NFA to DFA Transformation for Finite Languages*

Kai Salomaa and Sheng Yu

Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 5B7

Abstract. We consider the number of states of a DFA that is equivalent to an n-state NFA accepting a finite language. We first give a detailed proof for the case where the finite languages are over a two-letter alphabet. It shows that $O(2^{n/2})$ is the (worst-case) optimal upper-bound on the number of states of a DFA that is equivalent to an n-state NFA accepting a finite language. The main result of this paper is a generalization of the above result. We show that, for any n-state NFA accepting a finite language over an arbitrary k-letter alphabet, $n, k > 1$, there is an equivalent DFA of $O(k^{n/(\log_2 k + 1)})$ states, and show that this bound is optimal in the worst case.

1 Introduction

It is well-known that for each positive integer $n$, there exists a regular language $L$ such that $L$ is accepted by an $n$-state NFA and any complete DFA accepting $L$ requires at least $2^n$ states [3]. However, the same statement is not true if $L$ is required to be finite. In [2], Mandl showed that for each n-state NFA accepting a finite language over a two-letter alphabet, there exists an equivalent DFA which has $O(2^{n/3})$ states; more specifically, no more than $2^{n/3 + 1} - 1$ states if $n$ is even and $3 \cdot 2^{n/3} - 1$ states if $n$ is odd. In [2], it was also shown that these bounds are optimal in the worst case. However, there have been no corresponding results concerning finite languages over an arbitrary $k$-letter alphabet, $k \geq 2$. Also, the proofs in [2] for the two-letter alphabet case are rather sketchy.

In this paper, we first give detailed proofs for the two-letter alphabet case. Then, as the main result of this paper, we give the optimal upper-bounds for the general cases of the problem, i.e., for the cases where finite languages are over an arbitrary $k$-letter alphabet, $k \geq 2$. Specifically, we show that for any n-state NFA accepting a finite languages over a $k$-letter alphabet, $k \geq 2$, we can construct an equivalent DFA of $O(k^{n/(\log_2 k + 1)})$ states; and we show that for each $k$-letter alphabet, $k \geq 2$, and for each $n \geq 2$, there exists a finite language $L$ accepted by an $n$-state NFA such that the number of states of any DFA accepting $L$ is

* This research is supported by the Natural Sciences and Engineering Research Council of Canada grants OGP0041630.
\( \Omega(k^{n/(\log_2 k+1)}) \). One may observe that this bound, as a function of \( k \), approaches to \( 2^n \) when \( k \) becomes larger.

This paper is organized as follows. In the next section, we introduce the basic notations and definitions and also review some necessary background. In Section 3, we prove preliminary results that are necessary to the proofs in the next two sections. In Section 4, we give detailed proofs for the optimal upper-bounds for finite languages over a two-letter alphabet. In Section 5, we prove the general result, i.e., that for an arbitrary \( k \)-letter alphabet \( \Sigma, k > 1 \), the optimal upper-bound on the number of states of a DFA that is equivalent to a \( n \)-state NFA accepting a finite language over \( \Sigma \) is \( O(k^{n/(\log_2 k+1)}) \).

2 Basic notations and definitions

A deterministic finite automaton (DFA) \( A \) is a quintuple \((Q, \Sigma, \delta, s, F)\), where

- \( Q \) is the finite set of states;
- \( \Sigma \) is the input alphabet;
- \( \delta : Q \times \Sigma \rightarrow Q \) is the state transition function;
- \( s \in Q \) is the starting state; and
- \( F \subseteq Q \) is the set of final states.

Note that the transition function \( \delta \) is not necessarily a total function, i.e., it may not be defined for every pair of state and input symbol in \( Q \times \Sigma \). If \( \delta \) is a total function, we call \( A \) a complete DFA.

For convenience, we define \( \delta^+ : Q \times \Sigma^+ \rightarrow Q \) recursively by

1. \( \delta^+(q, a) = \delta(q, a) \)
2. \( \delta^+(q, xa) = \delta(\delta^+(q, x), a) \)

for \( q \in Q, a \in \Sigma, \) and \( x \in \Sigma^+ \).

A nondeterministic finite automaton \( A \) is a quintuple \((Q, \Sigma, \delta, s, F)\) where \( Q, \Sigma, s, \) and \( F \) are defined exactly the same way as for a DFA, and \( \delta : Q \times \Sigma \rightarrow 2^Q \) is the transition function, where \( 2^Q \) denotes the power set of \( Q \).

Similarly, we define \( \delta^+ : Q \times \Sigma^+ \rightarrow 2^Q \) for an NFA \( A \) by

1. \( \delta^+(q, a) = \delta(q, a) \)
2. \( \delta^+(q, xa) = \cup_{p \in \delta^+(q, x)} \delta(p, a) \).

A DFA or an NFA such that every state is reachable from the starting state and reaches a final state is called a reduced DFA or NFA, respectively.

The language accepted by a finite automaton \( A \), either an NFA or a DFA, is denoted \( L(A) \). We say that two finite automata \( A \) and \( B \) are equivalent if they accept exactly the same language, i.e., \( L(A) = L(B) \).

Note that according to the above NFA definition, an NFA does not have any \( \lambda \)-transitions. We call a nondeterministic finite automaton that has \( \lambda \)-transitions a \( \lambda \)-NFA. It has been shown in [9] that each \( \lambda \)-NFA is equivalent to an NFA with the same number of states. Therefore, the subsequent results on the number of states of NFA apply to \( \lambda \)-NFA as well.

The standard algorithm that transforms an NFA to an equivalent DFA is called the subset construction algorithm. Let \( A = (Q, \Sigma, \delta, s, F) \) be an arbitrary