Syntactic Pattern Generation and Matching

Pawan Kumar Patnaik[1], Venkata Padmavati Metta[2] , Jyoti Singh[3]
[1]*Department of Computer Science and Engineering, Bhilai Institute of Technology, Durg, India*
[2]*Department of Computer Science and Engineering, Bhilai Institute of Technology, Durg, India*
[3]*Chhattisgarh Professional Examination Board, Raipur, India*
[1]*pawanpatnaik37@gmail.com*, [2]*vmetta@gmail.com*, [3]*jsraipur13@gmail.com*

Abstract

In this paper a new model of Syntactical pattern generation are introduced. Methodological considerations on crucial issues in areas of string and hexagonal pattern based syntactical methods are created. During this paper one of the significant work of the analysis is that the recognition of three-dimensional pattern using hexagonal online tessellation acceptor.

Keywords: *Syntactic pattern generation, hexagonal picture, online acceptor.*

1. Introduction

Pattern recognition techniques are used to automatically classify physical objects (handwritten characters, tissue samples) or abstract multidimensional patterns (*n* points in *d* dimensions) into known or possibly unknown categories. Several techniques of pattern recognition, mainly based on numerical similarity measures are intensively studied and successfully put into practice. But there also exist a completely different approach, known as structural pattern recognition. Its main concern is how a pattern can be described and interpreted as a composition of simple sub-patterns (pattern primitives). The structural pattern generation and recognition through syntactic methods is based on formal language theory. Syntactic pattern recognition deals with image recognition (Bunke. H. and Sanfeliu. A. 1990). First the image data is decoded into the string or array of text that represents the image formally. After that, all well approved techniques of formal language theory, parsing and generation of sequences can be used.

Pattern matching has a wide range of applications in the fields of pattern recognition, image processing, computer vision etc. For one dimensional text, this problem is referred as string matching. String matching is a very well-explored area in computer science. The string matching problem is to find all occurrences (or the first occurrence) of a pattern *P* of length *m* in a given a text *T* of length *n*. String matching has got its applications in the fields of text editing, text searching, data base search, artificial intelligence, information retrieval etc. For two or more dimensional data, this problem is referred to as pattern matching. In recent years multidimensional string matching has become a field of more and more interest, largely motivated by problems in low-level image processing. The two dimensional string matching problem is to find the occurrences of a two dimensional pattern in a two dimensional text. Most known two-dimensional matching algorithms have a rectangular text and rectangular pattern as their input. These matching algorithms perform a row by row analysis followed by some column processing. These techniques are only efficient if all the rows are of equal length, hence the necessity for rectangular patterns. It has applications in digital picture processing, picture databases, computer vision and many other fields. Formally, the two dimensional string matching problem is defined by a text T [n, n] and pattern P [m, m]. The text and pattern contain symbols from an alphabet $\sum$ and we have to find all occurrences of P in T. In many applications like

computational biology, it is desirable to find the approximate matches of the pattern in the given text rather than the exact match. The Honeycomb conjecture states that the hexagonal tiling is the best way to divide a surface into regions of equal area with the least total perimeter. This research presents a novel method for generating and matching of hexagonal patterns. The hexagonal two-dimensional pattern can be viewed as three dimensional data, with data along *x*-axis, *y*-axis and *z*-axis. The hexagonal pattern matching problem is to find all occurrences of the patterns arrays in a hexagonal or rectangular text array.

This paper is organized as follows. Section 2 describes Related work. Section 3 deals with the hexagonal array, hexagonal online tessellation acceptor and notions of hexagonal online tessellation acceptor. Finally, the conclusions have been drawn in section 4.

2. Related Work

There are many well-known string matching algorithms like Knuth Morris Pratt and Boyer Moore. There is also a string matching algorithm that constructs a finite automaton for the pattern and finds all the occurrences of the pattern in the text. There are many variations to this string matching problem. One such variation is to find the occurrences of the pattern P in the text T allowing mismatches. This problem is also called as approximate string matching. This problem can be defined as follows: Given a text T of length *n*, a pattern P of length *m* and an integer *k*, to find all occurrences of the pattern P in the text T with at most *k* mismatches. Landau. G.M. and Vishkin. U. [15] have given an algorithm which runs in $O(k(mlogm+n))$ time.

Landau. G.M. and Vishkin. U. [16] have given an algorithm which runs in $O(m+nk^2)$ time for an alphabet set whose size is fixed. For general input the algorithm runs in $O(mlogm + nk^2)$ time. In both the cases, the space requirement is $O(m)$. Several algorithms have been presented for approximate string matching that use $O(kn)$ comparisons in the worst case or in the average case by taking advantage of the properties of the dynamic programming paradigm. Yates. R.A.B. and Perleberg. C.H. [31] have given an algorithm for approximate string matching that runs in linear time for most practical cases. Yates. R.A.B. and Navarro. G. [29] have given a new algorithm for on-line approximate string matching. Their algorithm is based on the simulation of a non deterministic finite automaton built from the pattern and using the text as input.

In syntactic pattern matching, the text will be generated by a grammar G. For one dimension, the grammar G is generally known as string grammars. Myers.E.W. and Miller.W.[17] have given an algorithm for the approximate matching of regular expressions. Myers. G.[18] has given an algorithm for approximately matching context free language. The algorithm generalizes the Cocke-Younger-Kasami (CYK) algorithm. Reghizzi. S. C. and Pradella. M. [22] have given a new parsing algorithm, which extends the Cocke, Kasmi and Younger's classical parsing technique for string languages and preserves polynomial time complexity.

The natural generalization of the multiple pattern matching is the one in which the pattern is a two dimensional array of symbols *P* of size m_1 x m_2 *and* the text *T* of size n_1 x n_2. The problem is to determine whether the pattern occurs as a sub array of the text *T* or not. Bird.R.S.[3] has given the first algorithm to solve this problem where he has assumed that the pattern *P* is of size *m* x *m* and text *T* is of size *n* x *n*. He used KMP as the base algorithm. The algorithm runs in $O(n^2 + m^2)$ time, uses $O(n^2 + m)$ space.

Krithivasan. K. and Sitalakshmi. R. [13] have given an improved algorithm to that of Bird. R.S. [3] and their algorithm runs better when the alphabet size is large. Yates. R.

A. B. and Regnier. M. [32] have given an algorithm that finds the occurrences of the pattern *P* of size *m* x *m* in a text *T* of size *n* x *n* in expected time of $O(n^2/m)$.

Karkkainen. J. and Ukkonen. E. [9] have given an algorithm for two and higher dimensional pattern matching in optional expected time. Their algorithm runs in $O(n^2/m^2 \log_c m^2)$ (where *c* is the alphabet size) which is optimal by Yao's lower bound result (Yao. A.C. 1979). Krithivasan. K. and Sitalakshmi. R. [12] have also given an algorithm for 2-dimensional pattern matching in presence of errors. Here they reported the occurrences of pattern *P* in the text *T* allowing up to *k* mismatches. Ranka. S. and Heywood. T. [21] have improved this algorithm. The approximate two dimensional pattern matching problem has also been considered by Park. K. [19].

Polcar. T. and Melichar. B. [20] has given an algorithm that transforms a special type of non-deterministic two-dimensional online tessellation automata into deterministic one is presented. Zdarek. J. and Melichar. B. [33] have given a general concept of two-dimensional pattern matching using conventional (one-dimensional) finite automata.

Terrier. V. [26] has considered the relationships between the classes of two-dimensional languages defined by deterministic on-line tessellation automata and by real time two-dimensional cellular automata with Moore and Von Neumann neighborhood. The algorithm generalizes the result known for one-dimensional cellular automata to two-dimensional cellular automata with Von Neumann neighborhood: the class of real time cellular automata is closed under rotation of 180° if and only if a real time cellular automaton is equivalent to linear time cellular automata.
Matrix grammar is studied in (Siromoney. G. Siromoney. R. and Krithivasan. K. 1972) and is used as a generation mechanism to generate rectangular arrays. With this type of grammar, first a horizontal string of intermediates is derived and then the vertical columns of the array are derived. In (Siromoney. G. Siromoney. R. and Krithivasan. K. 1972), all four types of grammars of Chomsky hierarchy are considered in the horizontal direction and only regular grammars are considered in the vertical direction.

Inoue. K. and Nakamura. A. [14] has given a new type acceptor called the "two-dimensional on line tessellation acceptor" (denoted by "2-ota") and to examine several properties of the 2-ota. Toda. M. Inoue. K. and Takanami. I. [27] have given an algorithm that can be solved array matching problem efficiently by using 2-dimensional online tessellation acceptor. The array matching problem can be solved in exactly $m+n-1$ steps for *m* by *n* text arrays.

Cece. G. and Giorgetti. A. [6] have introduced the notion of simulation over a class of automata which recognize 2D languages (languages of arrays of letters). This class of two-dimensional On-line Tessellation Automata accepts the same class of languages as the class of tiling systems, considered as the natural extension of classical regular word languages to the 2D case.

There are many algorithms for syntactic pattern matching and they considered rectangular pattern array. Till now there is no research has been done in designing the pure context free grammar for generating a set of hexagonal image patterns. This research will make use of pure context free grammars to generate hexagonal patterns which then will be recognized by a hexagonal online tessellation acceptor.

3. Design of Hexagonal Online Tessellation Acceptor

In this section we review here notions of hexagonal pictures and hexagonal array. Let $\sum$ be a finite alphabet of symbols. A hexagonal picture P over $\sum$ is a hexagonal array of symbols of $\sum$. The set of all hexagonal arrays of the alphabet $\sum$ is denoted by $\sum^{**H}$

Example. 1

A hexagonal picture over the alphabet {a, b, c, d} is shown in Figure.1

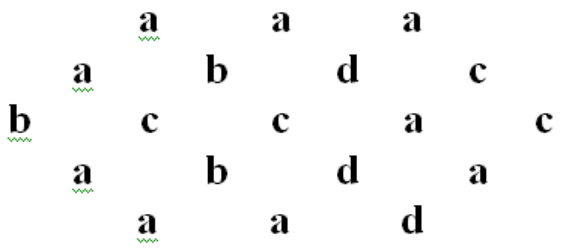


Figure. 1 Hexagonal picture

With respect to triad of triangular axes x, y and z, the co-ordinates of each element of the hexagonal picture in the above figure are shown in Figure. 2.

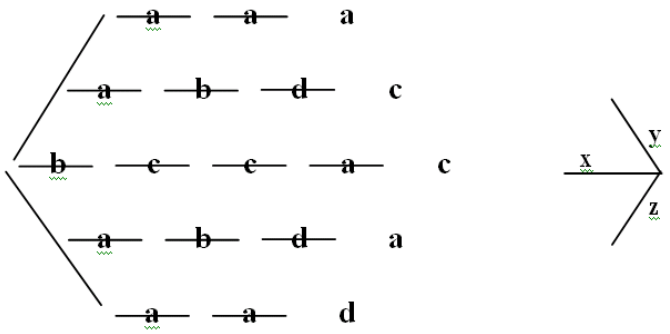


Figure. 2 Co-ordinates of elements of hexagonal picture

Let P be a hexagonal picture in $\sum^{**H}$; P (i, j, k) is the symbol in the position (i, j, k); we denote the number of elements in the *x*, *y*, *z* directions of P by *x*(P), *y*(P) and *z*(P) respectively. The triplet (*x* (P), *y* (P), *z* (P)) is called the size of the hexagonal picture P. The size of the empty picture Є is obviously (0, 0, 0).

If $h \in \sum^{**H}$, then *h* is the hexagonal array obtained by surrounded by * is shown in Figure. 3.

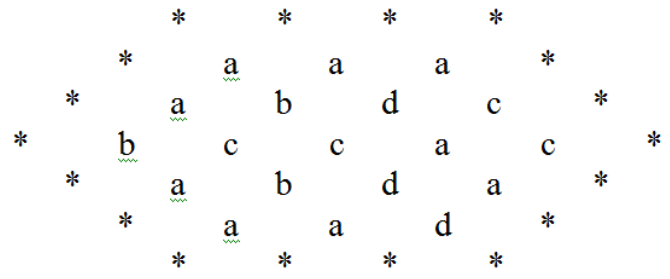


Figure. 3 A hexagonal array surrounded by *

A hexagonal online tessellation acceptor (in short, H-ota) is defined as a 7- tuple M = (K, E^3, ∑U {*}, δ, q_e, q_0, F) where,

1. K is a finite set of states;
2. E^3 is a set of all 3-tuples of integers;
3. ∑ is a finite set of input symbols and (*) is the boundary symbol not in ∑ ;
4. δ: K × K^3 × (∑ U {*}) → $2^{K'}$ U {{q_0}} where (K' = K^3 – {(q_e , q_0, q_0)}) is the cell state transition function;
5. q_e Є K is the motive state;
6. q_0 Є K is the quiescent state;
7. F ⊆ K – {q_e, q0} is a set of final states.

The cell transition function δ prescribes state transition of cells in E^3. M is called deterministic if the image under δ of every element in K^3 × (∑ U {*}) is a singleton; otherwise non-deterministic.

Example:

Let M = ({q_e, q_0, q_1, q_2}, E^3, {a,*}, δ, q_e, q_0, F = {q_3}) be deterministic H-ota, where

δ(q_e , q_0 , q_0 , q_0 , a) = δ(q_0 , q_0 , q_0 , q_1 , a) = δ (q_0 , q_0 , q_1 , q_0 , a) = δ (q_0 , q_2 , q_2 , q_2 , a) = δ (q_0 , q_2 , q_0 , q_1 , a) = δ (q_0 , q_2 , q_1 , q_0 , a) = q_1

and

δ(q_0 , q_1 , q_1 , q_1 , a) = δ(q_0 , q_1 , q_0 , q_2 , a) = δ(q_0 , q_1 , q_2 , q_0 , a) = q_2.

When the pattern *h* over {a} written on to the tape x as shown in Figure. 4 is presented to M, M enters the state configurations as shown in Figure. 5(a), …, Figure. 5(h) respectively at time t=0, …, t=7.

The run of M on *h* is the tape corresponding to the highlighted area enclosed by the heavy solid line. The halting state of the machine is shown in Figure. 5(h). The final state q_2 in the cell (2, 4) indicates that M accepts the input *h* on tape x.

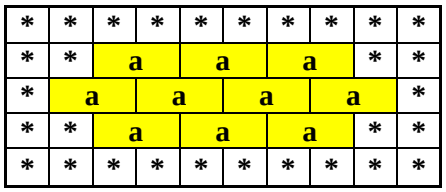
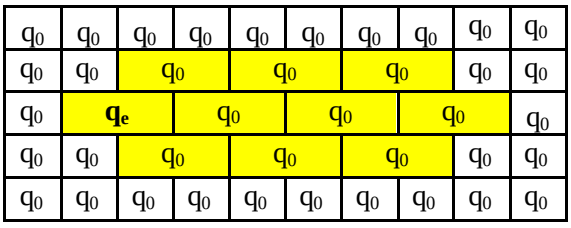
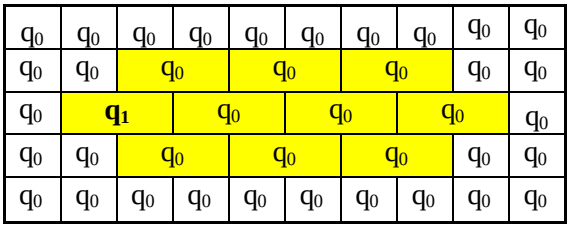


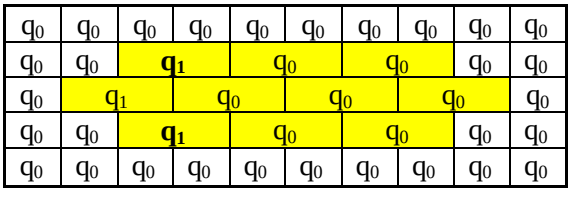
Figure.4 3 by 4 tape x over {a}



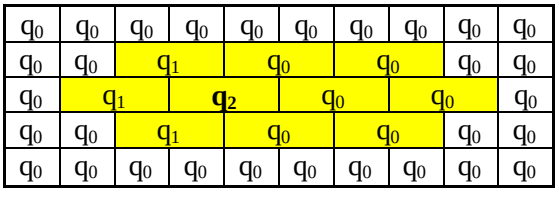
(a) t = 0



(b) t = 1



(c) t = 2



(d) t = 3

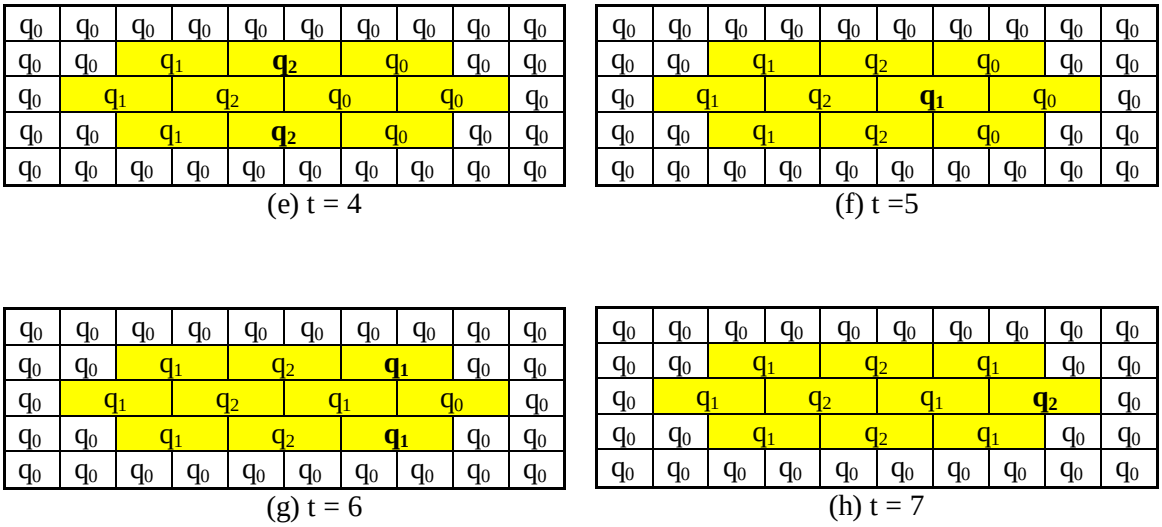


Figure.5 State configurations of M (when the tape x as shown in Figure. 4 is presented to M)

The transition functions at different times (t=0, 1, 2…) for the above figures can be stated as follows:

At t = 1:

$\delta (q_e , q_0 , q_0 , q_0 , a) = q_1$

At t = 2:

$\delta (q_0 , q_0 , q_0 , q_1 , a) = q_1$

$\delta (q_0 , q_0 , q_1 , q_0 , a) = q_1$

At t = 3:

$\delta (q_0 , q_1 , q_1 , q_1 , a) = q_2$

At t = 4:

$\delta (q_0 , q_1 , q_0 , q_2 , a) = q_2$

$\delta (q_0 , q_1 , q_2 , q_0 , a) = q_2$

At t = 5:

$\delta (q_0 , q_2 , q_2 , q_2 , a) = q_1$

At t = 6:

$\delta (q_0 , q_2 , q_0 , q_1 , a) = q_1$

$\delta (q_0 , q_2 , q_1 , q_0 , a) = q_1$

At t = 7:

$\delta (q_0 , q_1 , q_1 , q_1 , a) = q_2$ (*Final State*)

4. Conclusion

It has been concluded that syntactic methods play an important role in pattern generation and detection. One of the significant work of the proposed research is the recognition of three-dimensional pattern using hexagonal online tessellation acceptor. It has applications in pattern matching and object recognition. This enables to develop more efficient algorithms for recognizing three dimensional objects. This in-turn has applications in modeling proteins - antibody reactions which are used in drug designing.

References

[1] Aho. A. and Corasick M.J. 1975. *Efficient string matching: An aid to bibliographic search.* Communication of the ACM. 18(6): 333-340.

[2] Aho. A. and Ullman J.D. 1972. Theory of parsing, translation and compiling. Volume 1: Prentice Hall Inc. ISBN No.: 0-13-914556-7

[3] Bird. R.S. 1977. *Two dimensional pattern matching*. Information Processing Letters. 6:168-170.

[4] Boyer. R. and Moore. S. 1977. *A fast string searching algorithm*. Communication of the ACM. 20(10):762-772.

[5] Bunke. H. and Sanfeliu. A. 1990. *Syntactic and structural pattern recognition*: Theory and Applications. World Scientific.

[6] Cece. G. and Giorgetti. A. 2011. *Simulations over two-dimensional on-line tessellation automata.* Lecture Notes in Computer Science. 6795:141–152.

[7] Harrison. M.A. 1978. Introduction to Formal Language Theory. Addison-Wesley.1st edition. ISBN No.:0201029553

[8] Hopcroft. J.E. and Ullman. J.D. Introduction to automata theory, languages and computation.2007. Narosa Publishing House. ISBN No.: 8185015961

[9] Karkkainen. J. and Ukkonen. E. 1994. *Two and higher dimensional pattern matching in optimal expected time*. In Proc. of fifth annual ACM-SIAM Symposium on Discrete Algorithms. Arlington, VA, USA. January 23 – 25. 715-723.

[10] Knight. J.R. and Myers. E.W. 1995. *Approximate regular expression pattern matching with concave gap penalties.* Algorithmica, 14(1):85-121.

[11] Knuth. D.E., Morris. J.H. and Pratt V.R. 1977. *Fast pattern matching in strings.* Society for Industrial Applied Mathematics Journal on Computing. 6(2):323-350.

[12] Krithivasan. K. and Sitalakshmi. R. 1987. *Efficient two-dimensional pattern matching in the presence of errors.* Information Science 43:169-184.

[13] Krithivasan. K. and Sitalakshmi. R. 1986. *An efficient two dimensional pattern matching algorithm,* Technical report, CS-TR-1724, Centre for Automation Research, University of Maryland.

[14] Inoue. K. and Nakamura. A. 2009. *Some properties of two dimentional online tessellation acceptors.* Information Science 13**:** 95-121.

[15] Landau. G.M. and Vishkin. U. 1986. *Efficient string matching with k mismatches*. Theoretical Computer Science. 43(3):239-249.

[16] Landau. G.M. and Vishkin. U. 1988. *Fast string matching with k differences*. Journal of Computer and Systems Sciences. 37:63-78.

[17] Myers. E.W. and Miller. W. 1989. *Approximate matching of regular expressions.* Bulletin of Mathematical Biology. 51(1):5-37.

[18] Myers. G. 1995. *Approximately matching context-free languages*. Information Processing Letters.54:85-92.

[19] Park. K.1996. *Analysis of two-dimensional approximate pattern matching algorithms*. Lecture Notes in Computer Science. 1075:335-360.

[20] Polcar. T. and Melichar. B. 2005. *Two-dimentional pattern matching by two-dimentional online tessellation automata.* Lecture Notes in Computer Science .3317:327-328.

[21] Ranka. S. and Heywood. T. 1991. *Two-dimensional pattern matching with k mismatches. Pattern Recognition*. 24(1):31-40.

[22] Reghizzi. S. C. and Pradella. M. 2008. *A CKY parser for picture grammars*.Information Processing Letters. 105:213-217.

[23] Siromoney. G., Siromoney. R. and Krithivasan. K. 1972. *Abstract families of matrices and picture languages.* Computer Graphics and Image Processing. 1(3): 284-307.

[24] Siromoney. G.Siromoney. R. and Krithivasan. K. 1973. *Picture languages with array rewriting rules.*Information and Control. 22(5):447-470.

[25] Subramanian. K.G.Saravanan. R. and Robinson. T.2007. *P systems for array generation and application to kolam patterns*.natural computing.22: 47-54.

[26] Terrier. V. 2003. *Two-dimensional cellular automata and deterministic on-line tessalation automata*. Theoretical Computer Science 301(1-3): 167 – 186.

[27] Toda. M. Inoue. K. and Takanami. I.1983. *Two dimensional Pattern Matching by Two Dimensional On-line tessellation acceptors*. Theoretical Computer Science .24(2): 179-194.

[28] Yao. A.C. 1979. *The complexity of pattern matching for a random string*. Society for Industrial Applied Mathematics Journal on Computing.8:368-387.

[29] Yates. R.A.B. and Navarro. G. 1996.*A faster algorithm for approximate string matching*. In Proc. Of 7th Symposium CPM 96, Laguna Beach, California, June10-12. 1-23.

[30] Yates. R.A.B. and Navarro. G. 1996. *A faster heuristic for approximate string matching*. In Proc. of WSP'96. 47-63.

[31] Yates. R.A.B. and Perleberg. C.H. 1996. *Fast and practical approximate string matching*. Information Processing Letters.59: 21-27.

[32] Yates. R.A.B. and Regnier. M. 1993. *Fast Two-dimensional pattern matching*. Information Processing Letters. 45:51-57.

[33] Zdarek. J. and Melichar. B. 2008. *Two-dimensional pattern matching by finite automata*. Lecture Notes in Computer Science. 3845:329-340