УДК 519.712

К.В. ШАХБАЗЯН, Ю.Г. ШУКУРЯН

ВХОЖДЕНИЯ В МОНОИДАХ ТРЕКОВ

Ключевые слова: *сравнение с образцом, моноид треков, граф зависимости.*

ВВЕДЕНИЕ

Для двух треков трека-объекта t и трека-образца p, принадлежащих заданному свободному частично-коммутативному моноиду $M(\Sigma, D)$ треков, ставятся следующие задачи.

• Определить, входит ли трек-образец $p \in M(\Sigma, D)$ в трек-объект $t \in M(\Sigma, D)$. Трек p входит в трек t, если граф зависимости p можно получить из графа зависимости t удалением некоторых вершин и примыкающих к ним дуг.

• Подсчитать для заданного слова-представления y трека t и целого положительного w число w-трековых окон слова y, в которые входит трек-образец p.

• Подсчитать число минимальных факторов трека-объекта t, в которые входит трек-образец p.

Проблема сравнения некоторого объекта (target) с образцом (pattern) для массивов структурированных данных привлекает большое внимание, и ее решение важно для развития методов вскрытия данных (date mining). Одним из наиболее общих формализмов для моделирования структурированных данных являются графы. Однако при рассмотрении графов общего вида возникают проблемы, связанные с эффективностью. Как правило, подобные задачи имеют очевидные переборные алгоритмы и поэтому здесь речь идет о нахождении эффективных алгоритмов.

Проблема сравнения объекта с образцом ставится обычно в одной из следующих постановок:

1) найти вхождение образца в объект — массив данных — в виде фактора этого массива, т.е. целиком, без прерываний;

2) найти вхождение образца в объект в виде подпоследовательности, т.е. с возможными прерываниями;

3) при заданном размере окна — фактора объекта — найти вхождения образца в виде подпоследовательностей, ограниченных временными условиями, т.е. вхождения в окна-факторы заданного размера;

4) подсчитать число окон, содержащих образец.

Последняя постановка тесно связана с проблемой частых эпизодов (frequent episodes), где под эпизодом понимается вхождение образца внутрь окна, т.е. внутрь заданного интервала времени.

Проблемы сравнения с образцом в свободных моноидах (для слов) разрабатывалась многими авторами. Широко известен алгоритм Кнута, решающий задачу поиска образца как фактора. Работы [1–7] эффективно решают задачи вхождения слова-образца в окно слова-объекта в качестве подпоследовательности — при заданном размере окна.

Задачи вхождения, относящиеся к деревьям, рассматривались в [7–10]. Здесь предложены эффективные алгоритмы решения задачи о частых эпизодах при различных понятиях окна в деревьях.

Задача вхождения трека-образца в объект, как фактора, в моноидах треков рассмотрена в [11], где предложен линейный алгоритм ее решения.

Моноиды треков (конечно-порожденные частично коммутативные моноиды) — признанный инструмент для описания структур событий в параллельных и распределенных системах [13], а также используются при исследовании задач теоретического программирования [13, 14].

© К.В. Шахбазян, Ю.Г. Шукурян, 2010

Данная работа предлагает эффективные алгоритмы для решения некоторых задач вхождения и подсчета окон в моноидах треков. Она состоит из четырех частей. После предварительных сведений во второй части предлагается два алгоритма для проблемы вхождения, когда образец p и основной трек t представлены словами $x \in p$ и $y \in t$. Первый алгоритм заключается в построении автомата по заданному слову $x \in p$. Этот автомат сканирует слово $y \in t$, допуская слово y, если t содержит трек p. Второй алгоритм параллельно сканирует слова, представляющие трек-образец и трек-объект, читая каждый символ x один раз, но трансформируя y и возвращаясь к его началу. Сложность алгоритма — $O(|t||p|)$. Третья часть представляет алгоритм, который подсчитывает число w-трек-окон заданного представления-слова y трека t, включающих трек-образец. В последней части описан алгоритм, который подсчитывает число минимальных факторов t, включающих образец p.

ПРЕДВАРИТЕЛЬНЫЕ СВЕДЕНИЯ

В этом разделе приводятся основные понятия и обозначения, подробнее см. [12].

Пусть Σ — конечный алфавит, $D \subseteq \Sigma \times \Sigma$ — рефлексивное и симметричное отношение зависимости, $I = (\Sigma \times \Sigma) \setminus D$ — отношение независимости и перестановочности. Отношение I индуцирует отношение эквивалентности $\underset{I}{\approx}$ на Σ^*. Два слова $x, y \in \Sigma^*$ эквивалентны относительно $\underset{I}{\approx}$, если существует последовательность $z_1, \ldots, z_k$ слов таких, что $x = z_1$, $y = z_k$ и для всех i $(1 \le i < k)$ существуют слова z'_i, z''_i и буквы a_i, b_i, удовлетворяющие условиям $z_i = z'_i a_i b_i z''_i$, $z_{i+1} = z'_i b_i a_i z''_i$ и $(a_i, b_i) \in I$, т.е. два слова эквивалентны тогда и только тогда, когда одно может быть получено из другого допустимыми перестановками соседних независимых букв. Тогда множество $M(\Sigma, D)$ — трековый моноид, его элементами являются классы эквивалентных слов из Σ^* по отношению $\underset{I}{\approx}$, называемые треками. На треках определено умножение. Трек t обозначается $[x]$ для любого представляющего слова $x \in t$. Длина $|t|$ трека t есть длина любого его представления $x \in t$. Для любого $x \in \Sigma^*$ через $|x|$ обозначаем длину x, а $|x|_a$ — число вхождений буквы a в слово x.

Используем два представления треков: в виде графов зависимости и набора слов. Любой трек $t \in M(\Sigma, D)$ имеет единственное представление в виде отмеченного направленного ациклического графа G_t, обозначающего частичный порядок. Граф G_t определяется рекурсивно: G_t — пустой граф, $G_{t[a]}$ получается из графа G_t добавлением к нему вершины, отмеченной буквой a и новыми дугами, ведущими к этой вершине из всех вершин G_t, отмеченных символами, от которых зависит буква a. Граф G_t называется графом зависимости трека t. С графом G_t ассоциировано множество слов, индуцируемых всеми причинными порядками на G_t. Это множество слов формирует трек t.

Рассмотрим второе представление трека — с помощью набора слов. Пусть $\{\Sigma_1, \ldots, \Sigma_m\}$ — покрытие алфавита зависимости (Σ, D) кликами, т.е. семейство подмножеств Σ таких, что

$$\bigcup_{i=1}^{m} \Sigma_i = \Sigma, \; \Sigma_i \times \Sigma_i \subseteq D \; (i = 1, 2, \ldots, m)$$
$$(a, b) \in D \Leftrightarrow \exists\, i: \; a, b \in \Sigma_i .$$

Тогда трек $t \in M(\Sigma, D)$ может быть представлен m-кой слов $\pi(t) = \{\pi_1(t), \ldots, \pi_m(t)\}$, где $\pi_i(t) \in \Sigma_i^*$, и $\pi_i(t)$ — проекция произвольного слова $y \in t$ на Σ^* [12].

Считается, что для заданных двух треков: $p, t \in M(\Sigma, D)$ p есть префикс t, если $t = pq$, где $q \in M(\Sigma, D)$, $Pref(t)$ обозначает множество префиксов трека t. Если $t = pqr$, где $p, q, r \in M(\Sigma, D)$, то q называется фактором t. Заметим, что если $p \in Pref(t)$, то существуют $x, y \in \Sigma^*$, $t = [y]$, $p = [x]$ и x есть префикс слова y.

Пусть трек t задан своим представлением $\pi(t)$. Тогда любой префикс s трека t (относительно $\pi(t)$) можно представить массивом целых чисел $\underline{s} = (k_1, \ldots, k_m) \in \in N^m$, имея ввиду, что $k_i = |\pi_i(s)|$, $(i = 1, 2, \ldots, m)$, и $\pi_i(s)$ есть префикс длины k_i слова $\pi_i(t)$. Таким образом, $(0, 0, \ldots, 0) = \underline{e}$ представляет пустой префикс e, в то время как полный трек t представляется набором $\underline{t} = (n_1, n_2, \ldots, n_m)$, где $n_i = |\pi_i(t)|$ [12].

Граф префиксов трека t есть ориентированный ациклический граф $G^{Pref\,(t)} = (V, U, \mu)$ с множеством отмеченных дуг, где $V = \{\underline{s} \,/\, s \in Pref\,(t)\}$, U — множество упорядоченных пар $(\underline{s}, \underline{r})$, μ — отмечающая функция для дуг и дуга $(\underline{s}, \underline{r})$ имеет метку $\mu((\underline{s}, \underline{r})) = a$, $a \in \Sigma$, если $s, r \in Pref\,(t)$, $s = r[a]$, где $[a]$ — трек, содержащий единственное однобуквенное слово a.

Для моноида $M = M(\Sigma, D)$ M-автомат $A = (M, Q, \delta, q_0, F)$ представляется конечным множеством Q состояний, начальным состоянием $q_0 \in Q$, подмножеством $F \subseteq Q$ заключительных состояний, функцией переходов δ из $Q \times M$ в Q, удовлетворяющими следующим условиям:

$$\forall q \in Q\colon \delta(q, e) = q \ \ (e \text{ — пустой трек}),$$

$$\forall q \in Q,\ \forall t_1, t_2 \in M(\Sigma, D)\colon \delta(q, t_1, t_2) = \delta(\delta(q, t_1), t_2).$$

Множество $T \subseteq M = M(\Sigma, D)$, распознаваемое автоматом A, определяется следующим образом: $T = \{t \in M \,/\, \delta(q_0, t) \in F\}$.

Автомат Зеленки (асинхронный автомат относительно M) есть M-автомат $A = (M, Q, \delta, q_0, F)$, который удовлетворяет следующим дополнительным условиям:

1) состояние Q есть прямое произведение $Q = \prod_{i=1}^{n} Q_i$;

2) с каждым $a \in \Sigma$ ассоциировано множество индексов $K(a) \subseteq \{i \,/\, i \in \{1, 2, \ldots, n\}\}$, такое что $K(a) \cap K(b) = \varnothing$ тогда и только тогда, когда $(a, b) \in I$;

3) функция переходов δ задается набором отображений $\{\delta_a : \Pi_{i \in K(a)} Q_i \to \Pi_{i \in K(a)} Q_i\}_{a \in \Sigma}$.

Далее фиксируем моноид $M = M(\Sigma, D)$, покрытие кликами $\{\Sigma_1, \ldots, \Sigma_m\}$ алфавита (Σ, D) и целое m — число клик в покрытии.

ВХОЖДЕНИЕ ТРЕКА-ОБРАЗЦА В ТРЕК-ОБЪЕКТ

Определим понятие вхождения треков и исследуем задачу вхождения.

Определение 1. Трек p входит в трек t, если для некоторых $t_i, p_i \in M(\Sigma, D)$, $i = 0, 1, \ldots, n$, справедливо

$$t = t_0\ p_1\ t_1 \ldots t_{n-1}\ p_n\ t_n\,, \tag{1}$$

$$p = p_1\ p_2 \ldots p_n\,. \tag{2}$$

В этом случае считается, что t содержит p и записываем $p \subset t$. Представление (1) называем вхождением.

Пусть r — некоторый префикс трека $t\colon r \in Pref(t)$, $R_t(r)$ — множество меток дуг, выходящих из узла $\underline{r}$ в графе префиксов $G^{Pref(t)}$. Следовательно, $a \in R_t(t) \Leftrightarrow s = r[a] \in Pref(t)$.

С каждой буквой $a \in \Sigma$ свяжем множество индексов $\Im(a) = \{i \in \{1, 2, \ldots, m\} \mid a \in \Sigma_i\}$ таких, что $(a, b) \in I \Leftrightarrow \Im(a) \cap \Im(b) = \varnothing$.

Утверждение 1. Пусть трек t задан своим представлением $\pi(t) = \{\pi_1(t) \ldots \pi_m(t)\}$, где $\pi_i(t) = \pi_{i,1}(t) \ldots \pi_{i,n_i}(t) \in \Sigma^*$.

1.1. Если $r \in Pref(t)$, $z \in \Sigma^*$, то $s = r[z] \in Pref(t)$ тогда и только тогда, когда существует направленный путь в $G^{Pref}(t)$ из $\underline{r}$ в $\underline{s}$, который несет слово z.

Путь в $G^{Pref}(t)$ максимальный тогда и только тогда, когда он соответствует некоторому слову $z \in t$.

1.2. Для любого $r \in Pref(t)$, если $\underline{r} = (k_1, \ldots, k_m)$, множество $R_t(r)$ может быть вычислено как $R_t(r) = \{a \,/\, i \in \mathfrak{I}(a) \Rightarrow \pi_{i, k_i+1} = a\}$.

1.3. Для любого $r \in Pref(t)$ и $a \in R_t(r)$, если $\underline{r} = (k_i, \ldots, k_m)$, представление $\underline{s} = \underline{r[a]} = (k'_i, \ldots, k'_m)$ может быть вычислено как

$$k'_i = k_i \text{, если } i \notin \mathfrak{I}(a), \quad k'_i = k_i + 1, \text{ если } i \in \mathfrak{I}(a), \quad i = 1, 2, \ldots, m.$$

1.4. Для любых $r \in Pref(t)$ и $(a, b) \in I$, где $a \in R_t(r)$ и $s = r[a]$, справедливо

$$b \in R_t(s) \Leftrightarrow b \in R_t(t), \quad r[a][b] = r[b][a].$$

Доказательство следует из определения $G^{Pref(t)}$.

Пусть $p, t \in M(\Sigma, D)$. Построим два M-автомата, решающих проблему вхождения. Первый автомат $A(p)$ определяется образцом p и распознает трек-объект $t \in M(\Sigma, D)$ тогда и только тогда, когда p входит в t.

Второй автомат $B(t)$ определяется треком-объектом t и распознает образец $p \in M(\Sigma, D)$ тогда и только тогда, когда p входит в t.

Допустим, что $|t| > |p| > 1$ и что треки p и t заданы своими словарными представлениями $x \in p$, $y \in t$.

Утверждение 2. Для любого трека $p \in M(\Sigma, D)$ существует детерминированный асинхронный автомат Зеленки $A(p)$, который распознает $t \in M(\Sigma, D)$ тогда и только тогда, когда $p \subset t$.

Доказательство. Для построения M-автомата $A(p)$ рассмотрим граф префиксов для $p : G^{Pref}(p) = (V(p), E, \mu)$, где $V(p) = \{\underline{s} \,/\, s \in Pref(p)\}$. Добавим к множеству дуг E множество отмеченных петель $E' = \{(\underline{s}, \underline{s})_b \,/\, \underline{s} \in V(p), b \notin R_t(s), b \in \Sigma\}$, чтобы получить граф $G'(p)$:

$$G'(p) = (V(p), E \cup E', \mu'),$$

$$\mu'((\underline{s}, \underline{r})) = \mu((\underline{s}, \underline{r})), \text{ если } (\underline{s}, \underline{r}) \in E, \qquad \mu'((\underline{s}, \underline{s})_b) = b.$$

Тогда $A(p) = (M, V(p), \delta, q_0 = e, F = \{\underline{p}\})$, где δ определяется диаграммой переходов $G'(p) : \delta(\underline{s}, \underline{a}) = \underline{s[a]}$, если $a \in R_t(s)$, и $\delta(\underline{s}, a) = \underline{s}$, если $a \notin R_t(s)$. Из утверждения 1.1 заключаем, что автомат $A(p)$ решает настоящую задачу.

Заметим, что из утверждения 1.4 следует что для любого $s \in Pref(p)$ справедливо $sab = sba$, если $(a, b) \in I$. Следовательно, $A(p)$ — M-автомат. Очевидно, что $A(p)$ является также асинхронным автоматом Зеленки.

Утверждение 3. Существует online алгоритм для решения задачи вхождения $p \subset t$ со временной сложностью $O(m|t|)$ для любых $p, t \in M(\Sigma, D)$, представленных словами $x \in p$ и $y \in t$. Пространственная сложность — $O(m|p|)$.

Алгоритм 1 для задачи вхождения

Шаг 1. Обработка трека p: online сканированием строится проекция $\pi(p)$. Согласно утверждениям 1.2 и 1.3 этот шаг требует времени $O(|p|)$. (Построение автомата $A(p)$ не требуется.)

Шаг 2. Обработка трека t — моделирование работы автомата $A(p)$ на слове y. Сканируя *online* слово $y \in t$, алгоритм проходит путь на графе $G'(p)$ следующим образом.

Начиная с состояния $\underline{e} = (0, \ldots, 0)$, если состояние $\underline{s} = (k_1, \ldots, k_m) \in G'(p)$ достигнуто и текущая буква y есть a, то имеются следующие возможности:

1) $\underline{s} = \underline{p}$, тогда ВЫХОД: $p \subset t$;

2) a есть последняя буква слова y и $\underline{s[a]} \neq \underline{p}$, тогда ВЫХОД: $p \not\subset t$;

3) $a \in R_p(s)$; тогда переход к следующему состоянию $\underline{r} = \underline{s[a]}$;

4) $a \notin R_p(s)$; тогда следующее состояние остается прежним $\underline{s}$.

Конец алгоритма 1.

Этот алгоритм используется далее для последующих алгоритмов в разд. 3, 4.

Утверждение 4. Пусть $p, t \in M(\Sigma, D)$.

4.1. $p \subset t$ тогда и только тогда, когда граф зависимости трека-образца G_p есть подграф графа зависимости трека-объекта G_t.

4.2. $p \subset t$ тогда и только тогда, когда для любого слова $y \in t$ существует слово $x \in p$ такое, что x есть подпоследовательность слова y.

Доказательство. Ясно, что 4.2 следует из 4.1. Доказательство вытекает из определений.

Замечание. Из справедливости $p \subset t$ не следует, что для каждого слова $x \in p$ существует слово $y \in t$ такое, что x — подпоследовательность y. Это видно из следующего примера.

Пример. Рассмотрим моноид $M(\Sigma, D)$, где $\Sigma = \{a, b, c\}$, $D = \{(a, b), (b, c)\}$. Пусть $t = [abc]$. Трек t состоит из единственного слова $y = abc$. Если положить $p = [ac]$, то $ca \in p$, однако ca не является подпоследовательностью слова y.

Пусть $s, t \in M(\Sigma, D)$. Из утверждения 4.1 и определения вхождения $s \subset t$ тогда и только тогда, когда для любых слов $x = a_1 \ldots a_{|s|} \in s$, $y = b_1 \ldots b_{|t|} \in t$ существует вложение индексных множеств $\varphi_{x,y} : X_s \to X_t$. Здесь $X_s = \{1, \ldots, |s|\}$, $X_t = \{1, \ldots, |t|\}$ такое, что

1) $\varphi_{x,y}$ сохраняет метку: $a_i = b_{\varphi_{(x,y)}}(i)$;

2) $\varphi_{x,y}$ сохраняет порядок G_p: если $i <_{G_p} j$ и $(a_i, a_j) \in D$, то

$$\varphi_{x,y}(i) < \varphi_{x,y}(j). \tag{3}$$

Такие вложения называем корректными.

Пусть X — произвольное подмножество X_t. Обозначим $\overline{X} = \{k / \exists l \in X : k < l$, и $(y_k, y_l) \in D\}$.

Утверждение 5. Пусть $s, t \in M(\Sigma, D)$, $x = a_1 \ldots a_{|s|} \in s$, $y = b_1 \ldots b_{|t|} \in t$. Если $s \subset t$, $a \in \Sigma$, то $s[a] \subset t$ тогда и только тогда, когда

$$\exists i_0 \le |t| : b_{i_0} = a,\ i_0 \notin \varphi_{xy}(X_s) \cup \overline{\varphi_{xy}(X_s)}. \tag{4}$$

Доказательство. Допустим, $s[a] \subset t$. Согласно допущению для любого слова $x^a = a_1 \ldots a_j a a_{j+1} \ldots a_{|s|} \in s[a]$, где последнее вхождение буквы a выделено и для слова $y = b_1 \ldots b_{|t|} \in t$ существует корректное вложение $\varphi_{x^a, y} : X_{x^a} \to X_t$, где $X_{x^a} = \{1, \ldots, |s|+1\}$. Вычеркнем символ a из слова x^a. Очевидно результирующее слово $x = a_1 \ldots a_j\ a_{j+1} \ldots a_{|s|+1} \in s$ и, следовательно, вложение $\varphi_{x, y} : X_s \to X_y$, где $\varphi_{x, y}(i) = \varphi_{x^a, y}(i)$, если $i \le j$, и $\varphi_{x, y}(i) = \varphi_{x^a, y}(i+1)$, если $i > j$, корректно.

Положим $i_0 = \varphi_{x^a, y}(j+1)$. Допустим $i_0 \in \varphi_{x, y}(X_s) \cup \overline{\varphi_{x, y}(X_s)}$. Тогда (3) нарушено. Отсюда следует корректность (4).

Теперь допустим, что (4) верно. Тогда построим $\varphi_{x^a\ y}$ следующим образом:

$$\varphi_{x^a\ y}(i) = \varphi_{x,y}(i),\ \text{если}\ i < i_0,$$

$$\varphi_{x^a\ y}(i) = i_0,\ \text{если}\ i = i_0,$$

$$\varphi_{x^a\ y}(i) = \varphi_{x,y}(i-1),\ \text{если}\ i > i_0.$$

Отсюда следует, что $s[a] \subset t$.

Алгоритм 2 для задачи вхождения. Согласно утверждению 5 алгоритм сканирует слово $x = a_1 \ldots a_{|p|}$, читая каждый символ один раз и преобразует слово y в последовательность слов $y_\sigma = y, y_1, \ldots, y_{|p|}$. На i-м шаге ($i = 1, 2, \ldots, |p|$), когда читается a_i, слово y_{i-1} преобразуется следующим образом: если $y_{i-1} = b_1 \ldots b_n$, то следующее слово $y_i = b_1' \ldots b_n'$ получается из y_{i-1} удалением букв множества $\{b_j / (b_j, b_{i_0}) \in D\}$, $1 \le j \le i_0$, где b_{i_0} — первое вхождение буквы a_i в слово y_{i-1}, т.е. $a_i = b_{i_0}$.

Временна́я сложность алгоритма — $O(|p||t|)$, пространственная — $O(|t|+|p|)$.

Утверждение 6. Для любого трека $t \in M(\Sigma, D)$ существует детерминированный асинхронный автомат Зеленки $B(t)$, который распознает $p \in M(\Sigma, D)$ тогда и только тогда, когда $p \subset t$.

Доказательство. Для построения *M*-автомата $B(t)$ рассмотрим граф зависимости $G_t = (V_t, E_t, \lambda_t)$ и множество всех подмножеств множества V_t: $W_t = \{V' / V' \subset V_t\}$. Каждое подмножество представляет трек, входящий в t.

Пусть *M*-автомат $B(t) = (M, W_t, \delta, q_0, F)$, где $q_0 = V_t$, $F = W_t$, функция переходов определяется в соответствии с утверждением 5 следующим образом: если существует $v \in V'$ такое, что $\lambda(v) = a$, и если $v' <_{G_t} v \Rightarrow \lambda(v') \neq a$, то $\delta(V', a) = V' \setminus \{v' \le_{G_t} v, (\lambda(v'), \lambda(v)) \in D\}$.

Здесь $\le_G$ означает отношение предшествования в G_t. Из утверждения 5 заключаем, что $B(t)$ решает поставленную задачу.

Очевидно, что для любого $V' : \delta(V', [a][b]) = \delta(V', [b][a])$, т.е. $B(t)$ есть *M*-автомат. По теореме Зеленки существует синхронный автомат, распознающий $p \subset t$.

ПОДСЧЕТ *w*-ТРЕКОВЫХ ОКОН, СОДЕРЖАЩИХ ОБРАЗЕЦ

Задача. Даны два трека: $p, t \in M(\Sigma, D)$, представленных словами $x \in p$ и $y \in t$. Требуется подсчитать количество *w*-трековых окон, содержащих трек-образец p.

Определение 2. Пусть слово $y = b_1 \ldots b_n \in \Sigma^*$. Тогда *w*-окно размера w на слове y есть подслово $b_{i+1} \ldots b_{i+w}$ длины w. Трек $[b_{i+1} \ldots b_{i+w}]$ называется *w*-трековым окном размера w на слове y. Очевидно, что число *w*-окон, содержащих p, может быть различным для слов $y, y' \in t$, если $y \neq y'$.

Штампованный префикс трека $p \in M(\Sigma, D)$ есть упорядоченная пара $(\underline{r}, n)$, где $r \in Pref(p)$, $n \in N$. Конфигурация есть последовательность штампованных префиксов данного трека.

Утверждение 7. Существует алгоритм подсчета количества *w*-трековых окон слова $y \in t$, содержащих трек p. Временна́я сложность алгоритма — $O(mw|t|)$, пространственная — $O(|w||p|)$.

Идея алгоритма. Сканируя $y = b_1 \ldots b_n \in \Sigma^*$, алгоритм строит конфигурации C_i для каждой буквы b_i слова y. Штампованный префикс $(\underline{r}, u)$ $(r \in Pref(p))$ входит в конфигурацию C_i тогда и только тогда, когда трековое окно, т.е. трек $[b_{i-u+1} \ldots b_i]$ длины $u \le w$ содержит префикс r. По заданным C_i и b_{i+1} возможно подсчитать C_{i+1}. Если C_i содержит штампованный префикс $(\underline{p}, u)$, где $u \le w$, то *w*-трековое окно $[b_{i-w+1} \ldots b_i]$ содержит p.

Алгоритм подсчета *w*-трековых окон

Пусть $p, t \in M(\Sigma, D)$.

ВХОД — два слова: $x \in p$, $y \in t$.

ВЫХОД — *counter* (число *w*-трековых окон y, содержащих $[x]$)

Шаг 1. Предобработка $p = [x]$ — вычисление $\pi(p)$.

Инициализация: $counter := 0$; $C_0 := ((\underline{e}, 0))$.

Шаг 2. Обработка трека *t*.

for $i = 1, \ldots, |t|$

do

$a := b_i$; $C_i := ((\underline{e}, 0))$ (инициализация и вычисление C_i)

for all $(\underline{r}, u) \in C_{i-1}$

do

if $\underline{r[a]} = \underline{p}$ and $u = w - 1$ **then** $counter := counter + 1$

(*w*-трековое окно содержит *p*) **end if**;

if $a \in R_p(r)$, $u < w - 1$, **then** добавить штампованный

префикс $(\underline{r[a]}, u+1)$ к C_i, т.к. $y_{i-u+1} \dots y_i$ может быть началом требуемого w-трекового окна и подслово $y_{i-u+1} \dots y_i$ должно быть передано в C_{i+1} **end if**;
if $a \notin R_p(r)$, $u < w-1$ **then** добавить $(\underline{r}, u+1)$ к C_{i+1};
end do
end for all
if $a \in R_p(e)$ **then** добавить $(\underline{a}, 1)$ к C_i
(создать начало нового окна) **end if**
end do
end for.

ВЫХОД: *counter*

Конец алгоритма.

ПОДСЧЕТ ЧИСЛА МИНИМАЛЬНЫХ ФАКТОРОВ ОБЪЕКТА, СОДЕРЖАЩИХ ОБРАЗЕЦ

Пусть (1) и (2) верны, т.е. $p \subset t$. Фактор $f = p_1\, t_1 \dots t_{n-1}\, p_n$ трека t, содержащий p, называется минимальным, если каждый собственный фактор трека f не содержит p.

Задача. Даны два трека: $p, t \in M(\Sigma, D)$, представленных словами $x \in p$ и $y \in t$. Подсчитать число минимальных факторов трека t, содержащих трек p.

Утверждение 8. Существует алгоритм подсчета числа минимальных факторов трека t, содержащих трек p. Временна́я сложность алгоритма есть $O(m|t|^2)$, пространственная сложность — $O(m|t|)$.

Идея алгоритма. Пусть $t = qrs$ и $q, r, s, p \in M(\Sigma, D)$ и r — минимальный фактор трека-объекта t, содержащий трек-образец p. Тогда для каждого слова $y = b_1 \dots b_{|t|} \in t$ существует трековое окно (без ограничения на длину) $[b_k \dots b_i]$ такое, что $p \subset r \subset [b_k \dots b_i]$.

Очевидно, что число минимальных факторов трека t, если $t = [y]$, равно числу таких подслов слова y, собственные трек-подслова которых не содержат p. Таким образом, нужно найти и подсчитать количество таких минимальных подслов. Сканируя слово y, алгоритм ассоциирует с каждой буквой b_i слова y конфигурацию $C_i = (\underline{r_1}, \dots, \underline{r_l})$, $r_k \in Pref(p)$, $k = 1, \dots, l$, т.е. последовательность префиксов $p \in M(\Sigma, D)$ (не штампованных, как в предыдущей секции) таких, что $r_k = [b_{v_k} \dots b_i]$ и $v_1 < \dots < v_l < i$. Допустим, что для некоторого $j \in \{1, \dots, l\}$ справедливо

$$p \subset r_1, \dots, p \subset r_j, p \not\subset r_{j+1} \dots, p \not\subset r_l, \ 1 \le j \le l. \tag{5}$$

Тогда все $[b_{v_k} \dots b_i]$ для $k \le j$ содержат одинаковый минимальный трек-фактор трека t, трек-фактор y, содержащий минимальный трек-фактор t найден и счетчик должен быть увеличен на 1. При этом возможно рассматривать только трековые окна с $[b_k \dots b_l]$ с $b_k \in R_p(e)$.

Конфигурация C_{i+1} для j, удовлетворяющая (5), строится следующим образом:

If $C_i = (\underline{r_1}, \dots, \underline{r_l})$ **and** $b_{i+1} = a$, put $s_k = r_{j+k}[a]$ **if** $s_k = r_{j+k}[a] \in Pref(p)$ **and** $s_k = r_{j+k}$ **if** $r_{j+k}[a] \notin Pref(p)$ **for** $k = 1, \dots, l-j$. **Then**

$$C_{i+1} = (\underline{s_1}, \dots, \underline{s_{i-j}}) \text{ if } a \notin R_p(e),\ j < l,$$

$$C_{i+1} = (\underline{s_1}, \dots, \underline{s_{i-j}}, \underline{[a]}) \text{ if } a \in R_p(e),\ j < l,$$

$$C_{i+1} = \{\underline{e}\} \text{ if } j = l,\ a \notin R_p(e),$$

$$C_{i+1} = \{\underline{[a]}\} \text{ if } j = l,\ a \in R_p(e),$$

где e — пустой трек.

Сканируя слово y, можно подсчитать число минимальных факторов.

Замечание. Алгоритм подсчитывает только число минимальных факторов, но не сами минимальные факторы. Для того чтобы получить минимальные факторы t, необходимо дополнить алгоритм процедурой удаления из минимальных подслов лишних букв. Это увеличит сложность до $O(m|t|^3)$.

Алгоритм вычисления минимальных факторов

ВХОД: два слова: $x \in p$, $y \in t$.

ВЫХОД: *counter* — число минимальных факторов трека t, содержащих трек p.

Шаг 1. Предобработка $[x]$: вычисление $\pi(p)$.

Шаг 2. Инициализация $C_0 = (\underline{e})$; $counter := 0$;

for $i = 1, \ldots, |t|$ (для каждой буквы слова $y = b_1 \ldots b_{|t|}$ — построение C_{i+1}.)

$\varepsilon := 0$; $C_{i+1} := \varnothing$; $a := b_{i+1}$; (инициализация)

for all $s \in C_i$

do

if $\underline{p = s[a]}$ then $\varepsilon := 1$ (минимальный фактор найден) **else**

if $s[a] \in Pref(p)$ **then** add $\underline{s[a]}$ to C_{i+1} **else** add $\underline{s}$ to C_{i+1} **end**

if end if

end do

end for all

if $a \in R_p(e)$ **then** add $\underline{[a]}$ to C_{i+1}

else if $C_{i+1} = \varnothing$ **then** add $\underline{e}$ to C_{i+1} **end if end if**

$counter := counter + \varepsilon$

end for

Конец алгоритма.

Сложность: число префиксов в C_i ограничено $|t|$-длиной слова y. Следовательно, сложность алгоритма $O(m|t|^2)$.

ЗАКЛЮЧЕНИЕ

В настоящей статье приводятся алгоритмы решения следующих задач: решить, входит ли $p \in M(\Sigma, D)$ в $t \in M(\Sigma, D)$; подсчитать число w-трековых окон $y \in t$, которые содержат образец p; подсчитать число минимальных факторов трека t, которые содержат образец p.

СПИСОК ЛИТЕРАТУРЫ

1. Chi Y., Wang H., Yu P.S., Muntz R.R. Catch the moment: maintaining closed frequent itemsets over a date stream sliding window // Knowledge and Inform. Systems. — 2006. — **10**, N 3. — P. 265–294.
2. Li H-F., Shan M-K., Lee S-Y. DSM-FI: an efficient algorithm for mining frequent itemsets in data streams // Ibid. — 2008. — **17**, N 1.
3. Ba C., Ferrari M.H., Musicante M.A. Composing Web Services with PEWS : a trace-theoretical approach // 4-th Europ. Conf. on WEB Services. — 2006.
4. Mannila H., Toivonen H., Verkamo A. Discovering Frequent Episodes in Sequences // Proc. KDD Conf.— 1955.
5. Avellone A., Goldwurm M. Analysis of algorithms for the recgnition of rational and context-free trace languages // Theoretical Inform. and Appl. — 1998. — **32**. — P. 141–152.
6. Chi Y., Muntz R., Nijssen S., Kok J. Frequent Subtree Mining. — An Overview // Fundamenta Inform. — 2001. — **21**. — P. 161–198.
7. Crochemore M., Ilie L. Maximal repetitions in strings // J. of Comput. and System. Scie. — 2008. — **74.** — P. 796–807.
8. Boasson L., Cegielski P., Guessarian I., Matiyasevich Y. Window-Accumulated Subsequence Problem is linear // Annals of Pure and Appl. Logic. — 2001. — **113.** — P. 74–87.
9. Cegielski P., Guessarian I., Matiyasevich Y. Tree Inclusion Problems // RAIRO — Theoretical Inform. and Appl. — 2008. — **42.** — P. 5–20.
10. Guessarian I., Cegielski P. Tree Inclusions in Windows and Slices // CSIT Conf. — 2000.
11. Messner J. Pattern Matching in Trace Monoids // Symposium on Theoretical Aspects of Comput. Scie. — 1997.
12. Dikert V., Rozenberg G. The Book of Traces // Handbook of formal languages. — N.Y.: Springer-Verlag, 1997. — 568 p.
13. Letichevsky A.A. On the equivalence of automata over semigroup // Theoretic Cybernetics. — 1970. — **6.** — P. 3–71 (in Russian).
14. Shakhbazyan K.V., Shoukourian Yu.H. On process languages in finite graphs // J. of Mathemat. Scie. — 2000. — **101**, 4. — P. 205–215.

Поступила 19.05.2009