On the Incomplete Transition Complexity of some Basic Operations on Regular Languages

Eva Maia Nelma Moreira Rogério Reis
e-mail:{emaia,nam,rvr}@dcc.fc.up.pt
DCC-FC & CMUP, Universidade do Porto
Rua do Campo Alegre 1021, 4169-007 Porto, Portugal

Technical Report Series: DCC-2012-02
Version 1.3 March 2013
Old versions: 1.2 May 2012

On the Incomplete Transition Complexity of some Basic Operations on Regular Languages

Eva Maia, Nelma Moreira, Rogério Reis
{emaia,nam,rvr}@dcc.fc.up.pt

March 15, 2013

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 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 28) we summarize our results (in bold) as well as some known results for other descriptional complexity measures.

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. We can minimize a given DFA A if we calculate its quotient automaton by the equivalence relation $\approx$. Minimal DFAs are unique up to isomorphism. 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 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[1]. 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.*

Proof. Let A be a non-minimal initially connected DFA which recognizes L, and A_m be the minimal DFA that recognizes L. The quotient of A, $A/\approx$, is isomorphic to A_m. By construction, we know that $A/\approx$ has fewer transitions than A. Therefore A_m has the minimal number of transitions of any DFA (up to isomorphism) that recognizes L. $\square$

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

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

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 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 $\mathcal{L}(A) \cup \mathcal{L}(B)$.*

Proof. Consider $A = (Q, \Sigma, \delta_A, q_0, F_A)$ with m states, $B = (P, \Sigma, \delta_B, p_0, F_B)$ with n states and $C = (R, \Sigma, \delta_C, r_0, F_C)$, constructed with the algorithm above, such that $\mathcal{L}(C) = \mathcal{L}(A) \cup \mathcal{L}(B)$. The number of states in R is:

$$mn + m + n$$

because R is a set of pairs (α, β) where α is a state in Q or -1 and β is a state in P or -1; but α and β cannot be simultaneously -1. Thus R has $mn + m + n$ states because the number of possible αs is $m + 1$, the number of possible β is $n + 1$ and we need exclude the pair $(-1, -1)$. Therefore,

$$\begin{aligned} & (m+1)(n+1) - 1 = \\ = \; & mn + m + n + 1 - 1 \\ = \; & mn + m + n \end{aligned}$$

□

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

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

Proof. Consider $A = (Q, \Sigma, \delta_A, q_0, F_A)$ with m states, $B = (P, \Sigma, \delta_B, p_0, F_B)$ with n states and $C = (R, \Sigma, \delta_C, r_0, F_C)$, constructed with the algorithm above, such that $\mathcal{L}(C) = \mathcal{L}(A) \cup \mathcal{L}(B)$. We can count the number of transitions of DFA C in two ways:

1) If we use the Lemma 3.1 of Y. Gao *et al.* we know that for any $\tau \in \Sigma$:

$$\begin{aligned} itc_\tau(\mathcal{L}(A) \cup \mathcal{L}(B)) \leq \; & itc_\tau(\mathcal{L}(A)).itc_\tau(\mathcal{L}(B)) + itc_\tau(\mathcal{L}(A))(1 + isc(\mathcal{L}(B)) - itc_\tau(\mathcal{L}(B))) \\ & + itc_\tau(\mathcal{L}(B))(1 + isc(\mathcal{L}(A))) - itc_\tau(\mathcal{L}(A))) \end{aligned}$$

But we also know that $itc(L) = \sum_{\tau \in \Sigma} itc_\tau(L)$. Thus,

$$
\begin{aligned}
itc(\mathcal{L}(A)\cup\mathcal{L}(B)) &\leq \sum_{\tau\in\Sigma}(itc_\tau(\mathcal{L}(A)).itc_\tau(\mathcal{L}(B)) + itc_\tau(\mathcal{L}(A))(1+isc(\mathcal{L}(B))\\
&\quad -itc_\tau(\mathcal{L}(B))) + itc_\tau(\mathcal{L}(B))(1+isc(\mathcal{L}(A))) - itc_\tau(\mathcal{L}(A)))\\
itc(\mathcal{L}(A)\cup\mathcal{L}(B)) &\leq \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(A)).itc_\tau(\mathcal{L}(B)) + \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(A)) + \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(A))\\
&\quad isc(\mathcal{L}(B)) - \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(A))itc_\tau(\mathcal{L}(B)) + \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(B)) +\\
&\quad \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(B))isc(\mathcal{L}(A))) - \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(B))itc_\tau(\mathcal{L}(A)))\\
itc(\mathcal{L}(A)\cup\mathcal{L}(B)) &\leq \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(A)) + \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(A))isc(\mathcal{L}(B)) + \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(B))\\
&\quad + \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(B))isc(\mathcal{L}(A))) - \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(B))itc_\tau(\mathcal{L}(A)))\\
itc(\mathcal{L}(A)\cup\mathcal{L}(B)) &\leq \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(A)) + isc(\mathcal{L}(B))\sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(A)) + \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(B))\\
&\quad + isc(\mathcal{L}(A))\sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(B)) - \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(B))itc_\tau(\mathcal{L}(A)))
\end{aligned}
$$

But, $itc(\mathcal{L}(i)) = \sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(i)),\ i \in \{A, B\}$ and $isc(\mathcal{L}(A)) = m$ and $isc(\mathcal{L}(B)) = n$. Thus,

$$
\begin{aligned}
itc(\mathcal{L}(A)\cup\mathcal{L}(B)) &\leq itc(\mathcal{L}(A)) + isc(\mathcal{L}(B))itc(\mathcal{L}(A)) + itc(\mathcal{L}(B)) + isc(\mathcal{L}(A))itc(\mathcal{L}(B))\\
&\quad -\sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(B))itc_\tau(\mathcal{L}(A)))\\
itc(\mathcal{L}(A)\cup\mathcal{L}(B)) &\leq itc(\mathcal{L}(A)) + n\ itc(\mathcal{L}(A)) + itc(\mathcal{L}(B)) + m\ itc(\mathcal{L}(B))\\
&\quad -\sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(B))itc_\tau(\mathcal{L}(A)))\\
itc(\mathcal{L}(A)\cup\mathcal{L}(B)) &\leq itc(\mathcal{L}(A))(1+n) + itc(\mathcal{L}(B))(1+m) \qquad (1)\\
&\quad -\sum_{\tau\in\Sigma} itc_\tau(\mathcal{L}(B))itc_\tau(\mathcal{L}(A)))
\end{aligned}
$$

2) Consider the τ-transitions of the DFA A named by α_i $(1 \leq i \leq t(\tau, A))$ and the undefined τ-transitions named by $\bar{\alpha}_i$ $(1 \leq i \leq u(\tau, A) + 1)$. Consider also the τ-transitions of the DFA B named by β_j $(1 \leq j \leq t(\tau, B))$ and the undefined τ-transitions named by $\bar{\beta}_j$ $(1 \leq j \leq u(\tau, B) + 1)$. We need to consider one more undefined transition in each DFA which corresponds to the state -1 added to Q and P in the union algorithm, defined in the previous section. Each of the τ-transitions of DFA C can only have one of the following three forms:

- (α_i, β_j)
- $(\bar{\alpha}_i, \beta_j)$

- $(\alpha_i, \bar{\beta}_j)$

Thus, the number of τ-transitions of the DFA C is:

$$t(\tau, A)t(\tau, B) + t(\tau, A)(u(\tau, B) + 1) + (u(\tau, A) + 1)t(\tau, B) \tag{2}$$

because $t(\tau, A)t(\tau, B)$ is the number of pairs of the form (α_i, β_j); $t(\tau, A)(u(\tau, B) + 1)$ is the number of pairs of the form $(\bar{\alpha}_i, \beta_j)$; and $(u(\tau, A) + 1)t(\tau, B)$ is the number of pairs of the form $(\alpha_i, \bar{\beta}_j)$.

We know that $u(\tau, A) = m - t(\tau, A)$ and $u(\tau, B) = n - t(\tau, B)$, then the number of τ-transitions is:

$$t(\tau, A)t(\tau, B) + t(\tau, A)(n - t(\tau, B) + 1) + t(\tau, B)(m - t(\tau, A) + 1)$$

Therefore the number of transitions of the DFA C is:

$$\sum_{\tau \in \Sigma} t(\tau, A)t(\tau, B) + t(\tau, A)(n - t(\tau, B) + 1) + t(\tau, B)(m - t(\tau, A) + 1)$$

And if $itc_\tau(\mathcal{L}(A)) = t(\tau, A)$ and $itc_\tau(\mathcal{L}(B)) = t(\tau, B)$ this number is equal to (1).

□

3.1.1 More on Transition Complexity

We can use the unique representation of two DFAs A and B to obtain an upper bound for the number of transitions of the DFA C accepting $\mathcal{L}(A) \cup \mathcal{L}(B)$. For that we do the cartesian product of a special unique representation of A and a special unique representation of B; and of this product we remove the pair $(-1, -1)$. Then, the number of pairs we obtain as result represents the upper bound for the number of transitions of the DFA C. In the following we present the algorithm which compute this process.

```

def split_repr(r, n, k):
  d={}.fromkeys([x for x in range(0,k)],[])
  for i in range(0, len(r)):
        p=i % k
        d[p]=d[p]+ [r[i]]
    return d


def prod_cart(s0, s1):
    p=[]
    for i in s0.keys():
        l0=s0[i]+[-1]
        l1=s1[i]+[-1]
        p+=[(x,y) for x in l0 for y in l1 if x+y!=-2]
    return p

s0=split_repr(r0,n0,k0)
s1=split_repr(r1,n1,k1)
p=prod_cart(s0,s1)
```

The function `split_repr` splits the unique string representation into its projections. For example, if we have the string [0,1,-1,-1,2,-1,-1,-1,-1], three states (n) and a three letter alphabet (k), the function returns the dictionary $\{0 : [0, -1, -1], 1 : [1, 2, -1], 2 : [-1, -1, -1]\}$, where the keys are the alphabet symbols and each value is a list of the correspondent projections.

The `prod_cart` function receives two dictionaries, as the previous one. We add -1 to each list of the dictionaries; then we build a pair with the correspondent values of each dictionary, if at least one of the projections is different to -1 (line 15). Thus we obtain a list with all combinations of the projections.

With this algorithm, the number of pairs we obtain, for each projection($\tau \in \Sigma$), for two DFAs A with m states and B with n states is:

- $t(\tau, A)t(\tau, B)$
  number of pairs where both components of the pair is different of -1;
- $t(\tau, A).(n - t(\tau) + 1, B)$
  number of pairs where the first component of the pair is different from -1 and the second component is equal to -1.
- $t(\tau, B).(m - t(\tau, A) + 1)$
  number of pairs where the first component of the pair is equal to -1 and the second component is different from -1.

But this is the same we have in the proof of Lemma 3.1.in [6], and in the section above.

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.

3.2.1 Case 1: $m \geq 2$ and $n \geq 2$.

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).

Proposition 4. *DFA A is minimal.*

Proof. Consider $x, y \in \Sigma^\star$ such that $\hat{\delta}(0, x) = i$ and $\hat{\delta}(0, y) = j,\ i \neq j$. It is clear that $xb^{m-1-i}a \in \mathcal{L}(A)$ but $yb^{m-1-i}a \notin \mathcal{L}(A)$. Then x and y are in different left quotients induced by $\mathcal{L}(A)$. □

Proposition 5. *DFA B is minimal.*

Proof. Consider $x, y \in \Sigma^\star$ such that $\hat{\delta}(0, x) = i$ and $\hat{\delta}(0, y) = j,\ i \neq j$. It is clear that $xa^{n-1-i} \in \mathcal{L}(B)$ but $ya^{n-1-i} \notin \mathcal{L}(B)$. Then x and y are in different left quotients induced by $\mathcal{L}(B)$. □

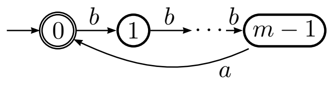

Figure 1: DFA A with m states.

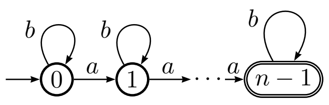

Figure 2: DFA B with n states.

Proposition 6. *For any integers $m \geq 2$ and $n \geq 2$, exist an m-state DFA A and an n-state DFA B (Fig. 1 and Fig. 2) such that any DFA accepting $\mathcal{L}(A) \cup \mathcal{L}(B)$ needs at least $mn + m + n$ states.*

Proof. Consider the pairs (i, j) which represents the states of a DFA C, constructed with the previous algorithm, accepting $\mathcal{L}(A) \cup \mathcal{L}(B)$. Then for each of these pairs (i, j) where $i \in \{0, \ldots, m-1, -1\}$ and $j \in \{0, \ldots, n-1, -1\}$ except the case when both, i and j, are -1, there exists a word

$$w = \begin{cases} (b^{m-1}a)^j b^i & if\ i \neq -1 \wedge j \neq -1 \\ (b^{m-1}a)^n b^i & if\ i \neq -1 \wedge j = -1 \\ b^m a^j & if\ i = -1 \wedge j \neq -1 \end{cases}$$

which represents each state of C. Thus there are at least $mn + m + n$ distinct left quotients (states). $\square$

Proposition 7. *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 (Fig. 1 and Fig. 2) such that any DFA accepting $\mathcal{L}(A) \cup \mathcal{L}(B)$ has, at least, $(r + 1)(s + 1)$ transitions.*

Proof. Consider the DFA C constructed by the previous algorithm, such that $\mathcal{L}(C) = \mathcal{L}(A) \cup \mathcal{L}(B)$. This DFA is minimal and has $mn + m + n$ states (Proposition 6). If we name the defined and undefined transitions of the DFA A and B as in the case 2 of the proof of the Proposition 3 then 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 $(\bar{\alpha}_i, \beta_j)$.
- $mn + m + n - 1$ 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 $(\bar{\alpha}_i, \beta_j)$.

Consequently,

$$\begin{aligned} & mn + n - m + 1 + mn + m + n - 1 \\ = \ & 2mn + 2n \\ = \ & 2(mn + n) \end{aligned}$$

Thus, the DFA C has $2(mn + n)$ transitions.
As $r = m$ and $s = 2n - 1 \Leftrightarrow n = \frac{s+1}{2}$. Therefore,

$$\begin{aligned} & 2(mn + n) \\ = \ & 2(r.\frac{s+1}{2} + \frac{s+1}{2}) \\ = \ & r.\frac{s+1}{2} + r + s + 1 \\ = \ & (r + 1)(s + 1) \end{aligned}$$

Thus, the DFA C has $(r+1)(s+1)$ transitions.

$\square$

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 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.*

3.2.2 Case 2: $\mathbf{m = 1}$ and $\mathbf{n \geq 2}$.

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.

Proposition 8. *For any integer $n \geq 2$, exists an 1-state DFA A (defined above) and an n-state DFA B (Fig. 2) such that any DFA accepting $\mathcal{L}(A) \cup \mathcal{L}(B)$ needs at least $2n + 1$ states.*

Proof. Consider the pairs (i, j) which represents the states of a DFA C, constructed by the previous algorithm, accepting $\mathcal{L}(A) \cup \mathcal{L}(B)$. Then for each of these pairs (i, j) where $i \in \{0, -1\}$ and $j \in \{0, \ldots, n-1, -1\}$ except the case when both, i and j, are -1, there exists a word

$$w = \begin{cases} a^j & \text{if } i \neq -1 \wedge j \neq -1 \\ ba^j & \text{if } i = -1 \wedge j \neq -1 \\ a^n & \text{if } i \neq -1 \wedge j = -1 \end{cases}$$

which represents each state of C. Thus there are at least $2n + 1$ distinct states. Note that there is only one pair such that $i \neq -1 \wedge j = -1$.

$\square$

Proposition 9. *For any integer $n \geq 2$, exists an 1-state DFA A (defined above) with one transition and an n-state DFA B (Fig. 2) with $s = 2n - 1$ transitions such that any DFA accepting $\mathcal{L}(A) \cup \mathcal{L}(B)$ has, at least, $2(s+1)$ transitions.*

Proof. Consider the DFA C, constructed with the previous algorithm, such that $\mathcal{L}(C) = \mathcal{L}(A) \cup \mathcal{L}(B)$. If we name the defined and undefined transitions of the DFA A and B as in the case 2 of the proof of the Proposition 3 then the DFA C has:

- $2n$ a-transitions because there exist $n-1$ a-transitions of the form (α_i, β_j) $i = 1$ and $1 \leq j \leq n-1$, where α_i represents the a-transitions of DFA A and β_j represents the a-transitions of the DFA B; 2 a-transition of the form $(\alpha_i, \bar{\beta}_j)$ $1 \leq j \leq 2$ where $\bar{\beta}_j$ represents the undefined a-transitions of the DFA B; and $n-1$ a-transitions of the form $(\bar{\alpha}_i, \beta_j)$ $i = 1$ where $\bar{\alpha}_i$ represents the a-undefined transition of the DFA A.

- $2n$ b-transitions because by this symbol only exist transitions of the form $(\bar{\alpha}, \beta_j)$ $1 \leq j \leq n$ where $\bar{\alpha}$ represents the undefined b-transition of the DFA A and β_j represents the b-transitions of the DFA B.

Consequently,

$$2n + 2n = 4n$$

Thus, the DFA C has $4n$ transitions. As $r = 1$ and $s = 2n - 1 \Leftrightarrow n = \frac{s+1}{2}$. Therefore,

$$\begin{aligned} & 4n \\ = \; & 4(\frac{s+1}{2}) \\ = \; & 2(s+1) \end{aligned}$$

Thus, the DFA D has $2(s+1)$ transitions. Note that $r = 1$, thus $2(s+1) = (r+1)(s+1)$. $\square$

$\square$

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.*

3.2.3 Example

The automata 3 and 4 are an example of the worst-case family for $n = 2$ and $m = 2$:

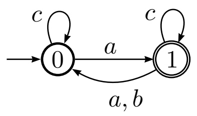

Figure 3: Union: Example of DFA A with m=2

This DFA A is a-complete and c-complete ($t(a, A) = 2$, $t(c, A) = 2$); and it is b-incomplete ($t(b, A) = 1$).

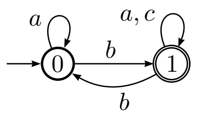

Figure 4: Union: Example of DFA B with n=2

This DFA B is a-complete and b-complete ($t(a, B) = 2$, $t(b, B) = 2$); and it is c-incomplete ($t(c, B) = 1$).

Consider the DFA C resulting of the union operation. If we calculate the number of τ-transitions ($\tau \in \Sigma$) using the above formula (2), we obtain:
$t(a, C) = 2 * 2 + 2 * 1 + 1 * 2 = 8$
$t(b, C) = 1 * 2 + 1 * 1 + 2 * 2 = 7$
$t(c, C) = 2 * 1 + 2 * 2 + 1 * 1 = 7$
Thus, C has 22 transitions.
We can also use the Proposition 3 and we obtain

$$\begin{aligned} itc(\mathcal{L}(C)) \; \leq \; & 5(1+2) + 5(1+2) - (2.2 + 1.2 + 2.1) \\ & 5.3 + 5.3 - (4 + 2 + 2) \\ & 22 \end{aligned}$$

The diagram 5 illustrates the DFA C resulting from the union operation:

This DFA C is a-complete ($t(a, C) = 8$); it is b-incomplete and c-incomplete - ($t(b, C) = 7$, $t(c, C) = 7$). As C is minimal we can say that $itc(\mathcal{L}(C)) = 22$.

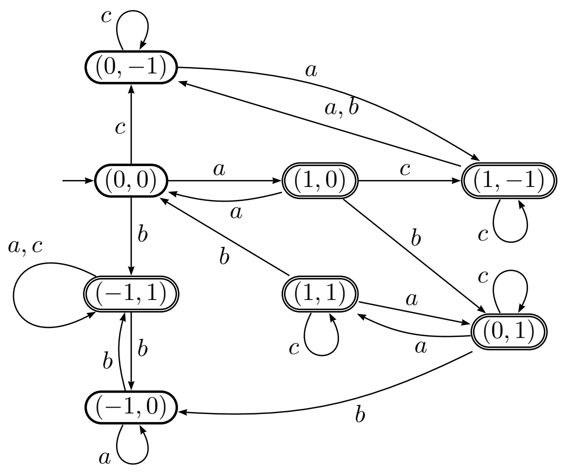

Figure 5: Union: DFA C resulting from the union operation

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.

Lemma 10. *The extended transition function for the DFA C ($\hat{\delta}_C$), resulting from the algorithm above, is defined in the follwing way:*

$$\begin{aligned}
\hat{\delta}_C(\langle q, T\rangle, \epsilon) &= \epsilon \\
\hat{\delta}_C(\langle q, T\rangle, w) &= \begin{cases} \langle \hat{\delta}_A(q, w), \hat{\delta}_B(T, w) \cup \{p_0\}\rangle & if\ \hat{\delta}_A(q, w) \downarrow\ \wedge\ \hat{\delta}_A(q, w) \in F_A \\ \langle \hat{\delta}_A(q, w), \hat{\delta}_B(T, w)\rangle & if\ \hat{\delta}_A(q, w) \downarrow\ \wedge\ \hat{\delta}_A(q, w) \notin F_A \\ \langle -1, \hat{\delta}_B(T, w)\rangle & if\ \hat{\delta}_A(q, w) \uparrow\ \wedge\ \hat{\delta}_B(T, w) \neq \emptyset \\ \uparrow & otherwise \end{cases}
\end{aligned}$$

Proof. If $w = \epsilon$ is trivial that $\hat{\delta}_C(\langle q, T\rangle, \epsilon) = \epsilon$. Let us prove by induction on the size of word w. If $w = \sigma$, $\sigma \in \Sigma$ we have that:

$$\begin{aligned}
\hat{\delta}_C(\langle q, T\rangle, \sigma) &= \delta_C(\langle q, T\rangle, \sigma) \\
&= \begin{cases} \langle \delta_A(q, \sigma), \delta_B(T, \sigma) \cup \{p_0\}\rangle & if\ \delta_A(q, \sigma) \downarrow\ \wedge\ \delta_A(q, \sigma) \in F_A \\ \langle \delta_A(q, \sigma), \delta_B(T, \sigma)\rangle & if\ \delta_A(q, \sigma) \downarrow\ \wedge\ \delta_A(q, \sigma) \notin F_A \\ \langle -1, \delta_B(T, \sigma)\rangle & if\ \delta_A(q, \sigma) \uparrow\ \wedge\ \delta_B(T, \sigma) \neq \emptyset \\ \uparrow & otherwise \end{cases}
\end{aligned}$$

Assume that $\hat{\delta}_C(\langle q,T\rangle, w)$, $w \neq \epsilon$, is defined as above and let us prove that the same happens for $\hat{\delta}_C(\langle q,T\rangle, w\sigma)$.

$$\begin{array}{rcl}
\hat{\delta}_C(\langle q,T\rangle, w\sigma) & = & \delta_C(\hat{\delta}_C(\langle q,T\rangle, w), \sigma) \\
& = & \begin{cases} \delta_C(\langle \hat{\delta}_A(q,w), \hat{\delta}_B(T,w) \cup \{p_0\}\rangle, \sigma) & if\ \hat{\delta}_A(q,w) \downarrow\ \wedge\ \hat{\delta}_A(q,w) \in F_A \\ \delta_C(\langle \hat{\delta}_A(q,w), \hat{\delta}_B(T,w)\rangle, \sigma) & if\ \hat{\delta}_A(q,w) \downarrow\ \wedge\ \hat{\delta}_A(q,w) \notin F_A \\ \delta_C(\langle -1, \hat{\delta}_B(T,w)\rangle, \sigma) & if\ \hat{\delta}_A(q,w) \uparrow\ \wedge\ \hat{\delta}_B(T,w) \neq \emptyset \\ \uparrow & otherwise \end{cases} \\
& = & \begin{cases} \langle \delta_A(\hat{\delta}_A(q,w),\sigma), \delta_B(\hat{\delta}_B(T,w) \cup \{p_0\}, \sigma) \cup \{p_0\}\rangle & if\ q' \downarrow\ \wedge\ q' \in F_A \\ \langle \delta_A(\hat{\delta}_A(q,w),\sigma), \delta_B(\hat{\delta}_B(T,w), \sigma)\rangle & if\ q' \downarrow\ \wedge\ q' \notin F_A \\ \langle \delta_A(-1,\sigma), \delta_B(\hat{\delta}_B(T,w), \sigma)\rangle & if\ q' \uparrow\ \wedge\ T' \neq \emptyset \\ \uparrow & otherwise \end{cases} \\
& & \text{where } q' = \delta_A(\hat{\delta}_A(q,w),\sigma) \text{ and } T' = \delta_B(\hat{\delta}_B(T,w),\sigma). \\
& = & \begin{cases} \langle \hat{\delta}_A(q,w\sigma), \hat{\delta}_B(T,w\sigma) \cup \delta_B(\{p_0\}) \cup \{p_0\}\rangle & if\ q'' \downarrow\ \wedge\ q'' \in F_A \\ \langle \hat{\delta}_A(q,w\sigma), \hat{\delta}_B(T,w\sigma)\rangle & if\ q'' \downarrow\ \wedge\ q'' \notin F_A \\ \langle -1, \hat{\delta}_B(T,w\sigma)\rangle & if\ q'' \uparrow\ \wedge\ T'' \neq \emptyset \\ \uparrow & otherwise \end{cases} \\
& & \text{where } q'' = \hat{\delta}_A(q,w\sigma) \text{ and } \hat{\delta}_B(T,w\sigma).
\end{array}$$

Thus, the lemma holds. □

Proposition 11. *DFA C recognizes the language $\mathcal{L}(A)\mathcal{L}(B)$.*

Proof. We need to prove that if a word pertains to $\mathcal{L}(A)\mathcal{L}(B)$ then the DFA C accepts the word.

Consider $w \in \mathcal{L}(A)\mathcal{L}(B)$ such that $w = w'w''$, where $w' \in \mathcal{L}(A)$ and $w'' \in \mathcal{L}(B)$. Thus,

$$\begin{array}{rcl}
\hat{\delta}_C(r_0', w) & = & \hat{\delta}_C(r_0, w'w'') = \\
& = & \hat{\delta}_C(\langle q_0, T\rangle, w'w'') = \\
& = & \hat{\delta}_C(\hat{\delta}_C(\langle q_0, T\rangle, w'), w'') = \\
& = & \hat{\delta}_C(\langle q', T'\rangle, w'') = \\
& & \text{where } p_0 \in T' \text{because } q' \in F_A, \text{ by assumption.} \\
& & \text{As we know that } p_0 \in T' \text{ and } w'' \in \mathcal{L}(B) \text{ we have that} \\
& = & \langle q'', T''\rangle \\
& & \text{where } p_f \in T'' \text{ and } p_f \in F_B.
\end{array}$$

Therefore, the word is accepted by the DFA C.

Let us prove that if the DFA C accepts a word then the word pertains to $\mathcal{L}(A)\mathcal{L}(B)$. If a word is accepted by the DFA C then exists a state $\langle q,T\rangle \in R$ such that $T \cap F_B \neq \emptyset$, which implies that $T \neq \emptyset$. As $T \neq \emptyset$ we know that exists a state $\langle q',T'\rangle \in R$ and $w' \in w$ such that $\hat{\delta}_C(r_0, w') = \langle q', T'\rangle$ where $q' \in F_A$ and $p_0 \in T'$ — $w' \in \mathcal{L}(A)$. Thus, if $w = w'w''$, $\hat{\delta}_C(\langle q',T'\rangle, w'') = \langle q,T\rangle$. As $T \cap F_B \neq \emptyset$, thus $w'' \in \mathcal{L}(B)$. Therefore, $w \in \mathcal{L}(A)\mathcal{L}(B)$. □

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

Proposition 12. *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 $\mathcal{L}(A)\mathcal{L}(B)$.*

Proof. R is a set of pairs (α, β) where α is a state in Q or -1; and the β is a subset of P. R does not contain the pairs in which α is a final state of A and the β does not contain the initial state of B. Thus,

(1) $(m+1)2^n$ corresponds to the set of pairs such that the first component of each pair is a state in Q or -1; and the second component is a subset of P.

(2) $f(A)2^{n-1}$ corresponds to the pairs in which the first component is a final state of A and the second component does not contain the initial state of B. This pairs are not in R but they are counted in (1) and because of this we need to remove them.

(3) 1 corresponds to the pair $(-1, \emptyset)$, which is not in R and because of this we also remove it.

Therefore, the number of states is:

$$(m+1)2^n - f(A)2^{n-1} - 1$$

Note that we don't need to take in account the dead state of the DFA B, because it corresponds to the empty set in the second component of the pair which is not relevant in R. ☐

Corollary 1. *The formula in Proposition 12 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 13. *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*

$$itc(L_1L_2) \leq |\Sigma|(m+1)2^n - |\Sigma_c^{L_2}|(f2^{n-1}+1) - \sum_{\tau \in \Sigma_i^{L_2}} (2^{u_\tau} + f2^{itc_\tau(L_2)}) - \\ - \sum_{\tau \in \Sigma_{ii}} \bar{u}_\tau 2^{u_\tau} - \sum_{\tau \in \Sigma_{ic}} \bar{u}_\tau.$$

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 case 2 of the proof of the Proposition 3 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. In conclusion,

$$\begin{cases} (m+1)2^n - 2^{u(\tau,B)} - f(A)2^{t(\tau,B)} & A\ \tau\text{-complete} \wedge B\ \tau\text{-incomplete} \\ (m+1)2^n - 1 - f(A)2^{n-1} & A\ \tau\text{-complete} \wedge B\ \tau\ \text{-complete} \\ (m+1)2^n - (\bar{u}(\tau,A)+1)2^{u(\tau,B)} - f(A)2^{t(\tau,B)} & A\ \tau\text{-incomplete} \wedge B\ \tau\text{-incomplete} \\ (m+1)2^n - (\bar{u}(\tau,A)+1) - f(A)2^{n-1} & A\ \tau\text{-incomplete} \wedge B\ \tau\text{-complete} \end{cases} \tag{3}$$

Thus the incomplete transition complexity is:

$$\begin{aligned} itc(L_1L_2) \quad \leq \quad & \sum_{\tau\in\Sigma_{ci}} (m+1)2^n - 2^{u_\tau} - f2^{itc_\tau(L_2)} + \sum_{\tau\in\Sigma_{cc}} (m+1)2^n - 1 - f2^{n-1} + \sum_{\tau\in\Sigma_{ii}} \\ & (m+1)2^n - (\bar{u}_\tau+1)2^{u_\tau} - f2^{itc_\tau(L_2)} + \sum_{\tau\in\Sigma_{ic}} (m+1)2^n - (\bar{u}_\tau+1) - f2^{n-1} \end{aligned}$$

If we simplify this inequality we obtain:

$$\begin{aligned} itc(L_1L_2) \quad \leq \quad & |\Sigma_{ci}|(m+1)2^n - \sum_{\tau\in\Sigma_{ci}} (2^{u_\tau} + f2^{itc_\tau(L_2)}) + |\Sigma_{cc}|((m+1)2^n - (1+f2^{n-1})) + |\Sigma_{ii}|(m+1)2^n \\ & - \sum_{\tau\in\Sigma_{ii}} ((\bar{u}_\tau+1)2^{u_\tau} + f2^{itc_\tau(L_2)}) + |\Sigma_{ic}|((m+1)2^n - f2^{n-1} - 1) - \sum_{\tau\in\Sigma_{ic}} \bar{u}_\tau \\ itc(L_1L_2) \quad \leq \quad & |\Sigma|(m+1)2^n - |\Sigma_{ic}|(f2^{n-1}+1) - \sum_{\tau\in\Sigma_{ci}} (2^{u_\tau} + f2^{itc_\tau(L_2)}) - \\ & |\Sigma_{cc}|(1+f2^{n-1}) - \sum_{\tau\in\Sigma_{ii}} ((\bar{u}_\tau+1)2^{u_\tau} + f2^{itc_\tau(L_2)}) - \sum_{\tau\in\Sigma_{ic}} \bar{u}_\tau \\ itc(L_1L_2) \quad \leq \quad & |\Sigma|(m+1)2^n - |\Sigma_c^{L_2}|(f2^{n-1}+1) - \sum_{\tau\in\Sigma_i^{L_2}} (2^{u_\tau} + f2^{itc_\tau(L_2)}) - \\ & - \sum_{\tau\in\Sigma_{ii}} \bar{u}_\tau 2^{u_\tau} - \sum_{\tau\in\Sigma_{ic}} \bar{u}_\tau. \end{aligned}$$

Thus the inequality in the proposition holds.

□

4.2 Worst-case Witnesses

The following results show that the complexity upper bounds found in Propositions 12 and 13 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.

4.2.1 Case 1: $m \geq 2$ and $n \geq 2$.

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. 6).

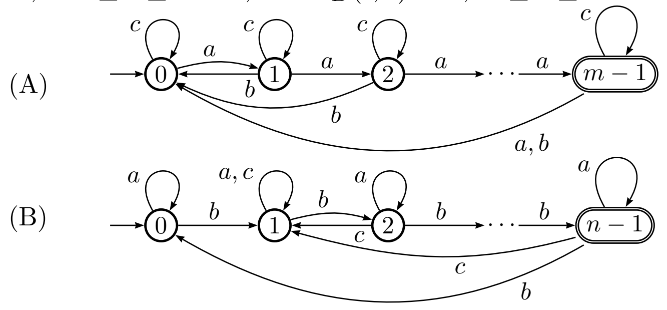

Figure 6: DFA A with m states and DFA B with n states.

Proposition 14. *DFA A is minimal.*

Proof. Consider $x, y \in \Sigma^\star$ such that $\hat{\delta}(0, x) = i$ and $\hat{\delta}(0, y) = j$, , $i \neq j$. It is clear that $xa^{m-1-i} \in \mathcal{L}(A)$ but $ya^{m-1-i} \notin \mathcal{L}(A)$. Then x and y are in different left quotients induced by $\mathcal{L}(A)$. □

Proposition 15. *DFA B is minimal.*

Proof. Consider $x, y \in \Sigma^\star$ such that $\hat{\delta}(0, x) = i$ and $\hat{\delta}(0, y) = j$, , $i \neq j$. It is clear that $xb^{n-1-i} \in \mathcal{L}(B)$ but $yb^{n-1-i} \notin \mathcal{L}(B)$. Then x and y are in different different left quotients induced by $\mathcal{L}(B)$. □

Proposition 16. *For any integers $m \geq 2$ and $n \geq 2$, exist an m-state DFA A and an n-state DFA B such that any DFA accepting $\mathcal{L}(A)\mathcal{L}(B)$ needs at least $(m+1)2^n - 2^{n-1} - 1$ states.*

Proof. (The proof follows the proof of the theorem 2.1 of [18]) Consider the DFA A and B which are shown in Fig. 6. We can verify that
$\mathcal{L}(A) = \{xy \mid x \in (c^\star a^\star \{b\})^\star, y \in \{a, c\}$ and $\#_a(y) = m-1 \bmod m\}$
and
$\mathcal{L}(B) \cap \{a, b\}^\star = \{x \in \{a, b\}^\star \mid \#_b(x) = (n-1) \bmod n\}$
Let us consider the concatenation of $\mathcal{L}(A)$ and $\mathcal{L}(B)$, *i.e.*, $\mathcal{L}(A)\mathcal{L}(B)$.

Fact. For $m \geq 2$, $\mathcal{L}(A) \cap \Sigma^\star\{b\} = \emptyset$, *i.e.* we never have a b at the end of a word which is accepted by the DFA A.

From each $x \in \{a, b\}^\star$, we define

$$S(x) = \{i \mid x = uv \text{ such that } u \in \mathcal{L}(A) \textit{ and } i = \#_b(v) \bmod n\}$$

Consider $x, y \in \{a, b\}^\star$ such that $S(x) \neq S(y)$. Let $k \in S(x) - S(y)$ (or $S(y) - S(x)$). Then it is clear that $xb^{n-1-k} \in \mathcal{L}(A)\mathcal{L}(B)$ but $yb^{n-1-k} \notin \mathcal{L}(A)\mathcal{L}(B)$. Thus x and y are in different left quotients induced by $\mathcal{L}(A)\mathcal{L}(B)$.

For each $x \in \{a, b\}^\star$, define $T(x) = max\{|z| | \ x = yz \textit{ and } z \in a^\star\}$, *i.e.*, $T(x)$ is the highest number of a's at the end of a word x. Consider $\alpha, \beta \in \{a, b\}^\star$ such that $S(\alpha) = S(\beta)$ and $T(\alpha) > T(\beta) \bmod m$. Let $i = T(\alpha) \bmod m$ and $w = a^{m-1-i}b^{n-1}$. We want to prove that

$\alpha w \in \mathcal{L}(A)\mathcal{L}(B)$. We know that α ends with i a's and $i \geq 1$. Thus, to reach the final state of the DFA A we only need to have more $m-1-i$ a's. The remaining word consists in $n-1$ b's, so it ends up in the final state of the DFA B. Therefore $\alpha w \in \mathcal{L}(A)\mathcal{L}(B)$. $\beta w \notin \mathcal{L}(A)\mathcal{L}(B)$ because it have at least less one a than αw.

Notice that there does not exist a word $w \in \Sigma^\star$ such that $0 \notin S(w)$ and $T(w) = m-1$, since the fact that $T(w) = m-1$ guarantees that $0 \notin S(w)$ because $S(w) = \{0 \mid w = uv \textit{ such that } u \in \mathcal{L}(A) \textit{ and } i = \#_b(v) \mod n\}$, *i.e.*, if w have $m-1$ a's at the end then it not has b's thus i=0.

For each subset $s = \{i_1, \ldots, i_t\}$ of $\{0, \ldots, n-1\}$, where $i_1 > \cdots > i_t$, and an integer $j \in \{0, \ldots, m-1, -1\}$ except the case when both $0 \in s$ and $j = m-1$ are true and the case when both $s = \emptyset$ and $j = -1$ are true, there exists a word

$$x = \begin{cases} a^{m-1}b^{i_1} \cdots a^{m-1}b^{i_t}a^j & \text{if } j \neq -1 \\ a^{m-1}b^{i_1} \cdots a^{m-1}b^{i_t}b^n & \text{if } j = -1 \end{cases}$$

such that $S(x) = s$ and $T(x) = j$. Thus, there are at least $(m+1)2^n - 2^{n-1} - 1$ distinct left quotients. □

Proposition 17. *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 (Fig. 6) such that any DFA accepting $\mathcal{L}(A)\mathcal{L}(B)$ has, at least, $(r+1)2^{\frac{s+1}{3}} + 3.2^{\frac{s-2}{3}} - 5$ transitions.*

Proof. Consider the DFA C such that $\mathcal{L}(C) = \mathcal{L}(A)\mathcal{L}(B)$, and, as in Proposition 13, 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.
- $(m+1)2^n - 2^{n-1} - 2$, c-transitions. This is analogous to the previous cases.

Thus,

$$\begin{aligned} & (m+1)2^n - 2^{n-1} - 1 + (m+1)2^n - 2^{n-1} - 2 + (m+1)2^n - 2^{n-1} - 2 \\ = \; & 3m2^n + 3.2^n - 3.2^{n-1} - 5 \\ = \; & 3(m2^n + 2^n - 2^{n-1}) - 5 \\ = \; & 3(m2^n + 2^{n-1}) - 5 \end{aligned}$$

Therefore the DFA C has $3(m2^n + 2^{n-1}) - 5$ transitions. As $r = 3m-1 \Leftrightarrow m = \frac{r+1}{3}$ and $s = 3n-1 \Leftrightarrow n = \frac{s+1}{3}$. Thus,

$$\begin{aligned} & 3.m2^n + 3.2^{n-1} - 5 \\ = \; & 3(\frac{r+1}{3})2^{\frac{s+1}{3}} + 3.2^{\frac{s+1}{3}-1} - 5 \\ = \; & (r+1)2^{\frac{s+1}{3}} + 3.2^{\frac{s-2}{3}} - 5 \end{aligned}$$

Therefore the DFA C has $(r+1)2^{\frac{s+1}{3}} + 3.2^{\frac{s-2}{3}} - 5$ transitions. □

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.*

4.2.2 Case 2: $m = 1$ and $n \geq 2$.

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 $P = \{0, \dots, n-1\}$, $F_B = \{n-1\}$, $\delta_B(i, a) = i$ 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. 7).

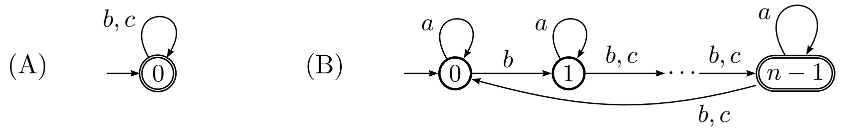

Figure 7: DFA A with 1 state and DFA B with n states.

Proposition 18. *DFA A is minimal.*

Proposition 19. *DFA B is minimal.*

Proof. Consider $x, y \in \Sigma^\star$ such that $\hat{\delta}(0, x) = i$ and $\hat{\delta}(0, y) = j$, $i \neq j$. It is clear that $xb^{n-1-i} \in \mathcal{L}(B)$ but $yb^{n-1-i} \notin \mathcal{L}(B)$. Then x and y are in different left quotients induced by $\mathcal{L}(B)$. □

Proposition 20. *For any integer $n \geq 2$, exist a 1-state DFA A and an n-state DFA B such that any DFA accepting $\mathcal{L}(A)\mathcal{L}(B)$ needs at least $2^{n+1} - 2^{n-1} - 1$ states.*

Proof. Consider the DFA $C = (R, \Sigma, \delta, 0, F)$, constructed by the previous algorithm, such that $\mathcal{L}(C) = \mathcal{L}(A)\mathcal{L}(B)$. The DFA A and the DFA B are defined in the beginning of this section. 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 $s \in \{-1, 0\}$, $c = \{i_1, \dots, i_k\}$, $1 \leq k \leq n$, 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 $s = 0$, $i_1 = 0$.

Let f be a final state of the form (s, c), where $c = \{i_1, \dots, i_{k-1}, n-1\}$ and $\bar{c} = P \setminus c$. Let us construct a word w of size n, such that $\delta(0, w) = f$. We will count the positions (starting with zero) of the word w from the last to the first. If f has $s = -1$, w has an a in the position i_1; c's in the positions $j \in \bar{c} \setminus \{i_1 - 1\}$ if $i_1 \neq 0$ or $j \in \bar{c}$ otherwise; all the other positions are b's. For example, if $n = 5$, $c = \{4\}$ and $\bar{c} = \{0, 1, 2, 3\}$ then $w = abccc$. If f has $s = 0$ the word has c's in all positions $i_j - 1$, $1 \leq j \leq k - 1$ for all $i_j \in \bar{c}$; all the other positions are $b's$. For example, if $c = \{0, 4\}$, $\bar{c} = \{1, 2, 3\}$ and $n = 5$ then $w = bbccc$. Now, consider the non-final states p which have the same form (s, c), but $i_k \neq n - 1$ and $\bar{c} = \{0, \dots, n-2\} \setminus c$. The word w for these two types of non-final states is constructed with the same rules described above for final states.

It was proved that all states are reachable from initial state. Now let us prove that all states are pairwise distinguishable. Final states are trivially distinguishable from non-final states. We need to prove that states of the same kind are distinguishable. Consider $x, y \in \Sigma^\star$ such that $\hat{\delta}(0, x) = q$ and $\hat{\delta}(0, y) = p$, $q \neq p$. Suppose that q and p are final. There are three cases to consider. Let $q = (0, \{0, i_2, \dots, i_k, n-1\})$ and $p = (0, \{0, j_2, \dots, j_{k'}, n-1\})$. Suppose $k \geq k'$ and $i \in \{0, i_2, \dots, i_k, n-1\} \setminus \{0, j_2, \dots, j_{k'}, n-1\}$. Then $xc^{n-1-i} \in \mathcal{L}(C)$ but $yc^{n-1-i} \notin \mathcal{L}(C)$. If $q = (-1, \{i_1, \dots, i_k, n-1\})$ and $p = (-1, \{j_1, \dots, j_{k'}, n-1\})$, we can take

i as before and then $xb^{n-1-i} \in \mathcal{L}(C)$ but $yb^{n-1-i} \notin \mathcal{L}(C)$. If $q = (0, \{0, i_2, \ldots, i_k, n-1\})$ and $p = (-1, \{j_1, \ldots, j_{k'}, n-1\})$, then $xc^nb^{n-1} \in \mathcal{L}(C)$ but $yc^nb^{n-1} \notin \mathcal{L}(C)$. Now suppose that q and p are non-final. Let $q = (0, \{0, i_2, \ldots, i_k\})$ and $p = (0, \{0, j_2, \ldots, j_{k'}\})$. Consider, without lost of generality, $k \geq k'$ and $i \in \{0, i_2, \ldots, i_k\} \setminus \{0, j_2, \ldots, j_{k'}\}$. It is clear that $xc^{n-1-i} \in \mathcal{L}(C)$ but $yc^{n-1-i} \notin \mathcal{L}(C)$. If $q = (-1, \{i_1, \ldots, i_k\})$ and $p = (-1, \{j_1, \ldots, j_{k'}\})$, we can take $i \in \{i_1, \ldots, i_k\} \setminus \{j_1, \ldots, j_{k'}\}$ and then $xb^{n-1-i} \in \mathcal{L}(C)$ but $yb^{n-1-i} \notin \mathcal{L}(C')$. Finally, if $q = (0, \{0, i_2, \ldots, i_k\})$ and $p = (-1, \{j_1, \ldots, j_{k'}\})$, clearly $xc^nb^{n-1} \in \mathcal{L}(C)$ but $yc^nb^{n-1} \notin \mathcal{L}(C)$. Thus C is minimal and has $2^{n+1} - 2^{n-1} - 1$ states.

Thus we have $2^{n-2} + 2^{n-1}$ final states and $2^{n-2} + 2^{n-1} - 1$ non-final states. Then

$$\begin{aligned}
2^{n-2} + 2^{n-1} + 2^{n-2} + 2^{n-1} - 1 &= 2^n + 2^{n-1} - 1 \\
&= 2^{n-1}(2+1) - 1 \\
&= 2^{n-1}(2^2 - 1) - 1 \\
&= 2^{n+1} - 2^{n-1} - 1
\end{aligned}$$

Therefore, a DFA accepting $\mathcal{L}(A)\mathcal{L}(B)$ needs at least $2^{n+1} - 2^{n-1} - 1$ states. □

Proposition 21. *For any integer $n \geq 2$, exist a 1-state DFA A (defined above) with 2 transitions and an n-state DFA (B Fig. 7). with $s = 3n - 1$ transitions such that any DFA accepting $\mathcal{L}(A)\mathcal{L}(B)$ has, at least, $3(2^{\frac{s+4}{3}} - 2^{\frac{s-2}{3}}) - 4$ transitions.*

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.*

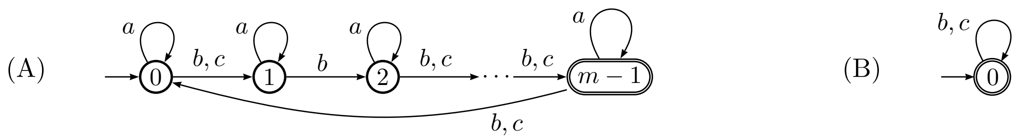

Figure 8: DFA A with m states and DFA B with 1 state.

4.2.3 Case 3: $m \geq 2$ and $n = 1$.

Define $A = (P, \Sigma, \delta_A, 0, F_A)$ 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. 8).

Proposition 22. *DFA A is minimal.*

Proof. Consider $x, y \in \Sigma^\star$ such that $\hat{\delta}(0, x) = i$ and $\hat{\delta}(0, y) = j$, $i \neq j$. $xb^{n-1-i} \in \mathcal{L}(A)$ but $yb^{n-1-i} \notin \mathcal{L}(A)$. Then x and y are in different left quotients induced by $\mathcal{L}(A)$. □

Proposition 23. *DFA B is minimal.*

Proposition 24. *For any integers $m \geq 2$, there exists an m-state DFA A and a 1-state DFA B such that any DFA accepting $\mathcal{L}(A)\mathcal{L}(B)$ needs at least $2m$ states.*

Proof. Consider the DFA $C = (Q, \Sigma, \delta, 0, F)$, constructed with the previous algorithm, such that $\mathcal{L}(C) = \mathcal{L}(A)\mathcal{L}(B)$. The DFA A is represented in Fig. 8. By construction we know that C have two kinds of p states:

- final states which are of the form $(\alpha, \{0\})$ where $\alpha \in \{0, \ldots, m-1\}$ or $(-1, \{0\})$.
- non-final states which are of the form $(\alpha, \emptyset)$ where $\alpha \in \{0, \ldots, m-2\}$.

For any state p we can find a word w for which $\delta(0, w) = p$. If p is a final state of the form $(\alpha, \{0\})$ where $\alpha \in \{0, \ldots, m-1\}$ then $w = b^{m+x}$. In case p has the form $(-1, \{0\})$ then $w = b^{m+1}c$. Finally, if p is a non-final state then $w = b^x$.

Let us prove that the final states are distinguishable:

- The states of the form $(\alpha, \{0\})$ where $\alpha \in \{0, \ldots, m-1\}$ are not equivalent because they correspond to the states of the DFA E which is minimal.x
- The state $(-1, \{0\})$ is not equivalent to the other final states because it is the only state which is τ-incomplete.

Now, consider the non-final states $(i, \{0\}), (j, \{0\})$ and $x, y \in \Sigma^\star$ such that $\hat{\delta}(0, \alpha) = (i, \{0\})$ and $\hat{\delta}(0, \beta) = (j, \{0\})$. It is clear that $xa^{i+1}b^{m-1-i}a^{i+1} \in \mathcal{L}(A)\mathcal{L}(B)$ but $ya^{i+1}b^{m-1-i}a^{i+1} \notin \mathcal{L}(A)\mathcal{L}(B)$. Then x and y are in different left quotients induced by $\mathcal{L}(A)\mathcal{L}(B)$.

Thus, a DFA accepting $\mathcal{L}(A)\mathcal{L}(B)$ needs at least $2m$ states. □

Proposition 25. *For any integer $m \geq 2$ exists an m-state DFA A (Fig. 8). 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 $2r$ transitions.*

Proof. Similar to the proof of the Proposition 3. □

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.*

4.2.4 Why |Σ|=3?

Let us show with an example why with a two letter alphabet we cannot build a DFA which state complexity reaches the upper bound.

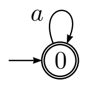

Figure 9: Concatenation: Example of DFA A with $m = 1$ and $|k| = 2$

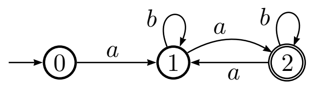

Figure 10: Concatenation: Example of DFA A with n=3 and —k—=2

The states of the DFA resulting of the concatenation operation are:

0. $(0, \{0\})$
1. $(0, \{0, 1\})$
2. $(0, \{0, 1, 2\})$

3. $(-1, \{1\})$
4. $(-1, \{1, 2\})$
5. $(-1, \{2\})$

The pairs considered in the upper bound and which do not appear are:

- $(0, \{0, 2\})$ because $\delta((0, \{1\}), \tau) = (0, \{0, 2\})$ but $(0, \{1\})$ can not appear because in the first component we have the final state of DFA A and because of this we need to have the initial state of DFA B in the second component.
- $(-1\{0, 1\})$, $(-1\{0, 2\})$, $(-1\{0\})$ and $(-1\{0, 1, 2\})$ can not appear because 0 can only appear in the second position of the pair if 0 is also in the first position of the pair because in the DFA B any transition goes to 0.

  Because of this we need one more letter which guarantees the pairs that are missing.

4.2.5 Example

The automata in Fig. 11 and Fig. 12 are an example from the worst-case family, with $n = 2$ and $m = 2$:

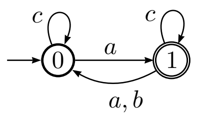

Figure 11: Concatenation: Example of DFA A with m=2

This DFA A is a-complete and c-complete ($t(a, A) = 2,\ t(c, A) = 2$); and it is b-incomplete ($t(b, A) = 1$).

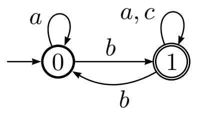

Figure 12: Concatenation: Example of DFA B with n=2

This DFA B is a-complete and b-complete ($t(a, B) = 2,\ t(b, B) = 2$); and it is c-incomplete ($t(c, B) = 1$). If we calculate the number of transitions using the above formula (3) , we obtain:

$$\begin{aligned}
t(a, C) =&\ (2+1) * 2^2 - 1 - 1 * 2^{2-1} \ = \ 3 * 4 - 1 - 2 \ = \ 12 - 3 \ = \ 9 \\
t(b, C) =&\ (1+1+1) * 2^2 - 2 - 1 * 2^{2-1} \ = \ 3 * 4 - 2 - 2 \ = \ 14 - 4 \ = \ 8 \\
t(c, C) =&\ (2+1) * 2^{1+1} - 2^1 - 1 * 2^1 \ = \ 3 * 2^2 - 2 - 2 \ = \ 14 - 4 \ = \ 8
\end{aligned}$$

Thus, C has 25 transitions.
We can also use the Proposition 13 and we obtain:

$$\begin{aligned}
itc(\mathcal{L}(C)) \ \leq &\ \ 3.3.2^2 - 2.(1.2^1 + 1) - (2^1 + 1.2^1) - 1 \\
&\ \ 36 - 11 \\
&\ \ 25
\end{aligned}$$

The diagram 13 illustrates the DFA C resulting from the concatenation operation:

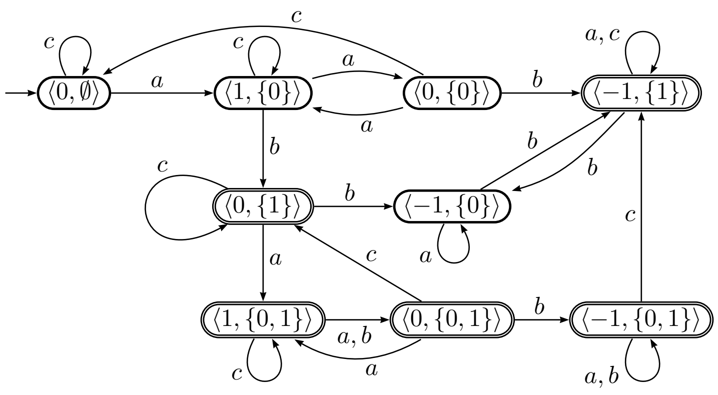

Figure 13: Concatenation: DFA C resulting from the concatenation operation

This DFA is a-complete ($t(a, C) = 9$); it is b-incomplete and c-incomplete - ($t(b, C) = 8$, $t(c, C) = 8$). Thus, it has 25 transitions. As C is minimal we can say that $itc(\mathcal{L}(C)) = 25$

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 $\mathcal{L}(A)^\star$.
We can verify that

Proposition 26. *DFA A' recognizes the language $\mathcal{L}(A)^\star$.*

Proof. Let w be a word of the language $\mathcal{L}(A)^\star$. We know that $w = \varepsilon$ or $\exists n \in \mathbb{N}\, \exists\, w_1, \ldots, w_n \in \mathcal{L}(A)$ such that $w = w_1 w_2 \cdots w_n$.

Let us prove by induction on the size of the word w — n. For $w \in \Sigma^\star$ we prove that if $w = w_1 \cdots w_n$ and $w_i \in (L(A))$ then $w \in \mathcal{L}(\mathcal{A}^\star)$

If $w = \varepsilon$ we have $\hat{\delta}'(q'_0, \varepsilon) = \{q'_0\}$ which is a final state of A'. Thus, the word is accepted by the DFA A'.

If n=1 then $w \in \mathcal{L}(A)$ and $\hat{\delta}'(q'_0, w) = \{\hat{\delta}(q_0, w), q_0\}$ because $\hat{\delta}(q_0, w) \in F$, and thus $\hat{\delta}'(q'_0, w) \in F'$.

Assume that $\forall 1 \leq m \leq n\ \forall w_1, \ldots, w_m \in \mathcal{L}(A) : w_1 \cdots w_m \in \mathcal{L}(A')$, and let us prove that $\forall w_1, \ldots, w_n \in \mathcal{L}(A) : w_1 \cdots w_n \in \mathcal{L}(A')$. We know that $\hat{\delta}'(q'_0, w_1) = \{\hat{\delta}(q_0, w_1), q_0\}$ because $\hat{\delta}(q_0, w_1) \in F$. Thus,

$$\begin{aligned} \hat{\delta}'(q'_0, w) &= \hat{\delta}'(q'_0, w_1 \cdots w_n) \\ &= \hat{\delta}'(\{\hat{\delta}(q_0, w_1), q_0\}, w_2 \cdots w_n) \\ &= \hat{\delta}'(\{\hat{\delta}(q_0, w_1)\}, w_2 \cdots w_n) \cup \hat{\delta}'(\{q_0\}, w_2 \cdots w_n) \in \mathcal{L}(A) \\ &\quad \text{by induction hypothesis.} \end{aligned}$$

We proved that if the word w pertains to the language $\mathcal{L}(A)^\star$ then the DFA A' accepts the word w.

We need to prove that all words accepted by A' are words pertaining to $\mathcal{L}(A)^\star$. A state of the DFA A' is final if and only if it includes a final state of A or it is q'_0. A word is accepted by the DFA A' if its recognizing process ends in a final state. Let *trace of execution* be the sequence of states for which a word w passes through during the recognizing process. Note that the trace of execution can include many final states. Let us proceed by induction on number (n) of final states in the trace of execution.

If $n = 1$ the process ends in the unique final state in the trace of execution, and thus the word is accepted by the automaton A.

Assume that if the trace of execution has $n-1$ final states then the word pertains to the language $\mathcal{L}(A')$, and let us prove that if the trace of execution has n final states in it, the word also pertains to the language. Let w be the word $w_1 \cdots w_{n-1} w_n$.

By induction hypothesis we know that $w_1 \cdots w_{n-1}$ pertains to $\mathcal{L}(A)^\star$ and we also know that w_n pertains to $\mathcal{L}(A)$. But then it also pertains to $\mathcal{L}(A)^\star$. Thus, w pertains to the language. □

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

Proposition 27. *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.*

Proof. Consider $A = (Q, \Sigma, \delta, q_0, F)$ and $A' = (Q', \Sigma, \delta', q'_0, F')$ constructed by the previous algorithm such that $\mathcal{L}(A') = (\mathcal{L}(A))^\star$. Note that Q' is defined as the union of 3 different sets. Thus, the number of states in Q' is:

$$2^{n-1} + 2^{n-l-1}$$

The states generated by the second set of Q' are the non-empty parts of Q disjoint from F_0. So in this set we have $2^{n-l} - 1$ states (we also remove the empty set).

The states in the third set of Q' are the parts of Q that contains q_0 and are non-disjoint from F_0. Those are at most $(2^l - 1)2^{n-l-1}$:

- $2^l - 1$ corresponds to the number of sets with only final states from which we exclude the empty set because 2^l takes it in account .
- 2^{n-l-1} corresponds to the number of sets with only non-final and non-initial states.

Therefore the number of states is:

$$\begin{aligned}
1 + 2^{n-l} - 1 + (2^l - 1)2^{n-l-1} &= 2^{n-l} + (2^l - 1)2^{n-l-1} \\
&= 2^{n-l} + 2^l 2^{n-l-1} - 2^{n-l-1} \\
&= 2^{n-l} + 2^{n-1} + 2^{n-l-1} \\
&= 2^{n-1} + 2^{n-l}(1 - 2^{-1}) \\
&= 2^{n-1} + 2^{n-l}2^{-1} \\
&= 2^{n-1} + 2^{n-l-1}
\end{aligned}$$

□

Corollary 2. *The formula in Proposition 27 is maximal when $l = 1$.*

Proposition 28. *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. To simplify the notation, we omit the A in the following measures $t(\tau, A)$, $\bar{u}(\tau, A)$, $u(\tau, A)$ and $i(\tau, A)$. Consider the DFA A' such that $\mathcal{L}(A') = (\mathcal{L}(A))^\star$ and A' is constructed using the algorithm below.

Following the analyze done to the states, the set of τ-transitions of A' is the disjoint union of:

1. the set of τ-transitions leaving the initial state of A, i_τ;
2. The set of transitions that exclude the τ-transitions leaving the final states of A:
   - if A is τ-complete, A' has $(2^{t(\tau)-l}) - 1$ τ-transitions of this form: $t(\tau) - l$ is the number of τ-transitions leaving the non-final states of A. Then $(2^{t(\tau)-l}) - 1$ is the number of sets formed with this transitions without the empty set.
   - if A is τ-incomplete, A' has $(2^{t(\tau)-l+u(\tau)} - 2^{\bar{u}(\tau)})$ τ-transitions of this form: $t(\tau) - l + u(\tau)$ is the number of τ-transitions leaving the non-final states of A. Then $(2^{t(\tau)-l+u(\tau)} - 2^{\bar{u}(\tau)})$ is the number of sets consisted of this transitions minus the number of sets composed only of the undefined transitions.
3. The set of transitions which include the transitions leaving the final states of A and the transitions leaving the non-final states of A. We not need to consider the transition leaving the initial state of A because, by construction, we know that whenever a transition of A' includes a transition leaving a final state of A then that also includes the transition leaving the initial state of A.
   - if A is τ-complete, then A' has $(2^l - 1).2^{t(\tau)-l-1}$ τ-transitions of this form.

- if A is τ-incomplete, then A' has $(2^l - 1).2^{t(\tau)-l-1+u(\tau)}$ τ-transitions of this form.

In conclusion,

1. the number of τ-transitions of A' if A is τ-complete is:

$$\begin{aligned} & i(\tau) + 2^{t(\tau)-l} - 1 + (2^l - 1)(2^{t(\tau)-l-1}) = \\ & = 2^{t(\tau)-f(\tau)-1} + 2^{t(\tau)-1} \\ & = 2^{n-l-1} + 2^{n-1} \end{aligned}$$

2. the number of τ-transitions of A' if A is τ-incomplete is:

$$\begin{aligned} & i(\tau) + (2^{t(\tau)-l+u(\tau)}) - 2^{\bar{u}(\tau)} + (2^l - 1)2^{t(\tau)-l-1+u(\tau)} = \\ & = i(\tau) + 2^{n-l} - 2^{\bar{u}(\tau)} + 2^{n-1} - 2^{n-l-1} \\ & = i(\tau) + 2^{n-1} - 2^{n-l-1} - 2^{\bar{u}(\tau)} \end{aligned}$$

Therefore,

$$\begin{aligned} & \sum_{\tau\in\Sigma_c} (2^{n-l-1} + 2^{n-1}) + \sum_{\tau\in\Sigma_i} (i(\tau) + 2^{n-1} - 2^{n-l-1} - 2^{\bar{u}(\tau)}) \\ & = \sum_{\tau\in\Sigma} (2^{n-l-1} + 2^{n-1}) + \sum_{\tau\in\Sigma_i} i(\tau) - \sum_{\tau\in\Sigma_i} 2^{\bar{u}(\tau)} \\ & = |k|(2^{n-l-1} + 2^{n-1}) + \sum_{\tau\in\Sigma_i} i(\tau) - \sum_{\tau\in\Sigma_i} 2^{\bar{u}(\tau)} \end{aligned}$$

Note that if A' is τ-complete then $i(\tau)$ is equal to 1, $t(\tau)$ is equal to n.

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 27 and Proposition 28 are reached. The following automaton family has $\Sigma = \{a, b\}$. Using Myhill-Nerode theorem, it is easy to prove that these automata are minimal.

Define $A = (Q, \Sigma, \delta_A, 0, F_A)$ with $Q = \{0, \dots, n-1\}$, $F_A = \{n-1\}$, $\delta_A(i, a) = i+1 \bmod n$ for $0 \le i \le n-1$, and $\delta_A(i, b) = i+1 \bmod n$ for $1 \le i \le n-1$ (see Fig.14).

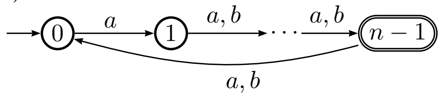

Figure 14: DFA A with n states.

Proposition 29. *DFA A is minimal.*

Proof. Consider $x, y \in \Sigma^\star$ such that $\hat{\delta}(0, x) = i$ and $\hat{\delta}(0, y) = j$, $i, j \in [0, n-1]$, $i \neq j$. It is clear that $xa^{n-1-i} \in \mathcal{L}(A)$ but $ya^{n-1-i} \notin \mathcal{L}(A)$. Then x and y are in different left quotients induced by $\mathcal{L}(A)$. □

Proposition 30. *For any integer $n \ge 2$, exists a n-state DFA A such that any DFA accepting $(\mathcal{L}(A))^\star$ needs at least $2^{n-1} + 2^{n-2}$ states.*

Proof. (Similar to the proof of the Theorem 3.3 of [18]) For $n = 2$ it is clear that $\mathcal{L} = \{w \in \{a,b\}^\star | \#_a(w) \textit{ is odd}\}$ is accepted by a two-state DFA, and $\mathcal{L}^\star = \{\varepsilon\} \cap \{w \in \{a,b\}^\star | \#_a(w) \geq 1\}$ cannot be accepted with less than 3 states. For $n > 2$, we consider the automaton family A which is shown in Fig. 14. We construct the DFA $A' = (Q', \Sigma, \delta', F')$ from A exactly as described in the algorithm above. We want to show that $L(A') = (L(A))^\star$ and that A' is minimal. $L(A') = (L(A))^\star$ is true by construction. To prove the minimality of A' we need to show that:

- every state is reachable from the start state. Because each state of A' ($q \in Q'$) is a subset of states of Q of A, we proceed by induction on the size (s) of this set of states – $|q|$. If $|q| = 1$ we have:

$$q = \{1\} \quad = \quad \delta'(q_0', a) \tag{4}$$
$$q = \{i\} \quad = \quad \delta'(\{i-1\}, a) \text{ for each } 1 < i < n-1. \tag{5}$$

  Note that we obtain $q = \{0\}$ from $\delta'(\{n-1, 0\}, b)$, which has size 2, but we obtain the state $\{n-1, 0\}$ from $\delta'(\{n-2\}, a)$ which is already considered in (5). Then we have all states such that $|q| = 1$. Assume that if $|q| < s$ then q is reachable, and let us prove that if $|q| = s$ then it is also reachable. Consider q where $|q| = s$. Let $q = \{i_1, i_2, \ldots, i_s\}$ such that $0 \leq i_1 < i_2 < \cdots < i_s < n-1$ if $n-1 \notin q$, $0 = i_1 < i_2 < \cdots < i_{s-1} < i_s = n-1$ otherwise. There are three cases to consider:

  (I) $\{n-1, 0, i_3, \ldots, i_s\} = \delta'(\{n-2, i_3-1, \ldots, i_s-1\}, a)$ where the state $\{n-2, i_3-1, \ldots, i_s-1\}$ contains $s-1$ states.

  (II) $\{0, 1, i_3, \ldots, i_s\} = \delta'(\{n-1, 0, i_3-1, \ldots, i_s-1\}, a)$ where state the state $\{n-1, 0, i_3-1, \ldots, i_s-1\}$ is considered in case (I).

  (III) $\{t, i_2, \ldots, i_s\} = \delta'(\{0, i_2-t, \ldots, i_s-t\}, a^t)$, $t > 0$, where and the state $\{0, i_2-t, \ldots, i_s-t\}$ is considered in case (II).

- each state defines a different left quotients induced by $\mathcal{L}(A')$.

  Consider p, $q \in Q'$, $p \neq q$ and $i \in p - q$. Then $\delta'(p, a^{n-1-i}) \in F'$ but $\delta'(q, a^{n-1-i}) \notin F'$.

□

Proposition 31. *For any integer $n \geq 2$, exists an n-state DFA A (Fig. 14) with $r = 2n-1$ transitions such that any DFA accepting $\mathcal{L}(A)^\star$ has, at least, $2^{\frac{r+1}{2}} + 2^{\frac{r-1}{2}} - 2$ transitions.*

Proof. Consider the DFA A' such that $\mathcal{L}(A') = (\mathcal{L}(A))^\star$. 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 28 and 2^{n-2} a–transitions which corresponds to case 3 of Proposition 28.

- $2^{n-1} - 2 + 2^{n-2}$ b–transitions because it has $2^{n-2+1} - 2$ b–transitions which corresponds to case 2 of Proposition 28, and 2^{n-3+1} b–transitions which corresponds to case 3.

The family of automata A' have $2^{n-1} - 2^{n-2} + 2^{n-1} - 2 + 2^{n-2} = 2^n + 2^{n-1} - 2$ transitions. As $r = 2n-1 \Leftrightarrow n = \frac{r+1}{2}$, thus

$$\begin{aligned} & 2^n + 2^{n-1} - 2 \\ = \; & 2^{\frac{r+1}{2}} + 2^{\frac{r-1}{2}} - 2 \end{aligned}$$

Finally, the family of automata A' have $2^{\frac{r+1}{2}} + 2^{\frac{r-1}{2}} - 2$ transitions. $\square$

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.*

5.2.1 Example

The DFA A in Figure 15 is an example from the worst-case family, with $n = 3$:

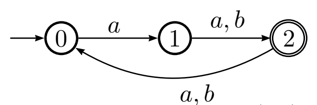

Figure 15: Star: Example of DFA A with n=3

This DFA A is a-complete ($t(a) = 3$); and it is b-incomplete ($t(b) = 2$).

Let us calculate the transition complexity of $\mathcal{L}(A)^\star$ using the Proposition 31:

$$\begin{aligned} tc(\mathcal{L}(A)^\star) \; \leq \; & 2^{3-1}(2^{-1} + 1) + 0 - 2^{3-1}(2^{1-2} - 2^{-1} - 1) \\ & 2 + 2^2 - 2 + 2 + 2^2 \\ & 10 \end{aligned}$$

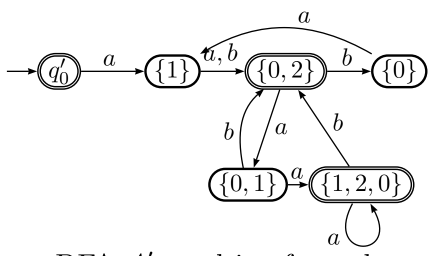

Figure 16: Star: DFA A' resulting from the star operation

The diagram in Figure 16 shows the DFA resulting from the star operation. This DFA is a-complete ($t(a) = 6$); it is b-incomplete b ($t(b) = 4$). Thus, it has 10 transitions. As it is minimal we can say that $itc(\mathcal{L}(A)) = 10$.

6 Incomplete Transition Complexity of the Reversal

It is known that the reversal of a complete DFA reaches upper bound 2^n. This upper bound occurs because of the subset construction which is implicit in the reversal operation. Because of this the DFA resulting of the reversal has a state which corresponds to the $\emptyset$, which is a dead state. Therefore, if we permit that the resulting automata is not complete the state complexity is $2^n - 1$. Consequently the transition complexity is $|k|(2^n - 1)$.

Note that the worst case of the reversal operation is always the complete DFAs, because if the DFA A has less transitions, then the DFA resulting of the reversal operation never has more than 2^n states.

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_1L_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.

7 Unary Languages

A DFA that accepts a unary language has a non-cyclic part (tail) and a cyclic part (loop). Note that if the DFA is not complete it only has the tail and it represents a finite language. Thus, the worst case state complexity of operations occurs when the DFA of the operands are complete. In 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.

8 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.

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.

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

Operation	itc	ntc
$L_1 \cup L_2$	$\mathbf{itc(L_1)(1+n) + itc(L_2)(1+m) -}$ $\sum_{\tau\in\boldsymbol{\Sigma}} \mathbf{itc_\tau(L_2) itc_\tau(L_1)}$	$ntc(L_1) + ntc(L_2) +$ $s(L_1) + s(L_2)$
$L_1 \cap L_2$	$itc(L_1)itc(L_2)$	$\sum\limits_{\tau\in\Sigma} ntc_\tau(L_1)ntc_\tau(L_2)$
L^C	$\|\Sigma\|(itc(L)+2)$	$\|\Sigma\|2^{ntc(L)+1}$
		$2^{\frac{ntc(L)}{2}-2} - 1$
L_1L_2	$\boldsymbol{\|\Sigma\|(m+1)2^n - \|\Sigma_c^{L_2}\|(f\ 2^{n-1}+1) - \sum\limits_{\tau\in\Sigma_i^{L_2}} (2^{u_\tau}}$ $\boldsymbol{+f\ 2^{itc_\tau(L_2)}) - \sum\limits_{\tau\in\Sigma_{ii}} \bar{u}_\tau 2^{u_\tau} - \sum\limits_{\tau\in\Sigma_{ic}} \bar{u}_\tau}$	$ntc(L_1) + ntc(L_2) +$ $f_{in}(L_1)$
$L^\star$	$\boldsymbol{\|\Sigma\|(2^{m-l-1} + 2^{m-1}) + \sum\limits_{\tau\in\Sigma_i} (i_\tau - 2^{\bar{u}_\tau})}$	$ntc(L) + f_{in}(L)$
L^R	$\boldsymbol{\|\Sigma\|(2^m - 1)}$	$ntc(L) + f(L)$

Table 2: Transition Complexity.

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 partial 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 (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)