ON-LINE TESSELLATION AUTOMATA AS A TEXT MATCHING PROBLEMS SOLVER

A.V. Kocharyan and K.V. Shahbazyan

The goal of this work is to investigate the possibilities of utilizing On-line Tessellation Automata (OTA) to solve a class of Text Matching Problems (TMP). We understand a TMP as a problem of matching two texts, i.e., sequences of words in a finite alphabet. The comparability of two texts we understand as realizability of given EMSO-formula that captures the relative properties of these texts. We consider a class of TMP for which a uniform approach can be applied to obtain OTA solving problems in linear time.

INTRODUCTION

The goal of this work is to investigate the possibilities of utilizing On-line Tessellation Automata (OTA) to solve a class of Text Matching Problems (TMP). TMP is a generalization of Pattern Matching Problems [1-5] where two *words* (sequences of *letters* in a finite alphabet) are compared. We understand a TMP as a problem of matching two *texts* (sequences of *words* in a finite alphabet). The comparability of two texts we understand as realizability of given existential monadic second-order (EMSO)-formula that captures relative properties of these texts. We consider a class of TMP for which a uniform approach can be applied to obtain OTA solving problems in linear time.

OTA is a cellular automata destined for recognition of matrix languages [6,7,8]. It is an area of elementary finite automata where a wave of computations passes once diagonally across the array (for exact definitions see [6,7,8], and in the same journal [9, section 1.3]). It was shown in [6] that a matrix language is recognizable by OTA iff it is definable in EMSO logic. So OTA's are equivalent to EMSO formulas.

We formulate our results in terms of OTA's. To this end we identify a pair of texts (T_1, T_2) with a matrix, and a set of texts pairs with a matrix language. We suggest an algorithm that constructs OTA for a problem if notions of words and texts comparability are separately expressed in terms of OTA's.

Notice that to recognize any (m,n) -matrix OTA performs $m+n$ steps and the time of one step depends only of the number of instructions to be done for one transition of elementary finite automaton. So we obtain the automaton that solves TMP for two texts T_1, T_2 on-line in $|T_1|+|T_2|$ steps where $|T|$ is the length of T.

This paper is a continuation of the paper [9] in the same journal, therefore we use some definitions introduced in [9]. There OTA's are analysed as recognizers of natural's sequences . Here OTA's are investigated as recognizers of properties of word sequences - of the texts. A generalization of the Theorem 1 from [9] gives us the possibility to construct OTA's for a class of TMP. This is the class of TMP's that can be formulated by EMSO formula over the signature equipped by OTA -recognizable binary predicates.

The paper is organised as follows: first we give preliminary definitions, and then formulate a generalization of Theorem 1 from [9]. Finally we illustrate the possibilities of OTA on several examples.

§1. DEFINITIONS

1.1 Words and texts.

Let A be a finite non-empty alphabet. We denote the letters of A by symbols a, b, c with subscripts , and words over A by symbols u, v, t, p.

A word $u = a_1...a_m$ is a *factor* of a word $v = b_1...b_n$ iff there exists an integer j such that $a_i = b_{j+i}$ for $1 \le i \le m$. A size w *-window* of v is a size w factor $b_{i+1}...b_{i+w}$ of v, there are $n-w+1$ w -windows in v. A word u is a *serial episode* (or *subsequence*) of v iff there exist integers $1 \le i_1 < i_2 < ... < i_m \le n$ such that $a_j = b_{i_j}$ for $1 \le j \le m$. If $i_m - i_1 \le w$ then u is a *serial episode* of v in a w *-window*. A word u is a *parallel episode* of v if each letter of u is a factor of v, i.e., iff there exist integers $1 \le i_1, i_2, ..., i_m \le n$ such that $a_j = b_{i_j}$ for $1 \le j \le m$. If $|i_l - i_k| \le w$ then u is a *parallel episode* of v in a w *-window*.

A *text* over alphabet A is a sequence of words over A. We denote texts by T, S, V, U with subscripts.

A text $U = u_1, ..., u_m$ is a *factor* of text $V = v_1, ..., v_n$, iff there exists an integer j

such that $u_i = v_{j+i}$ for $1 \leq i \leq m$. A size w *-window* of V is a size w factor $v_{i+1}, \ldots, v_{i+w}$ of V, there are $n - w + 1$ w-windows in V. A text U is a *serial episode* (or *subsequence*) of V iff there exist integers $1 \leq i_1 < i_2 < \ldots < i_m \leq n$ such that $u_j = v_{i_j}$ for $1 \leq j \leq m$. If $i_m - i_1 \leq w$ then U is an episode of V in a w-window. A text U is a *parallel episode* of V iff each word u_i is a factor of V, i.e., if there exist integers $1 \leq i_1, i_2, \ldots, i_m \leq n$ such that $u_j = v_{i_j}$ for $1 \leq j \leq m$. If $|i_l - i_k| \leq w$ then U is a *parallel episode* of V in a w *-window*.

We denote by $|U| = n + 1 + \sum |u_i|$ the legth of the text U, where $|u_i|$ is the length of the word u_i

Example 1. Consider the text $V =$ „an alphabet is a finite non-empty set A" (blanks are separators). Then the text „non-empty set" is a factor of V. The text „empty set " is neither a factor nor an episode of V. The text „finite set" is a serial episode of V in 3-window. The text „non-empty alphabet" is a parallel episode of V in 5-window.

1.2. Binary predicates over words and related OTA.

We associate with each pair of words (u, v) over alphabet A a matrix $M_{u,v}$ over alphabet $\Sigma = \{0, 1\}$ such that if $u = a_1 \ldots a_m$ and $v = b_1 \ldots b_n$ then

$$M_{u,v}(i,j) = \begin{cases} 1 & if \quad a_i = b_j, \\ 0 & if \quad a_i \neq b_j, \end{cases} \qquad i = 1, \ldots, m, \; j = 1, \ldots, n.$$

Notice that the matrix $M_{u,v}$ is constructed on the base of letters identity predicate. It is possible to consider in the definition of $M_{u,v}$ any other binary predicate over letters of A instead of identity. We propose this one because in Pattern Matching Problems we meet mostly the identity.

We express notion of words comparability by binary OTA-recognizable predicates over words.

Associate to each binary predicate P over words on alphabet A a matrix language $L_P = \{M_{u,v} | P(u,v) = 1\}$. If there exists OTA $\mathcal{A}_P$ that recognises the language L_P, then we call predicate P *OTA-recognizable*. In this case L_P can be defined also by EMSO formula over the signature $\sigma = (S_1, S_2, R)$ where R is unary predicate.

Here are several examples of OTA-recognizable predicates over words. It is straightforward to check their deterministic OTA-recognizability.

1. Identity predicate $P_=(u,v)$.

$$P_=(u,v) = \begin{cases} 1, & \text{if} \quad u = v, \\ 0, & \text{if} \quad u \neq v. \end{cases}$$

We denote by $\mathcal{A}_{P_=}$ the OTA recognizing this predicate. It recognizes a matrix over alphabet $\{0,1\}$ iff its main diagonal consists only of 1's.

2. Predicate „the word u is a factor of the word v".

$$P_{factor}(u,v) = \begin{cases} 1, & \text{if } u \text{ is a factor of } v, \\ 0, & \text{otherwise.} \end{cases}$$

The corresponding OTA $\mathcal{A}_{P_{factor}}$ recognizes a (m,n) -matrix M over alphabet $\{0,1\}$ iff there is such an integer i that the matrix composed of columns $i, i+1, ..., i+m-1$ of M has only 1's on the main diagonal .

3. Predicate „the word u is a serial episode of the word v".

$$P_{sepisode}(u,v) = \begin{cases} 1, & \text{if } u \text{ is serial episode of } v, \\ 0, & \text{otherwise.} \end{cases}$$

The corresponding OTA $\mathcal{A}_{P_{sepisode}}$ recognizes a (m,n) -matrix M over alphabet $\{0,1\}$ iff there are such integers $1 \leq i_1 < i_2 < ... < i_m \leq n$ that the matrix composed of columns $i_1, i_2, ..., i_m$ of M has only 1's on the main diagonal .

4. Predicate „the word u is a parallel episode of the word v".

$$P_{pepisode}(u,v) = \begin{cases} 1, & \text{if the word } u \text{ is a parallel episode of the word } v, \\ 0, & \text{otherwise.} \end{cases}$$

The corresponding OTA $\mathcal{A}_{P_{pepisode}}$ recognizes a (m,n) -matrix M over alphabet $\{0,1\}$ iff M has at least one 1 in each row.

5. Predicate „the word u is a permutation of letters of the word v".

$$P_{permut}(u,v) = \begin{cases} 1, & \text{if the word } u \text{ is a permutation of letters of the word } v, \\ 0, & \text{otherwise.} \end{cases}$$

The corresponding OTA $\mathcal{A}_{P_{permut}}$ recognizes a (m,n) -matrix M over alphabet $\{0,1\}$ iff

M has unique 1 in each row and in each column.

6. Let P be OTA-recognizable binary predicate over words. We define another binary predicate over words W_P^w depending on P and window size w.

$$W_P^w(u,v) = \begin{cases} 1, & \text{if there exists in } v \text{ at least one size } w \text{ window } t \\ & \text{such that } P(u,t) = 1, \\ 0, & \text{otherwise.} \end{cases}$$

The defined predicate W_P^w is deterministic OTA-recognizable. We denote the corresponding OTA by $\mathcal{A}W_P^w$.

7. In particular, predicates $W_{P_=}^w$, $W_{P_{factor}}^w$, $W_{P_{episode}}^w$, $W_{P_{pepisode}}^w$, $W_{P_{permut}}^w$ are deterministic OTA-recognizable.

Observe, that if a matrix $M_{u,v}$ is given, then any mentioned OTA solves corresponding Pattern Matching Problem. The OTA's $\mathcal{A}_{P_{factor}}$, $\mathcal{A}_{P_{episode}}$ $\mathcal{A}_{P_{pepisode}}$ solve the problems known as pattern matching, episode matching, parallel episode matching [1-5]. These automata solve the problems without precomputing. The correspondent window problems [1,2] can be solved by OTA's $\mathcal{A}W_{P_{factor}}^w$, $\mathcal{A}W_{P_{episode}}^w$, $\mathcal{A}W_{P_{pepisode}}^w$. These problems require joint preprocessing of window size w and of OTA $\mathcal{A}_P$ to obtain OTA solving the problem.

For many problems of data mining it is interesting to count the number of such windows t that $P(u,t) = 1$. The elementary automata of OTA are finite automata therefore the counting is possible only if the number of windows is limited. To count arbitrary number of windows a counter is necessary, that can't belong to OTA.

1.3. Matrix code of texts pair. Information matrix.

If $T = t_1, ..., t_n$ is a text then we call the word $C(T) = *t_1 * ... * t_n*$ over $A \cup \{*\}, * \notin A$ the *code* of T.

A *matrix code* of a pair of texts (U,V) (denoted by $MC_{U,V}$) is a $(|U|,|V|)$ -matrix over $A \cup \{*\}$ such that the first row of $MC_{U,V}$ contains $C(V)$ and the first column contains $C(U)$. The other entries of $MC_{U,V}$ are arbitrary letters from A.

Let binary predicates $P_1, ..., P_r$ over words be fixed. To any pair (U,V) of texts where $U = u_1, ..., u_m, V = v_1, ..., v_n$ we associate (m,n) -matrix $IM_{U,V}$ whose entries are r -bit vectors $IM_{U,V}(i,j) = (P_1(u_i,v_j)...P_r(u_i,v_j))$, $1 \le i \le m$, $1 \le j \le n$. This matrix we call an *information matrix* of the pair (U, V). We use the information matrices as a tool for defining the languages of texts pairs.

§2. GENERALIZATION OF THEOREM 1 [9]. EXAMPLES.

Let the solution of a TMP for texts pair (U, V) be represented as two following steps:

1. matching of words:

Given matrix code $MC_{U,V}$, compute the values of predicates $P_1, ..., P_r$ to receive information matrix $IM_{U,V}$.

2. matching of texts:

Given information matrix $IM_{U,V}$, compute the value of predicate $\mathcal{P}$ that gives the solution of the TMP.

Then we can use the Theorem $1'$ that states the existence of OTA $\mathcal{D} = \mathcal{D}(P_1, ..., P_r, \mathcal{P})$ solving the TMP.

Theorem $1'$. Let $P_1, ..., P_r$ and their complements $\bar{P}_1, ..., \bar{P}_r$ be OTA recognizable predicates and OTA's $\mathcal{A}_{P_1}, ..., \mathcal{A}_{P_r}, \mathcal{A}_{\bar{P}_1}, ..., \mathcal{A}_{\bar{P}_r}$ be the corresponding recognizing automata.

Let $\mathcal{P}$ be a language of r -bit matrices, which is OTA -recognizable by OTA $\mathcal{A}_{\mathcal{P}}$.

There exists an algorithm Ω that constructs an OTA

$\mathcal{D} = \Omega(\mathcal{A}_{P_1}, ..., \mathcal{A}_{P_r}, \mathcal{A}_{\bar{P}_1}, ..., \mathcal{A}_{\bar{P}_r}, \mathcal{A}_{\mathcal{P}})$ such that

1) The input of $\mathcal{D}$ is $MC_{U,V}$ - the $(|U|, |V|)$ - matrix code of text pair;

2) $\mathcal{D}$ recognizes matrix code $MC_{U,V}$ for given pair (U, V) of texts iff $\mathcal{A}_{\mathcal{P}}$ recognizes the information matrix IM_{UV};

3) OTA $\mathcal{D}$ recognizes $MC_{U,V}$ on-line in $|U| + |V|$ steps;

4) Two-dimensional cellular automaton $\mathcal{D}$ in this case degenerates into one-dimensional cellular automaton, i.e., a systolic automaton. Its length is $min\{|U|, |V|\}$, its inputs are the codes of texts $C(U)$ and $C(V)$.

5) If $\mathcal{A}_{P_1}, ..., \mathcal{A}_{P_r}, \mathcal{A}_{\bar{P}_1}, ..., \mathcal{A}_{\bar{P}_r}, \mathcal{A}_{\mathcal{P}}$ are deterministic OTA's then $\mathcal{D}$ is deterministic.

Proof: The proof of the Theorem $1'$ is the same as in [9]. The only difference consists in A_{prel} automaton. For every pair of texts $U = u_1, ..., u_m$, $V = v_1, ..., v_n$ and their matrix code $MC_{U,V}$ this automaton constructs the cell-matrix M (see section 1.2 in [9]) so that the cell $M < k, j >= M_{u_k, v_j}$, $1 \le k \le m$, $1 \le j \le n$. Then OTA $\mathcal{L}$ imitates the run of all $\mathcal{A}_{P_s}, \mathcal{A}_{P_s}$, $s = 1, ..., r$ on the cells, i.e., on matrices M_{u_i, v_j}. At last OTA $\mathcal{D}$ realizes a composition of $\mathcal{A}_{prel}, \mathcal{L}, \mathcal{A}_{\mathcal{P}}$. ◇

Let us bring some examples of TMPs which we can solve by our method.

Let the number of predicates be $r = 1$. We can use any of 10 above mentioned predicates in place of P_1, either of $\mathcal{P}$, that gives us 100 combinations and therefore the correspondent OTA's to solve 100 different TMP's. Evidently we can extend the set of predicates and simultaneously extend the number of TMP's.

Example 2. Given the texts $U = u_1, ..., u_m$, $V = v_1, .., v_n$, problem consists in finding whether the text U is a parallel episode of the text V.

In view of Theorem 1′ the problem can be solved by automaton $\mathcal{D}(P_=, P_{pepisode})$.

If $V =$,,an alphabet is a finite non-empty set A", and $U =$,,finite set", or $U =$,,set finite" (blanks are used as separators) then the problem has positive answer. If $U =$,,empty set " then the answer is negative.

Example 3. Given the texts $U = u_1, ..., u_m$, $V = v_1, .., v_n$ and a number w, problem consists in finding whether in the text V there is a size w window $T = v_k, ..., v_{k+w-1}$ such that

1) every word u_i of the text U is a factor of some word v_{j_i} of T, i.e., for $i = 1, ..., m$, $k \leq j_i \leq k + w - 1$, the word u_i is a factor if v_{j_i}.

2) the text $v_{j_1}, ..., v_{j_m}$ is a parallel episode of the window T.

In view of Theorem 1′ automaton $\mathcal{D}(P_{factor}, W^w_{P_{pepisode}})$ solves the problem.

If $V =$,,an alphabet is a finite non-empty set A", and $U =$,,empty set" then the answer is positive for $w \geq 2$. If $U =$,,non empty" then answer is positive for $w \geq 1$.

Example 4. Given the texts $U = u_1, ..., u_m$, $V = v_1, .., v_m$, problem consists in finding whether in the text V is a permutation of words of the text U.

In view of Theorem 1′ the problem can be solved by automaton $\mathcal{D}(P_=, P_{permut})$.

If $V =$,,an alphabet is a finite non-empty set A", and $U =$,,a finite non-empty set A is an alphabet" then answer is positive.

§3. CONCLUSION

Authors have implemented algorithm Ω as a computer program that constructs OTA $\mathcal{D}$ to solve TMP defined by predicates $P_1, ..., P_r, \mathcal{P}$. Inputs of the program are deterministic OTA's $\mathcal{A}_{P_1}, \ldots, \mathcal{A}_{P_r}$ and $\mathcal{A}_{\mathcal{P}}$ as programs that compute transitions of OTA's without storing the automaton. Output of the program is executable code of OTA $\mathcal{D}$ that computes transitions of OTA $\mathcal{D}$ without storing $\mathcal{D}$.

As it is mentioned in Theorem $1'$ the two-dimentional cellular automaton $\mathcal{D}$ for a TMP degenerates into one-dimensional one, i.e., systolic automaton. Therefore our program can be used to receive the definition of corresponding systolic array.

REFERENCES

1. L. Boasson, P. Cegielski, I. Guessarian, Y. Matiyasevich. Window accumulated subsequence matching is linear. Annals of Purc and Applied Logic Vol. 113(2001),pp.59-80.

2. L. Boasson, P. Cegielski, I. Guessarian, Y. Matiyasevich. Multiple Serial Episodes Matching. CSIT 2005, pp.26-38.

3. G. Das, R. Fleischer, L. Gasienic, D. Gunopoulos, J Kärkkäinen. Episode matching, Proc. 1997 Combinatorial Pattern Mutching Conf., LNCS 1264, Springer-Verlag, Berlin (1997), pp. 12-27.

4. Z. Galil. String matching in real time. J.Assoc. Comput. Mac. Vol.28,(1981), pp.134-149.

5. G. Kucherov, M. Rusinovitch. Matching a Set Strings with variable Length Don't Cares. TCS, Vol 178, (1997), pp.129-154.

6. K. Inoue and A. Nakamura. Some properties of two-dimensional on-line tessellation acceptors. *Information Sciences*,vol.13, p.95-121, 1977.

7. K. Inoue and A. Nakamura. A Survey of two-dimensional automata theory. In *Proc. 5th Int. Meeting of Young Computer Scientists*.J Dasson and J.Kelemen (Eds.), p. 72-91. Lecture Notes in Computer Science 381, Springer-Verlag, Berlin, 1990.

8. D. Giammarresi and A. Restivo. Two-dimensional languages. In *Handbook of Formal Laguage Theory*, v.3, Springer-Verlag, N.Y.,1996.

9. K. V. Shahbazyan, Yu. H. Shoukourian. The Finite-state Recognizability of sequences of Integers. In this journal.

17 January, 2006

Institute for Informatics and Automation Problems
National Academy of Sciences of Armenia
E-mail: armkoch@ipia.sci.am, shahb@ipia.sci.am