

---

# Finite Automata and Two-dimensional Pattern Matching

Jan Žďárek

*Thanks: Bořivoj Melichar*

zdarekj@fel.cvut.cz



PSC Prague Stringology Club

Czech Technical University in Prague

---

# Content

- ✿ Introduction
- ✿ 2D exact pattern matching
- ✿ 2D approximate pattern matching using the 2D Hamming distance and  $KS$  distance
- ✿ Summary.

# Objective

- ✿ There are many interesting results in the domain of matching by finite automata.
- ✿ Is there any possibility to reuse at least some of these results for matching in multiple dimensions?

There is a class of tasks solvable by finite automata (FA).  
For a given task one can

- ✿ construct DFA,
- ✿ run DFA over an input,
- ✿ obtain a solution.

*Fact:* In practice NFA is usually constructed much easier than DFA.

*Problem 1:* We cannot run NFA directly because of its nondeterminism.

*Solution A of P. 1:* Transformation of NFA to DFA using the standard subset construction (Hopcroft, Ullman – 1979) It may lead up to the exponential blow up of states ( $|Q_{DFA}| \leq 2^{|Q_{NFA}|}$ ).

*Advantage:* Once we have DFA, it runs in  $\mathcal{O}(|T|)$  time.

# Finite Automata (2)

N.B.: For PM automata it is *much* better, e.g.  $\mathcal{O}(|\Sigma|^k |P|^{k+1})$  states in case of PM using the Hamming distance.

(Polcar, Melichar – to be published)

*Solution B of P. 1:* Simulation of NFA runs always slower than DFA but has lower memory requirements and simple preprocessing.

Basic simulation method runs in time  $\mathcal{O}(n |Q_{NFA}|^2)$  and space  $\mathcal{O}(|\Sigma| |Q_{NFA}|^2)$ .

(Holub – 2000)

Implementation using bit vectors runs in time

$\mathcal{O}\left(n |Q_{NFA}| \left\lceil \frac{|Q_{NFA}|}{w} \right\rceil\right)$  and space  $\mathcal{O}\left(|\Sigma| |Q_{NFA}| \left\lceil \frac{|Q_{NFA}|}{w} \right\rceil\right)$ .

( $w$  is the length of a computer word measured in bits.)

# Finite Automata (3)

method	space complexity	time complexity
Exact string matching		
direct construction of DFA	$\mathcal{O}(m \Sigma )$	preprocess $\mathcal{O}(m \Sigma )$ , run $\mathcal{O}(n)$
bit parallelism	$\mathcal{O}(\frac{m}{w} + m \Sigma )$	preprocess $\mathcal{O}(m + \frac{m}{w} \Sigma )$ , run $\mathcal{O}(\frac{m}{w}n)$
dynamic programming	$\mathcal{O}(m)$	preprocess $\mathcal{O}(m)$ , run $\mathcal{O}(mn)$
Approximate string matching	Hamming distance	
NFA (construction)	both $\mathcal{O}((km - \frac{k^2}{2} + \frac{3k}{2}) \Sigma  + km)$	
bit parallelism	$\mathcal{O}((k\lceil \frac{m}{w} \rceil +  \Sigma m))$	preprocess $\mathcal{O}((m + \lceil \frac{m}{w} \rceil \Sigma  + k\lceil \frac{m}{w} \rceil)$ run $\mathcal{O}((kn\lceil \frac{m}{w} \rceil))$
dynamic programming	$\mathcal{O}(m)$	preprocess $\mathcal{O}(m)$ , run $\mathcal{O}(mn)$

*(Part of the table summarizing the previous results: Holub, Špiller – 2004)*

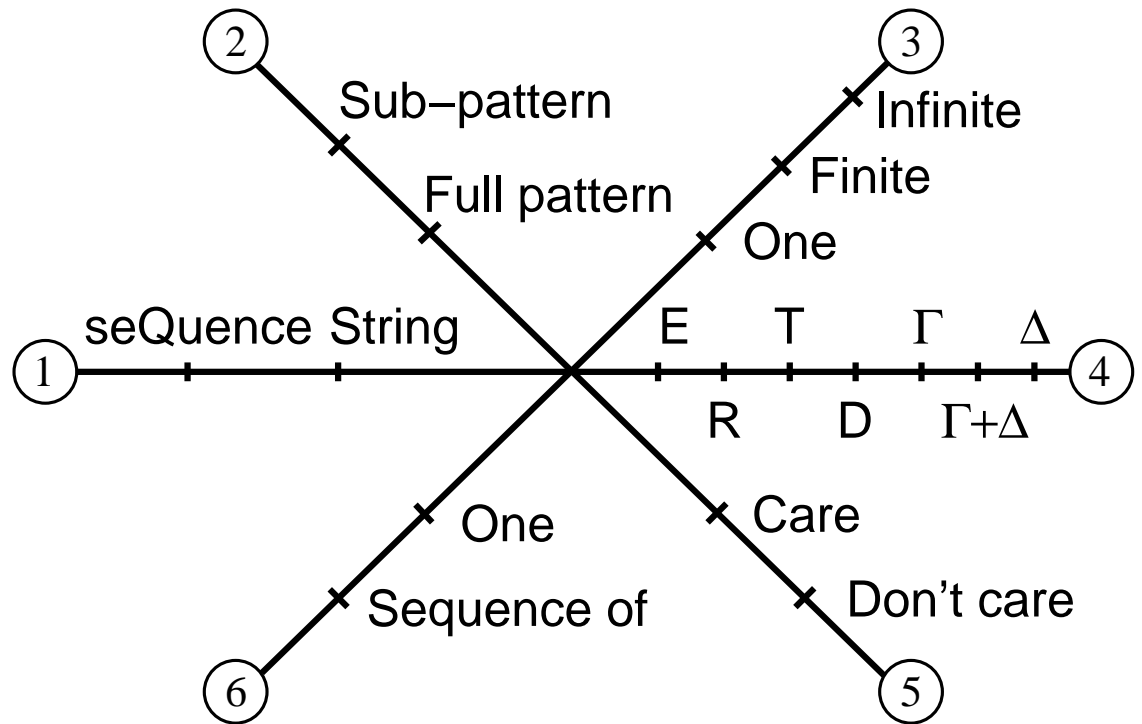
# Finite Automata (4)

- ➡ All (1D) pattern matching problems can be solved by finite automata.  
*(Melichar, Holub – 1997)*

# Classification of 1D Matching Problems

Classification of 1D pattern matching problems:

1 – nature of the pattern	2 – integrity of the pattern	3 – number of patterns	4 – way of matching	5 – importance of symbols	6 – number of instances
<i>S</i>	<i>F</i>	<i>O</i>	<i>E</i>	<i>C</i>	<i>O</i>
<i>Q</i>	<i>S</i>	<i>F</i>	<i>R</i>	<i>D</i>	<i>S</i>
		<i>I</i>	<i>D</i>		
			<i>T</i>		
			$L(\Gamma)$		
			$G(\Delta)$		
			$H(\Gamma+\Delta)$		



$2 \cdot 2 \cdot 3 \cdot 7 \cdot 2 \cdot 2 = 336$  described problems.

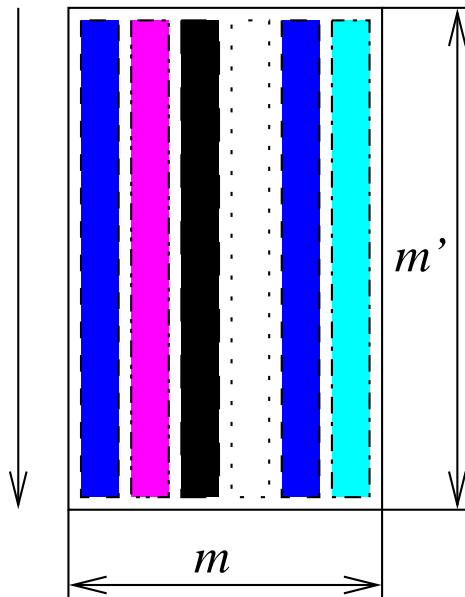
- ➔ To solve two or more dimensional problems by finite automata, appropriate transformation is needed.

## Idea of **linear reduction**:

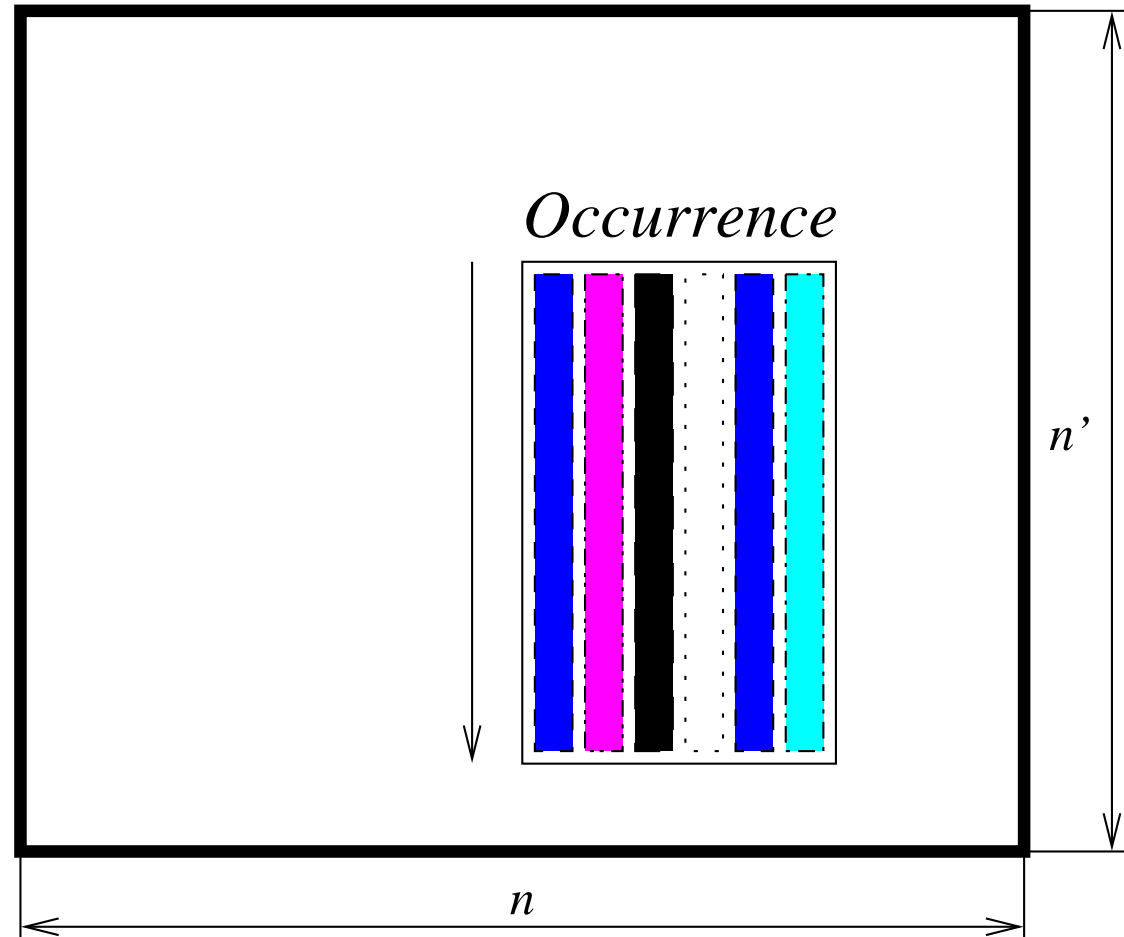
- ✿ multiple automata will be used,
- ✿ their results (preprocessed text) passed among them.
- ✿ The dimension of the problem is reduced by one in each step in the last step classical pattern matching can be used.

# Idea of 2D Exact Pattern Matching

Pattern Array

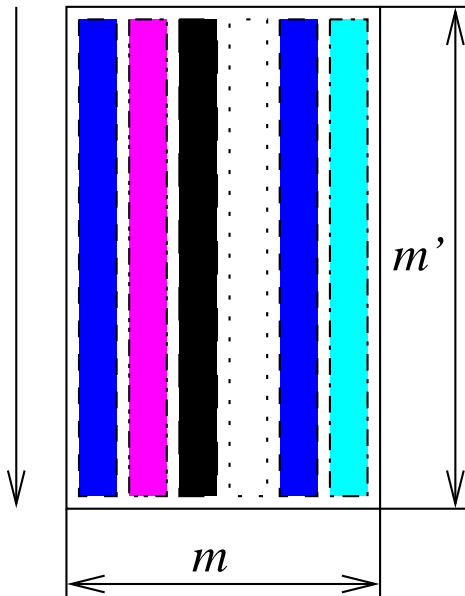


Text Array

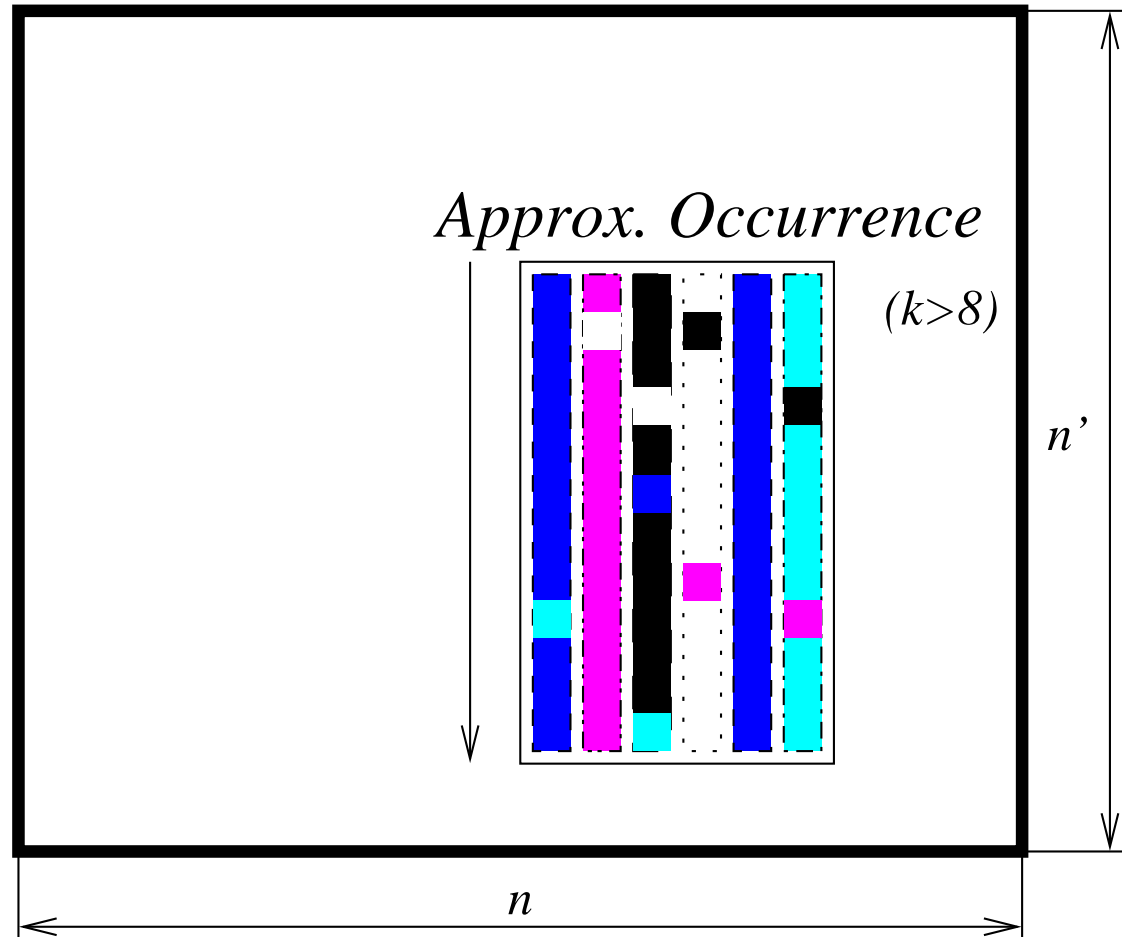


# Idea of 2D Approximate Matching

Pattern Array



Text Array



# Problems of 2D Pattern Matching (1)

Let  $P$  (pattern) and  $T$  (text) be pictures over some alphabet  $\Sigma$  and let  $P \in \Sigma^{m \times m'}$ ,  $T \in \Sigma^{n \times n'}$ ,  $n \geq m$ ,  $n' \geq m'$ .

- ✿ A two-dimensional pattern matching problem: to locate picture  $P$  as a sub-picture inside of  $T$ .
- ✿ A two-dimensional occurrence of  $P$  in  $T$ :
  - \* **exact**, if  $P$  is included in  $T$  as a sub-picture,
  - \* **approximate**, if for some sub-picture  $X$  of  $T$   $2D\text{-dist}(P, X)$  is minimal or  $2D\text{-dist}(P, X) \leq k$ .

( $2D\text{-dist}$  is a function defining a 2D distance between two pictures.)  
2D approximate pattern matching using some **2D edit distance**  $k$ ,  $k \in \mathbb{N}_0$  means to find all occurrences of  $P$  in  $T$  with equal or less than  $k$  errors.

# Problems of 2D Pattern Matching (2)

Let  $row_i(a)$  denotes the  $i^{th}$ -row of picture  $a$  and  $col_i(a)$  denotes the  $i^{th}$ -column of  $a$ .

2D approximate pattern matching using the 2D Hamming distance  $k$ , (2D PM with  $k$  mismatches):

to find all occurrences of  $P$  in  $T$  with equal or less than  $k$  mismatching symbols,  $k \in \mathbb{N}_0$ ,  $k < mm'$ .

The 2D Hamming distance can be described by this formula:

$$H(X, P) = \sum_{i=1}^{n'} ed_H (col_i(X), col_i(P))$$

Function  $ed_H$  computes the Hamming edit distance (*Hamming – 1950*) comparing two rows  $row_i(X)$  and  $row_i(P)$  or two columns  $col_i(X)$  and  $col_i(P)$ .

# Problems of 2D Pattern Matching (3)

---

*Krithivasan and Sitalakshmi – 1987*: Extension of *the edit distance* (*Levenshtein – 1965*) for two dimensions. It allows *replace, insert, delete* edit operations.

Given two images of the same shape, their *KS* 2D-distance is the sum of edit distances of corresponding row or column pictures.

The *KS* distance computed using columns of two compared pictures can be described by this formula:

$$KS(X, P) = \sum_{i=1}^{n'} ed_L (col_i(X), col_i(P))$$

Function  $ed_L$  computes the Levenshtein edit distance comparing two columns (or rows) of pictures  $X$  and  $P$ .

# Problems of 2D Pattern Matching (4)

---

*Observe:* to be able to sum the number of 2D errors in an occurrence, the length of matched rows or columns should be fixed.

# Problems of 2D Pattern Matching (5)

---

*Observe:* to be able to sum the number of 2D errors in an occurrence, the length of matched rows or columns should be fixed.

*Question:* Having the length of 1D pictures fixed, are  $KS$  and Hamming distances between them still different?

# Problems of 2D Pattern Matching (6)

---

*Observe:* to be able to sum the number of 2D errors in an occurrence, the length of matched rows or columns should be fixed.

*Question:* Having the length of 1D pictures fixed, are  $KS$  and Hamming distances between them still different?

Consider  $p = a(ba)^i$ , and one *delete* (or *insert*) at position **2**, i.e.  $x = a(ab)^i c$ ,  $i > 0$ ,  $c \in \Sigma$ .

# Problems of 2D Pattern Matching (7)

*Observe:* to be able to sum the number of 2D errors in an occurrence, the length of matched rows or columns should be fixed.

*Question:* Having the length of 1D pictures fixed, are *KS* and Hamming distances between them still different?

Consider  $p = a(ba)^i$ , and one *delete* (or *insert*) at position **2**, i.e.  $x = a(ab)^i c$ ,  $i > 0$ ,  $c \in \Sigma$ .

Then

$$ed_H(x, p) = 2i, \quad \forall i > 0 \wedge c \neq a$$

$$ed_L(x, p) = 1 + 1, \quad c \neq a$$

# 2D Matching Using Finite Automata (1)

---

Let the pattern  $P \in \Sigma^{**}$  be viewed as a sequence of strings. Without loss of generality let these strings be its columns, i.e. the pattern is  $P \in L_P^{m\oplus}$ ,  $L_P \subseteq \Sigma^{m'}$  is a 1D language of columns of  $P$ .

Our method uses two finite automata:

1. (Nondeterministic) finite automaton (transducer) **preprocessing** text  $T$  as set of columns, and producing text  $T'$  of the same shape and size as  $T$ .
2. (Nondeterministic) finite automaton **searching** for a *representing string* of  $P$  in rows of  $T'$ ; (1D) occurrences of this string determine positions of 2D occurrences of  $P$  in  $T$ .

# 2D Matching Using Finite Automata (2)

---

## Algorithm

The strategy of searching for  $P$  in text  $T$ .

- \* Let  $\Pi$  be the set of all (distinct) columns of  $P$ , treated as individual strings.
- \* Build the dictionary matching automaton  $M(\Pi) = (Q, \Sigma, \Sigma', \delta, I, F)$  (transducer) of type  $SFF?CO$ .
  - \* Its final states represent some pattern from  $\Pi$  and
  - \* columns of  $P$  are identified with final states of the automaton  $M(\Pi)$ .
- \* Automaton  $M(\Pi)$  is applied to each column of  $T$  and new picture  $T'$  is generated. Its elements are determined by run of  $M(\Pi)$ .

# 2D Matching Using Finite Automata (3)

---

## Algorithm

- \* Create a *string*  $R$  over the set of final states representing the pattern  $P$ :  $i^{\text{th}}$  symbol of  $R$  is the final state identified with  $i^{\text{th}}$  column of  $P$ .
- \* Build the string matching automaton  $M'$  of type  $SFO?CO$ .
- \* Rest of the entire process consists of locating  $R$  inside the rows of  $T'$ , reporting eventual (1D) occurrences of  $R$  in  $T'$  and therefore also 2D occurrences of  $P$  in the original  $T$ .  
 $T' \in Q^{**}$ ,  $T' \in L_F^{n\oplus}$ ,  $L_F \subseteq F^{n'}$  is a 1D language of columns of  $T'$ .

# 2D Exact Pattern Matching (1)

Our automata-based method of 2D exact pattern matching is

- ✿ based on the idea of BIRD and BAKER.
- ✿ Searching for a set of strings and reducing dimension of the problem by *SFFECO* automaton  $A$ .
  - \* Set  $\Pi$  of (distinct) strings of pattern array  $P$  (its columns or rows).
  - \* Transducer  $A(\Pi) = (Q, \Sigma, Q, \delta, \{q_0\}, F)$  accepting language  $L(A)$ ,  $L(A) = \Sigma^* \Pi = \{wp; w \in \Sigma^*, p \in \Pi\}$ .
  - \*  $\forall q_1 \in Q, a \in \Sigma : \delta(q_1, a) \ni \{(q_2, q_2)\} \iff \exists \delta(q_1, a) \ni q_2$  in the *SFFECO* automaton.  $a \in \Sigma, q_2 \in Q$ .

# 2D Exact Pattern Matching (2)

✿ Searching for occurrences of  $R$  by *SFOECO* automaton  $A'$ .

✿  $A' = (\{q_0, q_1, \dots, q_m\}, \Sigma', \delta', \{q_0\}, \{q_m\})$ , accepting language

$$L(A') = \Sigma'^* R = \{wR; w \in \Sigma'^*, R \in (\Sigma' \setminus (\Sigma' \setminus F))^m\}.$$

$\Sigma'$  is the output alphabet and  $F$  is a set of labels of final states of automaton  $A$  from the preprocessing step.

# 2D Exact Pattern Matching (3)

Let  $P \in \Sigma^{m \times m'}$ ,  $T \in \Sigma^{n \times n'}$ .

▣▣▣▣➔ Asymptotic time complexity:  $\mathcal{O}(\log |\Sigma| |T|)$ .

▣▣▣▣➔ Asymptotic space complexity:  $\mathcal{O}(|T|)$ ,

Optimized processing needs only  $\mathcal{O}(\log |\Sigma| |P| + n)$  space.

# 2D Approximate Pattern Matching

Our model of 2D approximate pattern matching using the 2D Hamming distance is composed of two *NFA*'s.

They cannot be used directly – require simulation:

- ✿ Set  $\Pi$  of (distinct) strings of pattern array  $P$  (its columns or rows), the maximum number of 2D errors allowed  $k$ ,  $k \in \mathbb{N}$ ,  $k < mm'$ .

$A(\Pi) = (Q, \Sigma, \Sigma', \delta, I, F)$  accepting language  $L(A)$ ,

$L(A) = \Sigma^* H_k(\Pi) = \{uv; u, v \in \Sigma^*, ed_H(v, p) \leq \min(k, |p| - 1) \wedge p \in \Pi\}$ .


# 2D Approximate Pattern Matching

Let an ordered couple  $(s, x)$  be the label of a final state of  $A(\Pi)$ ,  $\emptyset \notin Q$  be a special symbol not among the labels of states.

Let  $s$  be a number of a string in  $\Pi$ ,  $1 \leq s \leq |\Pi|$ ,  $x$  be the number of errors found in the particular occurrence of  $p_s \in \Pi$ .

The output mapping of transducer  $A(\Pi)$  constructs  $T'$  so that:  
 $\forall i, j, s \ 1 \leq i \leq n, 1 \leq j \leq n', 1 \leq s \leq |\Pi|$ :

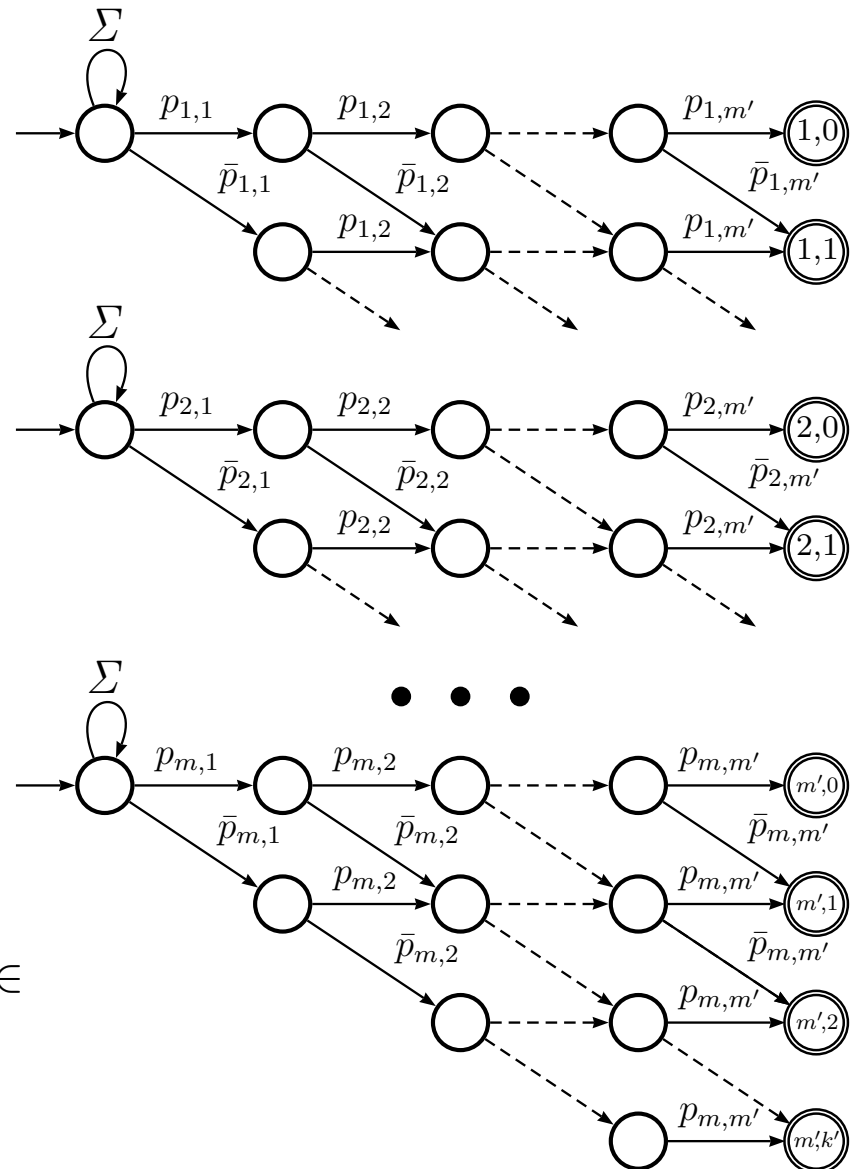
$$T'[i, j, s] = \begin{cases} x ; q \text{ active, } q \in F, q = (s, x), \\ \emptyset ; s^{\text{th}} \text{ sub-automaton has no active final state.} \end{cases}$$

 Preprocessing – dynamic programming in  
 $O(mm'nn') = O(|P||T|)$  time.

# 2D Approximate PM – Preprocessing

*SFFRCO* automaton for the approximate matching of set of strings using the Hamming distance accepts finite set of strings  $\Pi$ , each  $p_i \in \Pi$ , with at most  $k' = \min(|p_i| - 1, k)$  mismatches.

$L(A_\Pi) = \Sigma^* H_k(\Pi)$ , where  $H_k(\Pi) = \{x; x \in \Sigma^*, ed_H(x, p) \leq \min(k, |p| - 1) \wedge p \in \Pi\}$ .



# 2D Approximate Pattern Matching

---

For the matching phase an automaton is required for the error counting approximate pattern matching of representing string  $R$ .

Here  $SFO\Gamma CO$  automaton should be used, that is a finite automaton able to match pattern  $R$  using the  $\Gamma$  distance.

Definition: Let  $\Sigma'$  be an ordered alphabet.

$\Gamma$  *distance*  $ed_{\Gamma}(v, w)$  between two strings  $v, w \in \Sigma'^*$ ,  $|v| = |w|$ ,

is 
$$\sum_{i=1}^{|v|} |v[i] - w[i]| .$$

(Cambouropoulos, Crochemore, Iliopoulos, Mouchard, Pinzon – 1999)

# 2D Approximate Pattern Matching

Let the distances between two symbols, final states' labels of  $M(\Pi)$ , be defined as follows:

$$|(s, x) - (t, y)| = \begin{cases} |x - y| & ; \quad s = t, \\ m' & ; \quad ((s, x) \vee (t, y)) = \emptyset, \\ m' & ; \quad s \neq t. \end{cases}$$

Let function  $f(T'[i, j], s)$  be

$$f(T'[i, j], s) = \begin{cases} T'[i, j, s] & ; \quad T'[i, j, s] \neq \emptyset, \\ |p_s| & ; \quad T'[i, j, s] = \emptyset, \quad p_s \in \Pi. \end{cases}$$

# 2D Approximate Pattern Matching

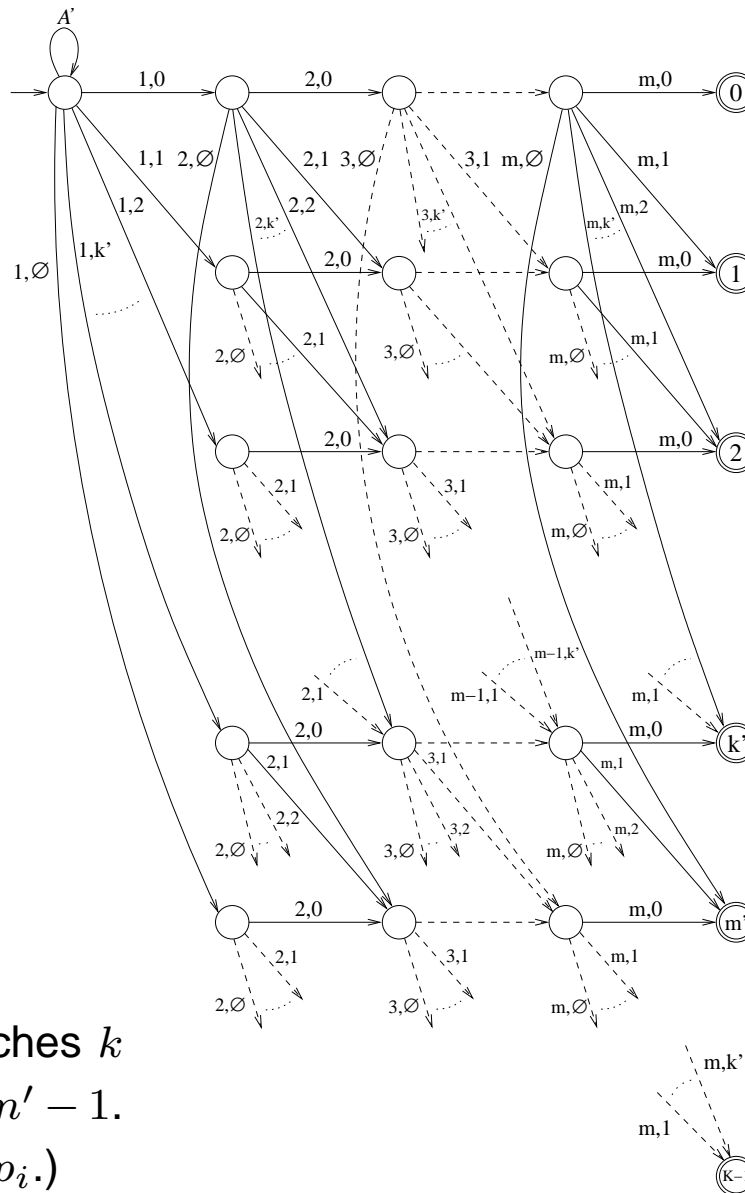
The matching phase:

Let  $A'$  be  $A' = (Q', \Sigma', \delta', \{q'_0\}, F')$ .

$\delta'$  is a mapping  $Q' \times f(cur\_pos, s) \mapsto 2^{Q'}$ , where  $cur\_pos$  is the current element of  $T'$  under the head of  $A'$ ,  $s$  is the number of a string from  $\Pi$  expected at the current position.

- ✿ Matching – simulation in  $\mathcal{O}(m|T|)$  time.
- ➡ Asymptotic time complexity:  $\mathcal{O}(|P||T|)$ .
- ➡ Asymptotic space complexity:  $\mathcal{O}(m|T|)$ ,  
optimized processing needs only  $\mathcal{O}(mn + |P|n)$ .

# 2D Approximate PM – Matching



The maximum number of mismatches  $k$  can be  $k = K - 1 = |P| - 1 = mm' - 1$ . (“ $i, \emptyset$ ” denotes no match of string  $p_i$ .)

# 2D Approximate Pattern Matching

---

Using different automaton for preprocessing of  $T$  we have obtained a new model of 2D approximate pattern matching using the  $KS$  distance.

As in the previous solution for 2D Hamming distance, this model requires simulation too. The complexities remain the same.

- ➡ Asymptotic time complexity:  $\mathcal{O}(|P||T|)$ .
- ➡ Asymptotic space complexity:  $\mathcal{O}(m|T|)$ .

# Summary and Open Problems

- ✿ Finite automata can be used even for multidimensional pattern matching.
- ✿ Method of 2D exact pattern matching.
- ✿ Methods of 2D approximate pattern matching for two distances: 2D Hamming and  $KS$ .

## Open problems:

- ✿ Models for more distances.
- ✿ Possibilities of application of weighted automata to deal with some problems encountered in practice.

There exist different approaches to the problem, for the comparison only some serial deterministic approaches are taken.

## ➡ 2D exact pattern matching

- ✿ First result: **Bird, Baker** (independently in 1977–8).  
Based on Knuth-Morris-Pratt and Aho-Corasick algorithms.  $\mathcal{O}(\log |\Sigma| nn')$  asymptotic time complexity.
  - \* Our result is equivalent in terms of time complexity and improves the space complexity.
- ✿ **Amir, Benson and Farach** (1992)  
(with preprocessing given by **Galil and Park** (1992))  
linear worst-case time, alphabet-independent algorithm.
  - \* Our result is alphabet dependent, it is linear provided that the alphabet is fixed.

## ⇒ 2D approximate pattern matching

✿ **Krithivasan and Sitalakshmi (1987).**

$\mathcal{O}\left(km\left(m' \log m' + \frac{mm'}{k} + nn'\right)\right)$  time complexity.

✿ **Ranka and Heywood (1991)**  $\mathcal{O}\left((k + a)(b \log b + n^2)\right)$  time complexity, where  $a = \min(m, m')$  and  $b = \max(m, m')$ .

✿ **Amir and Landau (1991), Crochemore and Rytter (1994)**  $\mathcal{O}\left((k + \log |\Sigma|)n^2\right)$  time complexity.

✿ **Our result:**  $\mathcal{O}(m^2n^2)$ . (Consider  $k \rightarrow m^2$  above.)