Approximate Regular Expression Matching

Pavel Mužátko

Department of Computer Science and Engineering,
Faculty of Electrical Engineering,
Czech Technical University,
Karlovo nám. 13,
121 35 Prague 2,
Czech Republic

Abstract. We extend the definition of Hamming and Levenshtein distance between two strings used in approximate string matching so that these two distances can be used also in approximate regular expression matching. Next, the methods of construction of nondeterministic finite automata for approximate regular expression matching considering both mentioned distances are presented.

1 Introduction

The notions from the theory of approximate string matching will be used for describing the problem of approximate regular expression matching. Approximate string matching is defined as follows: A text T, a pattern P, and an integer k are given. All occurrences of a substring X should be found such that the distance $D(P, X)$ between the string X and the pattern P is less or equal to k.

There are two basic types of distances called Hamming distance and Levenshtein distance. The Hamming distance (notation D_H) between two strings of equal length is the number of positions with mismatching symbols in this two strings. The Levenshtein or edit distance (notation D_L) between two strings P and X, not necessarily of equal length, is the minimal number of editing operations insert, delete, and replace needed to convert P into X.

If the Hamming distance is used then the approximate string matching is referred as string matching with k mismatches. If the Levenshtein distance is used then the approximate string matching is referred as string matching with k differences. Similarly, the notions regular expression matching with k mismatches and regular expression matching with k differences will be used.

2 Definition of Regular Expressions

Definition 1
A regular expression V over an alphabet A is defined as follows:

1. $\emptyset, \varepsilon, a$ are regular expressions for all $a \in A$.

2. If x, y are regular expressions over A then:

   (a) $(x + y)$ (union)

   (b) $(x.y)$ (concatenation)

   (c) $(x)^*$ (closure)

   are regular expressions over A.

Definition 2
A value h(x) of a regular expression x is defined as follows:

1. $h(\emptyset) = \emptyset, h(\varepsilon) = \{\varepsilon\}, h(a) = \{a\}$,

2. $h(x + y) = h(x) \cup h(y)$,
   $h(x.y) = h(x).h(y)$,
   $h(x^*) = (h(x))^*$.

The value of a regular expression is a regular language, a set of patterns. Unnecessary parentheses in regular expressions can be avoided by the convection for precedence of regular operations. The highest precedence has the closure operator, the lowest precedence has the union operator.

3   Regular Expression Matching with k Mismatches

The Hamming distance D_H^R between a regular expression V with a value $h(V)$ and a string X can be defined by using the Hamming distance D_H between two strings as follows:

$D_H^R = \min_{w \in h(V) \wedge |w| = |X|} D_H(w, X)$

Now, the construction of a nondeterministic finite automaton accepting patterns with the postfix generated by a given regular expression with k mismatches is presented.

Let a regular expression V over an alphabet A is given, and $M = (Q_0, A, q_0, \delta_0, F_0)$ is a nondeterministic finite automaton accepting the language $L = h(V)$. Let the automaton M has m states. The automaton $M_H^R = (Q, A, q_0, \delta, F)$ accepting patterns with the postfix from $h(V)$ with k mismatches will be constructed by interconnecting the $k + 1$ clones $M_0, ..., M_k$ of the automaton M.
Each state of the automaton M_H^R is labeled by $q_{i,j}$, where i is the number of the clone, $0 \leq i \leq k$, j is the number of the state inside the clone M_i, $0 \leq j \leq m - 1$. The mapping δ of the automaton M_H^R will be defined in the following way:

1. All transitions defined in the automata $M_0, ..., M_k$ will be also included in the automaton M_H^R.
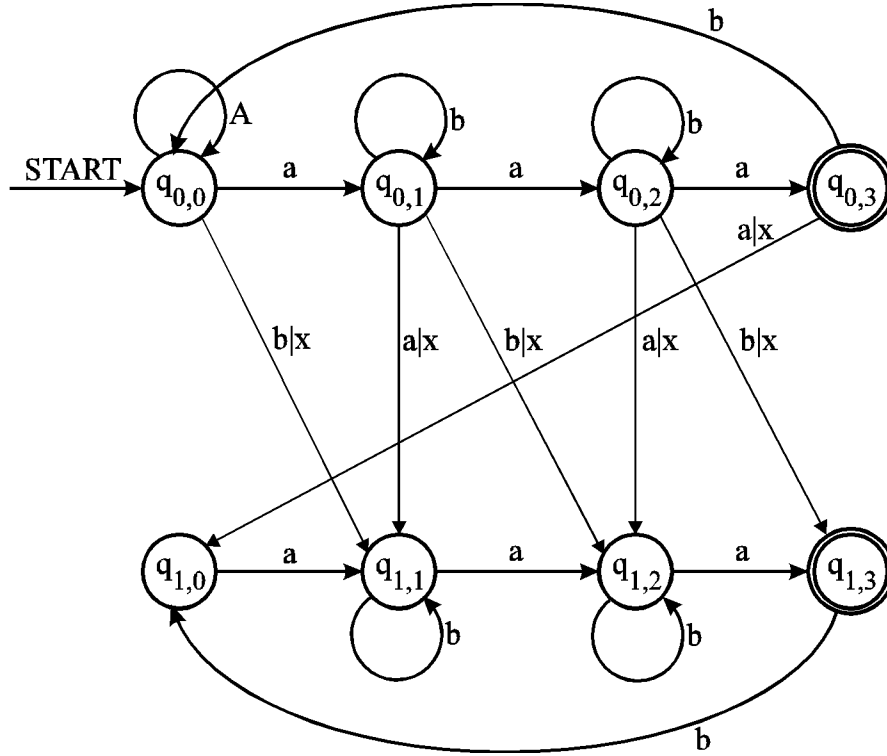
Figure 1: Nondeterministic automaton H_1.

2. Error transitions will be added. For each state $q_{i,j}$ $(0 \leq i \leq k - 1, 0 \leq j \leq m - 1)$ and for each such a symbol $a \in A$, for which $\delta(q_{i,j}, a)$ is defined, define $\delta(q_{i,j}, \overline{a}) = \delta(q_{i+1,j}, a)$, where $\overline{a}$ denotes all symbols from the alphabet A except the symbol a.

3. A self loop for all symbols from the alphabet A will be added for the state $q_{0,0}$.

The initial state of the automaton M_H^R is the state $q_{0,0}$. The set of final states $F = F_0 \cup F_1 \cup ... \cup F_k$.

The number of states of the automaton M_H^R is $m(k + 1)$.

Example 1

A transition diagram of a nondeterministic automaton H_1 accepting with 1 mismatch patterns with the postfix described by the regular expression $V = ab^*ab^*a(bab^*ab^*a)^*$ over the alphabet $A = \{a, b, x\}$ can be found in Fig. 1. This automaton accepts all strings with a postfix X such that $D_H^R(V, X) \leq 1$. The result of searching in the text $aabxabaa$ can be described as follows: $aab_{(1)}x_{(1)}a_{(1)}ba_{(1)}a_{(0,1)}$. The number in parentheses shows the number of mismatches occurred when a final state of the automaton H_1 was reached.

4 Regular Expression Matching with k Differences

The Levenshtein distance D_L^R between a regular expression V with a value $h(V)$ and a string X can be defined by using the Levenshtein distance D_L between two strings

as follows:

$$D_L^R = \min_{w \in h(V)} D_L(w, X)$$

Let V be again a regular expression over an alphabet A and M is a nondeterministic finite automaton accepting the language $L = h(V)$. We will construct a nondeterministic finite automaton M_L^R accepting with k differences all patterns with the postfix from $h(V)$. This automaton will be as in the previous case constructed by interconnecting the $k + 1$ clones $M_0, ..., M_k$ of the automaton M.
Each state of the automaton M_L^R is again labeled by $q_{i,j}$, where i is the number of the clone, $0 \leq i \leq k$, j is the number of the state inside the clone M_i, $0 \leq j \leq m - 1$. The mapping δ of the automaton M_L^R is defined in the following way:

1. All transitions defined in the automata $M_0, ..., M_k$ will be also included in the automaton M_L^R.

2. Replace transitions will be added. For each state $q_{i,j}$ ($0 \leq i \leq k - 1$, $0 \leq j \leq m - 1$) and for each such a symbol $a \in A$, for which $\delta(q_{i,j}, a)$ is defined, define $\delta(q_{i,j}, \overline{a}) = \delta(q_{i+1,j}, a)$, where $\overline{a}$ denotes all symbols from the alphabet A except the symbol a.

3. Delete transitions will be added. For each state $q_{i,j}$ and for each symbol $a \in A$ ($0 \leq i \leq k - 1$, $0 \leq j \leq m - 1$) $\delta(q_{i,j}, \varepsilon) = \delta(q_{i+1,j}, a)$.

4. Insert transitions will be added. For each state $q_{i,j}$ ($0 \leq i \leq k - 1$, $0 \leq j \leq m - 1$), and for each symbol $a \in A$ $\delta(q_{i,j}, a) = q_{i+1,j}$. All replace transitions between states, where insert transitions are also defined (e.g. the replace transitions between the states $q_{i,j}$ and $q_{i+1,j}$), can be removed.

5. A self loop for all symbols from the alphabet A will be added for the state $q_{0,0}$.

The initial state of the automaton M_L^R is the state $q_{0,0}$. The set of final states $F = F_0 \cup F_1 \cup ... \cup F_k$.

The number of states of the automaton M_L^R is $m(k + 1)$.

Example 2

A transition diagram of a nondeterministic automaton L_1 accepting with maximaly 1 difference patterns with the postfix defined by the regular expression $V = ab^*ab^*a(bab^*ab^*a)^*$ over the alphabet $A = \{a, b, x\}$ can be found in Fig. 2. Delete transitions are depicted as dashed lines. This automaton accepts all strings with a postfix X such that $D_L^R(V, P) \leq 1$.

The result of searching in the text *abxaa* can be described as follows:

$abxa_{(R)}a_{(R,I)}.$

The symbol in parentheses determines the operation needed to convert some pattern from $h(V)$ to the string read when a final state was reached.

The notation (R, I) has the following meaning:
The string *abbaa* $\in h(V)$ can be converted to the string *abxaa* by using one replace operation. The string *abaa* $\in h(V)$ can be converted to the string *abxaa* by using one insert operation.

Example 3

Let us consider the input text *abbbabab*. We are interested in finding all occurrences of
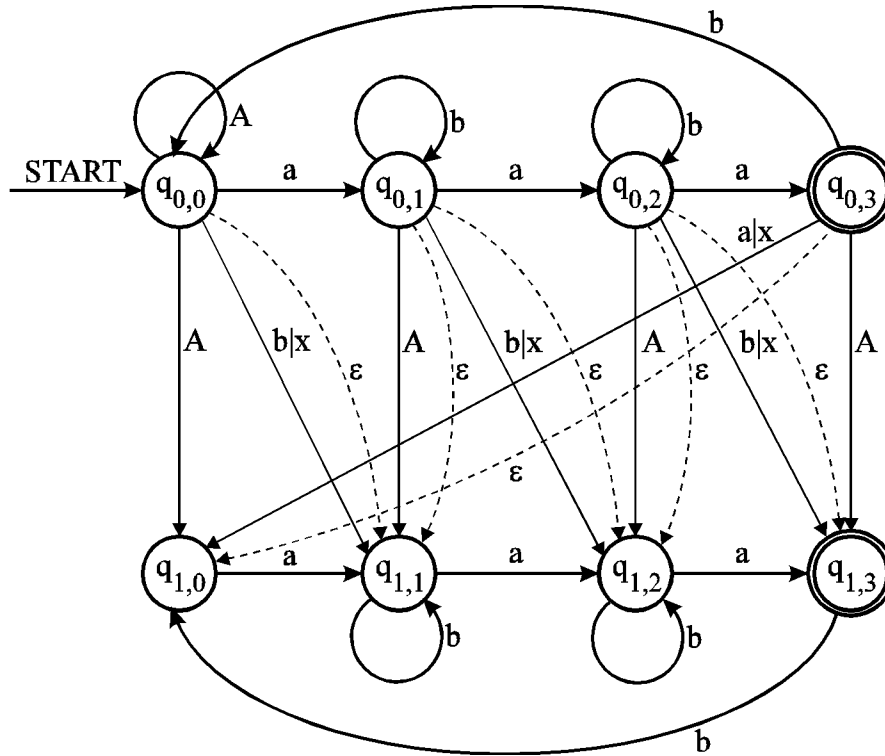
Figure 2: Nondeterministic automaton L_1.

strings with the postfix X such that $D_L^R(V, X) \leq 1$, where V is the regular expression from the previous example. The automaton L_1 will be used. The result can be described as follows:

$$abbba_{(R,D)}b_{(R,D)}a_{(0,R,D,I)}b_{(R,D,I)}.$$

The symbol 0 denotes the occurrence of a string from $h(V)$.

5   Conclusion

Both the nondeterministic automata M_H^R and M_L^R have to be deterministicaly simulated for practical purpose. But during the process of creating of equivalent deterministic finite automata the number of states can rise exponentialy, while the deterministic simulation of a nondeterministic automaton is of a high time complexity. It seems that this problem can be solved by constructing of a hybrid deterministic-nondeterministic finite automaton, but the problem is still open.

References

[1]   Aho, A., Ullman, J.: The Theory of Parsing, Translation, and Compiling. Vol. I: Parsing, Prentice Hall, Englewood Cliffs, New York 1992.

[2]   Melichar, B.: Approximate string matching by finite automata. In: Computer Analysis of Images and Patterns, LNCS 970, Springer 1995, pp. 132 – 137.