Incomplete Transition Complexity of some Basic Operations *

Eva Maia**, Nelma Moreira, Rogério Reis

CMUP & DCC, Faculdade de Ciências da Universidade do Porto
Rua do Campo Alegre, 4169-007 Porto, Portugal
e-mail:{emaia,nam,rvr}@dcc.fc.up.pt

Abstract. Y. Gao *et al.* studied for the first time the transition complexity of Boolean operations on regular languages based on not necessarily complete DFAs. For the intersection and the complementation, tight bounds were presented, but for the union operation the upper and lower bounds differ by a factor of two. In this paper we continue this study by giving tight upper bounds for the concatenation, the Kleene star and the reversal operations. We also give a new tight upper bound for the transition complexity of the union, which refutes the conjecture presented by Y. Gao *et al.*.

1 Introduction

The descriptional complexity of regular languages has recently been extensively investigated. For deterministic finite automata (DFA), the complexity measure usually studied is the state complexity (number of states of the complete minimal DFA) [15, 16, 1, 8, 2, 17], while for nondeterministic finite automata (NFA) both state and transition complexity were considered [5, 12, 7, 9, 8], being this last one a more interesting measure. Considering complete DFAs (when the transition function is total) it is obvious that the transition complexity is 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 analysers, discrete event systems, or any application that uses dictionaries where compact automaton representations are essential [11, 4, 3]. Thus, it makes sense to study the transition complexity of regular languages based on not necessarily complete DFAs.

Y. Gao *et al.* [6] studied for the first time the transition complexity of Boolean operations on regular languages based on not necessarily complete DFAs. For the intersection and the complementation, tight bounds were presented, but

* This work was partially funded by the European Regional Development Fund through the programme COMPETE and by the Portuguese Government through the FCT under projects PEst-C/MAT/UI0144/2011 and CANTE-PTDC/EIA-CCO/101904/2008.

** Eva Maia is funded by FCT grant SFRH/BD/78392/2011.

for the union operation the upper and lower bounds differ by a factor of two. Nevertheless, they conjectured a tight upper bound for this operation.

In this paper, we continue this study by extending the analysis to the concatenation, the Kleene star and the reversal operations. For these operations tight upper bounds are given. We also give a tight upper bound for the transition complexity of the union, which refutes the conjecture presented by Y. Gao *et al.*. The same study was made for unary languages. The algorithms and the witness language families used in this work, although new, are based on the ones of Yu *et al.* [18] and several proofs required new techniques. In the Tables 1 and 2 (page 11) we summarize our results (in bold) as well as some known results for other descriptional complexity measures. All the proofs not present in this paper, can be found in an extended version of this work[1].

2 Preliminaries

We recall some basic notions about finite automata and regular languages. For more details, we refer the reader to the standard literature [10, 14, 13].

A *deterministic finite automaton* (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, q_0 in Q is the initial state, $F \subseteq Q$ is the set of final states, and δ is the transition function mapping $Q \times \Sigma \to Q$. The transition function can be extended to sets — $2^Q \times \Sigma \to 2^Q$. A DFA is *complete* if the transition function (δ) is total. In this paper we consider the DFAs to be not necessarily complete. For $s \in Q$ and $\tau \in \Sigma$, if $\delta(s, \tau)$ is defined we write $\delta(s, \tau) \downarrow$, and $\delta(s, \tau) \uparrow$, otherwise, and, when defining a DFA, an assignment $\delta(s, \tau) = \uparrow$ means that the transition is undefined.

The transition function is extended in the usual way to a function $\hat{\delta} : Q \times \Sigma^\star \to Q$. This function can also be used in sets — $\hat{\delta} : 2^Q \times \Sigma^\star \to 2^Q$. The *language* accepted by A is $\mathcal{L}(A) = \{w \in \Sigma^\star \mid \hat{\delta}(q_0, w) \in F\}$. Two DFAs are *equivalent* if they accept the same language. For each regular language, considering a total transition function or not a total one, there exists a unique minimal complete DFA with a least number of states. The *left-quotient* of $L \subseteq \Sigma^\star$ by $x \in \Sigma^\star$ is $D_x L = \{z \mid xz \in L\}$. The equivalence relation $R_L \subseteq \Sigma^\star \times \Sigma^\star$ is defined by $(x, y) \in R_L$ if and only if $D_x L = D_y L$. 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 equal to the number of states of the minimal complete DFA. The *state complexity*, $sc(L)$, of a regular language L is the number of states of the minimal complete DFA of L. If the minimal DFA is not complete its number of states is the number of left quotients minus one (the sink state is removed).

The *incomplete state complexity* of a regular language L ($isc(L)$) is the number of states of the minimal DFA, not necessarily complete, that accepts L. Note that $isc(L)$ is either equal to $sc(L) - 1$ or to $sc(L)$. The *incomplete transition complexity*, $itc(L)$, of a regular language L is the minimal number of transitions

[1] http://www.dcc.fc.up.pt/Pubs/TReports/TR12/dcc-2012-02.pdf

over all DFAs that accepts L. Whenever the model is explicitly given we refer only to *state* or *transition* complexity, by omitting the term incomplete[2] . When we talk about the minimal DFA, we refer the DFA with the minimal number of states and transitions because we have the following result:

Proposition 1. *The state-minimal DFA, not necessarily complete, which recognizes L has the minimal number of transitions of any DFA that recognizes L.*

A transition labeled by $\tau \in \Sigma$ is named by *τ-transition* (represented by $\delta(s, \tau)$, where $s \in Q$) and the number of τ-transitions of a DFA A is denoted by $t(\tau, A)$. The *τ-transition complexity* of L, $itc_\tau(L)$ is the minimal number of τ-transitions of any DFA recognizing L. In [6, Lemma 2.1] it was showed that the minimal DFA accepting L has the minimal number of τ-transitions of any DFA accepting L. From this and Proposition 1 follows that $itc(L) = \sum_{\tau \in \Sigma} itc_\tau(L)$.

The *state complexity of an operation* on regular languages is the (worst-case) state complexity of a language resulting from the operation, considered as a function of the state complexities of the operands. The (worst-case) transition complexity of an operation is defined in the same way. Usually an *upper bound* is obtained by providing an algorithm, which given DFAs as operands, constructs a DFA that accepts the language resulting from the referred operation. The number of states or transitions of the resulting DFA are upper bounds for the state or the transition complexity of the operation, respectively. To prove that an upper bound is *tight*, for each operand we can give a family of languages (parametrized by the complexity measures), such that the resulting language achieves that upper bound. For determining the transition complexity of a language operation, we also consider the following measures and refined numbers of transitions. Given a DFA $A = (Q, \Sigma, \delta, q_0, F)$ and $\tau \in \Sigma$, let $f(A) = |F|$, $i(\tau, A)$ be the number of τ-transitions leaving the initial state q_0, $u(\tau, A)$ be the number of states without τ-transitions, i.e. $u(\tau, A) = |Q| - t(\tau, A)$, and $\bar{u}(\tau, A)$ be the number of non-final states without τ-transitions. Whenever there is no ambiguity we omit A from the above definitions. If $t(\tau, A) = |Q|$ we say that A is *τ-complete*, and *τ-incomplete* otherwise. All the above measures, can be defined for a regular language L, considering the measure values for its minimal DFA. Thus, we have, respectively, $f(L)$, $i_\tau(L)$, $u_\tau(L)$, and $\bar{u}_\tau(L)$. We also prove that the upper bounds are maximal when $f(L)$ is minimal.

3 Incomplete Transition Complexity of the Union

It was shown by Y. Gao *et al.* [6] that $itc(L_1 \cup L_2) \leq 2(itc(L_1)itc(L_2) + itc(L_1) + itc(L_2))$. The lower bound $itc(L_1)itc(L_2) + itc(L_1) + itc(L_2) - 1$ was given for particular ternary language families which state complexities are relatively prime. The authors conjectured, also, that $itc(L_1 \cup L_2) \leq itc(L_1)itc(L_2) + itc(L_1) + itc(L_2)$, when $itc(L_i) \geq 2$, $i = 1, 2$.

In this section we present an upper bound for the state complexity and we give a new upper bound for the transition complexity of the union of two regular

[2] In [6] the author refer $sc(L)$ and $tc(L)$ instead of $isc(L)$ and $itc(L)$.

languages. We also present families of languages for which these upper bounds are reached, witnessing that these bounds are tight.

3.1 An Upper Bound

In the following we describe the algorithm for the union of two DFAs that was presented by Y. Gao *et al.* [6, Lemma 3.1.]. Let $A = (Q, \Sigma, \delta_A, q_0, F_A)$ and $B = (P, \Sigma, \delta_B, p_0, F_B)$ be two DFAs ($-1 \notin Q$ and $-1 \notin P$). Let $C = (R, \Sigma, \delta_C, r_0, F_C)$ be a new DFA with $R = (Q \cup \{-1\}) \times (P \cup \{-1\})$, $r_0 = (q_0, p_0)$, $F_C = (F_A \times (Q \cup \{-1\})) \cup ((P \cup \{-1\}) \times F_B)$ and

$$\delta_C((q'_A, p'_B), \tau) = \begin{cases} (\delta_A(q'_A, \tau), \delta_B(p'_B, \tau)) & \text{if } \delta_A(q'_A, \tau) \downarrow \ \wedge \delta_B(p'_B, \tau) \downarrow, \\ (\delta_A(q'_A, \tau), -1) & \text{if } \delta_A(q'_A, \tau) \downarrow \ \wedge \delta_B(p'_B, \tau) \uparrow, \\ (-1, \delta_B(p'_B, \tau)) & \text{if } \delta_A(q'_A, \tau) \uparrow \ \wedge \delta_B(p'_B, \tau) \downarrow, \\ \uparrow & \text{otherwise,} \end{cases}$$

where $\tau \in \Sigma$, $q'_A \in Q \cup \{-1\}$ and $p'_B \in P \cup \{-1\}$. Note that $\delta_A(-1, \tau)$ and $\delta_B(-1, \tau)$ are always undefined, and the pair $(-1, -1)$ never occurs in the image of δ_C. It is easy to see that DFA C accepts the language $\mathcal{L}(A) \cup \mathcal{L}(B)$. We can determine the number of states and transitions which are sufficient for any DFA C resulting from the previous algorithm:

Proposition 2 ([6]). *For any m-state DFA A and any n-state DFA B, $mn + m + n$ states are sufficient for a DFA accepting $\mathcal{L}(A) \cup \mathcal{L}(B)$.*

Proposition 3. *For any regular languages L_1 and L_2 with $isc(L_1) = m$ and $isc(L_2) = n$, one has*

$$itc(L_1 \cup L_2) \leq itc(L_1)(1 + n) + itc(L_2)(1 + m) - \sum_{\tau \in \Sigma} itc_\tau(L_1) itc_\tau(L_2).$$

The proof of Proposition 3 follows from Lemma 3.1 in Y. Gao *et al.*.

3.2 Worst-case Witnesses

In this section, we show that the upper bounds given in Proposition 2 and Proposition 3 are tight. We consider two cases, parameterized by the state complexities of the language operands: $m \geq 2$ and $n \geq 2$; and $m = 1$ and $n \geq 2$ (or vice versa). Note that in all that follows we consider automaton families over a binary alphabet, $\Sigma = \{a, b\}$. Using Myhill-Nerode theorem, it is easy to prove that these automata are minimal because all their states correspond to different left quotients.

Theorem 1. *For any integers $m \geq 2$ and $n \geq 2$, exist an m-state DFA A with $r = m$ transitions and an n-state DFA B with $s = 2n - 1$ transitions such that any DFA accepting $\mathcal{L}(A) \cup \mathcal{L}(B)$ needs, at least, $mn + m + n$ states and $(r + 1)(s + 1)$ transitions.*

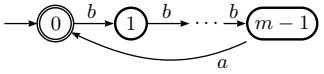

Fig. 1. DFA A with m states.

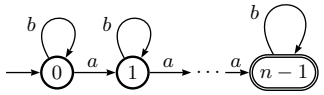

Fig. 2. DFA B with n states.

Proof. Let $A = (Q, \Sigma, \delta_A, 0, F_A)$ with $Q = \{0, \ldots, m-1\}$, $F_A = \{0\}$, $\delta_A(m-1, a) = 0$, and $\delta_A(i, b) = i + 1,\ 0 \leq i < m-1$; and $B = (P, \Sigma, \delta_B, 0, F_B)$ with $P = \{0, \ldots, n-1\}$, $F_B = \{n-1\}$, $\delta_B(i, a) = i + 1,\ 0 \leq i < n-1$, and $\delta_B(i, b) = i,\ 0 \leq i \leq n-1$ (see Fig. 1 and Fig. 2). Let C be the DFA constructed by the previous algorithm, which can be proved to be minimal. We only consider the part of the theorem corresponding to the transitions. We name τ-transitions of the DFA A by α_i $(1 \leq i \leq t(\tau, A))$ and the undefined τ-transitions named by $\bar{\alpha}_i$ $(1 \leq i \leq u(\tau, A) + 1)$, the τ-transitions of the DFA B by β_j $(1 \leq j \leq t(\tau, B))$ and the undefined τ-transitions by $\bar{\beta}_j$ $(1 \leq j \leq u(\tau, B) + 1)$. We need to consider one more undefined transition in each DFA that corresponds to the τ-transition of the state -1 added to Q and P in the union algorithm. Then, each of the τ-transitions of DFA C can only have one of the following three forms: (α_i, β_j), $(\bar{\alpha}_i, \beta_j)$, or $(\alpha_i, \bar{\beta}_j)$. Thus, the DFA C has: $mn + n - m + 1$ a-transitions because there exist $n-1$ a-transitions of the form (α_i, β_j); 2 a-transitions of the form $(\alpha_i, \bar{\beta}_j)$; and $m(n-1)$ a-transitions of the form $(\bar{\alpha}_i, \beta_j)$; and $mn + m + n - 1$ b-transitions because there exist $(m-1)n$ transitions of the form (α_i, β_j); $m-1$ b-transitions of the form $(\alpha_i, \bar{\beta}_j)$; and $2n$ b-transitions of the form $(\bar{\alpha}_i, \beta_j)$.

As $r = m$ and $n = \frac{s+1}{2}$ the DFA C has $(r+1)(s+1)$ transitions. □

The referred conjecture $itc(L_1 \cup L_2) \leq itc(L_1)itc(L_2) + itc(L_1) + itc(L_2)$ fails for these families because one has $itc(L_1 \cup L_2) = itc(L_1)itc(L_2) + itc(L_1) + itc(L_2) + 1$. Note that $r = itc(L_1)$ and $s = itc(L_2)$, thus $(r+1)(s+1) = (itc(L_1) + 1)(itc(L_2) + 1) = itc(L_1)itc(L_2) + itc(L_1) + itc(L_2) + 1$.

Theorem 2. *For any integer $n \geq 2$, exists an 1-state DFA A with one transition and an n-state DFA B with $s = 2n - 1$ transitions such that any DFA accepting $\mathcal{L}(A) \cup \mathcal{L}(B)$ has, at least, $2n + 1$ states and $2(s+1)$ transitions.*

Proof (Sketch). Let $A = (Q, \Sigma, \delta_A, 0, F_A)$ with $Q = \{0\}$, $F_A = \{0\}$, $\delta_A(0, a) = 0$, and consider the DFA B defined in the previous case. □

4 Incomplete Transition Complexity of the Concatenation

In this section we will show how many states and transitions are sufficient and necessary, in the worst case, for a DFA to accept the concatenation of two DFAs.

4.1 An Upper Bound

The following algorithm computes a DFA for the concatenation of a DFA $A = (Q, \Sigma, \delta_A, q_0, F_A)$, where $-1 \notin Q$ and $|Q| = n$, with a DFA $B = (P, \Sigma, \delta_B, p_0, F_B)$,

where $|P| = m$. Let $C = (R, \Sigma, \delta_C, r_0, F_C)$ be a new DFA with $R = (Q \cup \{-1\}) \times 2^P - F_A \times 2^{P-\{p_0\}}$, $r_0 = \langle q_0, \emptyset \rangle$ if $q_0 \notin F_A$ or $r_0 = \langle q_0, \{p_0\} \rangle$ if $q_0 \in F_A$, $F_C = \{\langle q, T \rangle \in R \mid T \cap F_B \neq \emptyset\}$, and for $a \in \Sigma$, $\delta_C(\langle q, T \rangle, a) = \langle q', T' \rangle$ with $q' = \delta_A(q, a)$, if $\delta_A(q, a) \downarrow$ or $q' = -1$ otherwise, and $T' = \delta_B(T, a) \cup \{p_0\}$ if $q' \in F_A$ or $T' = \delta_B(T, a)$ otherwise. DFA C recognizes the language $\mathcal{L}(A)\mathcal{L}(B)$.

The following results determine the number of states and transitions which are sufficient for any DFA C resulting from the previous algorithm.

Proposition 4. *For any m-state DFA A and any n-state DFA B, $(m+1)2^n - f(A)2^{n-1} - 1$ states are sufficient for any DFA accepting $\mathcal{L}(A)\mathcal{L}(B)$.*

Note that the minus one in the formula is due to the removal of the state $(-1, \emptyset)$.

Corollary 1. *The formula in Proposition 4 is maximal when $f(A) = 1$.*

Given an automaton A, the alphabet can be partitioned in two sets Σ_c^A and Σ_i^A such that $\tau \in \Sigma_c^A$ if A is τ-complete, or $\tau \in \Sigma_i^A$ otherwise. In the same way, considering two automata A and B, the alphabet can be divided into four disjoint sets Σ_{ci}, Σ_{cc}, Σ_{ii} and Σ_{ic}. As before, these notations can be extended to regular languages considering their minimal DFA.

Proposition 5. *For any regular languages L_1 and L_2 with $isc(L_1) = m$, $isc(L_2) = n$, $u_\tau = u_\tau(L_2)$, $f = f(L_1)$ and $\bar{u}_\tau = \bar{u}_\tau(L_1)$, one has*

$$itc(L_1 L_2) \leq |\Sigma|(m+1)2^n - |\Sigma_c^{L_2}|(f2^{n-1}+1) - \sum_{\tau \in \Sigma_i^{L_2}} (2^{u_\tau} + f2^{itc_\tau(L_2)}) - \\ - \sum_{\tau \in \Sigma_{ii}} \bar{u}_\tau 2^{u_\tau} - \sum_{\tau \in \Sigma_{ic}} \bar{u}_\tau.$$

Proof. Let A and B be the minimal DFAs that recognize L_1 and L_2, respectively. Consider the DFA C such that $\mathcal{L}(C) = \mathcal{L}(A)\mathcal{L}(B)$ and C is constructed using the algorithm described above. We name the τ-transitions of A and B as in the proof of the Theorem 1, with a slight modification: $1 \leq j \leq u(\tau, B)$. The τ-transitions of C are pairs (θ, γ) where θ is an α_i or $\bar{\alpha}_i$, and γ is a set of β_j or $\bar{\beta}_j$. By construction, C cannot have transitions where θ is an $\bar{\alpha_i}$, and γ is a set with only $\bar{\beta}_j$, because these would correspond to pairs of undefined transitions.

Let us count the number of τ-transitions of C. If $\tau \in \Sigma_{ci}$, the number of C τ-transitions is $(t(\tau, A)+1)2^{t(\tau,B)+u(\tau,B)} - 2^{u(\tau,B)} - f(A)2^{t(\tau,B)}$. The number of θs is $t(\tau, A)+1$ and the number of γs is $2^{t(\tau,B)+u(\tau,B)}$. From the product we need to remove the $2^{u(\tau,B)}$ sets of transitions of the form $(v, \emptyset)$ where v corresponds to the undefined τ-transition leaving the added state -1 of DFA A. If θ corresponds to a transition that leaves a final state of A, then γ needs to include the initial state of the B. Thus we also remove the $f(A)2^{t(\tau,B)}$ pairs. If $\tau \in \Sigma_{cc}$, C has $(t(\tau, A)+1)2^{t(\tau,B)} - 1 - f(A)2^{t(\tau,B)-1}$ τ-transitions. In this case, $u(\tau, B) = 0$. The only pair we need to remove is $(v, \emptyset)$ where v corresponds to the undefined τ-transition leaving the added state -1 of DFA A. Analogously, if $\tau \in \Sigma_{ii}$, C

has $(t(\tau,A)+u(\tau,A)+1)2^{t(\tau,B)+u(\tau,B)}-(\bar{u}(\tau,A)+1)2^{u(\tau,B)}-f(A)2^{t(\tau,B)}$ τ-transitions. Finally, if $\tau\in\Sigma_{ic}$, C has $(t(\tau,A)+u(\tau,A)+1)2^{t(\tau,B)}-(\bar{u}(\tau,A)+1)-f(A)2^{t(\tau,B)-1}$ τ-transitions.

Thus, after some simplifications, the right side of the inequality in the proposition holds. □

4.2 Worst-case Witnesses

The following results show that the complexity upper bounds found in Propositions 4 and 5 are tight. As in the previous section we need to consider three different cases, according to the state and transition complexities of the operands. All following automaton families have $\Sigma=\{a,b,c\}$. For these automata, it is easy to prove that they are minimal. It is also possible to prove that there cannot exist binary language families that reach the upper bounds.

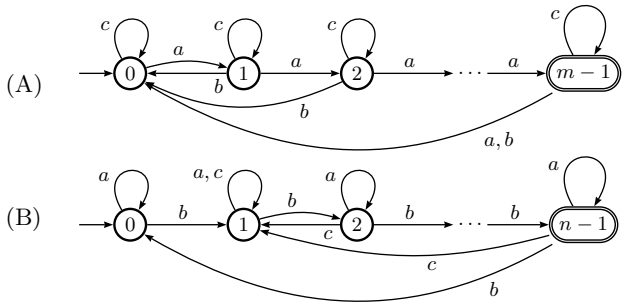

Fig. 3. DFA A with m states and DFA B with n states.

Theorem 3. *For any integers $m\geq 2$ and $n\geq 2$ exist an m-state DFA A with $r=3m-1$ transitions and an n-state DFA B with $s=3n-1$ transitions such that any DFA accepting $\mathcal{L}(A)\mathcal{L}(B)$ has, at least, $(m+1)2^n-2^{n-1}-1$ states and $(r+1)2^{\frac{s+1}{3}}+3.2^{\frac{s-2}{3}}-5$ transitions.*

Proof. Let $A=(Q,\Sigma,\delta_A,0,F_A)$ with $Q=\{0,\ldots,m-1\}$, $F_A=\{m-1\}$, and $\delta_A(i,a)=i+1 \bmod m$, if $0\leq i\leq m-1$, $\delta_A(i,b)=0$, if $1\leq i\leq m-1$, and $\delta_A(i,c)=i$ if $0\leq i\leq m-1$; and $B=(P,\Sigma,\delta_B,0,F_B)$ with $P=\{0,\ldots,n-1\}$, $F_B=\{n-1\}$, $\delta_B(i,a)=i$ if $0\leq i\leq n-1$, $\delta_B(i,b)=i+1 \bmod n$, if $0\leq i\leq n-1$, and $\delta_B(i,c)=1,\ 1\leq i\leq n-1$ (see Fig. 3). Consider the DFA C, constructed with the previous algorithm, such that $\mathcal{L}(C)=\mathcal{L}(A)\mathcal{L}(B)$ and which can be proved to be minimal. We only prove the part of the theorem correspondent to the number of transitions. As in Proposition 5, the transitions of C are pairs (θ,γ). Then, C has: $(m+1)2^n-2^{n-1}-1$, a-transitions. There are $m+1$ θs and 2^n γs, from which we need to remove the transition pair $(-1,\emptyset)$. If θ is a transition which leaves a final state of A, γ needs to include the transition that leaves the initial state of B. Thus, 2^{n-1} pairs are removed; $(m+1)2^n-2^{n-1}-2$, b-transitions. Here, the transition $(\bar{\theta},\emptyset)$ is removed; and $(m+1)2^n-2^{n-1}-2$, c-transitions. This is analogous to the previous cases. As $m=\frac{r+1}{3}$ and $n=\frac{s+1}{3}$ the DFA C has $(r+1)2^{\frac{s+1}{3}}+3.2^{\frac{s-2}{3}}-5$ transitions. □

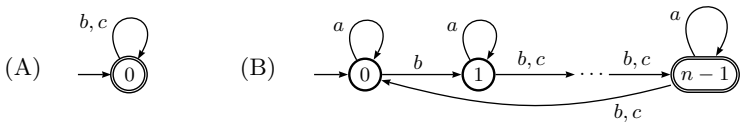

Fig. 4. DFA A with 1 state and DFA B with n states.

Theorem 4. *For any integer $n \geq 2$, exist a 1-state DFA A with 2 transitions and an n-state DFA B with $s = 3n-1$ transitions such that any DFA accepting $\mathcal{L}(A)\mathcal{L}(B)$ has, at least, $2^{n+1}-2^{n-1}-1$ states and $3(2^{\frac{s+4}{3}}-2^{\frac{s-2}{3}})-4$ transitions.*

Proof. Let $A = (Q, \Sigma, \delta_A, 0, F_A)$ with $Q = \{0\}$, $F_A = \{0\}$, $\delta_A(0, b) = \delta_A(0, c) = 0$; and define $B = (P, \Sigma, \delta_B, 0, F_B)$ with $P = \{0, \ldots, n-1\}$, $F_B = \{n-1\}$, $\delta_B(i, a) = i$ if $0 \leq i \leq n-1$, $\delta_B(i, b) = i+1 \bmod n$ if $0 \leq i \leq n-1$, and $\delta_B(i, c) = i+1 \bmod n$, if $1 \leq i \leq n-1$ (see Fig. 4). Consider the DFA $C = (R, \Sigma, \delta, 0, F)$, constructed by the previous algorithm, such that $\mathcal{L}(C) = \mathcal{L}(A)\mathcal{L}(B)$. One needs to prove that C is minimal, i.e. all states are reachable from the initial state and are pairwise distinguishable. The DFA C has states (s, c) with $s \in \{-1, 0\}$, $c = \{i_1, \ldots, i_k\}$, $1 \leq k \leq n$, and $i_1 < \cdots < i_k$. There are two kinds of states: final states where $i_k = n-1$; and non-final states where $i_k \neq n-1$. Note that whenever $s = 0$, $i_1 = 0$. Taking this form of the states into account is not difficult to prove that the DFA C is minimal. The proof of the second part of the theorem is similar to the proof of Theorem 3. □

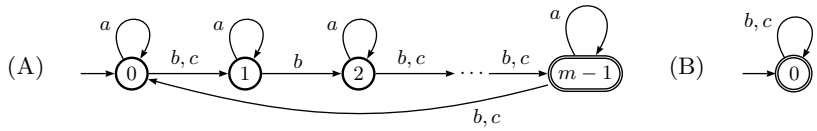

Fig. 5. DFA A with m states and DFA B with 1 state.

Theorem 5. *For any integer $m \geq 2$ exists an m-state DFA A. with $r = 3m-1$ transitions and an 1-state DFA B with 2 transitions such that any DFA accepting $\mathcal{L}(A)\mathcal{L}(B)$ has at least $2m$ states and $2r$ transitions.*

Proof (Sketch). Let $A = (P, \Sigma, \delta_A, 0, F_A)$ with $P = \{0, \ldots, n-1\}$, $F_A = \{m-1\}$, $\delta_A(i, X) = i$, if $0 \leq i \leq m-1$, $\delta_A(i, b) = i+1 \bmod m$, if $0 \leq i \leq m-1$, $\delta_A(i, c) = i+1 \bmod m$ if $i = 0$ or $2 \leq i \leq m-1$; and $B = (Q, \Sigma, \delta_B, 0, F_B)$ with $Q = \{0\}$, $F_B = \{0\}$, and $\delta_B(0, b) = \delta_B(0, c) = 0$ (see Fig. 5). □

5 Incomplete Transition Complexity of the Star

In this section we give a tight upper bound for the incomplete transition complexity of the star operation. The incomplete state complexity of star coincides with the one in the complete case.

5.1 An Upper Bound

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Let $F_0 = F \setminus \{q_0\}$ and suppose that $l = |F_0| \geq 1$. If $F = \{q_0\}$, then $\mathcal{L}(A)^\star = \mathcal{L}(A)$. The following algorithm obtains the kleene star of a DFA A. Let $A' = (Q', \Sigma, \delta', q_0', F')$ be a new DFA where $q_0' \notin Q$ is a new initial state, $Q' = \{q_0'\} \cup \{P \mid P \subseteq (Q \setminus F_0) \wedge P \neq \emptyset\} \cup \{P \mid P \subseteq Q \wedge q_0 \in P \wedge P \cap F_0 \neq \emptyset\}$, $F' = \{q_0'\} \ \cup \{R \mid R \subseteq Q \wedge R \cap F \neq \emptyset\}$, and for $a \in \Sigma$,

$$\delta'(q_0', a) = \begin{cases} \{\delta(q_0, a)\} & \text{if } \delta(q_0, a)\downarrow \ \wedge \delta(q_0, a) \notin F_0, \\ \{\delta(q_0, a), q_0\} & \text{if } \delta(q_0, a)\downarrow \ \wedge \delta(q_0, a) \in F_0, \\ \emptyset & \text{if } \delta(q_0, a)\uparrow. \end{cases}$$

and

$$\delta'(R, a) = \begin{cases} \delta(R, a) & \text{if } \delta(R, a) \cap F_0 = \emptyset, \\ \delta(R, a) \cup \{q_0\} & \text{if } \delta(R, a) \cap F_0 \neq \emptyset, \\ \emptyset & \text{if } \delta(R, a) \ = \emptyset. \end{cases}$$

We can verify that DFA A' recognizes the language $\mathcal{L}(A)^\star$.

The following results present upper bounds for the number of states and transitions for any DFA A' resulting from the algorithm described above.

Proposition 6. *For any integer $n \geq 2$ and any n-state DFA A, any DFA accepting $\mathcal{L}(A)^\star$ needs at least $2^{n-1} + 2^{n-l-1}$ states.*

Corollary 2. *The formula in Proposition 6 is maximal when $l = 1$.*

Proposition 7. *For any regular language L with $isc(L) = n$, $i_\tau = i_\tau(L)$, and and $\bar{u}_\tau = \bar{u}_\tau(L)$, one has*

$$itc(L^\star) \leq |\Sigma|(2^{n-1} + 2^{n-l-1}) + \sum_{\tau \in \Sigma_i} (i_\tau - 2^{\bar{u}_\tau})$$

Proof. Let A be a minimal DFA recognizing a language L. Consider the DFA A', constructed with the previous algorithm, such that $\mathcal{L}(A') = \mathcal{L}(A)^\star$. The number of τ-transitions of A' is the summation of:

1. i_τ τ-transitions leaving the initial state of A.
2. The number of sets of τ-transitions of A leaving only non-final states:
   (a) $(2^{t_\tau - l}) - 1$, if A is τ-complete, we have $t_\tau - l$ τ-transitions of this kind, and we remove the empty set.
   (b) $2^{t_\tau - l + u_\tau} - 2^{\bar{u}_\tau}$, if A is τ-incomplete: $t_\tau - l + u_\tau$ of this kind, and we subtract number of sets with only undefined τ-transitions of A.
3. The number of sets of τ-transitions of A leaving final and non-final states of A. We do not count the transition leaving the initial state of A because, by construction, if a transition of A' contains a transition leaving a final state of A then it also contains the one leaving the initial state of A.
   (a) $(2^l - 1)2^{t_\tau - l - 1}$, if A is τ-complete.
   (b) $(2^l - 1)2^{t_\tau - l - 1 + u_\tau}$, if A is τ-incomplete.

Thus, the inequality in the proposition holds. □

5.2 Worst-case Witnesses

Let us present an automaton family for which the upper bounds in Proposition 6 and Proposition 7 are reached. The following automaton family has $\Sigma = \{a, b\}$. Using Myhill-Nerode theorem, it is easy to prove that these automata are minimal.

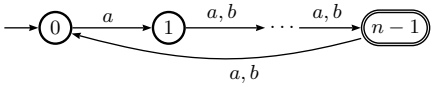

Fig. 6. DFA A with n states.

Theorem 6. *For any integer $n \geq 2$, exists an n-state DFA A with $r = 2n - 1$ transitions such that any DFA accepting $\mathcal{L}(A)^\star$ has, at least, $2^{n-1} + 2^{n-2}$ states and $2^{\frac{r+1}{2}} + 2^{\frac{r-1}{2}} - 2$ transitions.*

Proof. Let $A = (Q, \Sigma, \delta_A, 0, F_A)$ with $Q = \{0, \ldots, n-1\}$, $F_A = \{n-1\}$, $\delta_A(i, a) = i + 1 \bmod m$ for $0 \leq i \leq n-1$, and $\delta_A(i, b) = i + 1 \bmod m$ for $1 \leq i \leq n-1$ (see Fig.6). Consider the DFA A', constructed with the previous algorithm, such that $\mathcal{L}(A') = (\mathcal{L}(A))^\star$ and which can be proved to be minimal. We only analyse the transition complexity. The DFA A' has:

- $2^{n-1} + 2^{n-2}$ a–transitions because $i(a) = 1$, $2^{n-1} - 1$ a–transitions which corresponds to case 2 of Proposition 7 and 2^{n-2} a–transitions which corresponds to case 3 of Proposition 7.
- $2^{n-1} - 2 + 2^{n-2}$ b–transitions because it has $2^{n-2+1} - 2$ b–transitions which corresponds to case 2 of Proposition 7, and 2^{n-3+1} b–transitions which corresponds to case 3.

As $n = \frac{r+1}{2}$, A' has $2^{\frac{r+1}{2}} + 2^{\frac{r-1}{2}} - 2$ transitions. □

6 Final Remarks

It is known that considering complete DFAs the state complexity of the reversal operation reaches the upper bound 2^n, where n is the state complexity of the operand language. By the subset construction, a (complete) DFA resulting from the reversal has a state which corresponds to the $\emptyset$, which is a dead state. Therefore, if we remove that state the resulting automaton is not complete and the incomplete state complexity is $2^n - 1$. Consequently the transition complexity is $|\Sigma|(2^n - 1)$. Note that the worst case of the reversal operation is when the operand is complete.

In this paper we presented tight upper bounds for the incomplete state and incomplete transition complexities for the union, the concatenation, the Kleene star and the reversal of regular languages, with $|\Sigma| \geq 2$. Transition complexity bounds are expressed as functions of several more fine-grained measures of the operands, such as the number of final states, the number of undefined transitions or the number of transitions that leave initial state.

Operation	sc	isc	nsc
$L_1 \cup L_2$	mn	$\mathbf{mn + m + n}$	$m + n + 1$
$L_1 \cap L_2$	mn	mn	mn
L^C	n	$n + 1$	2^n
$L_1 L_2$	$m2^n - f_1 2^{n-1}$	$\mathbf{(m+1)2^n - f_1 2^{n-1} - 1}$	$m + n$
$L^\star$	$2^{m-1} + 2^{m-l-1}$	$\mathbf{2^{m-1} + 2^{m-l-1}}$	$m + 1$
L^R	2^m	$\mathbf{2^m - 1}$	$m + 1$

Table 1. State Complexity.

Table 1 and Table 2 summarize some of the results on state complexity and transition complexity of basic operations on regular languages, respectively. In Table 1 we present the state complexity, based on complete DFA (*sc*) [18], DFA (*isc*) (new results here presented and [6]); and NFAs (*nsc*) [7]. Nondeterministic transition complexity (*ntc*) of basic operations on regular languages was studied by Domaratzki and Salomaa [5, 12]. They also used refined number of transitions for computing the operational transition complexity. In Table 2, $s(L)$ is the minimal number of transitions leaving the initial state of any transition-minimal NFA M accepting L, and $f_{in}(L)$ is the number of transitions entering the final states of any transition-minimal NFA M accepting L. The upper bound for the nondeterministic transition complexity of the complementation is not tight, and thus we inscribe the lower and the upper bounds.

Operation	itc	ntc
$L_1 \cup L_2$	$\mathbf{itc(L_1)(1+n) + itc(L_2)(1+m)} - \sum_{\tau \in \mathbf{\Sigma}} \mathbf{itc}_\tau\mathbf{(L_2)itc}_\tau\mathbf{(L_1)}$	$ntc(L_1) + ntc(L_2) + s(L_1) + s(L_2)$
$L_1 \cap L_2$	$itc(L_1)itc(L_2)$	$\sum_{\tau \in \Sigma} ntc_\tau(L_1)ntc_\tau(L_2)$
L^C	$\lvert\Sigma\rvert(itc(L) + 2)$	$\lvert\Sigma\rvert 2^{ntc(L)+1}$
		$2^{\frac{ntc(L)}{2}-2} - 1$
$L_1 L_2$	$\lvert\mathbf{\Sigma}\rvert\mathbf{(m+1)2^n} - \lvert\mathbf{\Sigma_c^{L_2}}\rvert\mathbf{(f\ 2^{n-1}+1)} - \sum_{\tau \in \mathbf{\Sigma_i^{L_2}}} (\mathbf{2^{u_\tau}} + \mathbf{f\ 2^{itc_\tau(L_2)}}) - \sum_{\tau \in \mathbf{\Sigma_{ii}}} \mathbf{\bar{u}_\tau 2^{u_\tau}} - \sum_{\tau \in \mathbf{\Sigma_{ic}}} \mathbf{\bar{u}_\tau}$	$ntc(L_1) + ntc(L_2) + f_{in}(L_1)$
$L^\star$	$\lvert\mathbf{\Sigma}\rvert(\mathbf{2^{m-1-1}} + \mathbf{2^{m-1}}) + \sum_{\tau \in \mathbf{\Sigma_i}} (\mathbf{i}_\tau - \mathbf{2^{\bar{u}_\tau}})$	$ntc(L) + f_{in}(L)$
L^R	$\lvert\mathbf{\Sigma}\rvert(\mathbf{2^m - 1})$	$ntc(L) + f(L)$

Table 2. Transition Complexity.

In the case of unary languages, if a DFA is not complete it represents a finite language. Thus, the worst-case state complexity of operations occurs when the operand DFAs are complete. For these languages the (incomplete) transition complexity coincide with the (incomplete) state complexity. The study for union and intersection was made by Y. Gao *et al.* [6] and it is similar for the other operations studied in this article.

In future work we plan to extend this study to finite languages and to other regular preserving operations. In order to understand the relevance of these par-

tial transition functions based models, some experimental as well as asymptotic study of the average size of these models must be done.

Acknowledgements This, as many other subjects, was introduced to us by Sheng Yu. He will be forever in our mind.

References

1. Bordihn, H., Holzer, M., Kutrib, M.: Determination of finite automata accepting subregular languages. Theor. Comput. Sci. 410(35), 3209–3222 (2009)
2. Brzozowski, J.A.: Complexity in convex languages. In: Dediu, A.H., Fernau, H., Martín-Vide, C. (eds.) 4th LATA 2010 Proc. LNCS, vol. 6031, pp. 1–15. Springer (2010)
3. Cassandras, C.G., Lafortune, S.: Introduction to discrete event systems. Springer-Verlag (2006)
4. Daciuk, J., Weiss, D.: Smaller representation of finite state automata. In: Bouchou-Markhoff, B., Caron, P., Champarnaud, J.M., Maurel, D. (eds.) 16th CIAA 2011 Proc. LNCS, vol. 6807, pp. 118–129. Springer (2011)
5. Domaratzki, M., Salomaa, K.: Transition complexity of language operations. Theor. Comput. Sci. 387(2), 147–154 (2007)
6. Gao, Y., Salomaa, K., Yu, S.: Transition complexity of incomplete dfas. Fundam. Inform. 110(1-4), 143–158 (2011)
7. Holzer, M., Kutrib, M.: State complexity of basic operations on nondeterministic finite automata. In: Champarnaud, J.M., Maurel, D. (eds.) 7th CIAA 2002 Proc. LNCS, vol. 2608, pp. 148–157. Springer (2003)
8. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata. In: Dediu, A.H., Ionescu, A.M., Martín-Vide, C. (eds.) 3rd LATA 2009 Proc. LNCS, vol. 5457, pp. 23–42. Springer (2009)
9. Holzer, M., Kutrib, M.: Nondeterministic finite automata - recent results on the descriptional and computational complexity. Int. J. Found. Comput. Sci. 20(4), 563–580 (2009)
10. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
11. Owens, S., Reppy, J.H., Turon, A.: Regular-expression derivatives re-examined. J. Funct. Program. 19(2), 173–190 (2009)
12. Salomaa, K.: Descriptional complexity of nondeterministic finite automata. In: Harju, T., Karhumäki, J., Lepistö, A. (eds.) 11th Developments in Language Theory, DTL'2007. LNCS, vol. 4588, pp. 31–35. Springer (2007)
13. Shallit, J.: A Second Course in Formal Languages and Automata Theory. CUP (2008)
14. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 1, pp. 41–110. Springer (1997)
15. Yu, S.: State complexity: Recent results and open problems. Fundam. Inform. 64(1-4), 471–480 (2005)
16. Yu, S.: On the state complexity of combined operations. In: Ibarra, O.H., Yen, H.C. (eds.) 11th CIAA 2006 Proc. LNCS, vol. 4094, pp. 11–22. Springer (2006)
17. Yu, S., Gao, Y.: State complexity research and approximation. In: Mauri, G., Leporati, A. (eds.) 15th DLT 2011 Proc. LNCS, vol. 6795, pp. 46–57. Springer (2011)
18. Yu, S., Zhuang, Q., Salomaa, K.: The state complexities of some basic operations on regular languages. Theor. Comput. Sci. 125(2), 315–328 (1994)