Восемь вариантов конечных автоматов для проверки выполнения отношения покрытия итераций двух конечных языков. Часть I

Б. Ф. Мельников, Мэн Линцянь

Аннотация—**Задача, для которой рассматриваются все автоматы настоящей статьи, заключается в описании алгоритмов проверки т. н. бинарного отношение покрытия, рассматривавшегося во многих наших предыдущих статьях: оно выполняется для двух непустых конечных языков тогда и только тогда, когда любая итерация слов первого языка является префиксом некоторой итерации слов второго языка. Эту задачу можно рассматривать с нескольких разных точек зрения, что и делалось в предыдущих работах.**

В настоящей статье нас в первую очередь интересуют различные варианты реализации алгоритмов для проверки выполнения этого отношения, а более конкретно – применение для таких алгоритмов конечных автоматов. 2 варианта таких автоматов уже были ранее описаны в наших предыдущих работах, а в настоящей статье мы к ним добавляем ещё 6 вариантов.

Основная цель рассмотрения нескольких автоматов-алгоритмов – такая. Ранее нами был описан полиномиальный алгоритм построения одного из таких автоматов. Однако существование такого полиномиального алгоритма вовсе не означает, что мы можем полиномиально проверить выполнение необходимого нам условия – в связи со следующим обстоятельством. Если мы для ответа на поставленный вопрос строим недетерминированный автомат таким же образом, как мы описывали ранее, то он для положительного ответа должен принимать любое слово универсального языка – а последнее условие за полиномиальное время проверить нельзя. Однако мы также не можем утверждать, что подобного полиномиального алгоритма не существует – необходимы дополнительные исследования. Поэтому наличие четырёх описаний недетерминированных автоматов, отвечающих на поставленный вопрос о выполнении рассматриваемого бинарного отношения, даст бóльшие возможности для поиска возможного алгоритма, проверяющего основное условие статьи за полиномиальное время.

8 вариантов рассматриваемых в статье автоматов естественным образом получаются с помощью трёх различных дихотомий: одна из них – «обычная» (автомат детерминированный или недетерминированный), а две других непосредственно связаны с рассматриваемыми в статье вопросами проверки выполнения бинарного отношения пркрытия. Конкретно, эти две дихотомии таковы: во-первых, какой именно из двух рассматриваемых конечных языков является «главным» (именно для него мы применяем основной морфизм для проверки); во-вторых, рассматриваем ли мы требуемые покрытия каждого слова-итерации непосредственно при его возникновении – либо не сразу, а после формирования некоторого вспомогательного языка.

В представляемой первой части настоящей статьи мы кратко описываем все необходимые дихотомии, приводим связанные с ними вспомогательные обозначения, а также приводим новые варианты формулировок двух автоматов, рассматривавшихся в предыдущих статьях.

Ключевые слова—**формальные языки, итерации языков, морфизмы, конечные автоматы, алгоритмы.**

I. ВВЕДЕНИЕ И МОТИВАЦИЯ

Сначала сформулируем задачу, для которой и рассматриваются все автоматы, о которых идёт речь в названии настоящей статьи. Отношение покрытия – это рассматривавшееся во многих наших предыдущих статьях бинарное отношение $\preccurlyeq$, см. [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] и мн. др. Однако само это название («отношение покрытия») в статьях употреблялось не всегда: иногда вместо названия приводилось только определение в виде формул.

Приведём само такое формальное определение. Если для двух конечных языков A и B (в наших работах они чаще всего рассматриваются над «главным» алфавитом Σ и предполагаются непустыми и не содержащими пустого слова ε) выполняется условие

$$(\forall u \in A^*)\ (\exists v \in B^*)\ (u \in \mathrm{opref}(v)), \qquad (1)$$

то мы пишем $A \preccurlyeq B$ (либо $B \succcurlyeq A$). Всюду далее языки A и B будут использоваться именно в таком смысле, т. е. для описания условия (1) с некоторыми рассматриваемыми языками A и B.

(Если условия $A \preccurlyeq B$ и $A \succcurlyeq B$ выполняются одновременно, то мы пишем $A \Leftrightarrows B$. Однако в настоящей статье бинарное отношение $\Leftrightarrows$ – являющееся отношением эквивалентности – мы рассматривать не будем.)

В предыдущих работах мы также отмечали отличие используемых в последних работах обозначений – от наших же обозначений, применявшихся в статьях 1990-х годов, см. [11] и др. Конкретно, там соответствующие бинарные отношения определялись не для итерируемых конечных языков, а для *бесконечных* языков (ω-языков), являющихся результатом подобных итераций. А для последних легко устанавливается связь с бесконечными итерациями:

- условие (1) (т. е. $A \preccurlyeq B$) выполняется тогда и только тогда, когда $A^\omega \subseteq B^\omega$;
- а условие $A \Leftrightarrows B$ – тогда и только тогда, когда $A^\omega = B^\omega$.

(Все обозначения, связанные с ω-языками и бесконечными итерациями «обычных» языков, а также с некоторыми вариантами ω-автоматов, можно найти в [11], а также в недавней статье [12].)

Статья получена 16 августа 2023 г.

Борис Феликсович Мельников, Университет МГУ – ППИ в Шэньчжэне (bormel@smbu.edu.cn).

Мэн Линцянь, Университет МГУ – ППИ в Шэньчжэне (kostya45e@mail.ru).

Саму решаемую задачу, кратко описанную выше – *проверку условия выполнения отношение покрытия* $\trianglelefteq$ – можно рассматривать *с нескольких разных точек зрения*, что и делалось в процитированных работах. В настоящей статье нас в первую очередь интересуют различные варианты *реализации алгоритмов* для проверки выполнения этого отношения, а более конкретно – применение для таких алгоритмов конечных автоматов. Более того, *два* варианта таких автоматов были ранее описаны в [4], [5] [1] – а в настоящей статье мы к ним добавляем ещё несколько; всего, как следует из названия статьи, различных автоматов получается *восемь*.

Следующий пункт «мотивации к рассмотрению нескольких автоматов-алгоритмов» – такой. В статье [6] был описан *полиномиальный алгоритм построения* одного из таких автоматов (конкретно – автомата $\mathrm{NSPRI}(A, B)$) на основе заданных конечных языков A и B. Однако этот факт (существование такого полиномиального алгоритма) вовсе не означает, что мы можем *полиномиально проверить* выполнение необходимого нам условия $A \trianglelefteq B$ – в связи со следующим обстоятельством. Если мы таким же образом, как и в [6], строим именно автомат NSPRI, то он должен принимать любое слово языка Σ^* (именно для этого мы автомат и строим) – а это условие за полиномиальное время проверить нельзя.

В качестве комментария к последнему факту приведём с небольшими изменениями один абзац из [5]. Определить «полноту» некоторого языка [2], заданного, например, недетерминированным конечным автоматом, за полиномиальное время, вообще говоря, невозможно [3]: условие отсутствия «пустых клеток» в таблицу автомата является достаточным – но не необходимым и достаточным [4]. Даже в случае однобуквенного алфавита – что для наших проблем обычно неинтересно – подобная задача является NP-полной, см. формулировку в [13, упр. 10.14] [5]. Однако на основании приведённого комментария мы *ещё не можем утверждать, что полиномиального алгоритма проверки выполнения отношения* $\trianglelefteqslant$ *не существует*: для такого утверждения необходимы дополнительные исследования.

К последнему приведённому комментарию сто́ит также добавить, что если мы построим не автомат NSPRI, а автомат PRI (как было показано в предыдущих статьях, это делается аналогичными алгоритмами), то можно сказать, что само подобное построение, вообще говоря, неполиномиально: мы рассматриваем множество подмножеств слов (суффиксов одного из заданных языков).

Таким образом, наличие четырёх описаний недетерминированных автоматов, отвечающих на поставленный вопрос о выполнении отношения $A \trianglelefteq B$ (именно 4 автомата из 8 рассматриваемых в настоящей статье являются недетерминированными), даст бо́льшие возможности для поиска возможного алгоритма, проверяющего основное условие статьи за полиномиальное время. Например, может, как-то удастся воспользоваться тем, что в процессе построения автоматов возникают *не любые* подмножества множества суффиксов (или иногда префиксов) рассматриваемых языков.

А третий «пункт мотивации» – такой. В трёх частях статьи [7], [8], [9] была рассмотрена связь двух задач теории формальных языков: задачи извлечения корня из заданного языка и задачи построения для заданного языка оптимального инверсного морфизма. Но после последних публикаций [14], [15], [16], по-видимому, можно сказать, что теперь эти задачи «совершенно разделились»:

- при извлечении корня из языка – мы для получения результатов *не* употребляли отношение $\trianglelefteq$, хотя, конечно, в ней это отношение тоже выполняется, и, возможно, этим в будущем удастся воспользоваться;
- а при построении оптимального инверсного морфизма – мы употребляем не только это отношение, но и его различные частные случаи.

Поэтому для продолжения работы над задачами статьей [14], [15] (а возможно, и [16]) нам может понадобиться более подробное исследование отношения $\trianglelefteq$, в том числе – различных алгоритмов его построения и различных конечных автоматов, предназначенных для описания этих алгоритмов.

Приведём содержание статьи по разделам. Сразу скажем, что в статье не требуется обычный раздел «Предварительные све́дения»: основные определения и обозначения, связанные с бинарным отношением $\trianglelefteq$, уже были рассмотрены выше, а необходимые подробные обозначения, относящиеся к недетерминированным конечным автоматам, были приведены в трёх разделах недавно опубликованных статей [2], [3] (см. также [15]).

В разделе II кратко описаны все 8 вариантов рассматриваемых в статье автоматов; также вводятся обозначения, применяемые для этих автоматов далее. Эти 8 вариантов естественным образом получаются с помощью трёх различных дихотомий: одна из них – «обычная» (является ли автомат детерминированным), а две других непосредственно связаны с рассматриваемыми в статье вопросами проверки выполнения отношения $\trianglelefteq$.

Далее, в разделах III – X, мы последовательно описываем сами 8 вариантов автоматов (по одному в разделе). При этом в разделах III и IV приводятся переформулировки определений автоматов, уже встречавшихся в наших статьях ранее, а разделы начиная с V включены во вторую часть статьи. Для каждого варианта автомата приводятся примеры:

- для всех детерминированных автоматов – по два примера, с отрицательным и с положительным ответами;
- для всех недетерминированных автоматов – по одному примеру, с отрицательным ответом;

при этом все «отрицательные» примеры построены на основе пары языков, рассматривавшихся нами в предыдущих работах – и, по мнению авторов, эта пара языков даёт достаточно информативные примеры.

Раздел XI – заключение; в нём мы кратко описываем те возможные проблемы для дальнейшего решения, которые связаны с задачами, рассматриваемыми в настоящей статье.

[1] Здесь мы считаем, что описанные в этих статьях автоматы $\mathrm{NSPRI}^{\#}$ и NSPRI представляют собой один и тот же автомат.

[2] Т. е. его совпадение с языком $(\Sigma)^*$. В рассматриваемом случае это $(\Delta_A)^*$ для определённого в той статье алфавита Δ_A.

[3] А именно для этого и строится автомат NSPRI.

[4] Подтверждающий пример строится тривиально.

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

II. ВАРИАНТЫ АВТОМАТОВ: КРАТКОЕ ОПИСАНИЕ И ОБОЗНАЧЕНИЯ

В этом разделе кратко описаны все 8 вариантов рассматриваемых в статье автоматов; также вводятся обозначения, применяемые для этих автоматов далее. Эти 8 вариантов естественным образом получаются с помощью трёх различных дихотомий: одна из них – «обычная» (автомат детерминированный или недетерминированный), а две других непосредственно связаны с рассматриваемыми в статье вопросами проверки выполнения отношения $\trianglelefteqslant$.

При этом мы вводим не только обозначения для самих автоматов (и автоматы, и эти обозначения мы предполагаем применять и в дальнейших статьях) – но и специальные «двоичные номера» для настоящей статьи, причём эти номера мы применяем не только для автоматов, но и для относящихся к ним примеров.

Итак, приведём краткие описания и обозначения самих дихотомий.

1) Первая дихотомия связана с тем, рассматриваем ли мы требуемые покрытия каждого слова-итерации непосредственно при его возникновении – либо не сразу, а только после формирования некоторого вспомогательного языка. По-видимому, эти два варианта удобно называть «перегоняем» и «не доходим» соответственно. В применяемых далее обозначениях:
   - сверху над названием автомата символы $\rightarrow|$ означают, что мы «не доходим» (так и было в «старом» варианте автомата PRI, [4] и последующие публикации); двоичное обозначение *внутри настоящей статьи* – 0 в первой позиции [6];
   - сверху символы $|\rightarrow$ означают, что мы «перегоняем»; обозначение *внутри этой статьи* – 1 в первой позиции.
2) Вторая дихотомия связана с тем, какой именно из двух рассматриваемых конечных языков является «главным» (именно для этого мы применяем основной морфизм для проверки рассматриваемого отношения покрытия). В применяемых далее обозначениях:
   - снизу под названием автомата символ $\uparrow$ означает, что основным языком является «покрываемый» A (ранее всегда мы рассматривали только такие варианты) [7]; слова «главного» языка являются морфическими образами букв вспомогательного алфавита Δ (часто в наших обозначениях – Δ_A или Δ_B), эти буквы чвляются алфавитом строимого автомата, в таблицах автоматов они пишутся сверху; обозначение такого варианта дихотомии *внутри этой статьи* – 0 во второй позиции;
   - снизу под названием автомата символ $\downarrow$ означает, что основным языком является «покрывающий» B; обозначение *внутри этой статьи* – 1 во второй позиции.
3) Третья дихотомия связана с детерминированностью строимого автомата – т. е., «в старых обозначениях» ([4], [5]), PRI либо NSPRI. Точнее – поскольку в частных случаях автомат NSPRI может получаться детерминированным – с детерминированностью *алгоритма*, применяемого при построении. В детерминированном случае алгоритмы работают с множествами префиксов (суффиксов) нужного языка – а в недетерминированном с каждым словом этого языка по отдельности. В применяемых далее обозначениях:
   - детерминированный алгоритм (автомат) будет, как и ранее, называться PRI – но теперь с дополнительными верхними и нижними индексами, о которых уже было сказано в предыдущих пунктах; обозначение *внутри этой статьи* – 0 в третьей позиции;
   - аналогично, детерминированный алгоритм (автомат) будет, как и ранее, называться NSPRI; обозначение *внутри этой статьи* – 1 в третьей позиции.
4) Эта дихотомия связана не с автоматами (алгоритмами), а с двумя группами рассматриваемых далее примеров. А именно:
   - старый пример (постоянно нами рассматриваемый в разных статьях),
   $$\begin{aligned} A &= \{\,aaa, aabba, abba, bb\,\},\\ B &= \{\,aaaa, abb, abba, bbb\,\}; \end{aligned} \tag{2}$$
   как мы знаем, алгоритм (любой автомат) должен для этого примера дать отрицательный ответ; обозначение *внутри этой статьи* – 0 в четвёртой позиции (если таковая имеется);
   - новый пример,
   $$\begin{aligned} A &= \{\,ab, abaab, abaabab, abaaba\,\},\\ B &= \{\,abab, ababababaaba,\\ &\qquad abaaba, aba, abaab\,\}; \end{aligned} \tag{3}$$
   алгоритм (любой автомат) должен для этого примера дать положительный ответ; обозначение *внутри этой статьи* – 1 в четвёртой позиции.

   В обоих случаях «основной» алфавит $\Sigma = \{\,a, b\,\}$.

Ещё раз специально отметим, что четвёртая позиции двоичной записи при обозначениях автоматов – а не конкретных примеров с ними – не употребляется. И, конечно, все 8 вариантов автоматов для одного и того же рассматриваемого примера должны давать одинаковый результат.

Теперь в качестве примеров приведём два варианта *использования* наших обозначений.

- 000, или $\mathrm{PRI}^{\rightarrow|}_{\uparrow}(A,B)$, – это детерминированный автомат, проверяющий условие $A \trianglelefteqslant B$ для некоторых языков A и B. При этом проверка осуществляется для «основного» языка A путём описания для каждого слова $u \in A^*$ всех возможных слов языка $v \in B^*$, являющихся префиксами u, причём таких, что для каждого такого v существует некоторое $w \in B$, такое, что vw не является префиксом u. Все выбранные слова (w в этих обозначениях)

[6] Всего таких позиций – 3 когда мы рассматриваем *определения* автоматов, либо 4 когда мы рассматриваем *конкретные примеры* автоматов.

[7] Специально отметим, что язык A *всегда* является «покрываемым», а язык B «покрывающим»; неявно об этом уже было сказано ранее.

рассматриваются *как специальный язык* (а не по отдельности) – именно поэтому автомат является детерминированным. Фактически мы таким образом описываем автомат, определявшийся в предыдущих статьях как $\mathrm{PRI}(A, B)$.

- 1111 – это «внутреннее обозначение нашей статьи» конкретного примера недетерминированного автомата $\mathrm{NSPRI}^{\rightarrow|}_{\downarrow}(A, B)$, в котором основной язык – «покрывающий» B, причём A и B заданы согласно (3). Проверка осуществляется путём описания для каждого слова $u \in B^*$ всех возможных слов $v \in A^*$, таких что $uw = v$ для некоторого слова w (именно поэтому мы выше использовали термин «перегоняем»). Все выбранные слова (w в этих обозначениях) рассматриваются *по отдельности* (а не как специальный язык) – именно поэтому автомат является недетерминированным.

В обоих приведённых примерах было применено слово «описание»; конечно, каждый описываемый таким образом язык конечен – но число подобных описаний бесконечно, и именно поэтому мы применяем для этих описаний конечные автоматы; некоторые подробности см. в предыдущих работах, прежде всего в [4], [5].

В следующих 8 разделах (6 из них мы помещаем в часть II статьи), как уже было сказано выше, мы будем рассматривать строгие определения самих автоматов. При этом для близких по смыслу функций переходов мы будем использовать *одинаковые* обозначения Φ; понятно, что каждое такое обозначение «действует только внутри одного раздела».

Заранее отметим очевидность того, что каждый описываемый далее автомат действительно выполняет действия, необходимые для проверки рассматриваемого нами условия $A \trianglelefteqslant B$. Для одного из вариантов строгое доказательство этого факта было приведено в [2], [3] – но, повторим, корректность проводимых построений, по-видимому, очевидна и без такого доказательства, а просто на основе описаний автоматов, приведённых в начале каждого следующего раздела [8].

III. (1) Автомат 000, т. е. $\mathrm{PRI}^{\rightarrow|}_{\uparrow}$

Как уже было отмечено, 000, или $\mathrm{PRI}^{\rightarrow|}_{\uparrow}(A, B)$, – это детерминированный автомат, проверяющий условие $A \trianglelefteqslant B$ для некоторых языков A и B. В таком автомате проверка осуществляется для «основного» языка A путём описания для каждого слова $u \in A^*$ всех возможных слов языка $v \in B^*$:

- являющихся префиксами u,
- причём таких, что для каждого такого v существует некоторое $w \in B$, такое, что vw не является префиксом u [9].

Все выбранные слова (w в этих обозначениях) рассматриваются *как специальный язык* (а не по отдельности) – именно поэтому автомат является детерминированным.

Результат работы автомата-алгоритма определяется следующим образом:

- если из какого-либо его *достижимого* состояния имеется переход в состояние, обозначаемое $\varnothing$, то это означает, что имеется какое-то слово языка A^*, для которого нет возможности построить соответствующее *покрывающее его* слово языка B^*; очевидно, что в этом случае алгоритм должен дать отрицательный ответ;
- в противном случае – считаем, что алгоритм даёт положительный ответ [10].

Фактически мы таким образом неформально описали автомат, обозначавшийся в предыдущих статьях более простым способом: $\mathrm{PRI}(A, B)$, без каких-либо индексов.

Формальное описание автомата

Итак, в старых обозначениях описываемый автомат назывался $\mathrm{PRI}(A, B)$. Повторим поэтапное описание его определения согласно [4] – исключив некоторые комментарии той статьи [11]. Однако некоторые новые пояснения приведены на рис. 1 [12] – и, как надеются авторы, приведённые на нём пояснения на английском языке понятны без специального перевода. Рисунок относится как к рассматриваемому здесь детерминированному автомату, так и к приведённому в следующем разделе автомату недетерминированному, реализующему тот же самый алгоритм.

Во-первых, для пары непустых конечных языков $A, B \subseteq \Sigma^*$, таких что $A \not\ni \varepsilon$ и $B \not\ni \varepsilon$, язык

$$\underset{A-B}{\Phi}(u) \subseteq \Sigma^*$$

строится для некоторого слова $u \in \Sigma^*$ по следующему алгоритму – в нём используется вспомогательный язык D' и формируется язык-ответ D.

1) Для начала работы алгоритма полагаем
$$D' = \{u\} \cdot A, \quad D = \varnothing.$$
2) Если $D' = \varnothing$, то выход из алгоритма с ответом D.
3) Выбираем некоторое слово $v \in D'$, исключая его из D'.
4) Для выбранного v рассматриваем все варианты его представления в виде $v = uv'$, где $u \in B^+$,
$$D +: v'$$
(т. е. включаем каждое такое v' в язык D).

[8] Можно сказать, что этот абзац относится к каждому из последующих разделов – его можно рассматривать как одинаковое *завершение* каждого из них.

[9] В предыдущих статьях мы в некоторых интерпретациях этого автомата-алгоритма дополнительно добавляли следующее условие:

- такое слово vw должо быть префиксом некоторого (другого, «более длинного») слова из A^*.

Понятно, что добавление этого дополнительного условия непринципиально.

[10] В очередной раз (см. по этому поводу наши предыдущие публикации) отметим разницу между:

- *состоянием*, обозначаемым нами $\varnothing$ (это символизирует пустой язык) –
- и также обозначаемым с помощью $\varnothing$ *множеством состояний*.

Последнее множество может использоваться, например, в качестве множества входных или выходных состояний некоторого недетерминированного автомата.

[11] Отметим, что комментарии достаточно важные (в частности, они поясняют *два* разных варианта применения пустого множества $\varnothing$) – однако формально определения можно понять и без них.

[12] В части II аналогичные рисунки будут приведены и для других вариантов алгоритма.

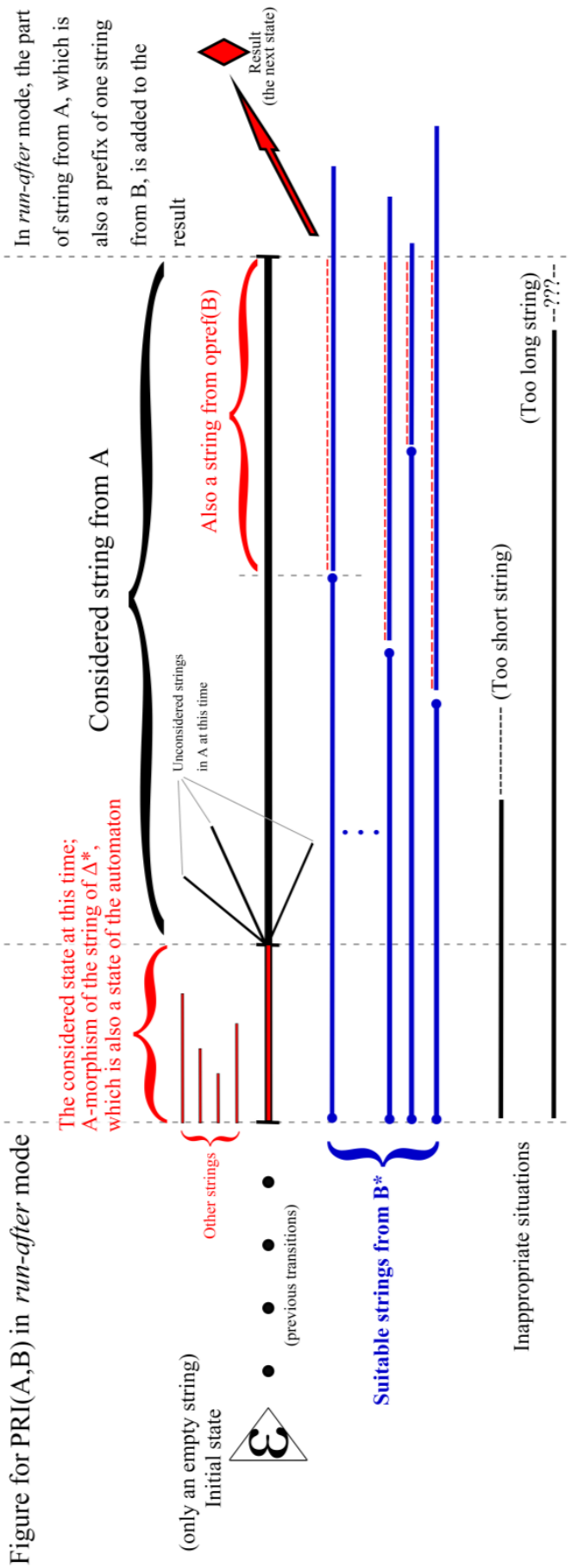

Рис. 1. Иллюстрация алгортма, соответствующего автоматам, обозначенным в статье 000 и 001. (Run-after mode, основной язык A.)

5) Если
$$(\exists w \in B)\,(v \in \operatorname{opref}(w)),$$
то $D \mathrel{+}= v$ (т. е. включаем v в язык D).
6) Переходим на п. 2.

Во-вторых, для языка $C \subseteq \Sigma^*$ определяем
$$\underset{A-B}{\Phi}(C) = \bigcup_{u \in C} \underset{A-B}{\Phi}(u).$$

В-третьих, детерминированный конечный автомат $\overline{\mathrm{PRI}}(A, B)$ для заданных конечных языков $A, B \subseteq \Sigma^*$ определяется следующим образом:
$$\overline{\mathrm{PRI}}(A, B) = \left(\overline{Q}_B, \Delta_A, \delta_{A-B}, \{\{\varepsilon\}\}, \{\varnothing\}\right),$$
где:
- $\Delta_A = \{0, 1, \ldots, n-1\}$, здесь n – число слов языка A;
- $\overline{Q}_B = \mathcal{P}(\operatorname{opref}(B))$;
- для некоторых $q_1, q_2 \in \overline{Q}_B$ и $a \in \Delta_A$ полагаем
$$q_1 \xrightarrow[\delta_{A-B}]{a} q_2$$
тогда и только тогда, когда
$$\underset{h_A(a)-B}{\Phi}(q_1) = q_2\,.$$

В-четвёртых, автомат, получающийся после удаления недостижимых состояний из автомата $\overline{\mathrm{PRI}}(A, B)$, будем обозначать
$$\mathrm{PRI}(A, B) = \left(Q_B, \Delta_A, \delta_{A-B}, \{\{\varepsilon\}\}, \{\varnothing\}\right),$$
где $Q_B \subseteq \mathcal{P}(\operatorname{opref}(B))$.

В новых обозначениях это $\mathrm{PRI}\overset{\rightarrow|}{\uparrow}(A, B)$.

Пример автомата, дающего отрицательный ответ

Пример автомата, построенного согласно описанному здесь алгоритму для упорядоченной пары языков (2), приведён на рис. 2. Как можно видеть, приведённый автомат – с точностью до указания входного и выходного состояний – не отличается от того, который мы в [4] назвали автоматом $\mathrm{PRI}(A, B)$ для тех же самых языков A и B.

Пример автомата, дающего положительный ответ

Пример автомата, построенного согласно описанному здесь алгоритму для упорядоченной пары языков (3), приведён на рис. 3: дополнительно добавленное состояние $\varnothing$ в этом автомате является недостижимым.

IV. (2) Автомат 001, т. е. $\mathrm{NSPRI}\overset{\rightarrow|}{\uparrow}$

Как уже было отмечено, 001, или $\mathrm{NSPRI}\overset{\rightarrow|}{\uparrow}(A, B)$, – это недетерминированный автомат, проверяющий условие $A \trianglelefteqslant B$ для некоторых языков A и B. Описание алгоритма похоже на приведённое в предыдущем разделе описание соответствующего детерминированного алгоритма – приведём изменённый вариант.

Аналогично автомату $\mathrm{PRI}\overset{\rightarrow|}{\uparrow}(A, B)$, необходимая проверка осуществляется для «основного» языка A – путём описания для каждого слова $u \in A^*$ всех возможных слов языка $v \in B^*$:
- являющихся префиксами u,
- причём таких, что для каждого такого v существует некоторое $w \in B$, такое, что vw не является префиксом u.

Все выбранные слова (w в этих обозначениях) рассматриваются *по отдельности* (а не как специальный язык) – именно поэтому автомат является недетерминированным.

Результат работы автомата-алгоритма определяется следующим образом:
- если из какого-либо его *достижимого* состояния имеется переход в состояние, обозначаемое $\varnothing$, то это означает, что имеется какое-то слово языка A^*, для которого нет возможности построить соответствующее *покрывающее его* слово языка B^*; очевидно, что в этом случае алгоритм должен дать отрицательный ответ;
- в противном случае – считаем, что алгоритм даёт положительный ответ.

Фактически мы таким образом неформально описали автомат, обозначавшийся в предыдущих статьях более простым способом: $\mathrm{NSPRI}(A, B)$, без каких-либо индексов.

Формальное описание автомата

Итак, в старых обозначениях описываемый автомат обозначался $\mathrm{NSPRI}(A, B)$. Мы также повторим поэтапное описание его определения согласно [5], исключив некоторые комментарии.

Сначала для заданных конечных языков $A, B \subseteq \Sigma^*$ определим (недетерминированный) конечный автомат $\mathrm{NSPRI}^{\#}(A, B)$ – следующим образом:
$$\mathrm{NSPRI}^{\#}(A, B) = (\, Q_B^{\#}, \Delta_A, \delta_{A-B}^{\#}, \{\varepsilon\}, \varnothing\,),$$
где:
- $Q_B^{\#} = \operatorname{opref}(B) \cup \{\varnothing\}$;
- 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
$$\underset{\{h_A(d)\}-B}{\Phi}(q_1) \ni q_2\,.$$

(В этом разделе функция Φ и функции переходов автоматов для всех вариантов их использования совпадают с аналогичными функциями из предыдущего раздела.)

Как мы показали в [5], язык автомата $\mathrm{NSPRI}^{\#}(A, B)$ совпадает с языком автомата $\mathrm{PRI}(A, B)$.

Для заданных конечных языков $A, B \subseteq \Sigma^*$ (недетерминированный) конечный автомат $\mathrm{NSPRI}(A, B)$ определяется следующим образом:
$$\mathrm{NSPRI}(A, B) = (\, \operatorname{opref}(B), \Delta_A, \delta_{A-B}^{\#}, \{\varepsilon\}, \operatorname{opref}(B)\,),$$
где функция переходов $\delta_{A-B}^{\#}$ совпадает с введённой в этом разделе. В новых обозначениях это $\mathrm{NSPRI}\overset{\rightarrow|}{\uparrow}(A, B)$.

Также в [5] мы показали, что для того, чтобы алгоритм давал положительный ответ, такой автомат должен определять полный язык над алфавитом Δ_A (т. е. язык $(\Delta_A)^*$).

	automaton 0000	aaa	aabba	abba	bb
0	Ø	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

Рис. 2. Автомат, обозначенный в статье 0000.

	automaton 0001	ab	abaab	abaaba	abaabab
0	Ø	0	0	0	0
1	{ε}	2	3	4	5
2	{ab}	6	2	7	6
3	{ε,ab}	5	3	4	5
4	{ε,a,aba}	3	3	4	5
5	{ε,ab,abab}	8	3	4	5
6	{ε,abab}	9	3	4	5
7	{ε,aba}	3	3	4	5
8	{ε,ab,abab,ababab}	10	3	4	5
9	{ab,ababab}	11	2	7	6
10	{ε,ab,abab,ababab, abababab}	12	3	4	5
11	{ε,abab,abababab}	13	3	4	5
12	{ε,ab,abab,ababab, abababab,ababababab}	12	3	4	5
13	{ab,ababab,abababababab}	11	2	7	6

Рис. 3. Автомат, обозначенный в статье 0001.

	automaton 0010	aaa	aabba	abba	bb
0	Ø	0	0	0	0
1	ε	2	0	1 9	4
2	aaa	5	1 9	0	0
4	bb	0	0	0	8
5	aa	9	0	0	0
8	b	0	0	0	1
9	a	1	0	0	1 16
16	abb	2 5	1 9	1 9	4

Рис. 4. Автомат, обозначенный в статье 0010.

Пример автомата, дающего отрицательный ответ

Пример автомата, построенного согласно описанному здесь алгоритму для упорядоченной пары языков (2), приведён на рис. 4.

Как можно видеть, приведённый автомат – с точностью до переобозначения состояний – не отличается от того, который мы в [4], [5] назвали автоматом NSPRI. Переобозначения состояний такое:

$$4 \longleftrightarrow 7, \quad 5 \longleftrightarrow 2, \quad 9 \longleftrightarrow 4, \quad 16 \longleftrightarrow 6;$$

здесь слева от каждой двойной стрелки – состояния «нового» автомата, справа – соответствующие им состояния «старого» автомата, остальные номера не изменились. Разница возникла из-за того, что номера состояниям мы давали в процессе их появления, точнее – появления равных им слов в подмножествах состояний автомата PRI, а алгоритмы рассмотрения последовательностей состояний в последнем автомате, применёные в этих двух статьях, различаются.

БЛАГОДАРНОСТИ

Настоящая работа была частично поддержана грантом научной программы китайских университетов “Higher Education Stability Support Program” (раздел “Shenzhen 2022 – Science, Technology and Innovation Commission of Shenzhen Municipality”) – 深圳市 2022 年高等院校稳定支持计划资助项目.

Список литературы

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

[2] Мельников Б.Ф., Мельникова А.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.

[3] Мельников Б.Ф., Мельникова А.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.

[4] Мельников Б.Ф. Варианты конечных автоматов, соответствующих бесконечным итерационным деревьям морфизмов. Часть I // International Journal of Open Information Technologies. 2021. Vol. 9, No. 7. P. 5–13.

[5] Мельников Б.Ф. Варианты конечных автоматов, соответствующих бесконечным итерационным деревьям морфизмов. Часть II // International Journal of Open Information Technologies. 2021. Vol. 9, No. 10. P. 1–8.

[6] Мельников Б.Ф., Мельникова А.А. Полиномиальный алгоритм построения конечного автомата для проверки равенства бесконечных итераций двух конечных языков // International Journal of Open Information Technologies. 2021. Vol. 9, No. 11. P. 1–10.

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

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

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

[10] Мэн Линцянь, Мельников Б.Ф. О группоиде первичного автомата PRI, определяемом для одного конечного языка // International Journal of Open Information Technologies. 2023. Vol. 11, No. 9. P. 1–12.

[11] Melnikov B.F. The equality condition for infinite catenations of two sets of finite words // International Journal of of Foundations of Computer Science. 1993. Vol. 4, No. 3. P. 267–273.

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

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

[14] Мельников Б.Ф. О комплексе задач исследования необходимых условий равенства бесконечных итераций конечных языков // International Journal of Open Information Technologies. 2023. Vol. 11, No. 1. P. 1–12.

[15] Мельников Б.Ф., Мельникова А.А. Применение лепестковых конечных автоматов для проверки выполнения частного случая гипотезы Zyu (для заданного конечного языка) // International Journal of Open Information Technologies. 2023. Vol. 11, No. 3. P. 1–11.

[16] Мельников Б.Ф., Мельникова А.А. О задачах извлечения корня из заданного конечного языка // International Journal of Open Information Technologies. 2023. Vol. 11, No. 5. P. 1–14.

Борис Феликсович МЕЛЬНИКОВ,
профессор Университета МГУ – ППИ в Шэньчжэне
(http://szmsubit.ru/),
email$_1$: bormel@smbu.edu.cn,
email$_2$: bf-melnikov@yandex.ru,
mathnet.ru: personid=27967,
elibrary.ru: authorid=15715,
scopus.com: authorId=55954040300,
ORCID: orcidID=0000-0002-6765-6800.

МЭН Линцянь,
студент-магистрант МГУ (https://www.msu.ru/),
email$_1$: 1120190010@smbu.edu.cn,
email$_2$: kostya45e@mail.ru,
ORCID: orcidID=0009-0009-2933-5684.

Eight variants of finite automata for checking the fulfillment of the coverage relation of iterations of two finite languages. Part I

Boris Melnikov, Meng Lingqian

***Abstract*—The task for which all automata of this paper are considered is to describe algorithms for checking the so-called binary coverage relation, which was considered in many of our previous papers: it is performed for two nonempty finite languages if and only if any iteration of words of the first language is a prefix of some iteration of words of the second language. This task can be viewed from several different points of view, which was done in previous works.**

In this paper, we are primarily interested in various variants for implementing algorithms to verify the fulfillment of this relationship, and more specifically, the use of finite automata for such algorithms. 2 variants of such automata have already been previously described in our previous works, and in this paper we add 6 more variants to them.

The main purpose of considering several automata-algorithms is as follows. Earlier, we described a polynomial algorithm for constructing one of these automata. However, the existence of such a polynomial algorithm does not mean that we can check the fulfillment of the condition we need in a polynomial way, due to the following circumstance. If we build a nondeterministic automaton to answer the question in the same way as we described earlier, then for a positive answer. it must accept any word of the universal language; the last condition cannot be checked in polynomial time. However, we also cannot say that such a polynomial algorithm does not exist: additional research is needed. Therefore, the presence of four descriptions of nondeterministic automata that answer the question about the fulfillment of the considered binary relation will give some new opportunities to search for a possible algorithm that checks the main condition of the paper in polynomial time.

8 variants of the automata considered in the paper are naturally obtained using three different dichotomies: one of them is the "usual" one (whether the automaton is deterministic), and the other two dichotomies are directly related to the issues of checking the implementation of the binary coverage relation considered in the paper. Specifically, these two dichotomies are as follows: first, which of the two finite languages under consideration is the "main" one (it is that for which we apply the basic morphism to check); secondly, do we consider the required covers of each iteration word directly when it occurs, or not immediately, but after the formation of some auxiliary language.

In the presented first part of this paper, we briefly describe all the necessary dichotomies, give the associated auxiliary designations, and also give new versions of the formulations of the two automata considered in previous papers.

***Keywords*—formal languages, iterations of languages, morphisms, finite automata, algorithms.**

References

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

[2] B Melnikov, A Melnikova. *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 (in Russian).

[3] B Melnikov, A Melnikova. *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 (in Russian).

[4] B Melnikov. *Variants of finite automata corresponding to infinite iterative morphism trees. Part I*, International Journal of Open Information Technologies, 2021, 9(7): 5–13 (in Russian).

[5] B Melnikov. *Variants of finite automata corresponding to infinite iterative morphism trees. Part II*, International Journal of Open Information Technologies, 2021, 9(10): 1–8 (in Russian).

[6] B Melnikov, A Melnikova. *A polynomial algorithm for constructing a finite automaton to check the equality of infinite iterations of two finite languages*, International Journal of Open Information Technologies, 2021, 9(11): 1–10 (in Russian).

[7 B Melnikov. *Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part I. Extracting the root from the language*, International Journal of Open Information Technologies, 2022, 10(4): 1–9 (in Russian).

[8] B Melnikov. *Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part II. Constructing an inverse morphism*, International Journal of Open Information Technologies, 2022, 10(5): 1–8 (in Russian).

[9] B Melnikov. *Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part III. The condition for the existence of a lattice*, International Journal of Open Information Technologies, 2022, 10(7): 1–9 (in Russian).

[10] Meng Lingqian, Melnikov B.F. About the groupoid of the primary automaton PRI defined for one finite language // International Journal of Open Information Technologies. 2023. Vol. 11, No. 9. P. 1–12 (in Russian).

[11] Melnikov B.F. The equality condition for infinite catenations of two sets of finite words // International Journal of of Foundations of Computer Science. 1993. Vol. 4, No. 3. P. 267–273.

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

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

[14] B Melnikov. *On the complex of problems for the study of the necessary conditions for the equality of infinite iterations of finite languages*, International Journal of Open Information Technologies, 2023, 11(1): 1–12 (in Russian).

[15] Melnikov B.F., Melnikova A.A. Application of petal finite automata to verify the fulfillment of a special case of the Zyu hypothesis (for a given finite language) // International Journal of Open Information Technologies. 2023, 11(3): 1–11 (in Russian).

[16] Melnikov B.F., Melnikova A.A. On the problems of extracting the root from a given finite language // International Journal of Open Information Technologies. 2023. Vol. 11, No. 5. P. 1–14 (in Russian).

Boris MELNIKOV,
Professor of Shenzhen MSU – BIT University, China
(http://szmsubit.ru/),
email: bf-melnikov@yandex.ru.

MENG Lingqian,
Master student of Moscow State University, Russia
(https://www.msu.ru/),
email: kostya45e@mail.ru.