Аппаратная конструкция для решения проблемы экспоненциального взрыва для одного класса регулярных языков

Бернадотт А., Галатенко А.

Аннотация

Известно, что язык, задаваемый регулярным выражением вида $\bigcup_{i=1}^{n} .*\alpha_i.*\beta_i.*$, где α_i, β_i — слова над некоторым алфавитом, в общем случае для распознавания конечным детерминированным автоматом требует экспоненциальное по n число состояний. В работе предлагается конструкция структурного автомата, распознающего языки из данного класса и имеющего полиномиальную пространственную сложность.

Ключевые слова: ДКА, структурный автомат, регулярный язык, экспоненциальный взрыв.

1. Введение

В настоящее время сетевые устройства могут проверять содержимое пакета с целью управления сетевым трафиком и сканирования на вредоносность.

В анализе сетевого трафика актуальной является скорость анализа, так как последний может значительно снижать пропускную способность сетевого узла. По скорости сканирования и по уровневой модели реализации можно выделить два основных класса анализаторов: 1) сетевые процессоры (англ. Network Processing Unit, NPU) — это программируемые микропроцессоры, оптимизированные для работы в сетевых устройствах в режиме обработки пакетов (packet processing); 2) интегральные схемы специального назначения, (англ. Application-Specific Integrated Circuit, ASIC) — это интегральные схемы, обладающие ограниченным функционалом, реализуемым под строго заданную задачу. Тогда как сетевой процессор является программируемым устройством (в данной статье устройство такого вида не рассматривается), которое обеспечивает высокий

уровень пластичности, ASIC обеспечивает высокую производительность, но не обеспечивает способности к репрограммируемости.

Оба типа устройства могут иметь интегрированные модули внешней и встроенной памяти. Количество встроенных модулей памяти с достаточной пропускной способностью ("быстрой памяти") ограничено значительно, а количество внешних элементов памяти с низкой пропускной способностью ограничено только размером устройства. Критичной является проблема объема и доступности встроенной памяти.

Один из основных методов анализа заключается в следующем. Эксперты задают сигнатуры злоумышленного поведения; анализатор просматривает сетевые пакеты и выявляет соответствие пакетов хотя бы одной сигнатуре. Для задания сигнатур используется аппарат регулярных выражений. По теореме Клини задача проверки принадлежности слова соответствующему регулярному языку может быть решена с помощью конечого детерминированного автомата с временной сложностью, линейной от длины слова. Однако число состояний автомата может расти экспоненциально с ростом длины регулярного выражения, то есть возможен экспоненциальный взрыв пространственной сложности. В результате возникает задача понижения пространственной сложности распознавания без существенного ухудшения временной сложности. Некоторые известные подходы к решению излагаются ниже.

Мы рассматриваем класс регулярных выражений вида $\bigcup_{i=1}^{n} .*\alpha_i.*\beta_i.*$, где α_i, β_i — слова в некотором алфавите, $n \in \mathbb{N}$. С одной стороны, в общем случае распознавание принадлежности слова такому языку требует экспоненциального по n числа состояний; с другой стороны, сигнатуры вида $.*\alpha.*\beta.*$ часто встречаются в практических приложениях. Для уменьшения пространственной сложности предлагается разделить автомат-распознаватель на два подавтомата — абстрактный, в котором функции переходов и выходов заданы таблично, и структурный. Доказывается, что возникающая конструкция корректно решает задачу распознавания и позволяет избежать экспоненциального взрыва.

Дальнейшее изложение построено следующим образом. Во втором разделе приводятся основные определения и дается краткий обзор по проблеме экспоненциального взрыва. В третьем разделе описывается предлагаемая конструкция. В четвертом разделе формулируются и доказываются основные утверждения, в пятом рассматривается несколько примеров. Наконец, шестой раздел является заключением.

2. Описание проблемы и основные определения

Напомним основные определения в соответсвии с источниками [1, 2].

Регулярный язык (событие или множество) над алфавитом A задаётся рекурсивно:

1) $\emptyset, \{\lambda\}$ и $\{a\}$ — регулярные языки, где λ — пустое слово, $a \in A$;

2) если M_1 и M_2 — регулярные языки, то объединения $M_1 \cup M_2$ и конкатенация $M_1 \cdot M_2$ языков M_1 и M_2, а также здезда Клини M_1^* — регулярные языки, где

$$M_1 \cup M_2 = \{\alpha_1 \cup \alpha_2 | \alpha_1 \in M_1, \alpha_2 \in M_2\},$$
$$M_1 \cdot M_2 = \{\alpha_1 \cdot \alpha_2 | \alpha_1 \in M_1, \alpha_2 \in M_2\},$$
$$M_1^* = \{\lambda\} \cup \{\alpha_1 \cdot \ldots \alpha_n | \alpha_i \in M_1, i \in \overline{1,n}, n \in N\}.$$

Регулярное выражение — язык над алфавитом $A \cup \{\cup, (,), \cdot, *\}$, задающий регулярное событие и определяемый как:

1) $\emptyset, \lambda, a$ — регулярные выражения, где $a \in A$;

2) если R_1, R_2 — регулярные выражения, то слова $(R_1 \cup R_2), (R_1 \cdot R_2), (R_1)*$ — регулярные выражения.

В дальнейшем мы будем предполагать, что операция $*$ имеет максимальный приоритет, а операция $\cup$ — минимальный, и не писать скобки, если результат определен однозначно. Кроме того, символ $\cdot$ будет иногда опускаться.

Регулярный язык, задаваемый регулярным выражением, определяется естественным образом.

На практике, в точности, в базах Snort, Zeek, Cisco, регулярные выражения представляют с использованием нотации PCRE (Perl Compatible Regular Expressions) [3]. Приведем PCRE —обозначения, которые будем использовать в работе:

“.” — произвольный символ из алфавита;

$R\{n\}$ — степень n языка R относительно конкатенации.

Проверка принадлежности слова регулярному языку, в классическом варианте, осуществляется принимающим абстрактным конечным автоматом (КА).

Абстрактным конечным автоматом называется набор $V = (A, Q, B, \varphi, \psi)$, где A, Q, B — конечные множества: входной алфавит, алфавит состояний и выходной алфавит, φ — функция переходов, $\varphi : Q \times A \to Q$, ψ — функция выходов, $\psi : Q \times A \to B$. Инициальный конечный автомат имеет отдельно выделенное начальное состояние, в котором начинается работа автомата: $V = (A, Q, B, \varphi, \psi, q_0)$, где q_0 — начальное состояние автомата [1]. Недетерминированный конечный автомат (НДКА) позволяет осуществлять переход по пустому слову и представляет собой набор $V = (A, Q, B, \varphi, \psi)$, где A, Q, B — конечные множества: входной алфавит, алфавит состояний и выходной алфавит, φ — функция переходов, $\varphi : Q \times A \cup \lambda \to 2^Q$, ψ — функция выходов, $\psi : Q \times A \cup \lambda \to B$. Детерминированный конечный автомат (ДКА) не имеет λ–переходов — переходов по пустому слову, и все переходы ДКА определены однозначно [1, 4].

Автомат может использоваться для представления языков: входное слово α считается принадлежащим некоторому языку L если и только если последний выданный автоматом символ лежит в принимающем множестве $B_0 \subset B$ (распознавание выходами) или состояние, в которое переводит автомат слово α, содержит элемент принимающего множества $Q_0 \subset Q$ (распознавание состояниями). Согласно теореме Клини, событие над алфавитом A представимо конечным автоматом тогда и только, когда является регулярным языком [1, 2].

КА, представляющий регулярный язык, может быть недетерминированным и детерминированным. Согласно теореме о эквивалентности детерминированных и недетерминированных конечных автоматов, всякий язык принимается некоторым НДКА тогда и только тогда, когда этот язык принимается некоторым ДКА [4]. Кроме того, существует алгоритм построения из НДКА эквивалентного ему ДКА [4, 5].

Основное преимущество недетерминированного автомата — низкая пространственная сложность, линейная по длине регулярного выражения; основной недостаток — высокая временная сложность, так как время обработки одного символа входного слова вообще говоря также линейно по длине регулярного выражения.

При переходе от недетерминированного автомата к детерминированному временная сложность обработки входного символа становится константной; с другой стороны, согласно теореме Лупанова, в случае, если входной алфавит содержит хотя бы два символа, мощность множества состояний вообще говоря экспоненциально возрастает [1, 6]. В качестве примера рассмотрим класс регулярных выражений $\bigcup_{i=1}^{n} . * \alpha_i . * \beta_i . *$, где

α_i, β_i — некоторые слова в алфавите A. Известно, что в общем случае для распознавание принадлежости слова соответствующему языку требуется экспоненциальное по n число состояний [7]. В базу сигнатур системы выявления вторжений Snort на 2019 год содержится 3936 выражений вида $.*\alpha.*\beta.*$ [8]; автомат, распознающий язык, порожденный такими выражениями, заведомо не помещается в память. Проблема экспоненциально растущего числа состояний получила название экспоненциального взрыва [1, 9, 6].

Проблемой экспоненциального взрыва занимаются более 20 лет. Глобально можно выделить выделить три подхода к её решению. Первый подход связан с ограничением на сигнатуры, задаваемые экспертами. Однако приведенный выше пример показывает, что даже очень простые, заведомо линейные по пространственной сложности регулярные выражения при объединении могут давать экспоненциальный рост.

Второй подход основан на изменении распознаваемого языка с целью упрощения распознающего автомата. При этом измененный язык отличается от исходного, поэтому возникают ошибки распознавания. В качестве примера можно привести предложенную Александровым идею оптимизации языков вида $\bigcup_{i=1}^{n} .*R_i.*R_i^{'}.*$ за счет замены пары слагаемых на слагаемое $(R_{i_1}|R_{i_2}).*(R_{i_1}^{'}|R_{i_2}^{'})$. Оценка выигрыша от такой замены приведена в работах [10, 11], оценка погрешности — в работе [12].

Третий подход заключается в модификации структуры конечного автмата. Примерами модификаций являются рассмотрение нескольких автоматов, работающий параллельно, сжатие диаграммы Мура за счет добавления переходов по умолчанию, добавлению просто устроенных недетерминированных или структурных компонет. Обзор модификаций приведен, например, в работе Александрова [7]. Отдельно отметим конструкцию, предложенную Кумаром и заключающуюся в добавлении к абстрактному автомату счетчиков и битовых масок и навешивании дополнительных логических выражений на ребра диаграммы перехода [9]. Такое решение позволяет решать задачу распознавания выражений вида $\bigcup_{i=1}^{n} .*\alpha_i.*\beta_i.*$, избегая экспоненциального взрыва пространственной сложности.

В данной работе предлагается решение, которое можно отнести к третьему типу. Автомат декомпозируется на две компоненты — “абстрактную” (задаваемую таблично; на практике таблицы значений записываются в память) и “структурную” (конфигурируемую за счет начального состояния задержек). В отличие от конструкции Кумара, используется немодифицированный автомат; основной выигрыш получается за счет

перехода от чисто абстрактного автомата к структурному. В дальнейшем предполагается провести накопление структурных компонент, позволяющих решить проблему экспоненциального взрыва для максимально широкого класса языков.

3. Аппаратная конструкция КА в контексте проблемы экспоненциального взрыва

Фиксируем конечный непустой входной алфавит A и рассмотрим класс языков $\bigcup_{i=1}^{n} . * \alpha_i . * \beta_i .*$, где α_i, β_i — непустые слова в алфавите A. Известно, что в общем случае для представления языка из данного класса требуется экспоненциальное по n число состояний конечного детерминированного автомата [7]. Ниже описывается конструкция, позволяющая избежать экспоненциального взрыва пространственной сложности за счет декомпозиции распознающего автомата на две компоненты — "абстрактную", функционирование которой задается таблично, а сложность определяется как объем памяти, требующейся для хранения таблицы значений, и "структурную", сложность которой определяется как число элементов в схеме. Заметим, что, с одной стороны, уже тривиальная схема из k элементов задержки задает автомат с 2^k состояниями; с другой стороны, почти все булевы функции от k переменных имеют экспоненциальную схемную сложность.

Сперва опишем идею конструкции на содержательном уровне. Общая схема представлена на рис. 1. Автомат состоит из трех блоков. Первый блок с помощью выходов распознает язык $L_1 \cup L_2$, где $L_1 = \bigcup_{i=1}^{n} . * \alpha_i$, $L_2 = \bigcup_{i=1}^{n} . * \beta_i$. Выходной алфавит представляет из себя множество двоичных векторов длины $2n$. Для $1 \leq i \leq n$ iая компонента выхода обращается в единицу тогда и только тогда, когда входное слово принадлежит языку $. * \alpha_i$; если же $n + 1 \leq i \leq 2 * n$, то равенство iой компоненты выхода единице эквивалентно принадлежности входного слова языку $. * \beta_i$. Первые n компонент выхода подаются на вход второго блока и включают счетчики, отсчитывающие длины соответствующих слов β_i. Если требуемая длина достигнута, единичный сигнал передается на вход блока 3, вычисляющего конъюнкции выходов первого блока с номерами от $n+1$ до $2n$ с соответствующими выходами второго блока. Выход конъюнкции подается на вход переключателя, запоминающего единичный сигнал. Таким образом, равенство выхода переключателя единице эквивалентно выполнению следующих условий:

1) во входном потоке встретилось слово α_i;

2) не менее, чем через $|\beta_i|$ тактов после этого во входном потоке встретилось слово β_i.

Несложно увидеть, что эти условия эквивалентны тому, что входное слово принадлежит языку $.*\alpha_i.*\beta_i.*$.

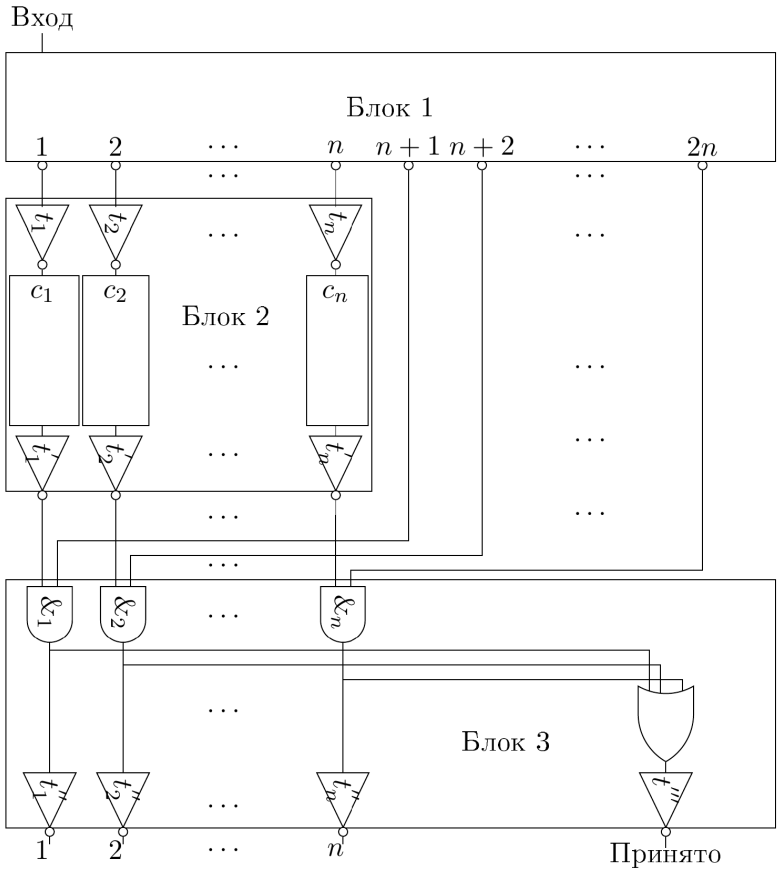

Рис. 1. Аппаратная конструкция, t_i, t'_i, t''_i, t''' — триггеры; c_i — счётчики.

Опишем аппаратную конструкцию более подробно. Распознающая аппаратная конструкция состоит из 3–х блоков. Блок 1 представляет собой ДКА, принимающий регулярный язык $(\bigcup_{i=1}^{n} .*\alpha_i) \bigcup (\bigcup_{i=1}^{n} .*\beta_i)$, который строится путём применения алгоритма Ахо и Корасик [13]. В данном ДКА словам, описываемым разными регулярными выражениями вида $.*\alpha_i$ и $.*\beta_i$, соответствуют различные принимающие состояния

ДКА. Блок 1 имеет 1 вход и $2n$ выходов. Первые n выходов блока 1 отвечают за распознавание языка $\bigcup_{i=1}^{n} . * \alpha_i$. То есть на i–ый выход, $i = 1, \ldots, n$, идет единичный сигнал тогда и только тогда, когда на данном такте вход блока 1 принадлежит соответствующему слагаемому из языка $\bigcup_{i=1}^{n} . * \alpha_i$. Сигнал от $\overline{1, n}$ выходов блока 1 идёт в блок 2 — на входы n счётчиков — выход, соответствующий номеру слова $. * \alpha_i$ из $\bigcup_{i=1}^{n} . * \alpha_i$, идёт на счётчик, соответствующий номеру этого же слова в блоке 2. Выходы блока 1 с номерами $\overline{n+1, 2n}$ отвечают за распознавание языка $\bigcup_{i=1}^{n} . * \beta_i$. Сигнал от этих $\overline{n+1, 2n}$ выходов идёт на блок 3 — на вторые входы соответствующих n конъюнкций — $\overline{\&1, \&n}$. На второй вход i–ого логического вентиля "конъюнкция" от $n + i$–ого выхода блока 1 идет единичный сигнал тогда и только тогда, когда на данном такте вход принадлежит соответствующему слагаемому — слово $. * \beta_i$ принимается, иначе выход равен 0.

Блок 2 — отсчитывает длины слов языка, представляющего объединение вторых пар слов: $\bigcup_{i=1}^{n} \beta_i$ начального регулярного языка $\bigcup_{i=1}^{n} . * \alpha_i . * \beta_i . *$. Блок 2 состоит из ряда n однополюсных однопозиционных переключателей, следующим за ним ряда n регистровых счётчиков и следующим за ним третьим рядом n однополюсных однопозиционных переключателей. Каждый счётчик соответствует своему слову β_i и осуществляет счёт до $|\beta_i|$. Счётчик представляет собой композицию включающего мультиплексора, функции сложения с константой 1 и компаратора. Величина, до которой ведется счет, задается через начальное состояние. Блок имеет n входов и n выходов. На входы блока 2 поступает сигнал от первых n выходов блока 1 — от выходов с номерами $\overline{1, n}$. Сигнал от соответствующего i–го входа бока 2 поступает на i–ый переключатель. При поступлении единичного сигнала на вход i–того переключателя, переключатель размыкается и генерирует на своём выходе логическую единицу. В обратном случае, переключатель находится в замкнутом состоянии и подаёт на выход логических ноль. Единичный сигнал от i–го переключателя активирует соответствующий i–ый счётчик. Переведенный в считающее состояние счётчик увеличивает свое значение на единицу каждый такт до достижения значения $|\beta_i|$. Достижение i–ым счётчиком максимального значения посылает на вход i–ого однополюсного однопозиционного переключателя из третьего ряда логическую единицу и размыкает переключатель. Данные переключатели в соответствии со своим номером посылают на выходы блока 2 сигналы: ноль, если соответствующий переключатель замкнут, единицу — если он разомкнут. После размыкания переключатель не может перейти в замкнутое состояние и посылает ло-

гическую единицу на выход блока 2 каждый последующий такт. Таким образом единичный сигнал на выход i блока 2 начинает поступать тогда и только тогда, когда блок 1 послал единицу на соотвествующий i–ый переключатель, i–ый счётчик достиг максимального значения и разомкнул i–ый переключатель.

Блок 3 — блок выходных сигналов конструкции. Блок состоит слоя $n+1$ логических вентилей (n логических вентилей типа "конъюнкция" и 1 логического вентиля типа "дизъюнкция") и следующего за ним слоя $n+1$ однополюсных однопозиционных переключателей. На вход блока 3 поступает сигнал от блока 2 на первые входы "конъюнкций" и блока 1, распознающего слова $\bigcup_{i=1}^{n} . * \beta_i$, на вторые входы "конъюнкций".

Таким образом, на выход блока 3 идет сигнал о распознанном слове вида $. * \alpha_i . * \beta_i$ от соответствующего i–того логического вентиля (конъюнкции) тогда и только тогда, когда i–тый счётчик блока 2 перешел в финальное состояние и отправил сигнал (логическую единицу) на первый вход i–того логического вентиля блока 3, и на второй вход данного логического вентиля блока 3 пришёл сигнал (логическая единица) из блока 1 о распознанном слове β_i. Соответсвующий i–ый логический вентиль "конъюнкция" посылает единичный сигнал на единственный соответсвующий себе i–ый однополюсный однопозиционный переключатель и переводит его в разомкнутое состояние. С этого такта на i–ый выход блока 3 i–ый однополюсный однопозиционный переключатель посылает логическую единицу.

Логическая единица на выходе "дизъюнкции" от всех "конъюнкций" сигнализирует единицей о распознанном слове из языка $\bigcup_{i=1}^{n} .*\alpha_i.*\beta_i.*$ и переводит соответсвующий себе $n+1$–ый переключатель в разомкнутое состояние. С этого такта $n+1$ выход блока 3 сигнализирует единицей о распознанном слове из языка $\bigcup_{i=1}^{n} . * \alpha_i . * \beta_i.*$, а соответствующая i конъюнкция с своим переключателем — об определенном i–ом принятом слове, соответствующему регулярному выражению $. * \alpha_i . * \beta_i.*$.

Описанная конструкция распознает соответствующий язык и позволяет избежать экспоненциального взрыва числа состояний ДКА, что перспективно для реализации на практике для сканирования трафика на вредоносность. В силу своей простоты и малого объема диаграмма автомата блока 1 может быть записана в память, а конфигурация блока 2 — задана начальными состояниями задержек. Таким образом, можно реализовывать схемы для классов языков, ограниченных параметрами n и m.

4. Основные результаты

Теорема 1 (О корректности работы)**.** *Представленная конструкция корректно принимает язык* $\bigcup_{i=1}^{n} .* \alpha_i .* \beta_i .*$*, где* α_i*,* β_i *— слова над алфавитом* A*. При этом* i*–ый выход конструкции равен* 1 *тогда и только тогда, когда входное слово лежит в* i*–ом слагаемом, а* $n+1$*–ый выход равен* 1 *тогда если и только если слово лежит в языке.*

Доказательство. Предположим, что слово $\gamma \in A^*$ принадлежит языку $.* \alpha_i .* \beta_i .*$ для некоторого i. Значит, γ имеет вид $\delta\alpha_i\delta'\beta_i\delta''$, где δ не содержит подслово α_i, а δ' — подслово β_i. Значит, после подачи на вход $\delta\alpha_i$ будет запущен счетчик номер i, и к моменту окончания подачи $\delta\alpha_i\delta'\beta_i$, во-первых, iый выход блока 2 станет равным единице, и, во-вторых, $2n+i$ый выход блока 1 станет равным 1. Следовательно, на выход iой конъюнкции блока 3 также появится единица, которая будет сохранена iым переключателем.

Обратно, пусть выход номер i третьего блока равнен единице. Рассмотрим момент, в который значение в первый раз изменилось с 0 на 1. Тогда в этот момент оба входа конъюнкции номер i третьего блока стали равными единице, то есть входное слово имеет вид $\delta''\beta_i$, и не менее $|\beta_i|$ тактов назад слово имело вид $\delta\alpha_i$. Следовательно, входное слово может быть записнано в виде $\delta\alpha_i\delta'\beta_i$, то есть оно принадлежит языку $.* \alpha_i .* \beta_i .*$. После этого момента выход останется равным 1, при этом любое надслово слова $\delta\alpha_i\delta'\beta_i$ также лежит в языке $.* \alpha_i .* \beta_i .*$.

Утверждение про выход номер $n+1$ вытекает из утверждения про первые n выходов.

Теорема доказана. □

Теорема 2 (Об отсутствии экспоненциального взрыва)**.** *Для распознавания языка* $\bigcup_{i=1}^{n} .* \alpha_i .* \beta_i .*$ *представленной конструкцией требуется* $O(mn\log_2(mn)+n)$ *бит памяти для хранения диаграммы блока 1, и* $O(n\log_2 m)$ *элементов для реализации блоков 2 и 3, где* m *— максимальная длина слов* α_i *и* β_i*.*

Доказательство. Сперва рассморим первый блок. Алгоритм Ахо и Корасик строит автомат, число состояний которого оценивается сверху как $2mn$. Для хранения диаграммы переходов требуется $|A| \times |Q|$ ячеек длины $]\log_2|Q|[$. Учитывая, что мощность алфавита является константой, эта величина имеет вид $O(mn\log_2(mn))$. Выход имеет размерность $2n$; следовательно, для хранения функции выходов достаточно $|A| \times |Q|$ ячеек длины n, то есть $O(mn \cdot n)$.

Рассмотрим второй блок. Он состоит из линейного по n числа триггеров и счётчиков. Триггер имеет константную сложность, счётчик — логарифмическую по m. В результате получаем сложность $O(n \log_2 m)$.

Наконец, третий блок очевидным образом линеен по n и константен по m.

Теорема доказана. □

Замечание 1. *Конструкция может быть естественным образом обобщена на классы языков, в которых число слов-сомножителей больше, чем 2.*

Замечание 2. *При изменении распознаваемого языка не обязательно перестраивать автомат; одно устройство способно обрабатывать произвольный язык из рассматриваемого класса при условии, что число слагаемых не превышает n, диаграмма автомата блока 1 помещается в память, а длины слов-сомножителей — в счётчики. Перенастройка производится за счет записи в память новой диаграммы и выставления соответствующих начальных состояний на счётчиках.*

5. Некоторые примеры

Рассмотрим язык $\bigcup_{i=1}^{n} . * a_i\{l\} . * a_i\{l\}$, где $a_i \in A$, l — фиксированная длина слов. ДКА для данного языка требует $O(nl \cdot 2^n)$ состояний [7]. Представленная в данной статье аппаратная конструкция требует ln состояний ДКА блока 1, $nl \cdot \log_2 2nl$ бит на таблицу переходов в памяти, n счётчиков в блоке 2 и $O(n)$ элементов блока 3.

6. Заключение

В данной статье рассмотрен класс языков с высокой практической значимостью, вызывающий экспоненциальный взрыв распознающего ДКА. Для этого класса языков предложена конструкция — комбинация абстрактного и структурного автомата, решающая проблему взрыва сложности распознающего устройства. Доказаны утверждения о корректности работы представленной конструкции и об отсутствии экспоненциального взрыва числа элементов данной конструкции.

Далее планируется расширить конструкцию с целью возможности ухода от проблемы экспоненциального взрыва для распознавания максимально широкого класса практически интересных языков.

Список литературы

[1] Кудрявцев В. Б., Алешин С. И., Подколзин А. С., *Введение в теорию автоматов*, Наука, Москва, 1985.

[2] Кудрявцев В. Б., Гасанов Э. Э., Подколзин А. С., *Основы теории интеллектуальных систем*, Макс Пресс, Москва, 2016.

[3] Документация для Perl-совместимых регулярных выражений, http://perldoc.perl.org/perlre.html.

[4] Хопкрофт Дж., Мотвани Р., Ульман Дж., *Введение в теорию автоматов, языков и вычислений*, Вильямс, Москва, 2017.

[5] Rabin M.O., Scott D., "Finite automata and their decision problems", *IBM J. Research and Development*, **3**:2 (1959), 115–125.

[6] Лупанов О.Б., "О сравнении двух типов конечных источников", Проблемы кибернетики, 1963, № 9, 321–326.

[7] Александров Д. Е., "Эффективные методы проверки сетевых пакетов", *Интеллектуальные системы. Теория и приложения*, **18**:1 (2014), 37–59.

[8] База регулярных выражений, http://www.snort.org/.

[9] Kumar, *A Thesis on Accelerationof Network Processing Algorithms*, Doctor of Science Thesis, Washington University, Saint Louis, Missouri, 2008.

[10] Александров Д. Е., "Об оценках автоматной сложности распознавания класса регулярных языков", *Интеллектуальные системы. Теория и приложения*, **18**:4 (2014), 121–146.

[11] Александров Д. Е., "Об уменьшении автоматной сложности за счет расширения регулярных языков", *Программная инженерия*, 2014, № 11, 26–34.

[12] Александров Д. Е., "Об оценках мощности некоторых классов регулярных языков", *Дискретная математика*, **27**:2 (2015), 3–21.

[13] Aho A.V., Corasick M.J., "Efficient string matching: an aid to bibliographic search", Communication of the ACM, **18**:6 (1975), 333-340.

Structural automaton design for solving the problem of exponential blowup for one class of regular languages

Bernadotte A., Galatenko A.

Аннотация

It is well known that recognition of a language specified by a regular expression of the form $\bigcup_{i=1}^{n} . * \alpha_i . * \beta_i .*$, where α_i, β_i are words over some alphabet, in the general case requires a deterministic automaton with the number of states which is exponential in n. We propose a design of a structural automaton of a polynomial spatial complexity that recognizes languages from a given class.

Keywords: DFA, structural automaton, regular language, exponential blowup.