Minimal Cover-Automata for Finite Languages[⋆]

Cezar Câmpeanu, Nicolae Sântean, and Sheng Yu

Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 5B7
{cezar,santean,syu}@csd.uwo.ca

Abstract. A cover-automaton A of a finite language $L \subseteq \Sigma^*$ is a finite automaton that accepts all words in L and possibly other words that are longer than any word in L. A minimal deterministic cover automaton of a finite language L usually has a smaller size than a minimal DFA that accept L. Thus, cover automata can be used to reduce the size of the representations of finite languages in practice. In this paper, we describe an efficient algorithm that, for a given DFA accepting a finite language, constructs a minimal deterministic finite cover- automaton of the language. We also give algorithms for the boolean operations on deterministic cover automata, i.e., on the finite languages they represent.

1 Introduction

Regular languages and finite automata are widely used in many areas such as lexical analysis, string matching, circuit testing, image compression, and parallel processing. However, many applications of regular languages use actually only finite languages. The number of states of a finite automaton that accepts a finite language is at least one more than the length of the longest word in the language, and can even be in the order of exponential to that number. If we do not restrict an automaton to accept the exact given finite language but allow it to accept extra words that are longer than the longest word in the language, we may obtain an automaton such that the number of states is significantly reduced. In most applications, we know what is the maximum length of the words in the language, and the systems usually keep track of the length of an input word anyway. So, for a finite language, we can use such an automaton plus an integer to check the membership of the language. This is the basic idea behind cover automata for finite languages.

Informally, a cover-automaton A of a finite language $L \subseteq \Sigma^*$ is a finite automaton that accepts all words in L and possibly other words that are longer than any word in L. In many cases, a minimal deterministic cover automaton of a finite language L has a much smaller size than a minimal DFA that accept

[⋆] This research is supported by the Natural Sciences and Engineering Research Council of Canada grants OGP0041630.

L. Thus, cover automata can be used to reduce the size of automata for finite languages in practice.

Intuitively, a finite automaton that accepts a finite language (exactly) can be viewed as having structures for the following two functionalities:

1. checking the patterns of the words in the language, and
2. controlling the lengths of the words.

In a high-level programming language environment, the length-control function is much easier to implement by counting with an integer than by using the structures of an automaton. Furthermore, the system usually does the length-counting anyway. Therefore, a DFA accepting a finite language may leave out the structures for the length-control function and, thus, reduce its complexity.

The concept of cover automata is not totally new. Similar concepts have been studied in different contexts and for different purposes. See, for example, [1,7,4,10]. Most of previous work has been in the study of a descriptive complexity measure of arbitrary languages, which is called "automaticity" by Shallit et al. [10]. In our study, we consider cover automata as an implementing method that may reduce the size of the automata that represent finite languages.

In this paper, as our main result, we give an efficient algorithm that, for a given finite language (given as a deterministic finite automaton or a cover automaton), constructs a *minimal* cover automaton for the language. Note that for a given finite language, there might be several minimal cover automata that are not equivalent under a morphism. We will show that, however, they all have the same number of states.

2 Preliminaries

Let T be a set. Then by $\#T$ we mean the cardinality of T. The elements of T^* are called strings or words. The empty string is denoted by λ. If $w \in T^*$ then $|w|$ is the length of x.

We define $T^l = \{w \in T^* \mid |w| = l\}$, $T^{\leq l} = \bigcup_{i=0}^{l} T^i$, and $T^{<l} = \bigcup_{i=0}^{l-1} T^i$. We say that x is a prefix of y, denoted $x \preceq_p y$, if $y = xz$ for some $z \in T^*$.The relation $\preceq_p$ is a partial order on T^*. If $T = \{t_1, \ldots, t_k\}$ is an ordered set, $k > 0$, the quasi-lexicographical order on T^*, denoted $\prec$, is defined by: $x \prec y$ iff $|x| < |y|$ or $|x| = |y|$ and $x = zt_iv$, $y = zt_ju$, $i < j$, for some $z, u, v \in T^*$ and $1 \leq i, j \leq k$. Denote $x \preceq y$ if $x \prec y$ or $x = y$.

We say that x is a prefix of y, denoted $x \preceq_p y$, if $y = xz$ for some $z \in T$.

A deterministic finite automaton (DFA) is a quintuple $A = (\Sigma, Q, q_0, \delta, F)$, where Σ and Q are finite nonempty sets, $q_0 \in Q$, $F \subseteq Q$ and $\delta : Q \times \Sigma \longrightarrow Q$ is the transition function. We can extend δ from $Q \times \Sigma$ to $Q \times \Sigma^*$ by

$$\overline{\delta}(s, \lambda) = s$$

$$\overline{\delta}(s, aw) = \overline{\delta}(\delta(s, a), w).$$

We usually denote $\overline{\delta}$ by δ.

The language recognised by the automaton A is $L(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$. For simplicity, we assume that $Q = \{0, 1, \ldots, \#Q - 1\}$ and $q_0 = 0$. In what follows we assume that δ is a total function, i.e., the automaton is complete.

Let l be the length of the longest word(s) in the finite language L. A DFA A such that $L(A) \cap \Sigma^{\leq l} = L$ is called a *deterministic finite cover-automaton* (DFCA) of L. Let $A = (Q, \Sigma, \delta, 0, F)$ be a DFCA of a finite language L. We say that A is a *minimal* DFCA of L if for every DFCA $B = (Q', \Sigma, \delta', 0, F')$ of L we have $\#Q \leq \#Q'$.

Let $A = (Q, \Sigma, \delta, 0, F)$ be a DFA. Then

a) $q \in Q$ is said to be accessible if there exists $w \in \Sigma^*$ such that $\delta(0, w) = q$,

b) q is said to be useful (coaccessible) if there exists $w \in \Sigma^*$ such that $\delta(q, w) \in F$.

It is clear that for every DFA A there exists an automaton A' such that $L(A') = L(A)$ and all the states of A' are accessible and at most one of the states is not useful (the sink state). The DFA A' is called a *reduced* DFA.

In what follows we shall use only reduced DFA.

3 Similarity Sequences and Similarity Sets

In this section, we describe the L-similarity relation on Σ^*, which is a generalisation of the equivalence relation $\equiv_L$ ($x \equiv_L y$: $xz \in L$ iff $yz \in L$ for all $z \in \Sigma^*$). The notion of L-similarity was introduced in [7] and studied in [4] etc. In this paper, L-similarity is used to establish our algorithms.

Let Σ be an alphabet, $L \subseteq \Sigma^*$ a finite language, and l the length of the longest word(s) in L. Let $x, y \in \Sigma^*$. We define the following relations:

(1) $x \sim_L y$ if for all $z \in \Sigma^*$ such that $|xz| \leq l$ and $|yz| \leq l$, $xz \in L$ iff $yz \in L$;

(2) $x \not\sim_L y$ if $x \sim_L y$ does not hold.

The relation $\sim_L$ is called *similarity* relation with respect to L.

Note that the relation $\sim_L$ is reflexive, symmetric, but not transitive. For example, let $\Sigma = \{a, b\}$ and $L = \{aab, baa, aabb\}$. It is clear that $aab \sim_L aabb$ and $baa \sim_L aabb$, but $aab \not\sim_L baa$.

The following lemma is obvious:

Lemma 1 *Let $L \subseteq \Sigma^*$ be a finite language and $x, y, z \in \Sigma^*$, $|x| \leq |y| \leq |z|$. The following statements hold:*

1. *If $x \sim_L y$, $x \sim_L z$, then $y \sim_L z$.*
2. *If $x \sim_L y$, $y \sim_L z$, then $x \sim_L z$.*
3. *If $x \sim_L y$, $y \not\sim_L z$, then $x \not\sim_L z$.*

If $x \not\sim_L y$ and $y \sim_L z$, we cannot say anything about the similarity relation between x and z.

Example 1. Let $x, y, z \in \Sigma^*$, $|x| \leq |y| \leq |z|$. We may have

1) $x \not\sim_L y$, $y \sim_L z$ and $x \sim_L z$, or

2) $x \not\sim_L y$, $y \sim_L z$ and $x \not\sim_L z$.

Indeed, if $L = \{aa, aaa, bbb, bbbb, aaab\}$ we have 1) if we choose $x = aa$, $y = bbb$, $z = bbbb$, and 2) if we choose $x = aa$, $y = bba$, $z = abba$.

Definition 1. *Let $L \in \Sigma^*$ be a finite language.*

1. *A set $S \subseteq \Sigma^*$ is called an L-similarity set if $x \sim_L y$ for every pair $x, y \in S$.*
2. *A sequence of words $[x_1, \ldots, x_n]$ over Σ is called a dissimilar sequence of L if $x_i \not\sim_L x_j$ for each pair i, j, $1 \le i, j \le n$ and $i \neq j$.*
3. *A dissimilar sequence $[x_1, \ldots, x_n]$ is called a canonical dissimilar sequence of L if there exists a partition $\pi = \{S_1, \ldots, S_n\}$ of Σ^* such that for each i, $1 \le i \le n$, $x_i \in S_i$, and S_i is a L-similarity set.*
4. *A dissimilar sequence $[x_1, \ldots, x_n]$ of L is called a maximal dissimilar sequence of L if for any dissimilar sequence $[y_1, \ldots, y_m]$ of L, $m \le n$.*

Theorem 1. *A dissimilar sequence of L is a canonical dissimilar sequence of L if and only if it is a maximal dissimilar sequence of L.*

Proof. Let L be a finite language. Let $[x_1, \ldots, x_n]$ be a canonical dissimilar sequence of L and $\pi = \{S_1, \ldots, S_n\}$ the corresponding partition of Σ^* such that for each i, $1 \le i \le n$, S_i is an L-similarity set. Let $[y_1, \ldots, y_m]$ be an arbitrary dissimilar sequence of L. Assume that $m > n$. Then there are y_i and y_j, $i \neq j$, such that $y_i, y_j \in S_k$ for some k, $1 \le k \le n$. Since S_k is a L-similarity set, $y_i \sim_L y_j$. This is a contradiction. Then, the assumption that $m > n$ is false, and we conclude that $[x_1, \ldots, x_n]$ is a maximal dissimilar sequence.

Conversely, let $[x_1, \ldots, x_n]$ a maximal dissimilar sequence of L. Without loss of generality we can suppose that $|x_1| \le \ldots \le |x_n|$. For $i = 1, \ldots, n$, define

$$X_i = \{y \in \Sigma^* \mid y \sim_L x_i \text{ and } y \notin X_j \text{ for } j < i\}.$$

Note that for each $y \in \Sigma^*$, $y \sim_L x_i$ for at least one i, $1 \le i \le n$, since $[x_1, \ldots, x_n]$ is a *maximal* dissimilar sequence. Thus, $\pi = \{X_1, \ldots, X_n\}$ is a partition of Σ^*. The remaining task of the proof is to show that each X_i, $1 \le i \le n$, is a similarity set.

We assume the contrary, i.e., for some i, $1 \le i \le n$, there exist $y, z \in X_i$ such that $y \not\sim_L z$. We know that $x_i \sim_L y$ and $x_i \sim_L z$ by the definition of X_i. We have the following three cases: (1) $|x_i| < |y|, |z|$, (2) $|y| \le |x_i| \le |z|$ (or $|z| \le |x_i| \le |y|$), and (3) $|x_i| > |y|, |z|$. If (1) or (2), then $y \sim_L z$ by Lemma 1. This would contradict our assumption. If (3), then it is easy to prove that $y \not\sim x_j$ and $z \not\sim x_j$, for all $j \neq i$, using Lemma 1 and the definition of X_i. Then we can replace x_i by both y and z to obtain a longer dissimilar sequence $[x_1, \ldots, x_{i-1}, y, z, x_{i+1}, \ldots, x_n]$. This contradicts the fact that $[x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n]$ is a maximal dissimilar sequence of L. Hence, $y \sim z$ and X_i is a similarity set.

Corollary 1. *For each finite language L, there is a unique number $N(L)$ which is the number of elements in any canonical dissimilar sequence of L.*

Theorem 2. *Let S_1 and S_2 be two L-similarity sets and x_1 and x_2 the shortest words in S_1 and S_2, respectively. If $x_1 \sim_L x_2$ then $S_1 \cup S_2$ is a L-similarity set.*

Proof. It suffices to prove that for an arbitrary word $y_1 \in S_1$ and an arbitrary word $y_2 \in S_2$, $y_1 \sim_L y_2$ holds. Without loss of generality, we assume that $|x_1| \leq |x_2|$. We know that $|x_1| \leq |y_1|$ and $|x_2| \leq |y_2|$. Since $x_1 \sim_L x_2$ and $x_2 \sim_L y_2$, we have $x_1 \sim_L y_2$ (Lemma 1 (2)), and since $x_1 \sim_L y_1$ and $x_1 \sim_L y_2$, we have $y_1 \sim_L y_2$ (Lemma 1 (1)).

4 Similarity Relations on States

Let $A = (Q, \Sigma, \delta, 0, F)$ be a DFA and $L = L(A)$. Then it is clear that if $\delta(0, x) = \delta(0, y) = q$ for some $q \in Q$, then $x \equiv_L y$ and, thus, $x \sim_L y$. Therefore, we can also define equivalence as well as similarity relations on states.

Definition 2. *Let $A = (Q, \Sigma, \delta, 0, F)$ be a DFA. We define, for each state $q \in Q$,*

$$level(q) = min\{|w| \mid \delta(0, w) = q\},$$

i.e., $level(q)$ is the length of the shortest path from the initial state to q.

Definition 3. *Let $A = (Q, \Sigma, \delta, 0, F)$ be a DFA and $L = L(A)$. We say that $p \equiv_A q$ (state p is equivalent to q in A) if for every $w \in \Sigma^*$, $\delta(s, w) \in F$ iff $\delta(q, w) \in F$.*

Definition 4. *Let $A = (Q, \Sigma, \delta, 0, F)$ be a DFCA of a finite language L. Let $level(p) = i$ and $level(q) = j$, $m = \max\{i, j\}$. We say that $p \sim_A q$ (state p is L-similar to q in A) if for every $w \in \Sigma^{\leq l-m}$, $\delta(p, w) \in F$ iff $\delta(q, w) \in F$.*

If $A = (Q, \Sigma, \delta, 0, F)$ is a DFA, for each $q \in Q$, we denote $x_A(q) = \min\{w \mid \delta(0, w) = q\}$, where the minimum is taken according to the quasi-lexicographical order, and $L_A(q) = \{w \in \Sigma^* \mid \delta(q, w) \in F\}$. When the automaton A is understood, we write x_q instead of $x_A(q)$ and L_q instead $L_A(q)$.

Lemma 2 *Let $A = (Q, \Sigma, \delta, 0, F)$ be a DFCA of a finite language L. Let $x, y \in \Sigma^*$ such that $\delta(0, x) = p$ and $\delta(0, y) = q$. If $p \sim_A q$ then $x \sim_L y$.*

Proof. Let $level(p) = i$ and $level(q) = j$, $m = \max\{i, j\}$, and $p \sim_A q$. Choose an arbitrary $w \in \Sigma^*$ such that $|xw| \leq l$ and $|yw| \leq l$. Because $i \leq |x|$ and $j \leq |y|$ it follows that $|w| \leq l - m$. Since $p \sim_A q$ we have that $\delta(p, w) \in F$ iff $\delta(q, w) \in F$, i.e. $\delta(0, xw) \in F$ iff $\delta(0, yw) \in F$, which means that $xw \in L(A)$ iff $yw \in L(A)$. Hence $x \sim_L y$.

Lemma 3 *Let $A = (Q, \Sigma, \delta, 0, F)$ be DFCA of a finite language L. Let $level(p) = i$ and $level(q) = j$, $m = \max\{i, j\}$, and $x \in \Sigma^i$, $y \in \Sigma^j$ such that $\delta(0, x) = p$ and $\delta(0, y) = q$. If $x \sim_L y$ then $p \sim_A q$.*

Proof. Let $x \sim_L y$ and $w \in \Sigma^{\leq l-m}$. If $\delta(p,w) \in F$, then $\delta(0,xw) \in F$. Because $x \sim_L y$, it follows that $\delta(0,yw) \in F$, so $\delta(q,w) \in F$. Using the symmetry we get that $p \sim_A q$. ¿

Corollary 2. *Let $A = (Q,\Sigma,\delta,0,F)$ be a DFCA of a finite language L. Let $level(p) = i$ and $level(q) = j$, $m = \max\{i,j\}$, and $x_1 \in \Sigma^i$, $y_1 \in \Sigma^j$, $x_2, y_2 \in \Sigma^*$, such that $\delta(0,x_1) = \delta(0,x_2) = p$ and $\delta(0,y_1) = \delta(0,y_2) = q$. If $x_1 \sim_L y_1$ then $x_2 \sim_L y_2$.*

Example 2. If x_1 and y_1 are not minimal, i.e. $|x_1| > i$, but $p = \delta(0,x_1)$ or $|y_1| > j$, but $q = \delta(0,y_1)$, then the conclusion of Corollary 2 is not true.

Let $L = \{a,b,aa,aaa,bab\}$, so $l = 3$ (Figure 1).

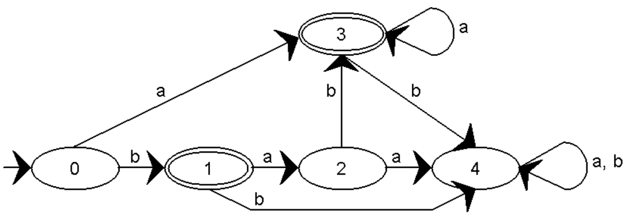

Fig. 1. A DFCA of L

we have that $b \sim_L bab$, but $b \not\sim_L a$.

Corollary 3. *Let $A = (Q,\Sigma,\delta,0,F)$ be a DFCA of a finite language L and $p,q \in Q$, $p \neq q$. Then $x_p \sim_L x_q$ iff $p \sim_A q$.*

If $p \sim_A q$, and $level(p) \leq level(q)$ and $q \in F$ then $p \in F$.

Lemma 4 *Let $A = (Q,\Sigma,\delta,0,F)$ be a DFCA of a finite language L. Let $s,p,q \in Q$ such that $level(s) = i$, $level(p) = j$, $level(q) = k$, $i \leq j \leq k$. The following statements are true:*

1. *If $s \sim_A p$, $s \sim_A q$, then $p \sim_A q$.*
2. *If $s \sim_A p$, $p \sim_A q$, then $s \sim_A q$.*
3. *If $s \sim_A p$, $p \not\sim_A q$, then $s \not\sim_A q$.*

Proof. We apply Lemma 1 and Corollary 3.

Lemma 5 *Let $A = (Q,\Sigma,\delta,0,F)$ be a DFCA of a finite language L. Let $level(p) = i$, $level(q) = j$, and $m = \max\{i,j\}$. If $p \sim_A q$ then $L_p \cap \Sigma^{\leq l-m} = L_q \cap \Sigma^{\leq l-m}$ and $L_p \cup L_q$ is a L- similarity set.*

The proof is left to the reader.The next lemma is obvious.

Lemma 6 *Let $A = (Q, \Sigma, \delta, 0, F)$ be a DFCA of a finite language L. Let $i = level(p)$ and $j = level(q)$, $i \leq j$. Let $p \sim_L q$.Let $w = w_1 \dots w_n \in \Sigma^{\leq l}$ and $p_i = \delta(0, w_1 \dots w_i)$, $1 \leq i \leq n$. Then $w \in L$ iff $x_k w_{k+1} \dots w_n \in L$ for $1 \leq k \leq n$.*

Lemma 7 *Let $A = (Q, \Sigma, \delta, 0, F)$ be a DFCA of a finite language L. If $p \sim_A q$ for some $p, q \in Q$, $i = level(p)$, $j = level(q)$ and $i \leq j$, $p \neq q$, $q \neq 0$. then we can construct a DFCA $A' = (Q', \Sigma, \delta', 0, F')$ of L such that $Q' = Q - \{q\}$, $F' = F - \{q\}$, and*

$$\delta'(s,a) = \begin{cases} \delta(s,a) & \textit{if } \delta(s,a) \neq q, \\ p & \delta(s,a) = q \end{cases}$$

for each $s \in Q'$ and $a \in \Sigma$. Thus, A is not a minimal DFCA of L.

Proof. It suffices to prove that A' is a DFCA of L. Let l be the length of the longest word(s) in L and assume that $level(p) = i$ and $level(q) = j$, $i \leq j$. Consider a word $w \in \Sigma^{\leq L}$. We now prove that $w \in L$ iff $\delta'(0, w) \in F'$.

If there is no prefix w_1 of w such that $\delta(0, w_1) = q$, then clearly $\delta'(0, w) \in F'$ iff $\delta(0, w) \in F$. Otherwise, let $w = w_1 w_2$ where w_1 is the shortest prefix of w such that $\delta(0, w_1) = q$. In the remaining, it suffices to prove that $\delta'(p, w_2) \in F'$ iff $\delta(q, w_2) \in F$. We prove this by induction on the length of w_2. First consider the case $|w_2| = 0$, i.e., $w_2 = \lambda$. Since $p \sim_A q$, $p \in F$ iff $q \in F$. Then $p \in F'$ iff $q \in F$ by the construction of A'. Thus, $\delta'(p, w_2) \in F'$ iff $\delta(q, w_2) \in F$. Suppose that the statement holds for $|w_2| < l'$ for $l' \leq l - |w_1|$. (Note that $l - |w_1| \leq l - j$.) Consider the case that $|w_2| = l'$. If there does not exist $u \in \Sigma^+$ such that $u \preceq_p w_2$ and $\delta(p, u) = q$, then $\delta(p, w_2) \in F - \{q\}$ iff $\delta(q, w_2) \in F - \{q\}$, i.e., $\delta'(p, w_2) \in F'$ iff $\delta(q, w_2) \in F$. Otherwise, let $w_2 = uv$ and u be the shortest nonempty prefix of w_2 such that $\delta(p, u) = q$. Then $|v| < l'$ (and $\delta'(p, u) = p$). By induction hypothesis, $\delta'(p, v) \in F'$ iff $\delta(q, v) \in F$. Therefore, $\delta'(p, uv) \in F'$ iff $\delta(q, uv) \in F$.

Lemma 8 *Let A be a DFCA of L and $L' = L(A)$. Then $x \equiv_{L'} y$ implies $x \sim_L y$.*

Proof. It is clear that if $x \equiv_L y$ then $x \sim_L y$. Let l be the length of the longest word(s) in L. Let $x \equiv_{L'} y$. So, for each $z \in \Sigma^*, xz \in L'$ iff $yz \in L'$. We now consider all words $z \in \Sigma^*$, such that $\mid xz \mid \leq l$ and $\mid yz \mid \leq l$. Since $L = L' \cap \Sigma^{\leq l}$ and $xz \in L'$ iff $yz \in L'$, we have $xz \in L$ iff $yz \in L$. Therefore, $x \sim_L y$ by the definition of $\sim_L$.

Corollary 4. *Let $A = (Q, \Sigma, \delta, 0, F)$ be a DFCA of a finite language L, $L' = L(A)$. Then $p \equiv_A q$ implies $p \sim_L q$.*

Corollary 5. *A minimal DFCA of L is a minimal DFA.*

Proof. Let $A = (Q, \Sigma, \delta, 0, F)$ be a minimal DFCA of a finite language L. Suppose that A is not minimal as a DFA for $L(A)$, then there exists $p, q \in Q$ such that $p \equiv_{L'} q$, then $p \sim_L q$. By Lemma 7 it follows that A is not a minimal DFCA, contradiction.

Remark 1. Let A be a DFCA of L and A a minimal DFA. Then A may not be a minimal DFCA of L.

Example 3. We take the DFA's of Figure 2.

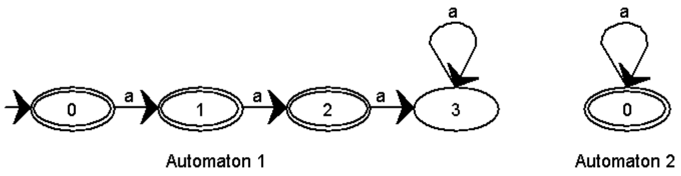

Fig. 2. Example

The DFA in Automaton 1 is a minimal DFA and a DFCA of $L = \{\lambda, a, aa\}$ but not a minimal DFCA of L, since the DFA in Automaton 2 is a minimal DFCA of L:

Theorem 3. *Any minimal DFCA of L has exactly $N(L)$ states.*

Proof. Let $A = (Q, \Sigma, \delta, 0, F)$ be DFCA of a finite language L, and $\#Q = n$.

Suppose that $n > N(L)$. Then there exist $p, q \in Q$, $p \neq q$, such that $x_p \sim_L x_q$ (because of the definition of $N(L)$). Then $p \sim_A q$ by Lemma 3. Thus, A is not minimal. A contradiction.

Suppose that $N(L) > n$. Let $[y_1, \ldots, y_{N(L)}]$ be a canonical dissimilar sequence of L. Then there exist i, j, $1 \leq i, j \leq N(L)$ and $i \neq j$, such that $\delta(0, y_i) = \delta(0, y_j) = q$ for some $q \in Q$. Then $y_i \sim_L y_j$. Again a contradiction.

Therefore, we have $n = N(L)$.

5 The Construction of Minimal DFCA

The first part of this section describe an algorithm that determines the similarity relations between states. The second part is to construct a minimal DFCA assuming that the similarity relation between states is known.

An ordered DFA is a DFA where $\delta(i, a) = j$ implies that $i \leq j$, for all states i, j and letters a.

5.1 Determining Similarity Relation between States

The aim is to present an algorithm which determines the similarity relations between states.

Let $A = (\Sigma, Q, 0, \delta, F)$ a DFCA of a finite language L. For each $s \in Q$ let $\gamma_s = \min\{w \mid \delta(s, w) \in F\}$, where minimum is taken according to the quasi-lexicographical order. Define $D_i = \{s \in Q \mid |\gamma_s| = i\}$, for each $i = 0, 1, \ldots$.

Lemma 9 *Let $A = (\Sigma, Q, 0, \delta, F)$ a DFCA of a finite language L, and $s \in D_i$, $p \in D_j$. If $i \neq j$ then $s \not\sim p$.*

Proof. We can assume that $i < j$. Then obviously $\delta(s, \gamma_s) \in F$ and $\delta(p, \gamma_s) \notin F$. Since $l \geq |x_s| + \gamma_s|$, $l \geq |x_p| + |\gamma_p|$, and $i < j$, it follows that $|\gamma_s| < |\gamma_p|$. So, we have that $|\gamma_s| \leq \min(l - |x_s|, l - |x_p|)$. Hence, $s \not\sim p$.

Lemma 10 *Let $A = (Q, \Sigma, 0, \delta, F)$ be a reduced ordered DFA accepting L, $p, q \in Q - \{\#Q - 1\}$, where $\#Q - 1$ is the sink state, and either $p, q \in F$ or $p, q \notin F$. If for all $a \in \Sigma$, $\delta(p, a) \sim_A \delta(q, a)$, then $p \sim_A q$.*

Proof. Let $a \in \Sigma$ and $\delta(p, a) = r$ and $\delta(q, a) = s$. If $r \sim_A s$ then for all $|w|$, $|w| < l - \max\{x_A(s), x_A(r)\}$, $x_A(r)w \in L$ iff $x_A(s)w \in L$. Using Lemma 2 we also have: $x_A(q)aw \in L$ iff $x_A(s)w \in L$ for all $w \in \Sigma^*$, $|w| \leq l - |x_A(s)|$ and $x_A(p)aw \in L$ iff $x_A(r)w \in L$ for all $w \in \Sigma^*$, $|w| \leq l - |x_A(r)|$.

Hence $x_A(p)aw \in L$ iff $x_A(q)aw \in L$, for all $w \in \Sigma^*$, $|w| \leq l - \max\{|x_A(r)|, |x_A(s)|\}$. Because $|x_A(r)| \leq |x_A(q)a| = |x_A(q)| + 1$ and $|x_A(s)| \leq |x_A(p)a| = |x_A(p)| + 1$, we get $x_A(p)aw \in L$ iff $x_A(q)aw \in L$, for all $w \in \Sigma^*$, $|w| \leq l - \max\{|x_A(p)|, |x_A(q)|\} - 1$.

Since $a \in \Sigma$ is chosen arbitrary, we conclude that $x_A(p)w \in L$ iff $x_A(q)w \in L$, for all $w \in \Sigma^*$, $|w| \leq l - \max\{|x_A(p)|, |x_A(q)|\}$, i.e. $x_A(p) \sim_A x_A(q)$. Therefore, by using Lemma 3, we get that $p \sim_A q$.

Lemma 11 *Let $A = (Q, \Sigma, 0, \delta, F)$ be a reduced ordered DFA accepting L such that $\delta(0, w) = s$ implies $|w| = |x_s|$ for all $s \in Q$. Let $p, q \in Q - \{\#Q - 1\}$, where $\#Q - 1$ is the sink state. If there exists $a \in \Sigma$ such that $\delta(p, a) \not\sim_A \delta(q, a)$, then $p \not\sim_A q$.*

Proof. Suppose that $p \sim_A q$. then for all $aw \in \Sigma^{l-m}$, $\delta(p, aw) \in F$ iff $\delta(q, aw) \in F$, where $m = \max\{level(p), level(q)\}$. So $\delta(\delta(p, a), w) \in F$ iff $\delta(\delta(q, a), w) \in F$ for all $w \in \Sigma^{l-m-1}$. Since $|x_{\delta(p,a)}| = |x_p| + 1$ and $|x_{\delta(q,a)}| = |x_q| + 1$ it follows by definition that $\delta(p, a) \sim_A \delta(q, a)$. This is a contradiction.

Our algorithm for determining the similarity relation between the states of a DFA (DFCA) of a finite language is based on Lemmas 10 and 11. However, most of DFA (DFCA) do not satisfy the condition of Lemma 11. So, we shall first transform the given DFA (DFCA) into one that does.

Let $A = (Q_A, \Sigma, 0, \delta_A, F_A)$ be a DFCA of L. We construct the minimal DFA for the language $\Sigma^{\leq l}$, $B = (Q_B, \Sigma, 0, \delta_B, F_B)$ ($Q_B = \{0, \ldots, l, l + 1\}$,

$\delta_B(i,a) = i+1$, for all i, $0 \leq i \leq l$, $\delta_B(l+1,a) = l+1$, for all $a \in \Sigma$, $F_B = \{0,\ldots,l\}$). The DFA B will have exact $l+2$ states.

Now we use the standard Cartesian product construction (see, e.g., [3], for details) for the DFA $C = (Q_C, \Sigma, q_0, \delta_C, F_C)$ such that $L(C) = L(A) \cap L(B)$, and we eliminate all inaccessible states. Obviously, $L(C) = L$ and C satisfies the condition of Lemma 11.

The next lemma is easy to prove and left for the reader.

Lemma 12 *For the DFA C constructed above we have $(p,q) \sim_C (p,r)$.*

Lemma 13 *For the DFA C constructed above, if $\delta_C((0,0),w) = (p,q)$, then $|w| = q$.*

Proof. We have $\delta_C((0,0),w) = (p,q)$, so $\delta_B(0,w) = q$ so $|w| = q$.

Now we are able to present an algorithm, which determines the similarity relation between the states of C. Note that Q_C is ordered by that $(p_A,p_B) < (q_A,q_B)$ if $p_A < q_A$ or $p_A = q_A$ and $p_B < q_B$. Attaching to each state of C is a list of similar states. For $\alpha, \beta \in Q_C$, if $\alpha \sim_C \beta$ and $\alpha < \beta$, then β is stored on the list of similar states for α.

We assume that $Q_A = \{0,1,\ldots,n\}$ and n is the sink state of A.

1. Generate the DFA B for the language $\Sigma^{\leq l}$.
2. Compute the DFA C such that $L(C) = L(A) \cap L(B)$ using the standard Cartesian product algorithm (see [3] for details).
3. Compute D_i of C, $0 \leq i \leq l$.
4. Initialize the similarity relation by specifying:
   - For all $(n,p),(n,q) \in Q_C$, $(n,p) \sim_C (n,q)$.
   - For all $(n,l+1-i) \in Q_C$, $(n,l+1-i) \sim_C \alpha$ for all $\alpha \in D_j$, $j = i,\ldots,l$, $0 \leq i \leq l$.
5. For each D_i, $0 \leq i \leq l$, create a list $List_i$, which is initialized to $\emptyset$.
6. For each $\alpha \in Q_C - \{(n,q) \mid q \in Q_B\}$, following the reversed order of Q_C, do the following: Assuming $\alpha \in D_i$.
   - For each $\beta \in List_i$, if $\delta_C(\alpha,a) \sim_C \delta_C(\beta,a)$ for all $a \in \Sigma$, then $\alpha \sim_C \beta$.
   - Put α on the list $List_i$.

Remark 2. The above algorithm has complexity $O((n \times l)^2)$, where n is the number of states of the initial DFA (DFCA) and l is the maximum accepted length for the finite language L.

5.2 The Construction of a Minimal DFCA

As input we have the above DFA C and, with each $\alpha \in Q_C$, a set $S_\alpha = \{\beta \in Q_C \mid \alpha \sim_C \beta$ and $\alpha < \beta\}$. The output is $D = (Q_D, \Sigma, \delta_D, q_0, F_D)$, a minimal DFCA for L.

```
We define the following:
$i = 0$, $q_i = 0$, $T = Q_C - S_i$, $(x_0 = \lambda)$;
while $(T \neq \emptyset)$ do the following:
    $i = i + 1$;
    $q_i = \min\{s \in T\}$,
    $T = T - S_{q_i}$, $(x_i = \min\{w \mid \delta_C(0, w) \in S_i\})$;
$m = i$;
```

Then $Q_D = \{q_0, \ldots, q_m\}$; $q_0 = 0$; $\delta_D(i, a) = j$, iff $k = \min S_i$ and $\delta_C(k, a) \in S_j$; $F_D = \{i \mid S_i \cap F_C \neq \emptyset\}$.

Note that the constructions of x_i above are useful for the proofs in the following only, where the *min* (minimum) operator for x_i is taken according to the lexicographical order. Let $X_i = \{(i, s) \mid (i, s) \in Q_C\}$ and $a_i = \#X_i$, $0 \leq i \leq l+1$.

Step 1. For all $1 \leq i \leq l+1$ do $b_i = a_i$, for all $(i, j) \in Q_C$ do $new((i, j)) = -1$. Set $m = 0$, $r = 0$ and $s = 0$.
Step 2. Put $S_m = \{(p, q) \in Q_C \mid (r, s) \sim_A (p, q)\}$.
Step 3. For all $(p, q) \in S_m$, perform $new((p, q)) = m$ and $b_p = b_p - 1$.
Step 4. Put $m = m + 1$.
Step 5. While $b_r = 0$ and $r \leq l+1$ do $r = r + 1$. If $r > l + 1$ then go to Step 7, else go to Step 6.
Step 6. Take the state $(r, s) \in A_r$ such that $new(r, s) \neq -1$, and s is the minimal with this property. Go to Step 2.
Step 7. $Q_D = \{0, \ldots, m - 1\}$, $F = \{i \mid new((p, q)) = i, (p, q) \in F_C\}$. For all $q \in Q_D$ and $a \in \Sigma$ set $\delta_D(q, a) = -1$.
Step 8. For all $p = 0, \ldots, l + 1$, $q = 0, \ldots, n$, $(p, q) \in Q_C$ and $a \in \Sigma$if $\delta_D(new(p, q), a) = -1$ define

$$\delta_D(new(p, q), a) = new(\delta_C((p, q), a)).$$

According to the algorithm we have a total ordering of the states Q_C: $(p, q) \leq (r, s)$ if $(p, q) = (r, s)$ or $p < r$ or $p = r$ and $q < s$. Hence $\delta_D(i, a) = j$ iff $\delta_D(0, x_i a) = j$. Also, using the construction (i.e. the total order on Q_C) it follows that $0 = |x_0| \leq |x_1| \leq \ldots \leq |x_{m-1}|$.

Lemma 14 *The sequence $[x_0, x_1, \ldots, x_{m-1}]$, constructed above is a cannonical L- dissimilar sequence.*

Proof. We construct the sets $X_i = \{w \in \Sigma^* \mid \delta(0, w) \in S_i\}$. Obviously $X_i \neq \emptyset$. From Lemma 2 it follows that X_i is a L- similarity set for all $0 \leq i \leq m - 1$.

Let $w \in \Sigma^*$. Because $(S_i)_{1 \leq i \leq m-1}$ is a partition of Q, $w \in X_i$ for some $0 \leq i \leq n - 1$, so $(X_i)_{0 \leq i \leq n-1}$ is a partition of Σ^* and therefore a cannonical L-dissimilar sequence.

Corollary 6. *The automaton D constructed above is a minimal DFCA for L.*

Proof. Since the number of states is equal to the number of elements of a cannonical L-dissimilar sequence, we only have to prove that D is a cover automaton for L. Let $w \in \Sigma^{\leq l}$. We have that $\delta_D(0, w) \in F_D$ iff $\delta_C((0, 0), w) \in S_f$ and $S_f \cap F_C \neq \emptyset$, i.e. $x_f \sim_C w$. Since $|w| \leq l$, $x_f \in L$ iff $w \in L$ (because C is a DFCA for L).

6 Boolean Operations

We shall use similar constructions as in [3] for constructing DFCA of languages which are a result of boolean operations between finite languages. The modifications are suggested by the previous algorithm. We first construct the DFCA which satisfies hypothesis of Lemma 11 and afterwards we can minimise it using the general algorithm. Since the minimisation will follow in a natural way we shall present only the construction of the necessarily DFCA.

Let $A_i=(Q_i,\Sigma,0,\delta_i,F_i)$, two DFCA of the finite languages L_i, $l_i=\max\{|w| \mid w\in L_i\}$, $i=1,2$.

6.1 Intersection

We construct the following DFA:

$A=(Q_1\times Q_2\times\{0,\dots,l\},\Sigma,\delta,(0,0,0),F)$, where

$l=\min\{l_1,l_2\}$, $\delta((s,p,q),a)=(\delta_1(s,a),\delta_2(p,a),q+1)$, for $s\in Q_1$, $p\in Q_2$, $q\leq l$, and $\delta((s,p,l+1),a)=(\delta_1(s,a),\delta_2(p,a),l+1)$ and $F=\{(s,p,q)\mid s\in F_1,p\in F_2,q\leq l\}$.

Theorem 4. *The automaton A constructed above is a DFA for $L=L(A_1)\cap L(A_2)$.*

Proof. We have the following relations: $w\in L_1\cap L_2$ iff $|w|\leq l$ and $w\in L_1$ and $w\in L_2$ iff $|w|\leq l$ and $w\in L(A_1)$ and $w\in L(A_2)$. The rest of the proof is obvious.

6.2 Union

We construct the following DFA:

$A=(Q_1\times Q_2\times\{0,\dots,l\},\Sigma,\delta,(0,0,0),F)$, where

$l=\max\{l_1,l_2\}$, $m=\min\{l_1,l_2\}$, $\delta((s,p,q),a)=(\delta_1(s,a),\delta_2(p,a),q+1)$, for $s\in Q_1$, $p\in Q_2$, $q\leq l$, and $\delta((s,p,l+1),a)=(\delta_1(s,a),\delta_2(p,a),l+1)$ and $F=\{(s,p,q)\mid s\in F_1 \text{ or } p\in F_2,\ q\leq m\}\cup\{(s,p,q)\mid s\in F_r \text{ and } m<q\leq l\}$, where r is such that $l_r=l$.

Theorem 5. *The automaton A constructed above is a DFA for $L=L(A_1)\cup L(A_2)$.*

Proof. We have the following relations: $w\in L_1\cup L_2$ iff $|w|\leq m$ and $w\in L_1$ or $w\in L_2$, or $m<|w|\leq l$ and $w\in L_r$ iff $|w|\leq m$ and $w\in L(A_1)$ or $w\in L(A_2)$, or $m<|w|\leq l$ and $w\in L(A_r)$. The rest of the proof is obvious.

6.3 Symmetric Difference

We construct the following DFA:

$A=(Q_1\times Q_2\times\{0,\dots,l\},\Sigma,\delta,(0,0,0),F)$, where

$l=\max\{l_1,l_2\}$, $m=\min\{l_1,l_2\}$, $\delta((s,p,q),a)=(\delta_1(s,a),\delta_2(p,a),q+1)$, for $s\in Q_1$, $p\in Q_2$, $q\leq l$, and $\delta((s,p,l+1),a)=(\delta_1(s,a),\delta_2(p,a),l+1)$ and $F=\{(s,p,q)\mid s\in F_1 \text{ or exclusive } p\in F_2,\ q\leq m\}\cup\{(s,p,q)\mid s\in F_r \text{ and } m<q\leq l\}$, where r is such that $l_r=l$.

Theorem 6. *The automaton A constructed above is a DFA for $L = L(A_1)\varDelta L(A_2)$.*

Proof. We have the following relations: $w \in L_1 \varDelta L_2$ iff $|w| \leq m$ and $w \in L_1$ or exclusive $w \in L_2$, or $m < |w| \leq l$ and $w \in L_r$ iff $|w| \leq m$ and $w \in L(A_1)$ or exclusive $w \in L(A_2)$, or $m < |w| \leq l$ and $w \in L(A_r)$. The rest of the proof is obvious.

6.4 Difference

We construct the following DFA:

$A = (Q_1 \times Q_2 \times \{0, \ldots, l\}, \Sigma, \delta, (0, 0, 0), F)$, where

$l = \max\{l_1, l_2\}$, $m = \min\{l_1, l_2\}$ and $\delta((s, p, q), a) = (\delta_1(s, a), \delta_2(p, a), q + 1)$, for $s \in Q_1$, $p \in Q_2$, $q \leq l$, and $\delta((s, p, l + 1), a) = (\delta_1(s, a), \delta_2(p, a), l + 1)$. If $l_1 < l_2$ then $F = \{(s, p, q) \mid s \in F_1 \text{ and } p \notin F_2,\ q \leq m\}$ and $F = \{(s, p, q) \mid s \in F_1 \text{ and } p \notin F_2,\ q \leq m\} \cup \{(s, p, q) \mid s \in F_1 \text{ and } m < q \leq l\}$, if $l_1 \geq l_2$.

Theorem 7. *The automaton A constructed above is a DFA for $L = L(A_1) - L(A_2)$.*

Proof. We have the following relations: $w \in L_1 - L_2$ iff $|w| \leq m$ and $w \in L_1$ and $w \notin L_2$, or $m < |w| \leq l$ and $w \in L_1$ iff $|w| \leq m$ and $w \in L(A_1)$ and $w \notin L(A_2)$, or $m < |w| \leq l$ and $w \in L(A_1)$. The rest of the proof is obvious.

Open Problems 1) Try to find a better algorithm for minimisation 2) or prove that any minimisation algorithm has complexity $\Omega(n^2)$. 3) Find a better algorithm for determining similar states 4) in any DFCA of L. 3) Find better algorithms for boolean operations on DFCA.

References

1. J.L. Balcàzar, J. Diaz, and J. Gabarrò, Uniform characterisations of non-uniform complexity measures, *Information and Control* 67 (1985) 53-89.
2. Y. Breitbart, On automaton and “zone” complexity of the predicate “tobe a kth power of an integer”,*Dokl. Akad. Nauk SSSR***196** (1971), 16-19[Russian]; Engl. transl.,Soviet Math. Dokl. **12** (1971), 10-14.
3. Cezar Câmpeanu. Regular languages and programming languages, *Revue Roumaine de Linguistique - CLTA*, 23 (1986), 7-10.
4. C.Dwork and L.Stockmeyer, A time complexity gap for two-way probabilistic finite-state automata, *SIAM Journal on Computing* 19 (1990) 1011-1023.
5. J. Hartmanis, H.Shank, Two memory bounds for the recognition of primes by automata, *Math. Systems Theory* **3** (1969), 125-129.
6. J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison Wesley (1979), Reading, Mass.
7. J. Kaneps, R. Freivalds, Minimal Nontrivial Space Space Complexity of Probabilistic One-Way Turing Machines, in Proceedings of Mathematical Foundations of Computer Science, Banská Bystryca, Czechoslovakia, August 1990, *Lecture Notes in Computer Science*, vol 452, pp. 355-361, Springer-Verlag, New York/Berlin, 1990.

8. J. Kaneps, R. Freivalds, Running time to recognise non-regular languages by 2-way probabilistic automata, in ICALP'91, *Lecture Notes in Computer Science*, vol 510, pp. 174-185, Springer-Verlag, New York/Berlin, 1991.
9. J. Paredaens, R. Vyncke, A class of measures on formal languages,*Acta Informatica*, **9** (1977), 73-86.
10. Jeffrey Shallit, Yuri Breitbart, Automaticity I: Properties of a Measure of Descriptional Complexity, *Journal of Computer and System Sciences*, **53**, 10-25 (1996).
11. A. Salomaa, *Theory of Automata*, Pergamon Press (1969), Oxford.
12. K. Salomaa, S. Yu, Q. Zhuang, The state complexities of some basic operations on regular languages, *Theoretical Computer Science* 125 (1994) 315-328.
13. B.A. Trakhtenbrot, Ya. M. Barzdin, Finite Automata: Behaviour and Synthesis, *Fundamental Studies in Computer Science*, Vol.1, North-Holland, Amsterdam, 1973.
14. S. Yu, Q. Zhung, On the State Complexity of Intersection of Regular Languages, *ACM SIGACT News*, vol. 22, no. 3, (1991) 52-54.
15. S. Yu, Regular Languages, Handbook of Formal Languages, Springer Verlag, 1995.