Algebra of Pattern Matching Automata

Václav Snášel, Tomáš Koutný

Department of Computer Science
Palacky University
Tomkova 40
771 00 Olomouc
Czech Republic

e-mail: {Vaclav.Snasel, Tomas.Koutny}@upol.cz

Abstract. In this paper we classify pattern matching problems using algebraic means. We construct an algebra of pattern matching automata in which finite automata are the elements and operations applied to them correspond to the creation of new pattern matching problems. We present several such operations and describe some identified properties of the algebra defined in this way.

Key words: pattern matching, finite automata, algebra

1 Introduction

In this paper we will classify pattern matching problems using algebraic means. We will define an algebra of pattern matching automata. Elements of this algebra will be automata and defined operations will correspond to the creation of pattern matching problems.

Melichar and Holub also deal with pattern matching problems classification in their work [MH97]. However, their approach is different. They describe pattern matching problems using 6 criteria and therefore they can locate them in 6D space. They show how to construct a finite automaton for each point of the space. It is possible to start from a simple automaton which performs an exact match and transform it into a more sophisticated one. This inspired us to create an algebra of finite automata and to define operations with them which would correspond to transformations resulting in different points of the 6D space. This may provide an answer to an interesting question: will the number of obtained points be finite after a multiple application of such operations or not?

We wanted to construct operations for each of the six axes of the space which would generate automata corresponding to the points already identified. Then we could apply these operations several times and identify the properties of the algebra defined by the set of finite automata and these operations. In this paper we present up-to-date results of our work. We created an algebra of pattern matching automata in which only several of the intended operations are defined. To support our theory, we created a program in which we implemented operations identified on the automata so far. Using the program we obtained experimental data which was useful for us when constructing proofs for our statements.

In the next part we will give a precise definition of the algebra of pattern matching automata. Following that, we will characterize the algebra using identities discovered. In chapter 3 we will describe the program used during our experiment. To conclude with, we will mention other possible implementations of finite automata which are more suitable for practical application and outline our further research goals.

2 Definition of the algebra of pattern matching automata

2.1 Notation

Notation not listed directly in this paper can be found in [MI91].

Definition: Nondeterministic automaton is the quantiple M defined as $M = (\Sigma, Q, \delta, q_0, F)$, where

Σ is a finite alphabet

Q is a finite set of states

$q_0 \in Q$ is the start state

$F \subset Q$ is a set of final states

δ, a transition relation, is a finite subset of $Q \times \Sigma^* \times Q$.

In the remaining part of the text we will use an extended alphabet Σ', which is derived from Σ as follows:

Let Σ be a finite set. Then we can use Σ^- to denote the set $\{\bar{x} | \bar{x} = \Sigma - \{x\} \forall x \in \Sigma\}$. Let's use Γ to denote the symbol corresponding to the whole set and ε to denote the symbol corresponding to an empty transition. Then we can define the extended alphabet Σ' over the set Σ as $\Sigma \cup \Sigma^- \cup \{\varepsilon\} \cup \{\Gamma\}$.

For the definition of the algebra of pattern matching automata we will need to define an operation $\oplus$: merger of automata.

Definition: Let A be a set of automata using the same alphabet Σ and the same start state q_0. Then for the automata $X = (\Sigma, Q^X, \delta^X, q_0, F^X) \in A$ and $Y = (\Sigma, Q^Y, \delta^Y, q_0, F^Y) \in A$ we will define the automaton $X \oplus Y$ in the following way:

$$X \oplus Y = (\Sigma, Q^Y \cup Q^X, \delta^X \cup \delta^Y, q_0, F^X \cup F^Y)$$

Note: $\delta^X \cup \delta^Y$ for our purposes means a union of relations.

In the following we will assume that set A is closed with respect to construction $\oplus$. Then $(A, \oplus)$ is an algebra with a binary operation.

Theorem 1. Algebra $(A, \oplus)$ satisfies the following identities:

$$a \oplus a = a$$

$$a \oplus b = b \oplus a$$

$$(a \oplus b) \oplus c = a \oplus (b \oplus c)$$

$$\forall a, b, c \in A$$

`Proof.`
The proof is obvious and ensues directly from the construction of operation $\oplus$. □

2.2 Definition of operations R, I, D

[MH97] introduced constructions $R(X,k)$ and $DIR(X,k)$.
These constructions correspond to the creation of a nondeterministic automaton X' from the automaton X which accepts a string $P = p_1 p_2 \ldots p_n$. Automaton X' accepts only those strings P' with the value of P to P' distance equivalent to k while using distance R or DIR. Distance $R(P,P')$ is called Hamming distance and is defined as the minimum number of symbol replacement operations in string P required for the conversion of string P into string P'. Distance $DIR(P,P')$ is called Levenshtein distance and is defined as the minimum number of operations of symbol deletion (D), insertion (I) or replacement (R) in P required for the conversion of string P into string P'.
Automaton X' is called $R-$trellis or $DIR-$trellis as the case may be. Their construction is described, for example, in [MH97] or [HO96] (see Fig. 1).

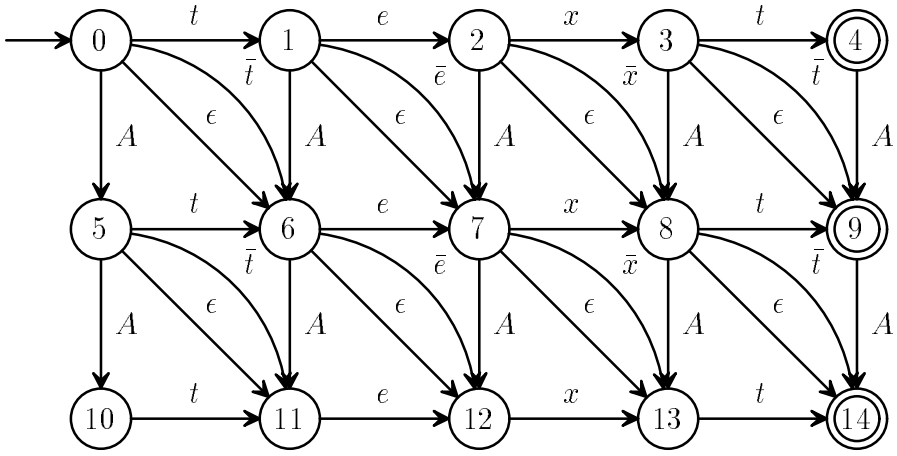

Figure 1: $DIR-$trellis for string "text", $k = 2$.

When defining the operations in our algebra we are not restricted to the set of automata which perform an exact match but we define operations R and DIR for any finite automaton X in a similar way as Mužátko in his generalization of regular expression matching automata in [MU96]. Since distance DIR corresponds to any combination of operations of deletion D, insertion I, or replacement R, it is possible to define each of these operations independently.

Definition: Let $X = (\Sigma, Q, \delta, q_0, F)$ be a finite automaton and let $k \in N$. The result of operations $D(X,k)$, $I(X,k)$ or $R(X,k)$ is an automaton X' which will be derived from automaton X through the following steps:

1. Automaton X' will contain $k+1$ clones of automaton X.

2. The states of automaton X' will be labelled $q_{i,j}$ where i is the sequence number of the clone and j is the sequence number of the state inside the original automaton X.

3. All transitions defined in the original automaton X will remain included in all its clones.

4. Error transitions will be added into automaton X' according to one of the following operations:

   Operation R: $\delta(q_{i,j}, \bar{a}) = \delta(q_{i+1,j}, a)$ shall be defined for each state $q_{i,j} (0 \leq i \leq k - 1, 0 \leq j \leq m-1)$ and for each symbol $a \in \Sigma$ for which transition $\delta(q_{i,j}, a)$ is defined. The symbol $\bar{a} \in \Sigma^-$ represents all symbols from alphabet Σ not equal to symbol a; or

   Operation D: $\delta(q_{i,j}, \varepsilon) = \delta(q_{i+1,j}, a)$ shall be defined for each state $q_{i,j}$ and for each symbol $a \in \Sigma, (0 \leq i \leq k - 1, 0 \leq j \leq m-1)$; or

   Operation I: $\delta(q_{i,j}, \Gamma) = q_{i+1,j}$ shall be defined for each state $q_{i,j}, (0 \leq i \leq k - 1, 0 \leq j \leq m-1)$.

5. Start state of automaton X' is state $q_{0,0}$. The alphabet of the automaton is the extended alphabet Σ'. The set of final states is the union of final states in all the clones: $F = F_0 \cup F_1 \cup \ldots F \cup F_k$.

Definition: Operations $D(X), R(X)$ and $I(X)$ correspond to operations $D(X,1), R(X,1)$ and $I(X,1)$ respectively.

Now we are ready to define the algebra of pattern matching automata.

Definition: The algebra of pattern matching automata is the algebra $\mathcal{A} = (A, D, I, R, *, +, \cdot, \oplus)$, where

A is a set of finite automata

$\oplus$ is an operation of merger

D is an operation of deletion

I is an operation of insertion

R is an operation of replacement

$*$ is an operation of closure

$+$ is an operation of union

$\cdot$ is an operation of concatenation

2.3 Properties of the algebra of pattern matching automata

Theorem 2. Let A be an algebra of pattern matching automata. Then for each automaton $X \in A$ it holds that

$$R(X,k) = R^k(X)$$

where $R^k(X)$ means $\underbrace{R(R(R(...R(X))...)))}_{k \ times}$.

Proof.
We will prove the theorem using mathematical induction.

1. According to the definition it holds: $R(X,1) = R(X)$.

2. Let's assume that $R(X,k) = R^k(X)$ holds.

3. If string $w_0r_0w_1r_1 \ldots w_kr_kw_{k+1}$ is accepted by automaton X, then automaton $R(X,k+1)$ accepts the string $w = w_0r'_0w_1r'_1 \ldots w_kr'_kw_{k+1}$, where $r'_i \in \{\bar{r}_i, r_i\}$. According to induction hypothesis, automaton R^k accepts string $w_0r'_0w_1r'_1 \ldots w_kr_kw_{k+1}$. After reading string $w_0r'_0w_1r'_1 \ldots w_k$, automaton $R^k(X)$ is in a state $q_{w_0r'_0w_1r'_1\ldots w_k}$. It is obvious from the construction of automaton $R(R^k(X))$ that if $r'_k = \bar{r}_k$, then there is a transition from state $q_{w_0r'_0w_1r'_1\ldots w_k,0}$ into state $q_{w_0r'_0w_1r'_1\ldots w_kr'_k,1}$ and hence automaton $R(R^k(X)) = R^{k+1}(X)$ accepts string w. If $r'_k = r_k$, it is obvious that automaton $R(R^k(X)) = R^{k+1}(X)$ accepts string w.

   In reverse, let's assume that automaton $R^{k+1}(X)$ accepts string $w = w_0r'_0w_1r'_1 \ldots w_kr'_kw_{k+1}$. Then again according induction hypothesis it holds that $R^{k+1}(X) = R(R(X,k))$. Automaton $R(X,k)$ accepts string $w_0r'_0w_1r'_1 \ldots w_kr_kw_{k+1}$. It is obvious from the construction of the trellis that automaton $R(R(X,k))$ accepts string $w_0r'_0w_1r'_1 \ldots w_kr'_kw_{k+1}$ and hence the languages accepted by both automata, $R^{k+1}(X)$ and $R(X,k+1)$, are equal. □

Similar proofs can be provided for the remaining operations, D and I. Figure 2 shows an example of an automaton for operation D and $k = 2$.

We have not defined an equivalent of the DIR construction in our algebra. It is not really necessary. The corresponding DIR operation in our algebra will result from suitable application of operation $\oplus$ to automata $D(X), I(X)$ and $R(X)$:

$$DIR(X) = D(X) \oplus I(X) \oplus R(X)$$

The correctness of such a definition ensues from the behaviour of operation $\oplus$.

It is also possible to define the following in a similar way:

$$DI(X) = D(X) \oplus I(X)$$

$$DR(X) = D(X) \oplus R(X)$$

$$RI(X) = R(X) \oplus I(X)$$

Theorem 3. For any automaton $X = (\Sigma, Q, \delta, q_0, F)$ it holds that

$$DIR(X,k) = D^k(X) \oplus I^k(X) \oplus R^k(X)$$

`Proof.`
Let Σ' be an extended alphabet over alphabet Σ.
$DIR(X,1) = D(X) \oplus I(X) \oplus R(X)$ ensues directly from the definition of the construction.
If string $w = w_0r_0w_1r_1 \ldots w_kr_kw_{k+1}$, where $r_i \in \Sigma$, is accepted by automaton X, then automaton $DIR(X,k)$ accepts string $w = w_0r'_0w_1r'_1 \ldots w_kr'_kw_{k+1}$, where $r'_i \in \Sigma'$. If $r'_0 = \Gamma$, then having read string $w_0r'_0$ using transitions from automaton $I^k(X)$, automaton $DIR(X,k)$ reaches state $q_{w_0r'_0}$ which corresponds to the state of automaton X after the acceptance of word w_0r_0. Similarly, it is possible to demonstrate that for any segment of string w', automaton $D^k(X) \oplus I^k(X) \oplus R^k(X)$ reaches the state

corresponding to the equivalent segment of string w.
It means that string w' transfers the automaton into a final state.
Let's suppose that string w is accepted by automaton X. The string accepted by automaton $D^k(X) \oplus I^k(X) \oplus R^k(X)$ will contain $k+1$ segments separated by symbols from set Σ'. It means that this string will be accepted by automaton $DIR(X,k)$ too. □

3 Program

During the phase of building the hypotheses we carried out some experiments using our own program written in Microsoft Visual Basic 5.0. In this program we implemented nondeterministic finite automata and operations with such automata.

We used the method of simulation of the nondeterministic finite automaton in a deterministic way. The usage of symbols from the extended alphabet over the original alphabet prevented the rapid increase of states and transitions which would otherwise become inevitable during multiple application of the operation to the original string searching automaton.
The aim of the implementation was to verify our hypotheses and which was why we didn't pay any special attention to the effectiveness of the implementation. In case where the speed of the algorithm is one of the main criteria, it is possible to use a different type of implementation as described, for example by Mohri in [MM95].

In the following we give an example of pseudocode which demonstrates the algorithm of determination of active states after reading a terminal from input.

```
EpsilonPath(states)
1  i <- 1
2  While <= states.Count Do
3     For Each trans In m_Transitions[states[i],Epsilon]
4        Union (states,trans)
5     i <- i + 1

NewStates(old_states, t)
1   new_states <- EmptySet
2   For Each index In t
3      For Each or_state In old_states
4         For Each trans In m_Transitions[or_state,index]
5            If trans <> EmptyState And trans <> PreStartState Then
6               Union (new_states,trans)
7   EpsilonPath (new_states)
8   NewStates <- new_states

Go(t)
1   t_col <- m_TableOfTerminals.TermToIndexes(t)
2   m_ActiveStates <- NewStates (m_ActiveStates, t_col)
```

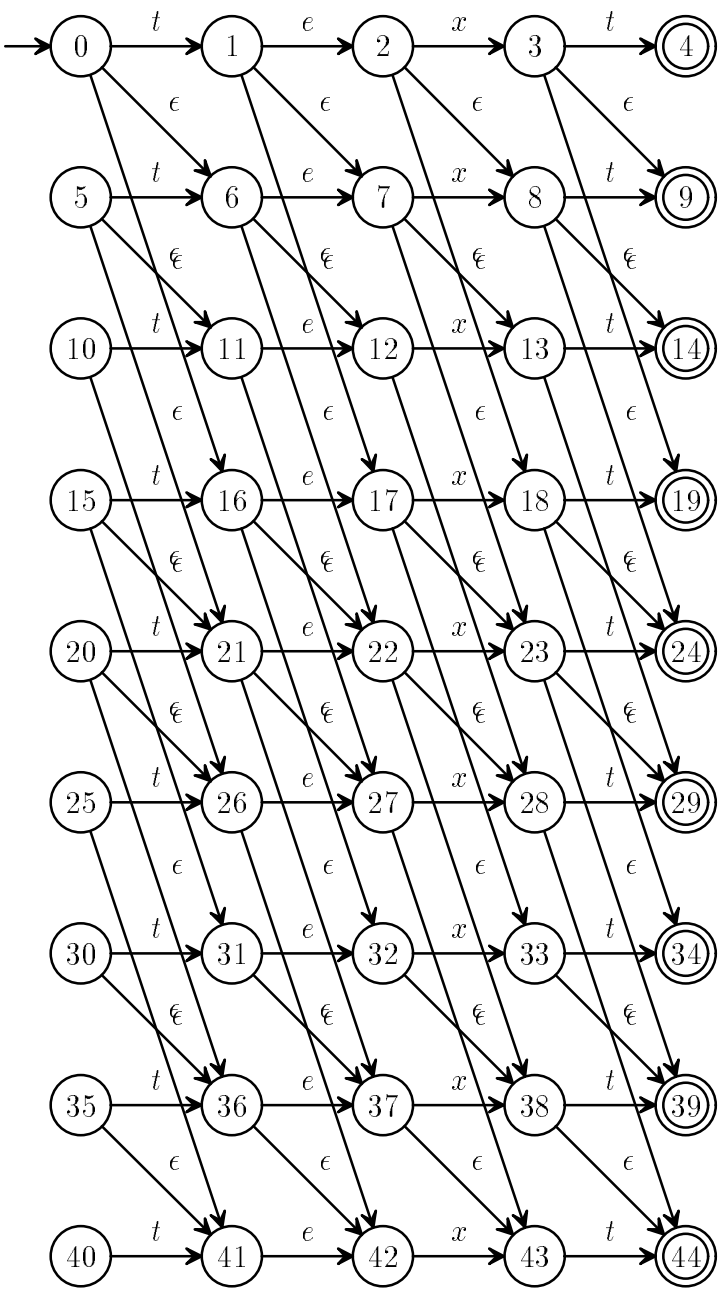

Figure 2: Double application of operation D on automaton performing exact match of string "text".

4 Conclusion

As we mentioned in the introduction we still find ourselves in the middle of the work. Our target is to create a description of pattern matching problems on an algebraic basis. This could be achieved by gradual addition of other operations which would correspond to the axes of the 6D space not included in our work yet. A further step of our research could be extension of the computing power of nondeterministic finite automata through the application of other models, such as multitape automata, and finding out whether they are suitable for pattern matching.

Another problem is the construction of an effective automaton (see [CR94]). As mentioned in the previous part of the paper, a construction scheme of the deterministic automaton for a regular expression can be found in [MM95]. We think that this scheme could be applicable for our purposes too.

References

[CR94] M. Crochemore, W. Rytter. *Text Algorithms.* Oxford University Press, 1994.

[HO96] J. Holub. *Reduced Nondeterministic Finite Automata for Approximate String Matching.* Proceedings of the Prague Stringology Club Workshop '96.

[LP81] H. R. Lewis, C. H. Papadimitriou. *Elements of the Theory of Computation.* Prentice Hall 1981.

[MH97] B. Melichar, J. Holub. *6D Classification of Pattern Matching Problems.* In this volume.

[MI91] B. Mikolajczak ed. *Algebraic and Structural Automata Theory.* North Holland 1991.

[MM95] M. Mohri. *Matching Patterns of an Automaton.* Combinatorial Pattern Matching, 6th Annual Symposium, CPM 95, Espoo, Finland, Springer Verlag 1995.

[MU96] P. Mužátko. *Approximate Regular Expression Matching.* Proceedings of the Prague Stringology Club Workshop '96.

[DP90] D. Perrin. *Finite Automata. Handbook of Theoretical Computer Science.* Elsevier Science Publishers 1990.