduced automata, we define a special algebraic system, the properties of which is proposed to consider in more detail in following publications.

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

References

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

[2] Melnikov B., Melnikova A. *Multidimensional minimization of nondeterministic finite automata. Part I. Auxiliary facts and algorithms* // News of higher educational institutions. Volga region. Physical and mathematical sciences. – 2011. – No. 4 (20). – P. 59–69 (in Russian).

[3] Melnikov B., Melnikova A. *Multidimensional minimization of nondeterministic finite automata. Part II. Basis algorithms* // News of higher educational institutions. Volga region. Physical and mathematical sciences. – 2012. – No. 1 (21). – P. 31–43 (in Russian).

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

[5] Abramyan M., Melnikov B. *An approach to algorithmizing the problem of vertex minimization of nondeterministic automata. Part I. Problem statement and the brief description of the basis methods* // IOP Conference Series: Materials Science and Engineering. Krasnoyarsk Science and Technology City Hall of the Russian Union of Scientific and Engineering Associations. 2020. C. 52055.

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

[7] Melnikov B., Melnikova A. *An approach to the classification of the loops of finite automata. Part I: Long corresponding loops* // International Journal of Open Information Technologies. – 2018. – Vol. 6. No. 9. – P. 9–14.

[8] Melnikov B., Melnikova A. *An approach to the classification of the loops of finite automata. Part II: The classification of the states based on the loops* // International Journal of Open Information Technologies. – 2018. – Vol. 6. No. 11. – P. 1–6.

[9] Melnikov B., Melnikova A. *Infinite trees in the algorithm for checking the equivalence condition of iterations of finite languages. Part I* // International Journal of Open Information Technologies. – 2021. – Vol. 9. 2021. – Vol. 9. No. 4. – P. 1–11 (in Russian).

[10] Melnikov B., Melnikova A. *Infinite trees in the algorithm for checking the equivalence condition of iterations of finite languages. Part II* // International Journal of Open Information Technologies. – 2021. – Vol. 9. 2021. – Vol. 9. No. 5. – P. 1–11 (in Russian).

[11] Melnikov B. *The equality condition for infinite catenations of two sets of finite words* // International Journal of Foundation of Computer Science. – 1993. – Vol. 4. No. 3. – P. 267–274.

[12] Aho A., Hopcroft J., Ullman J. *The Design and Analysis of Computer Algorithms*. – Massachusetts, Addison-Wesley. – 1974. – 470 p.

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

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

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

[16] Melnikov B., Korabelshchikova S., Dolgov V. *On the task of extracting the root from the language* // International Journal of Open Information Technologies. – 2019. – Vol. 7. No. 3. – P. 1–6.

[17] Alekseeva A., Melnikov B. *Iterations of finite and infinite languages and nondeterministic finite automata* // Vector of Science of Togliatti State University. – 2011. – No. 3 (17). – P. 30–33 (in Russian).

Boris MELNIKOV,
Professor of Shenzhen MSU – BIT University, China
(http://szmsubit.ru/),
email: bf-melnikov@yandex.ru,
mathnet.ru: personid=27967,
elibrary.ru: authorid=15715,
scopus.com: authorId=55954040300,
ORCID: orcidID=0000-0002-6765-6800.