Information and Computation 244 (2015) 1–22

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Incomplete operational transition complexity of regular languages ☆

Eva Maia *, Nelma Moreira, Rogério Reis

CMUP & DCC, Faculdade de Ciências da Universidade do Porto, Rua do Campo Alegre, 4169-007 Porto, Portugal

ARTICLE INFO

Article history:
Received 8 July 2014
Received in revised form 31 March 2015
Available online 20 August 2015

Keywords:
Descriptional complexity
Automata theory
Regular languages
Transition complexity
State complexity

ABSTRACT

The state complexity of basic operations on regular languages considering complete deterministic finite automata (DFA) has been extensively studied in the literature. But, if incomplete DFAs are considered, transition complexity is also a significant measure. In this paper we study the incomplete (deterministic) state and transition complexity of some operations for regular and finite languages. For regular languages we give a new tight upper bound for the transition complexity of the union, which refutes the conjecture presented by Y. Gao et al. For finite languages, we correct the published state complexity of concatenation for complete DFAs and provide a tight upper bound for the case when the *right* operand is larger than the *left* one. We also present some experimental results to test the behavior of those operations on the average case, and we conjecture that for many operations and in practical applications the worst-case complexity is seldom reached.

1. Introduction

In the last two decades the descriptional complexity of regular languages has been extensively investigated. For deterministic finite automata (DFA), the complexity measure usually studied is the state complexity, i.e. the number of states of the complete minimal DFA, [3,23,24,11,4,25], while for nondeterministic finite automata (NFA) both state and transition complexities were considered [10,7,19,12,11]. For NFAs transition complexity is generally considered a more interesting measure. Considering complete DFAs, where the transition function is total, the transition complexity is, obviously, the product of the alphabet size by the state complexity. But in many applications where large alphabets need to be considered or, in general, when very sparse transition functions take place, partial transition functions are very convenient. Examples include lexical analyzers, discrete event systems, or any application that uses dictionaries where compact automaton representations are essential, for instance for manipulation on large Unicode alphabets [2,17,6,18]. And, in many cases, only finite languages are needed. Thus, it makes sense to investigate the transition complexity of not necessarily complete DFAs.

In this paper we study the incomplete operational transition complexity of several operations on regular and finite languages. To be comprehensive we also analyze the state complexity of resulting languages. In general, transition complexity bounds depend not only on the complexities of the operands but also on other refined measures, as the number of unde-

☆ This work was partially funded by the European Regional Development Fund through the programme COMPETE and by the Portuguese Government through the FCT under project PEst-C/MAT/UI0144/2013 and project FCOMP-01-0124-FEDER-020486. Eva Maia is funded by FCT grant SFRH/BD/78392/2011.

* Corresponding author.
E-mail addresses: emaia@dcc.fc.up.pt (E. Maia), nam@dcc.fc.up.pt (N. Moreira), rvr@dcc.fc.up.pt (R. Reis).

http://dx.doi.org/10.1016/j.ic.2015.08.004
0890-5401/

Table 1
Incomplete transition complexity for regular and finite languages, where m and n are the (incomplete) state complexities of the operands, $f_1(m,n) = (m-1)(n-1)+1$ and $f_2(m,n) = (m-2)(n-2)+1$. The column $|\Sigma|$ indicates the minimal alphabet size for which the upper bound is reached.

Operation	Regular	$\|\Sigma\|$	Finite	$\|\Sigma\|$
$L_1 \cup L_2$	$2n(m+1)$	2	$3(mn-n-m)+2$	$f_1(m,n)$
$L_1 \cap L_2$	nm	1	$(m-2)(n-2)(2+\sum_{i=1}^{\min(m,n)-3}(m-2-i)(n-2-i))+2$	$f_2(m,n)$
L^C	$m+2$	1	$m+1$	1
L_1L_2	$2^{n-1}(6m+3)-5$, if $m,n \geq 2$	3	$2^n(m-n+3)-8$, if $m+1 \geq n$ See Theorem 16(5)	2 $n-1$
$L^\star$	$3.2^{m-1}-2$, if $m \geq 2$	2	$9 \cdot 2^{m-3} - 2^{m/2} - 2$, if m is odd $9 \cdot 2^{m-3} - 2^{(m-2)/2} - 2$, if m is even	3
L^R	$2(2^m-1)$	2	$2^{p+2}-7$, if $m = 2p$ $3 \cdot 2^p - 8$, if $m = 2p-1$	2

fined transitions or the number of transitions that leave the initial state. For both families of languages we performed some experimental tests in order to have an idea of the average-case complexity of those operations.

The paper is organized as follows. Section 2 recalls some useful definitions and notation. In Section 3, we study the state and transition complexity for the union, concatenation, Kleene star and reversal operations on regular languages. For all these operations tight upper bounds are given. The tight upper bound presented for the transition complexity of the union operation refutes the conjecture presented by Y. Gao et al. [8]. We also present the same study for unary regular languages. In Subsection 3.6 we analyze some experimental results. In the Section 4 we continue the line of research of the Section 3 considering finite languages. For the concatenation, we correct the upper bound for the state complexity of complete DFAs [5], and show that if the *right* operand is larger than the *left* one, the upper bound is only reached using an alphabet of variable size. We also present some experimental results for finite languages. The algorithms and the witness language families used in this work, although new, are based on the ones of Yu et al. [26]; several proofs required new techniques.

Table 1 presents a summary and a comparison of the obtained results for transition complexity on general and finite languages. Note that the values in the table are obtained using languages for which the upper bounds are reached. This paper expands the work presented in extended abstracts [16,15] with full proofs of theorems and experimental tests.

2. Preliminaries

We recall some basic notions about finite automata and regular languages. For more details, we refer the reader to the standard literature [13,22,21].

Given two integers $m, n \in \mathbb{N}$, let $[m,n] = \{i \in \mathbb{N} \mid m \leq i \leq n\}$ and $[m,n[= \{i \in \mathbb{N} \mid n \leq i < n\}$.

A DFA is a five-tuple $A = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is a finite input alphabet, δ is the transition function $\delta : Q \times \Sigma \to Q$, q_0 in Q is the initial state, and $F \subseteq Q$ is the set of final states. Let $|\Sigma| = k$, $|Q| = n$, and without loss of generality, we assume $Q = [0,n[$ with $q_0 = 0$. The transition function can be naturally extended to subsets of Q and words $w \in \Sigma^\star$. A DFA is called *complete* if the transition function δ is total. In this paper we consider the DFAs to be not necessarily complete, i.e. with partial transition functions. For $q \in Q$ and $\sigma \in \Sigma$, if $\delta(q,\sigma)$ is defined we write $\delta(q,\sigma)\downarrow$, and $\delta(q,\sigma)\uparrow$, otherwise, and, when defining a DFA, an assignment $\delta(q,\sigma) = \uparrow$ means that the transition is undefined. The *language* accepted by A is $\mathcal{L}(A) = \{w \in \Sigma^\star \mid \delta(q_0,w) \in F\}$. Two DFAs are *equivalent* if they accept the same language. For each regular language there exists a unique minimal complete DFA with the minimum number of states. The *left quotient* of $L \subseteq \Sigma^\star$ by $x \in \Sigma^\star$ is $D_xL = \{z \mid xz \in L\}$. The equivalence relation $R_L \subseteq \Sigma^\star \times \Sigma^\star$ is defined by $(x,y) \in R_L$ if and only if $D_xL = D_yL$. The *Myhill–Nerode Theorem* states that a language L is regular if and only if R_L has a finite number of equivalence classes, i.e., L has a finite number of left quotients. This number is the number of states of the minimal complete DFA, which is unique up to isomorphism. Using Myhill–Nerode theorem, it is easy to prove that an automaton is minimal if all its states correspond to different left quotients. Thus, to prove that a DFA is minimal it is enough to show that for each state q, there is a word w such that $\delta(q,w) \in F$ and $\delta(q',w) \notin F$ for all other states $q' \neq q$. We say that this word w distinguishes q from the other states. The *state complexity*, $\mathrm{sc}(L)$, of a regular language L is the number of states of the minimal complete DFA of L. If we consider the non-complete minimal DFA, its number of states is the number of left quotients minus one, due to the removal of the *dead state*, that we denote by Ω. The left quotient corresponding to Ω is the empty language. The *incomplete state complexity* of a regular language L ($\mathrm{isc}(L)$) is the number of states of the minimal DFA without states non-conducting to a final state (thus, not necessarily complete) that accepts L. Note that $\mathrm{isc}(L)$ differs at most by 1 from $\mathrm{sc}(L)$ ($\mathrm{isc}(L) \in \{\mathrm{sc}(L)-1, \mathrm{sc}(L)\}$). The *incomplete transition complexity*, $\mathrm{itc}(L)$, of a regular language L is the minimal number of transitions over all DFAs that accept L. Whenever the model is explicitly given we refer only to *state* or *transition* complexity, by omitting the term incomplete.[1] It is well known that the minimal DFA of a language has also the minimal number of transitions.

[1] In [8] the authors use the notation $\mathrm{sc}(L)$ and $\mathrm{tc}(L)$ instead of $\mathrm{isc}(L)$ and $\mathrm{itc}(L)$.