Результаты приведены в таблице (строки ШЗ-Ф), где представлены также результаты ПЛИС-реализации процедур шифрования и расшифрования шифра Закревского на основе перестраиваемого автомата, заданного таблицами переходов и выходов (строки ШЗ-Т), взятые из [1], и результаты реализации шифра AES [4]. Стоит отметить, что в работе [1] исследуется перестраиваемый автомат, у которого $n = 16$, $m = 8$, а длина ключа 123 бита. В данной работе $n = m = 16$ и длина ключа увеличена до 244 бит.

Сравнение ПЛИС-реализаций шифра Закревского на основе перестраиваемого автомата при табличном и формульном задании

Шифр	Ресурсоёмкость, Slices (S)	Производительность, Мбит/с (T)	Коэффициент эффективности T/S
ШЗ-Т (шифрование)	370	298	0,805
ШЗ-Т (расшифрование)	365	269	0,737
ШЗ-Ф (шифрование)	397	349	0,879
ШЗ-Ф (расшифрование)	398	366	0,920
AES	163	208	1,276

Из таблицы видно, что несмотря на некоторое усложнение конструкции (большее число состояний, большая длина ключа), производительность шифра Закревского на основе перестраиваемого автомата возросла на 17–36 %, при этом ресурсоёмкость увеличилась незначительно. Коэффициент эффективности реализации также увеличился, хотя и не достиг значения этой величины для AES. Однако AES опережает шифр Закревского на основе перестраиваемого автомата только за счёт меньшей ресурсоёмкости, что является существенным только для ПЛИС с небольшой логической ёмкостью (количеством вентилей).

В целом, проведённые исследования показывают, что аппаратная реализация шифра Закревского на основе перестраиваемого автомата, заданного формулами, пригодна к использованию на практике.

ЛИТЕРАТУРА

1. *Ковалев Д. С.* Реализация на ПЛИС шифра Закревского на основе перестраиваемого автомата // Вестник Сибирского государственного аэрокосмического университета им. акад. М. Ф. Решетнёва. 2014. № 1 С. 16–18.
2. *Закревский А. Д.* Метод автоматической шифрации сообщений // Прикладная дискретная математика. 2009. № 2. С. 127–137.
3. *Тренькаев В. Н.* Реализация шифра Закревского на основе перестраиваемого автомата // Прикладная дискретная математика. 2010. № 3. С. 69–77.
4. *Rouvroy G., Standaert F. X., Quisquater J. J., and Legat J. D.* Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications // Proc. Intern. Conf. Inform. Technology: Coding and Computing. 2004. V. 2. P. 583–587.

УДК 519.7

ПРИМЕНЕНИЕ КОНЕЧНОГО АВТОМАТА ДЛЯ ОДНОВРЕМЕННОГО ПОИСКА НЕСКОЛЬКИХ ДВОИЧНЫХ ШАБЛОНОВ В ПОТОКЕ ДАННЫХ

И. В. Панкратов

Рассматривается задача поиска булевых векторов в потоке данных. Предлагается метод построения конечного автомата, который ищет одновременно несколько векторов, совершая только две простые операции на каждый бит или группу

битов. При этом с увеличением количества искомых шаблонов объём требуемой памяти растёт медленнее, чем суммарная длина шаблонов, а трудоёмкость не изменяется совсем. Приводится оценка количества состояний автомата.

Ключевые слова: *поиск битовых последовательностей, поиск подстроки.*

Рассматриваемую задачу можно сформулировать так. Имеем двоичную последовательность (поток данных) и набор булевых векторов (далее называемых *шаблонами*) V_1, V_2, …, V_n длин k_1, k_2, …, k_n соответственно. Необходимо найти вхождения всех шаблонов в последовательность.

В работе предлагается строить конечный автомат, на вход которого подаются биты или байты потока, а на выходе получается информация о найденных в потоке шаблонах. На каждый бит или байт потока автомат совершает две простейших операции: изменение состояния по таблице переходов и определение выхода по таблице выходов. При этом автомат осуществляет одновременный поиск произвольного количества шаблонов любых длин.

Опишем алгоритм построения битового автомата.

Входной алфавит автомата — $\{0, 1\}$.

Выходной алфавит — множество булевых векторов $\{0, 1\}^n$, где каждому биту сопоставлен шаблон из набора, и этот бит принимает значение 1, если шаблон закончился в текущей позиции.

Строится автомат по индукции. Сначала вводится нулевое состояние, соответствующее ситуации, когда не найден ни один префикс ни одного шаблона. Затем алгоритм поочерёдно обрабатывает все состояния условной подачей на вход автомата 0 и 1, попутно сохраняя новые полученные состояния и строя таблицы переходов и выходов автомата. Обработав таким образом все состояния автомата, получаем таблицы переходов и выходов, а также таблицу соответствия номеров состояний их описаниям: состоянию s сопоставляется набор $T_s = (T_{s,1}, T_{s,2}, \ldots, T_{s,n})$, где $T_{s,j}$ содержит множество длин найденных префиксов шаблона номер j.

Обозначим l-й бит вектора V через $V[l]$, биты нумеруем с нуля. В выходном векторе F бит $F[j-1]$ соответствует j-му шаблону.

Алгоритм построения битового поискового автомата

Вход: набор булевых векторов (шаблонов) V_1, V_2, …, V_n, их длины k_1, k_2, …, k_n
Выход: функции переходов ψ и выходов φ поискового автомата

1. Запоминаем начальное состояние $T_0 := (\varnothing, \varnothing, \ldots \varnothing)$, $s := 0$.
2. Обрабатываем состояние s условной подачей нуля и единицы. Подаваемый бит обозначим b. Сначала зададим $b := 0$.
3. Строим новое состояние T, в которое автомат должен перейти после подачи бита b в состоянии T_s, и соответствующий выходной символ — вектор F; сначала полагаем $F := 00\ldots0 = 0^n$.
   Для каждого j от 1 до n:
   3.1. Зададим $A := T_{s,j} \cup \{0\}$, $B_j := \varnothing$.
   3.2. **Для всех** $l \in A$ **если** $b = V_j[l]$, **то** $B_j := B_j \cup \{l+1\}$.
   Теперь множество B_j содержит длины всех найденных префиксов вектора V_j после подачи бита b в состоянии T_s.
   3.3. **Если** $k_j \in B_j$, **то** вектор V_j найден; полагаем $B_j := B_j \setminus \{l+1\}$, $F[j-1] := 1$.

4. Получили набор $T := (B_1, B_2, \ldots, B_n)$, описывающий следующее состояние, и выходной символ автомата — вектор F. Ищем набор T среди имеющихся наборов $T_0, \ldots, T_s$.
   Если нашли, что $T = T_h$, **то** присваиваем $s' := h$;
   если такого состояния ещё нет, **то** добавляем его в таблицу в новую ячейку. Пусть номер этой ячейки s'. Тогда состояние $T_{s'} = T$.
5. Запоминаем $\psi(s, b) := s'$, $\varphi(s, b) := F$.
6. **Если** $b = 0$, **то** $b := 1$ и переход на шаг 3.
7. $s := s + 1$. **Если** в таблице есть состояние T_s, **то** переход на шаг 2.
8. Ответ: функции ψ и φ.

Имея автомат, принимающий на вход биты, можно построить автомат, принимающий на вход сразу пачки битов, например байты или полубайты. Будем называть такой автомат *байтовым*, а его входные векторы — байтами. Множество состояний у него такое же, как у битового автомата; входной алфавит $\{0, 1\}^q$, где q — число битов в байте; выходной алфавит сложнее, поскольку в байтовом автомате возможно нахождение сразу нескольких вхождений одного шаблона в различных позициях.

При наличии битового автомата таблицы переходов и выходов байтового автомата строятся просто: по очереди обрабатываются все состояния автомата и все возможные значения байта. Для каждой пары входного байта и состояния анализ происходит так: битовый автомат устанавливается в соответствующее состояние и на его вход побитно подаётся входной байт. Все выходные сигналы автомата собираются вместе и запоминаются с учётом того, при подаче какого бита был получен выходной сигнал. Новое состояние байтового автомата должно соответствовать состоянию, в которое перешёл битовый автомат после подачи всех битов байта.

Оценим количество состояний автомата.

У автомата, который ищет n векторов с длинами k_1, k_2, …, k_n, количество состояний ограничено сверху величиной $\sum\limits_{j=1}^{n} k_j - n + 1$. Эта оценка достижима; в частности, автомат, ищущий один шаблон длины k, всегда имеет ровно k состояний.

Таким образом, предлагаемый поисковый автомат позволяет с очень высокой эффективностью одновременно искать в потоке данных несколько двоичных шаблонов. С увеличением количества шаблонов объём требуемой памяти растёт медленнее, чем их суммарная длина, а трудоёмкость не изменяется совсем.

Полностью результаты представлены в [1].

ЛИТЕРАТУРА

1. *Панкратов И. В.* Одновременный поиск нескольких двоичных шаблонов в потоке с помощью конечного автомата // Прикладная дискретная математика. 2014. № 2. С. 119–125.