УДК 004, 004.3, 004.9, 519.6, 519.7, 519.8
DOI: 10.17587/prin.13.449-461

А. Бернадотт, канд. мед. наук, доц., bernadotte.alexandra@intsys.msu.ru,
Национальный исследовательский технологический университет "МИСиС",
ООО "Нейроспутник", Москва, Механико-математический факультет МГУ им. М. В. Ломоносова

Структурная модификация конечного автомата для решения проблемы экспоненциального взрыва

В ряде современных приложений, таких как системы обнаружения и предупреждения вторжений, экспертные знания формализуются в виде регулярных выражений, после чего проводится проверка принадлежности слова регулярному языку конечным автоматом. При этом задача понижения пространственной сложности при сохранении низкой временной сложности крайне актуальна. Представлен обзор современных решений данной задачи, основной идеей которых является переход от абстрактного конечного автомата, представленного таблично заданной функцией переходов, к структурному автомату, комбинирующему абстрактную часть, хранящуюся в памяти, и различные добавки типа битовых массивов и счетчиков.

Ключевые слова: *конечный автомат, экспоненциальный взрыв, сетевые системы обнаружения вторжения, структурная сложность, понижение сложности, регулярные языки, регулярные выражения*

Введение

Принадлежность слова регулярному языку используется в широком спектре сетевых служб и служб безопасности на большинстве сетевых устройств, таких как маршрутизаторы, межсетевые экраны, коммутаторы уровня 7 и сетевые системы обнаружения вторжения (ССОВ).

Данные ССОВ зачастую интегрированы с базами регулярных выражений, составляющих регулярный язык, описывающий признаки атак. Детектирование вторжения проводится путем проверки трафика на наличие слов, принадлежащих данному регулярному языку. Большинство ССОВ с открытым исходным кодом, такие как Snort, используют сигнатурные базы Perl-совместимых регулярных выражений для реализации анализа сетевого трафика. Эффективность использования памяти в алгоритмах обработки сетевого трафика в сетевых устройствах является критической характеристикой в силу предпочтительного для скорости использования данными устройствами преимущественно статической памяти с произвольным доступом (SRAM).

На 2022 г. проверка принадлежности слова в трафике регулярному языку большинства ССОВ представляет сложность при использовании классического метода — распознавания языка детерминированным конечным автоматом (ДКА). Причиной данной сложности является увеличение числа состояний распознающего ДКА, что критично для памяти системы обнаружения вторжений.

В практическом использовании конечные автоматы в составе ССОВ могут комбинироваться с дополнительными аппаратными элементами, типа счетчиков, битовых массивов, переходов между медленной и быстрой памятью и другими элементами. В данной работе приведен обзор вариантов структурной модификации конечного автомата для решения проблемы увеличение числа состояний распознающего ДКА.

Формальные понятия и определения

Основные определения даны в соответствии с работами [1, 2].

Язык M, $M \subseteq A^* \setminus \{\lambda\}$, называется регулярным, если его можно получить применением конечного числа операций объединения, произведения и итерации над регулярными языками вида $\varnothing$, $\{a\}$, $a \in$ A. Формально, регулярный язык над алфавитом A задается рекурсивно:

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

2) если M_1 и M_2 — регулярные языки, то объединения $M_1 \cup M_2 = \{\alpha_1 \cup \alpha_2 \mid \alpha_1 \in M_1, \alpha_2 \in M_2\}$ и конкатенация $M_1 \cdot M_2 = \{\alpha_1\alpha_2 \mid \alpha_1 \in M_1, \alpha_2 \in M_2\}$ языков M_1 и M_2, а также итерация или звезда Клини $M_1^* = \{\lambda\} \cup \{\alpha_1 \cdot \ldots \cdot \alpha_n \mid \alpha_i \in M_1,\ i \in \mathbb{N}\}$ — регулярные

языки, где регулярность языка устанавливается за конечное число операций объединения, конкатенации и итерации [1, 2].

Формулу, которая описывает последовательность операций, принято называть "регулярным выражением", задающим данный регулярный язык. Формально, регулярные выражения представляют собой слова над алфавитом $A \cup \{\cup, (,), \cdot, *\}$, определяемые следующим образом:

1) $\varnothing$, λ, 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.

Регулярный язык, задаваемый регулярным выражением, определяется естественным образом, путем выполнения последовательности операций объединения, конкатенации и итерации, задаваемой данным регулярным выражением.

На практике 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;

".*" — ноль или более вхождений произвольного символа из алфавита;

"a+" — одно или более вхождений "a";

"{n}" — число вхождений, равное n;

"{n, m}" — число вхождений от n до m;

"|" — логическое "объединение";

"^", "¬" — логическое "не";

"[]" — групповой символ;

"[0−9]" — любой цифровой символ;

"[A − Za − z0 − 9]" — любой буквенный или цифровой символ.

Формально, ДКА V называется набор $V = (A, Q, Q_B, \varphi)$, где $A \neq \varnothing$ — входной алфавит; $Q \neq \varnothing$ — конечное множество состояний; $Q_B \subseteq Q$ — множество заключительных состояний; φ — функция переходов; φ: $A \times Q \to Q$.

Инициальный конечный автомат имеет отдельно выделенное начальное состояние, в котором начинается работа автомата: $V = (A, Q, Q_B, \varphi)$, где q_0 — начальное состояние автомата; $q_0 \in Q$.

Недетерминированный конечный автомат (НДКА) позволяет осуществлять переход из одного и того же состояния под действием одной и той же буквы в разные состояния и представляет собой набор $V = (A, Q, Q_B, \varphi, q_0)$, где $A \neq \varnothing$ — входной алфавит; $Q \neq \varnothing$ — конечное множество состояний; $Q_B \subseteq Q$ — множество заключительных состояний; φ — функция переходов, φ: $A \times Q \to 2^Q \setminus \{\varnothing\}$; $q_0 \in Q$ — начальное состояние автомата.

Согласно теореме об эквивалентности детерминированных и недетерминированных конечных автоматов, всякий язык принимается некоторым НДКА тогда и только тогда, когда этот язык принимается некоторым ДКА [4, 5]. Кроме того, существует алгоритм построения из НДКА эквивалентного ему ДКА [4, 5].

Конечные автоматы рассматриваются как устройства, распознающие (и принимающие) некоторые множества входных слов над конечным алфавитом. Подмножество M, $M \subseteq A^* \setminus \{\lambda\}$, где λ — пустое слово, называют языком над алфавитом A [1, 2]. Пусть $V_q = (A, Q, Q_B, \varphi, q_0)$ — инициальный автомат, $Q_B \subseteq Q$. Множество M (язык M) = $\{\alpha: \alpha \in A^*, \varphi(q, \alpha) \in Q_B\}$ называется представимым в автомате V_q с помощью подмножества заключительных состояний Q_B, или говорим, что автомат V_q принимает M посредством Q_B. Множество M (язык M) называется представимым.

В российской литературе представление языка проводится не посредством состояний, а посредством выходов [1, 2]. Пусть $V_q = (A, Q, B, \varphi, \psi, q_0)$ — инициальный автомат, где $A \neq \varnothing$ — входной алфавит; $Q \neq \varnothing$ — конечное множество состояний; B — выходной алфавит; φ — функция переходов, φ: $A \times Q \to Q$; ψ — функция выходов; ψ: $A \times Q \to B$. Множество M (язык M) = $\{\alpha: \alpha \in A^*, \psi(q, \alpha) \in B'\ B' \subseteq B\}$ называется представимым в автомате V_q с помощью подмножества B' входных символов, или говорим, что автомат V_q принимает M посредством B'. Распознавание посредством выходов или посредством распознавания состояниями является эквивалентным [1, 2].

В прикладной задаче проверки принадлежности слова регулярному языку значимым является пространственная сложность реализующего проверку конечного автомата (КА) — число состояний распознающего КА (и объем необходимой для проверки памяти), и временная сложность распознавания — число переходов из одного состояния в другое (соответствующих тактам работы КА). Принято считать, что вычисление значения функции переходов автомата имеет константную сложность — сводится к извлечению значения из памяти.

Детерминированный конечной автомат имеет оптимальную по порядку временную сложность, однако число состояний автомата (пространственная сложность) может расти экспоненциально от длины регулярного выражения.

Недетерминированный конечный автомат обладает низкой пространственной сложностью, линейно зависимой от длины регулярного выражения. При этом временная сложность является основным недостатком НДКА, так как время обработки одного символа входного слова, вообще говоря, также линейно зависит от длины регулярного выражения.

При переходе от недетерминированного автомата к детерминированному временная сложность обработки входного символа становится константной. Однако в общем случае при входном алфавите мощности более или равной двум мощность множества состояний экспоненциально возрастает от длины регулярных выражений [1, 2, 6].

Связь между регулярным языком и автоматом описывается теоремой Клини: слово лежит в языке тогда и только тогда, когда заключительное состояние соответствующего автомата лежит в выделенном подмножестве.

Проблема экспоненциально растущего числа состояний конечного автомата

В прикладных задачах распознавания регулярного языка значительной проблемой является проблема экспоненциально растущего числа состояний распознающего ДКА от длины регулярных выражений распознаваемого языка [1, 6].

К признакам, вызывающим экспоненциальный взрыв, относятся: использование в регулярных выражениях сочетаний ".+" и ".*" и "считающих" выражений типа ".{*n*}" и "[$\neg a_i$]{*n*}".

Отдельное внимание уделяется проблеме экспоненциального взрыва для языка $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$, 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 ДКА [7].

Проблемой экспоненциального взрыва исследователи занимаются более 20 лет. Существуют следующие три основных подхода к решению этой проблемы с использованием КА.

- Ограничение на сигнатуры, задаваемые экспертами. Данный метод предполагает экспертный подбор регулярных выражений, позволяющий сократить число состояний данного КА. Минусом данного подхода являются низкая автоматизация подхода и его высокая трудоемкость. При этом возможно как расширение, так и сужение распознаваемого языка после экспертной работы.
- Модификация распознаваемого регулярного языка. Данный подход предполагает появление в решении практической задачи распознавания ошибок первого и второго рода. При этом метод дает значимое с практической точки зрения уменьшение сложности КА за счет расширения регулярных языков [8].
- Модификация структуры КА. Данный подход заключается в модификации структуры КА без изменения распознаваемого языка и может быть реализован:

— как модификация КА через применение алгоритмов сжатия [9];

— как модификация структуры КА через добавление специальных структурных элементов.

В настоящей работе предметом интереса является модификация структуры КА через добавление специальных структурных элементов, которая не требует изменения распознаваемого языка.

Существующие структурные расширения конечных автоматов

Самыми распространенным структурными элементами, расширяющими возможности КА, являются счетчики и дополнительные элементы памяти, хранящие определенные свойства КА. В данном разделе приведен список существующих решений, основной идеей которых является переход от абстрактного КА, представленного таблично заданной функцией переходов, к структурному автомату, комбинирующему абстрактную часть, хранящуюся в памяти, и различные добавки типа битовых массивов, счетчиков, подсхем. При этом пространственная сложность определяется как сумма объема памяти, занимаемой абстрактной частью, и сумма сложностей добавок (т. е. схемная сложность). Экспоненциальное число состояний перемещается в структурную часть, так что общая сложность остается приемлемой.

Введем термин "метасостояние конечного автомата". Метасостояние конечного автомата — дополнительный параметр КА в наборе, описывающий дополнительные структурные элементы КА.

Детерминированный абстрактным конечным автоматом с метасостоянием называется набор $V = (A, Q, Q_B, M, \varphi, \omega)$, где A, Q, Q_B, M — конечные множества: входной алфавит, алфавит состояний, подмножества заключительных состояний и алфавит метасостояний соответственно; φ — функция переходов, $\varphi: A \times Q \times M \to Q$; ω — функция смены метасостояния, $\omega: A \times Q \times M \to M$.

Хранение значения дополнительного параметра КА, отвечающего за выделение метасостояний, чаще всего осуществляется посредством отдельных битов, битовых массивов, битовых масок и счетчиков. Экспоненциальное число состояний перемещается в структурную часть, так что общая сложность остается приемлемой. Наглядно такое структурное перестроение будет изложено ниже.

Рассмотрим существующие принципиальные решения проблемы экспоненциального взрыва с использованием дополнительных структурных элементов.

Двойной конечный автомат с использованием быстрой и медленной памяти

Занимаясь проблемой экспоненциального взрыва в приложении к распознаванию вредоносного трафика, Кумар и соавторы [10] вводят несколько понятий, характеризующих недостатки использования классического ДКА в контексте проблемы экспоненциального взрыва. Так, "бессонницей" (в оригинале *insomnia*) авторы называют расточительное хранение всех состояний автомата, вне зависимости от вероятности их активации [10].

Действительно, при анализе трафика КА большинство состояний активируются редко, и вероятность активации состояния падает по мере удаления от начального состояния КА.

Авторами предлагается делить регулярное выражение на префиксную и суффиксную части, которые определяются вероятностью перехода КА в состояния, соответствующие суффиксной части. К высоковероятным состояниям относят состояния, описывающиеся префиксами регулярных выражений, к маловероятным состояниям относятся те, которые описываются хвостовыми частями (суффиксами) регулярных выражений. Авторами предлагается хранить переходы в высоковероятные состояния ДКА в быстрой памяти, а переходы в маловероятные состояния — "в состоянии сна" (в медленной памяти). Переключение из быстрой памяти в медленную должно осуществляться при необходимости — при срабатывании триггера. При этом триггером является распознавание префикса подслова быстрой частью ДКА.

Структурным элементом решения является медленная память с переключателями между быстрой и медленной памятью. Особенность такого хранения состояний дает выгоду по объему затрачиваемой быстрой памяти, при этом обращение к медленной памяти замедляет работу данной конструкции при сравнении с классическим ДКА. Однако предварительный статистический анализ трафика позволяет снизить необходимость обращения к медленной памяти. Таким образом, конструкция позволяет снизить объем затрачиваемой быстрой памяти без существенных временных потерь скорости работы.

На рис. 1 приведен пример двойного КА из работы Кумар и соавторов, принимающего язык L, $L = R_1 \cup R_2 \cup R_3$, представленный объединением следующих регулярных выражений:

$R_1 = .*[gh]d[\neg g]*ge$, $R_2 = .*fag[\neg i]*i[\neg j]*j$, $R_3 = .*a[gh]i[\neg l]*[ae]c$.

Модификация классического ДКА авторами предложена для прикладной задачи обнаружения вторжений при анализе трафика, и сама модификация ориентируется на предварительный анализ нормального и вредоносного трафика. Вопрос о разделении состояний на вероятные и маловероятные предлагается авторами решать эмпирически, а именно через рассмотрение распределения вероятностей различных входных символов для конкретного трафика, анализируемого устройством с описываемым автоматом [10—12].

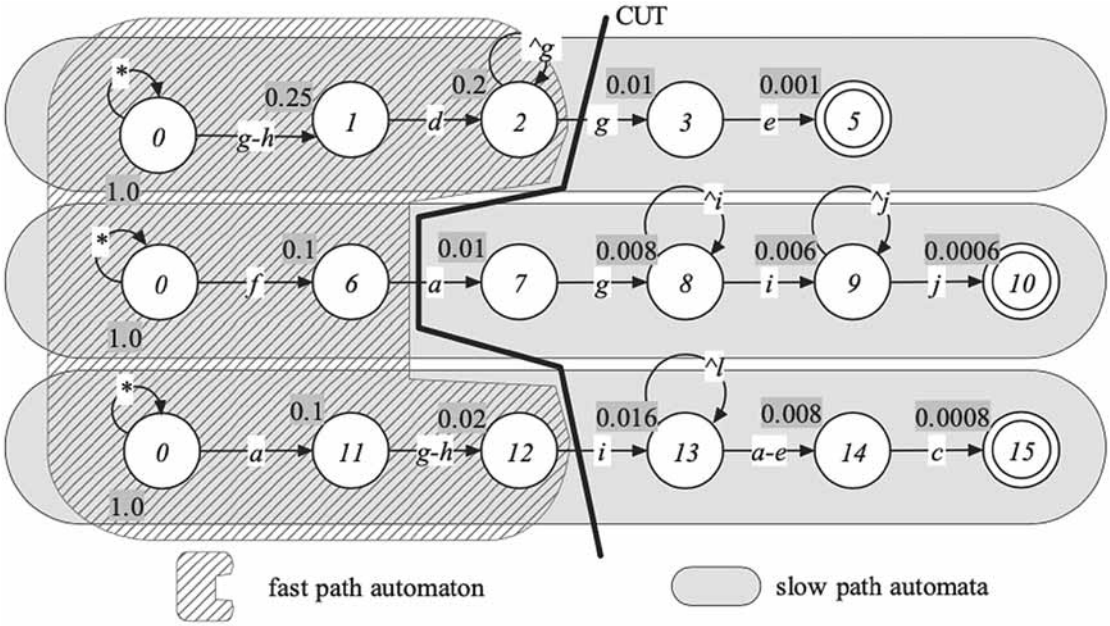

Рис. 1. Диаграмма двойного КА с изображением быстрой (слева) и медленной частей (справа) памяти. Диаграмма соответствует принимаемому языку L, $L = R_1 \cup R_2 \cup R_3$, $R_1 = .*[gh]d[\neg g]*ge$, $R_2 = .*fag[\neg i]*i[\neg j]*j$, $R_3 = .*a[gh]i[\neg l]*[ae]c$. Над стрелками перехода отмечена вероятность перехода, полученная при предварительном анализе трафика, двойной линией обозначены принимающие состояния [10]

Следует отметить, что в системе обнаружения вторжений, в состав которой входит предложенный двоичной КА, требуется предварительно вычислять вероятность совпадения префиксов разной длины при нормальном и аномальном трафике и на основе полученных данных хранить состояния в медленной или быстрой памяти. Таким образом, чтобы использовать двойной КА в прикладной задаче недостаточно использовать существующую базу регулярных выражений, маркирующих вредоносный трафик. Требуется "дообучить" систему на собранном датасете, где трафик будет размечен относительно частоты активации состояний соответствующего КА. Данное свойство несколько затрудняет использование интересного решения коллег.

Детерминированный конечный автомат с битовым массивом

Существует несколько решений, использующих ДКА в комбинации с хранением дополнительной информации в памяти в виде битового массива.

Детерминированный конечный автомат с памятью (битовым массивом). Неспособность КА запоминать "посещенные состояния" Кумар и соавторы называют "амнезией" (в оригинале *amnesia*) и предлагают хранить метасостояние автомата в виде нескольких флагов (битов) в дополнение к текущему состоянию автомата [10]. Автомат с такой модификацией авторами назван "ДКА с памятью" (в оригинале *History based Finite Automaton* или сокращенно — H-FA) [10].

Формально, ДКА с памятью представлен набором, H-FA = (A, Q, q_0, Q_B, φ, H), где A — входной алфавит; Q — алфавит состояний; q_0 — начальное состояние; Q_B — множество принимающих состояний; H — битовый массив; φ — функция перехода, φ: $A \times Q \times H \rightarrow Q \times H$ [10]. Битовый массив принимает соответствующее состоянию КА значение, равное 1, если в данное состояние ранее был осуществлен переход, и значение, равное 0, если такого перехода не было.

Детерминированный конечный автомат с памятью структурно расширяет ДКА с помощью вспомогательной памяти (битового массива), осуществляя работу сложных переходов в зависимости от хранимой в памяти информации. Выигрыш от использования битового массива заключается в уменьшении числа необходимых состояний и числа переходов за счет хранения дополнительной информации о "посещенных" состояниях. На рис. 2, *б* приведен пример ДКА с памятью (H-FA) из работы Кумар и соавторов, принимающий язык $L = R_1 \cup R_2$, $R_1 = .*ab[\neg a]*c$, $R_2 = .*def$, и дано сравнение с классическим ДКА (рис 2, *а*), демонстрирующее пространственную выгоду использования битового массива.

Модификация ДКА с памятью под названием HASIC, предложенная группой авторов [13], делает работу системы обнаружения вторжений в несколько раз более эффективной.

Расширенный детерминированный конечный автомат с битовым массивом. Другая группа исследователей [14] также использовала идею с битовым массивом для хранения метасостояния ДКА. Группа сфокусировала свое внимание на определенном классе языков, вызывающем экспоненциальный взрыв и накладывающем значительное ограничение на эффективность и скорость работы систем обнаружения вторжений — $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$, где α_i, β_i — слова над алфавитом A [14].

Представленный "расширенный ДКА" (в первоисточнике — *extended finite automata* (XFA)) решает проблему экспоненциального взрыва для языков класса: $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$, где слова α_i, β_i имеют определенные ограничения [14]. При этом расширенный ДКА позволяет снизить число состояний распознающего ДКА с $O(nl2^n)$ числа состояний нативного ДКА до $O(nl)$ состояний расширенного ДКА, где l — максимальная длина слов α_i, β_i. Формально расширенный ДКА — это (A, Q, D, φ, U_φ, (q_0, d_0), F), где A — входной алфавит; Q — алфавит состояний; φ: $Q \times A \rightarrow Q$ — функция перехода; D — битовый массив с начальными нулевыми значениями; U_φ: $Q \times A \times D \rightarrow D$ — функция обновления битового массива; (q_0, d_0) — начальные состояния из Q и D; $F \subseteq Q \times D$ — множество принимающих конфигураций.

Дополнительным структурным элементом данного решения является битовый массив, который фиксирует метасостояния ДКА в отношении распознанного подслова α_i. Снижение пространственной сложности реализуется через сохранение дополнительного бита в специальном структурном элементе — битовом массиве, при распознавании автоматом первого слагаемого в классе языков $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$, где слова α_i, β_i не являются подсловами друг друга [14].

Данная модификация расширенного ДКА позволяет "запомнить", что первое слагаемое α_i из $.*\alpha_i.*\beta_i.*$ было распознано, и перевести ДКА в соответствующее метасостояние. При нахождении расширенного ДКА в определенном метасостоянии и при распознавании второго слагаемого — β_i, происходит распознавание всего слова —$.*\alpha_i.*\beta_i.*$. При этом число хранимых бит в битовом массиве соответствует числу первых сомножителей.

Смит и коллеги применяют расширенный ДКА к еще одному классу языков. На рис. 3 показан

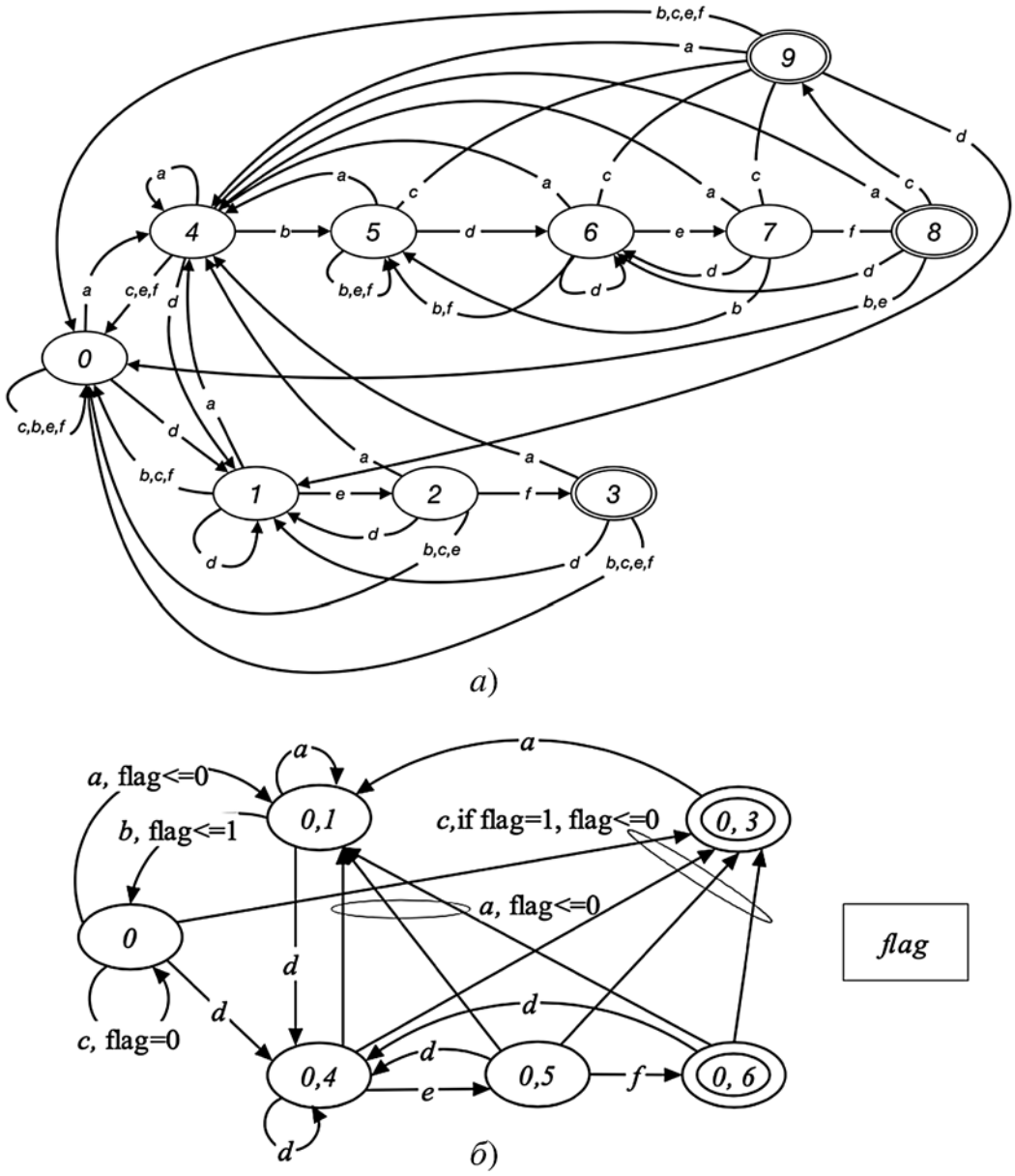

Рис. 2. Детерминированный конечный автомат, распознающий язык $L = R_1 \cup R_2$, R_1 =.*ab[¬a]*c, R_2 =.*def над алфавитом *A* (*а*) и ДКА с памятью, распознающий язык $L = R_1 \cup R_2$, R_1 =.*ab[¬a]*c, R_2 =.*def над алфавитом *A* (*б*). Память реализована хранимым битовым массивом. На рисунке бит, названный *flag*, принимает значение 1 при переходе из состояния 0,1 в состояние 0 по символу *b*, в обратном случае *flag* принимает значение 0 [10]

переход от обычного ДКА, распознающего язык $L = \{(.* ab.* cd) \cup (.* ef.* gh)\}$ над алфавитом {*a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*} к расширенному ДКА [14].

Такое решение требует существенных ограничений, накладываемых на распознаваемый язык. Авторы утверждают, что необходимо, чтобы слова α_i и β_i не являлись подсловами друг друга. Требует детального рассмотрения возможность ослабления данного ограничения. Так, если сначала проводить проверку на активный бит, а потом присваивать его (рис. 4), то подобное ограничение не требуется.

При этом у предложенного решения останется классическая проблема алгоритмов борьбы с экспоненциальным взрывом при рассмотрении данного класса языков $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$: если суффикс α_i является префиксом β_i, то данная представленная конструкция расширенного ДКА с битовым массивом расширяет принимаемый язык. Дадим название отмеченной проблеме: "проблема расширения языка". Примеры, иллюстрирующие данную проблему расширения языка, представлены далее.

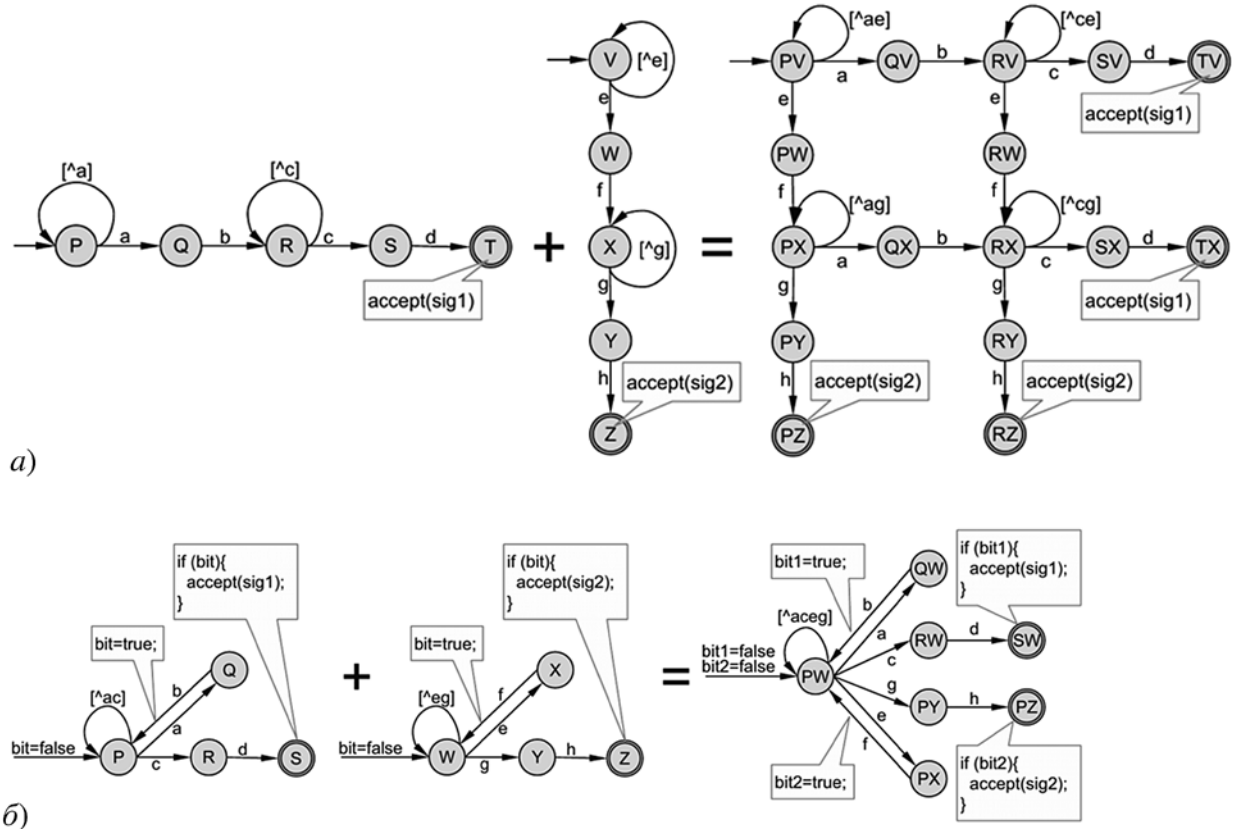

Рис. 3. ДКА (*а*) и расширенный ДКА (*б*), распознающие язык L = {(.ab*.**cd*) ∪ (.**ef*.**gh*)} над алфавитом {*a, b, c, d, e, f, g, h*} [14]**

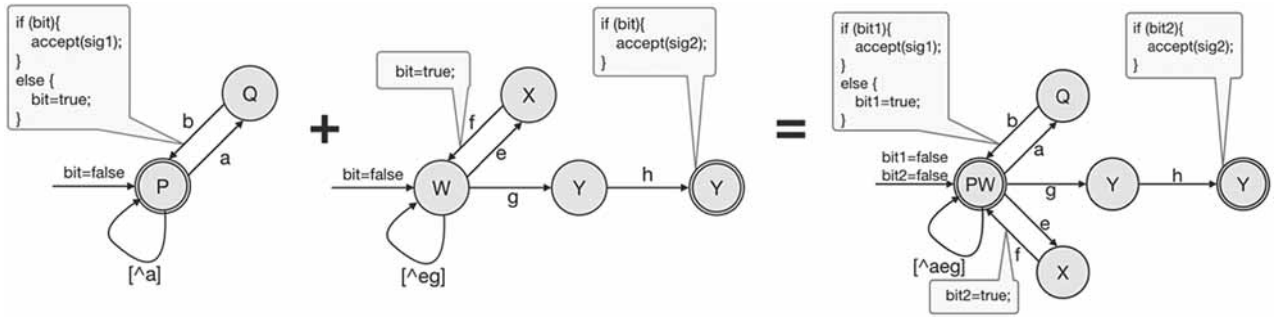

Рис. 4. Расширенный ДКА, распознающий язык L = {(.* *ab*.* *ab*) ∪ (.* *ef*.* *gh*)} над алфавитом {*a, b, c, d, e, f, g, h*}

Гибридные конечные автоматы

Сочетание ДКА и НДКА дает ощутимое преимущество для решения проблемы экспоненциального взрыва, позволяя комбинировать положительные с точки зрения экономии ресурсов и времени свойства детерминированности и недетерминированности. Существует несколько предложений использования комбинации ДКА и НДКА в контексте проблемы экспоненциального взрыва [15, 16].

Детерминированный конечный автомат с линейным автоматом. Лиу и Ву отмечают, что причина проблемы экспоненциального взрыва ДКА заключается в большом числе "независимых состояний" соответствующего НДКА (назовем "нативный НДКА", НДКА = (A, Q, Q_B, φ, q_0), из которого строится ДКА, где два состояния v_i и v_j нативного НДКА являются независимыми, если существуют три множества активных состояний α, β и γ, такие что $v_i \in \alpha$, $v_j \notin \alpha$, $v_i \notin \beta$, $v_j \in \beta$, v_i, $v_j \in \gamma$. Для решения проблемы экспоненциального взрыва авторами представлена конструкция, названная "двойной конечный автомат" [15]. Чтобы не путать с описанным ранее решением, назовем данную конструкцию "детерминированный конечный автомат с линейным автоматом".

Детерминированный конечный автомат с линейным автоматом состоит из ДКА и дополнительного структурного элемента — конечного линейного автомата. Классический ДКА данной конструкции расширяется по средствам "расширенных" и "условных" переходов, которыми данный ДКА взаимодействует с линейным. Такой ДКА авторы называют "расширенным ДКА". Линейный конечный автомат (ЛКА) — это НДКА, представляющий собой КА с переходами такими, что из каждого состояния возможен переход в себя и в единственное другое состояние, при этом в каждое состояние возможен переход из единственного состояния, отличного от данного. "Линейность" автомата позволяет упорядочить состояния и хранить переходы в виде битового массива (битовой маски), объемом $2|A||Q|$ бит, где A — входной алфавит; Q — алфавит состояний НДКА. Состояния упорядочены таким образом, что если существует переход из состояния u (представлено i-м битом в битовом массиве) в состояние v, то состояние v должно быть представлено $(i-1)$-м битом, находящимся в битовом массиве справа от i-го бита. Для каждого символа a_i из алфавита A формируются два битовых массива длиной равной $|Q|$: определяющие переход без смены состояния (битовая маска $self[a_i]$, где $self[a_i][j] = 1 \Leftrightarrow q_j \in \varphi(q_j, a_i)$), и переход в другое состояние по данному символу (битовая маска $next[a_i]$, где $\text{next}[a_i][j] = 1 \Leftrightarrow q_j - 1 \in \varphi(q_j, a_i)$).

Для иллюстрации на рис. 5 приведем решение авторов для языка $L = (.+B.^*) \cup (.^*[A-Q] + [I-Z])$. Решение авторов приводит к изменению пространственной сложности, определяемой числом состояний КА.

В один момент времени может существовать один активный битовый массив L, представляющий собой массив длиной $|Q|$ и фиксирующий метасостояние автомата, выражающееся в наборе активных состояний. Активный битовый массив L можно вычислить с помощью следующих побитовых операций при получении на вход символа a_i из алфавита A: $(L \cap self[a_i]) \cup [(L \cap next[a_i]) >> 1]$, где $\cap$, $\cup$ и $>>$ побитовые операторы "и", "или", и "сдвиг вправо" соответственно. Активный битовый массив является меткой расширенного перехода, объединяющего ДКА и ЛКА. Условный переход осуществляется при нахождении автомата в определенном метасостоянии, определяемом текущим активным битовым массивом. Каждый переход расширенного ДКА должен быть закодирован двумя битовыми массивами — кодирующими индекс следующего состояния расширенного ДКА и состояния ЛКА.

Для каждого символа из алфавита A, поступающего на вход, двойной КА выполняет следующие действия: 1) в соответствии с поступающим символом из алфавита A ЛКА и расширенный ДКА совершают переходы, при этом ЛКА определяет новое метасостояние двойного автомата, закодированное

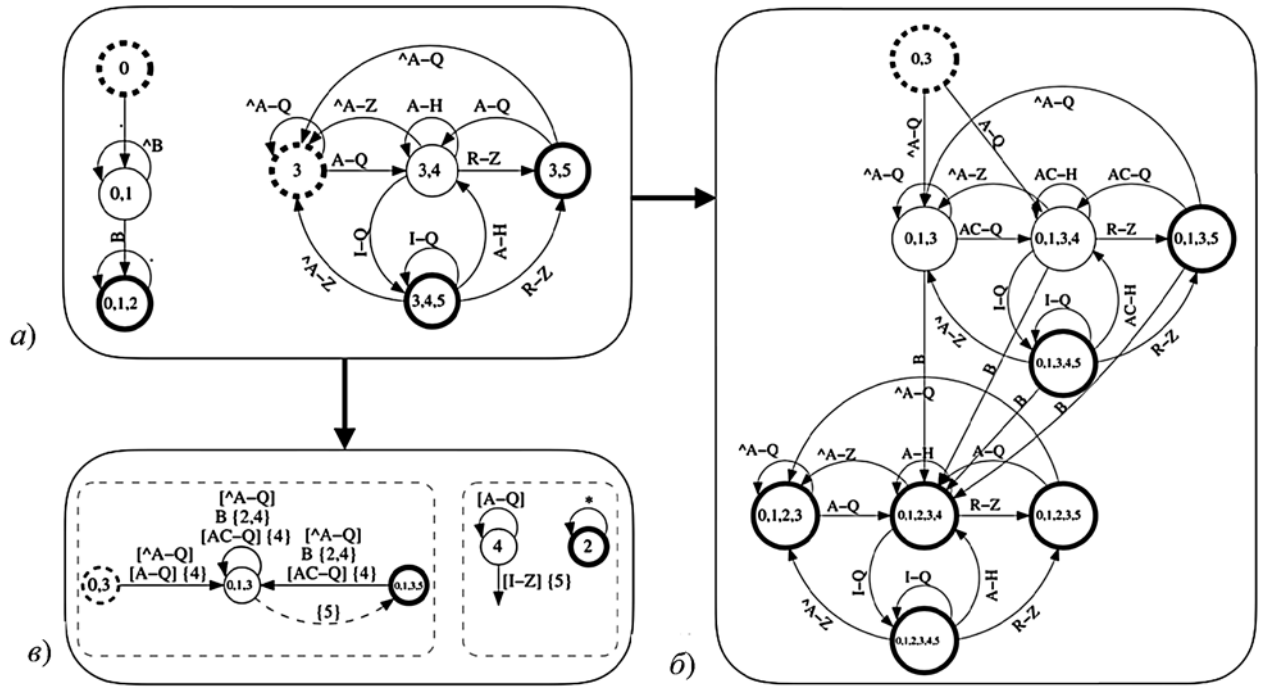

Рис. 5. Два ДКА, распознающих языки $L_1 = . + B.^*$ и $L_2 = .^*[A-Q] + [I-Z]$, требующих три и четыре состояния соответственно (*а*); ДКА, распознающий язык $L = (. + B.^*) \cup (.^*[A-Q] + [I-Z])$, представленный девятью состояниями (*б*); двойной КА с линейным автоматом, распознающий язык $L = (.+B.^*) \cup (.^*[A-Q] + [I-Z])$, представленный пятью состояниями (*в*) [15]

как "активный битовый массив *L*"; 2) дополнительные состояния ЛКА могут быть установлены как активные, если они указаны переходом, выполняемым расширенным ДКА; 3) расширенный ДКА совершает условный переход, если на это указывает переход, выполняемый ЛКА.

Метасостояние КА определяется активностью дополнительного линейного автомата, закодированного как активный битовый массив *L*. Таким образом, реализация расширенного ДКА с ЛКА позволяет использовать описанную ранее схему фиксации метасостояния через хранение битов.

Говоря о структурной организации конструкции, хранение битовых массивов может осуществляться через хранение в кеш-памяти (или встроенной памяти), обеспечивающей быстрый доступ. Однако само построение оптимального двойного КА с точки зрения минимизации числа состояний является задачей, не имеющей оптимальных алгоритмов.

Конечные автоматы со счетчиками

При реализации как НДКА, так и ДКА существует проблема неэффективного подсчета вхождения определенных подвыражений. Всякий раз, когда регулярное выражение содержит ограничение длины *k* для подвыражения типа *R*{*k*}, число состояний, требуемых подвыражением *R*, умножается на *k*. Большинство авторов предлагают решать данную проблему неэффективного использования КА через добавление счетчиков. Однако в данном обзоре решений приводится использование счетчиков и для решения проблемы экспоненциального взрыва.

Детерминированный конечный автомат с битовым массивом и счетчиками. Кумар и соавторы выделяют недостаток КА, связанный с появлением в принимаемом регулярном языке *R*{*k*} и *R*{*n*, *k*}, где *R* — регулярное выражение, специальным названием — "акалькулия" (в первоисточнике *acalculia*), в силу которой как НДКА, так и ДКА не могут эффективно подсчитывать вхождения определенных подвыражений [10].

Для решения проблемы экспоненциального взрыва авторами [10] предложен ДКА с блоком дополнительной памяти и блоком счетчиков (в первоисточнике — *History based counting finite Automata* или H-cFA), хранящих метасостояния автомата в виде набора битов. Кумар и соавторы дополняют конструкцию ДКА битовым массивом вместо тех состояний, которые вызывают экспоненциальный взрыв типа ".*" и ".+". Счетчики дополняют те состояния, которые соответствуют подвыражениям типа *R*{*k*} и *R*{*n, k*}, где *R* — регулярное выражение [10].

Формально, ДКА с памятью и счетчиками представлен набором H-cFA, H-cFA = (*A*, *Q*, *H*, *C*, Q_B, φ, q_0), где *A* — входной алфавит; *Q* — алфавит состояний; *H* — множество конфигураций, фиксирующих метасостояние автомата через дополнительный бит; *C* — множество конфигураций, фиксирующих метасостояние автомата через счетчики; φ — функция перехода, φ: $A \times Q \times H \times C \rightarrow Q \times H \times C$; Q_B — множество принимающих состояний [10]. Пространственный выигрыш осуществляется за счет уменьшения мощности алфавита состояний (и требуемого объема табличной организации переходов) и оптимизации хранения переходов в виде счетчиков.

На рис. 6 приведен пример ДКА с памятью и счетчиками (H-cFA) из работы Кумар и соавторов, принимающий язык $L = R_1 \cup R_2$, R_1 =.*ab*[¬*a*]{4}*c*, R_2 =.**def*.

Практическое применение авторами [10] ДКА с памятью и счетчиками на существующих датасетах систем обнаружения вторжений привело к уменьшению числа состояний на порядок по сравнению с классическим ДКА.

Следует обратить внимание на особенность представленных Кумаром и соавторами структурно-модифицированных автоматов: данные автоматы имеют ограничение на правильность распознавания языков, что ранее было названо "проблемой расширения языка". Данная проблема расширения языка приводит к ложным срабатываниям в системе обнаружения вторжений.

Авторы интересного решения ДКА с памятью и счетчиками не акцентировали внимание

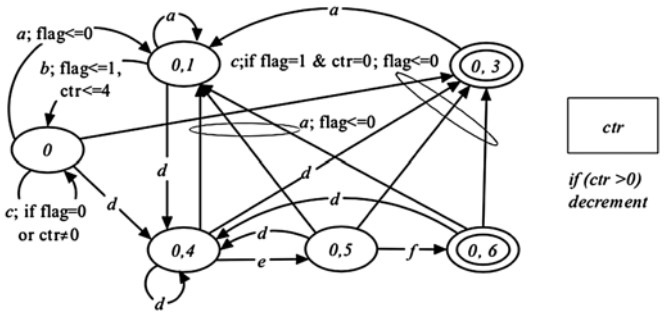

Рис. 6. Детерминированный конечный автомат с памятью и счетчиками (H-cFA), распознающий язык (.ab*[¬*a*]{4}*c*) ∪ (. * *def*) над алфавитом A. Счетчик на рисунке обозначен как *ctr*.**
Память реализована хранимым битовым массивом. На рисунке бит, названный *flag*, принимает значение 1 при переходе из состояния 0,1 в состояние 0 по символу *b*, в обратном случае *flag* принимает значение 0. Счетчик *ctr* принимает значение 4 тогда и только тогда, когда реализуется переход из состояния 0,1 в состояние 0 по символу *b*. Далее каждый такт счетчик уменьшает свое значение на единицу. Сброс состояния счетчика до нулевого происходит при переходе из состояния 0 в состояние 0,1 по символу *a*. Условный переход реализует переход из состояния 0 в 0,3 тогда и только тогда, когда бит *flag* принимает значение 1 и счетчик *ctr* принимает значение 0 [10]

на данной проблеме расширения языка. Однако незначительная модификация регулярного языка, который распознается ДКА с памятью (H-FA) на рис. 2, приводит тому, что предложенное решение дает ложные срабатывания в прикладной задаче обнаружения вторжений.

Так, преобразование языка $L = R_1 \cup R_2$, $R_1 = .*ab[\neg a]\{4\}c$, $R_2 = .*def$ посредством изменения первого регулярного выражения из $R_1 = .*ab[\neg a]\{4\}c$ в регулярное выражение $R' = .*ab[\neg a]\{4\}bc$, дает язык $L' = R'_1 \cup R_2$, где $R_2 = .*def$, при этом модифицированный ДКА с памятью (H-FA) будет принимать не L', а язык $L \cup L'$, что и будет обеспечивать ложные срабатывания системы обнаружения вторжений.

Детерминированный конечный автомат со счетчиками для решения проблемы экспоненциального взрыва и проблемы расширения языка. Продолжением идеи расширенного ДКА явилась конструкция авторов Бернадотт и Галатенко [17]. Предложение авторов ориентировано на решение проблемы экспоненциального взрыва для языков следующего класса: $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$, где слова α_i, β_i — непустые слова в алфавите A. При этом конструкция решает проблему расширения языка.

Данная конструкция позволяет избежать экспоненциального взрыва пространственной сложности через разбиение КА 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 [17].

Общая схема конструкции представлена на рис. 7. Автомат состоит из трех блоков. Блок 1 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 данного блока — множество двоичных век-

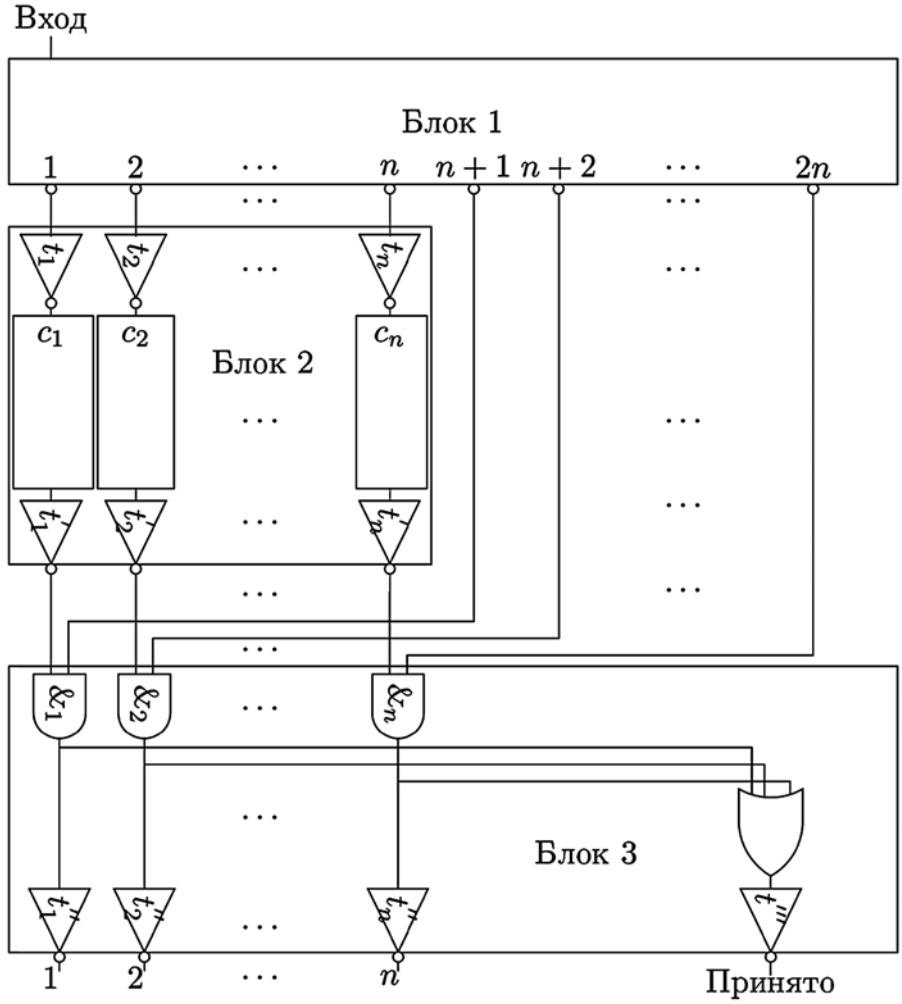

Рис. 7. Аппаратная конструкция, t_i, t'_i, t''_i t'''_i — триггеры; прямоугольники c_1, ..., c_n блока 2 — счетчики, отсчитывающие длину слов, соответствующих β_i языка $L_2 = \cup_{i=1}^{n} = 1.*\beta_i$ [17]

торов длины $2n$. Для $1 \leqslant i \leqslant n$, i-я компонента выхода обращается в 1 тогда и только тогда, когда входное слово принадлежит языку $.*\alpha_i$; если же $n+1 \leqslant i \leqslant 2n$, то равенство i-й компоненты выхода 1 эквивалентно принадлежности входного слова языку $.*\beta_i$. Первые n компонент выхода блока 1 подаются на вход блока 2 и включают счетчики, отсчитывающие длины соответствующих слов β_i. Если требуемая длина достигнута, единичный сигнал передается на вход блока 3, вычисляющего конъюнкции выходов блока 1 с номерами от $n+1$ до $2n$ с соответствующими выходами блока 2. Выход конъюнкции подается на вход переключателя, запоминающего единичный сигнал. Таким образом, равенство выхода переключателя 1 эквивалентно выполнению следующих условий: во входном потоке встретилось слово α_i и не менее чем через $|\beta_i|$ тактов после этого во входном потоке встретилось слово β_i. Несложно заметить, что эти условия эквивалентны тому, что входное слово принадлежит языку $.*\alpha_i.*\beta_i.*$ [17].

Данная конструкция распознает язык $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$ корректно, а отсутствие экспоненциального взрыва доказано авторами работы [17] в приведенной далее теореме.

Теорема 1 (Об отсутствии экспоненциального взрыва [17]). 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.

Конструкция очевидным образом обобщается на классы языков с большим числом сомножителей. Структурным элементом конструкции, позволяющим решить отмеченные задачи, являются счетчики. Триггеры конструкции, отмечающие принятые слова типа α_i, фиксируют метасостояния КА.

Заключение

В направлении решения задачи экспоненциального взрыва через добавление специальных структурных элементов особо успешными стали некоторые идеи и алгоритмы, давшие начало направлениям исследовательской работы и практического применения в сигнатурном анализе.

Во-первых, к таким успешным решениям относится использование счетчиков. При этом находки с использованием счетчиков касаются принципиально разных регулярных языков.

Во-вторых, идея фиксировать метасостояния КА через хранение дополнительной информации о состоянии КА открыла целое направление поиска в решении проблемы экспоненциального взрыва для реализации практического поиска по сигнатурам.

В-третьих, правильная комбинация НДКА и ДКА позволяет использовать скорость ДКА и экономичность НДКА, что было продемонстрировано на практике несколькими исследовательскими группами.

В-четвертых, экономичное отношение к КА в плане его расположения к быстрой и медленной памяти позволяет учитывать эмпирический опыт накопленного вредоносного трафика в системах обнаружения вторжений и позволяет значительно сократить потребности в быстрой памяти для подобного рода сканеров на основе КА.

Приведенные решения оптимизации ресурсов часто требуют полного перебора для нахождения оптимального решения. Кроме того, предложенные решения не всегда пластичны с точки зрения изменения регулярного языка для систем обнаружения вредоносного трафика, и требуют полной пересборки КА и перенастройки структурных элементов для адаптации к изменению набора вредоносных сигнатур.

Следует отметить, что решение проблемы экспоненциального взрыва для некоторых классов регулярных языков имеет существенную практическую пользу при внедрении предложенных математических решений в работающие системы анализа трафика на вредоносность.

Автор выражает благодарность А. В. Галатенко за комментарии и добавления, позволившие сделать работу сильнее.

Список литературы

1. **Кудрявцев В. Б., Алешин С. В., Подколзин А. С.** Введение в теорию автоматов, Москва: Наука, 1985, 320 с.

2. **Кудрявцев В. Б., Гасанов Э. Э. Подколзин А. С.** Основы теории интеллектуальных систем, М.: МАКС Пресс, 2016. 612 с.

3. **Документация** для Perl-совместимых регулярных выражений. URL://http://perldoc.perl.org/perlre.html

4. **Хопкрофт Дж., Мотвани Р., Ульман Дж.** Введение в теорию автоматов, языков и вычислений, М.: Вильямс, 2017. 528 с.

5. **Rabin M. O., Scott D.** Finite automata and their decision problems // IBM J. Research and Development. 1959. Vol. 3, No. 2. P. 115—125.

6. **Лупанов О. Б.** О сравнении двух типов конечных источников // Проблемы кибернетики. 1963. № 9. С. 321—326.

7. **Александров Д. Е.** Об оценках мощности некоторых классов регулярных языков // Дискретная математика. 2015. Том 27, № 2. С. 3—21.

8. **Александров Д. Е.** Об уменьшении автоматной сложности за счет расширения регулярных языков // Программная инженерия. 2014. № 11. С. 26—34.

9. **Бернадотт А.** Модификация конечного автомата через применение алгоритмов сжатия // Интеллектуальные системы. Теория и приложения. 2020. Том 24, № 3. С. 25—41.
10. **Kumar S., Chandrasekaran B., Turner J., Varghese G.** Curing regular expressions matching algorithms from insomnia, amnesia, and acalculia // ANCS '07 Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems. 2008. P. 155—164. DOI: 10.1145/1323548.1323574.
11. **Antonatos S., Anagnostakis K., Markatos E.** Generating realistic workloads for network intrusion detection systems // Proceedings of the 4th international workshop on Software and performance, ACM SIGSOFT Software Engineering Notes. 2004. Vol. 29, No. 1. P. 207—215. DOI: 10.1145/974044.974078.
12. **Zhang X., Liu H., Wang B., Huang J., Han X.** Generating realistic network traffic and interactive application workloads using container technology // Conference: 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017. P. 1021— 1025. DOI: 10.1109/ICCSN.2017.8230265.
13. **Liu A. X., Norige E., Kumar S.** A few bits are enough — ASIC friendly regular expression matching for high speed network security systems // 21st IEEE Int. Conf. Netw. Protocols (ICNP). 2013. P. 1—10.
14. **Smith R., Estan C., Jha S.** XFA: Faster Signature Matching with Extended Automata // IEEE Symposium on Security and Privacy. 2008. P. 187—201.
15. **Liu C., Wu J.** Fast Deep Packet Inspection with a Dual Finite Automata // IEEE Transactions on Computers. 2013. Vol. 62, No. 2. P. 310—321. DOI: 10.1109/TC.2011.231.
16. **Becchi M., Crowley P.** Extending finite automata to efficiently match Perl-compatible regular expressions // CoNEXT. ACM, 2008. P. 25—37.
17. **Бернадотт А., Галатенко А. В.** Аппаратная конструкция для решения проблемы экспоненциального взрыва для одного класса регулярных языков // Интеллектуальные системы. Теория и приложения. 2019. Том 23, № 4. С. 27—38.

Structural Modification of the Finite State Machine to Solve the Exponential Explosion Problem

Bernadotte A., bernadotte.alexandra@intsys.msu.ru,
Department of Information Technologies and Computer Sciences, National University of Science and Technology MISIS (NUST MISIS), Moscow, 119049, Russia Federation,
Faculty of Mechanics and Mathematics, Moscow State University, Moscow, 119991, Russia Federation

Corresponding author:

Alexandra Bernadotte, Associate Professor,
Department of Information Technologies and Computer Sciences, National University of Science and Technology MISIS (NUST MISIS), Moscow, 119049, Russian Federation,
Faculty of Mechanics and Mathematics, Moscow State University, Moscow, 119991, Russian Federation
E-mail: bernadotte.alexandra@intsys.msu.ru

Received on July 07, 2022
Accepted on August 10, 2022

In many modern applications, such as intrusion detection and prevention systems, expert knowledge can be formalized in the form of regular expressions. After this formalization, a finite automaton checks whether a word belongs to a regular language.

Deterministic finite automata have an optimal time complexity, but the number of automaton states (space complexity) can grow exponentially with the length of the regular expression. At the same time, the time complexity is the main disadvantage of non-deterministic finite automata. Therefore, reducing spatial complexity while maintaining low time complexity is highly relevant. In applied problems of regular language recognition, a significant problem is the problem of the exponentially growing number of states of the recognizing deterministic finite automaton depending on the length of regular expressions of the recognized language — the exponential explosion problem.

There are the following three main approaches to solving this problem using finite automata: 1) restriction on signatures given by experts; 2) regular language modification — this approach assumes the appearance in the solution of a practical problem of recognizing errors of the first and second kind; 3) finite automata modification without recognizing regular language changing. The third approach can be implemented as a finite automata modification through compression algorithms and particular structural elements.

The paper presents a review of modern solutions, the main idea of which is the transition from an abstract finite automaton, represented by a table-specified function, to a structural automaton that combines the abstract part stored in the memory and various structural elements such as bit arrays and counters.

Some ideas and algorithms have become especially successful when solving the exponential explosion problem by adding special structural elements.

First, such successful solutions include the use of counters. Second, the idea of storing additional information about the state machine. Third, the combination of non-deterministic and deterministic finite automata. Fourth, the

economic attitude to the state machine regarding its location to fast and slow memory allows us to consider the empirical experience of accumulated malicious traffic in intrusion detection systems.

Keywords: *finite state machine, exponential explosion problem, network intrusion detection systems, structural complexity, complexity reduction, regular languages, regular expressions*

For citation:

Bernadotte A. Structural Modification of the Finite State Machine to Solve the Exponential Explosion Problem, *Programmnaya Ingeneria,* 2022, vol. 13, no. 9, pp. 449—461.

DOI: 10.17587/prin. 13.449-461

References

1. **Kudryavtsev V. B., Aleshin S. V., Podkolzin A. S**. *Introduction to Automata Theory*, Moscow, Nauka, 1985, 320 p. (in Russian).

2. **Kudryavtsev V. B., Gasanov E. E., Podkolzin A. S.** *Fundamentals of the theory of intelligent systems*, Moscow, MAKS Press, 2016, 612 p. (in Russian).

3. **Documentation** for Perl-compatible regular expressions, available at: http://perldoc.perl.org/perlre.html

4. **Hopcroft J., Motwani R., Ulman J.** *Introduction to the Theory of Automata, Languages and Computing*, Moscow, Williams, 2017, 528 p. (in Russian).

5. **Rabin M. O., Scott D.** Finite automata and their decision problems*, IBM J. Research and Development*, 1959, vol. 3, no. 2, pp. 115—125 (in Russian).

6. **Lupanov O. B.** Comparison of two types of finite sources, *Problems of Cybernetics*, 1963, vol. 9, pp. 321—326 (in Russian).

7. **Aleksandrov D. E.** Estimates for the cardinality of some classes of regular languages*, Discrete Mathematics*, 2015, vol. 27, no. 2, pp. 3—21 (in Russian).

8. **Aleksandrov D. E.** On the reduction of automaton complexity by extending regular languages, *Programmnaya Ingeneria*, 2014, no. 11, pp. 26—34 (in Russian).

9. **Bernadotte A.** Modification of a finite automaton through the use of compression algorithms, *Intelligent systems. Theory and Applications*, 2020, vol. 24, no. 3, pp. 25—41 (in Russian).

10. **Kumar S., Chandrasekaran B., Turner J., Varghese G.** Curing regular expressions matching algorithms from insomnia, amnesia, and acalculia, *ANCS'07 Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems*, 2008, pp. 155—164. DOI: 10.1145/1323548.1323574.

11. **Antonatos S., Anagnostakis K., Markatos E.** Generating realistic workloads for network intrusion detection systems, *Proceedings of the 4th international workshop on Software and performance, ACM SIGSOFT Software Engineering Notes*, 2004, vol. 29, no. 1, pp. 207—215. DOI: 10.1145/974044.974078.

12. **Zhang X., Liu H., Wang B., Huang J., Han X.** Generating realistic network traffic and interactive application workloads using container technology, *Conference: 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, 2017, pp. 1021—1025. DOI: 10.1109/ICCSN.2017.8230265.

13. **Liu A. X., Norige E., Kumar S.** A few bits are enough — ASIC friendly regular expression matching for high speed network security systems, *21st IEEE Int. Conf. Netw. Protocols (ICNP)*, 2013, pp. 1—10.

14. **Smith R., Estan C., Jha S.** XFA: Faster Signature Matching with Extended Automata, *IEEE Symposium on Security and Privacy*, 2008, pp. 187—201.

15. **Liu C., Wu J.** Fast Deep Packet Inspection with a Dual Finite Automata, *IEEE Transactions on Computers*, vol. 62, no. 2, 2013, pp. 310—321. DOI: 10.1109/TC.2011.231.

16. **Becchi M., Crowley P.** Extending finite automata to efficiently match Perl-compatible regular expressions, *CoNEXT*. ACM, 2008, pp. 25—37.

17. **Bernadotte A., Galatenko A. V.** A hardware design for solving the exponential explosion problem for a class of regular languages, *Intelligent Systems. Theory and Applications*, 2019, vol. 23, no. 4, pp. 27—38 (in Russian).