Journal of Universal Computer Science, vol. 16, no. 5 (2010), 653-675
submitted: 15/6/09, accepted: 12/2/10, appeared: 1/3/10 © J.UCS

Orthogonal Concatenation: Language Equations and State Complexity

In Honor of Derick Wood's 70th Birthday

Mark Daley
(Department of Computer Science and Department of Biology
University of Western Ontario
London, Ontario N6A 5B7, Canada
daley@csd.uwo.ca)

Michael Domaratzki
(Department of Computer Science
University of Manitoba
Winnipeg, Manitoba R3T 2N2, Canada
mdomarat@cs.umanitoba.ca)

Kai Salomaa
(School of Computing, Queen's University
Kingston, Ontario K7L 3N6, Canada
ksalomaa@cs.queensu.ca)

Abstract: A language L is the orthogonal concatenation of languages L_1 and L_2 if every word of L can be written in a unique way as a concatenation of a word in L_1 and a word in L_2. The notion can be generalized for arbitrary language operations. We consider decidability properties of language orthogonality and the solvability of language equations involving the orthogonal concatenation operation. We establish a tight bound for the state complexity of orthogonal concatenation of regular languages.

Key Words: language operations, language equations, regular languages, state complexity, decidability
Category: F.4.3, F.1.3

1 Introduction

The Code Division Multiple Access (CDMA) multiplexing scheme used in radio communications allows for the simultaneous reception of transmissions from multiple senders by assigning each sender a waveform which, when superimposed with the waveforms of other senders, generates a signal which can be uniquely decomposed back in to the original waveforms. This is possible due to the additive nature of interfering radio waves and the careful choice of waveforms which are mutually orthogonal. In the case of binary signaling, we may represent waveforms as binary vectors of some finite size leading to a simple choice of those waveforms which are represented by orthogonal vectors in the sense of traditional linear algebra[1].

The theoretical and practical properties of CDMA have been investigated thoroughly in the engineering literature, including algebraic studies of CDMA variants (see, e.g., [Castaing and De Lathauwer 2004]). These studies, however, principally concern themselves with the details of radio communications, rather than abstract communications channels. In this paper we seek to generalize the notion of orthogonality to arbitrary word operations to perhaps facilitate the investigation of code-division multiplexing schemes in the more general context of abstract coding theory. Restrictions of operational orthogonality have been previously studied for the concatenation operation in the formal language theory literature on language equations.

We present here a general framework for studying orthogonality of operations on formal languages and demonstrate preliminary results for the operations of concatenation and shuffle on trajectories. In section 4 we consider language equations involving orthogonal concatenation. Concatenation is a fundamental operation on formal languages. The inverse operation of language decomposition and the computation of irreducible (or prime) components are important theoretical tools used in data classification [Czyzowicz et al. 2003], however, the complexity of language decomposition is not yet properly understood [Wood et al. 2007, Mateescu et al. 2002, Salomaa 2008]. For orthogonal concatenation the situation is even worse, and even the decidability status of the question whether a regular language has an orthogonal decomposition with respect to concatenation remains open.

In section 5 we study the state complexity of the orthogonal concatenation operation. The orthogonality condition may appear quite restrictive and, at first sight, one could expect that the state complexity is significantly lower than in the case of unrestricted concatenation [Jirásková 2005, Maslov 1970, Yu et al. 1994, Yu 1997]. However, we show that the worst-case state complexity of orthogonal concatenation differs from the state complexity of ordinary concatenation precisely by a factor of two.

The state complexity of a related, but essentially different, operation of *unique concatenation* was investigated by [Rampersad et al. 2009]. Whereas orthogonal concatenation is defined only for pairs of orthogonal languages, the unique concatenation of any two languages is defined to consist of those words that can be written in a unique way as a concatenation of words from the component languages [Rampersad et al. 2009].

A preliminary version of parts of this work was presented at TALE'07 [Daley et al. 2007] and at DCFS'08 [Daley et al. 2008].

[1] We refer the reader to [Rao and Dianat 2005] for an introductory text on CDMA.

2 Preliminaries

In the following Σ is a finite alphabet. The set of all words over Σ is Σ^* and Σ^+ is the set of non-empty words over Σ. The length of a word $w \in \Sigma^*$ is $|w|$ and ε is the empty word. The set of symbols of Σ occurring in a word w is $\mathrm{alph}(w)$. We write $u <_{\mathrm{p}} v$ if u is a proper prefix of v, $u, v \in \Sigma^*$. The powerset of a set A is $\mathcal{P}(A)$.

For $w \in \Sigma^*$ and $L \subseteq \Sigma^*$ we define

$$w^{-1}L = \{u \in \Sigma^* \mid wu \in L\}, \quad Lw^{-1} = \{u \in \Sigma^* \mid uw \in L\}.$$

The reversal of a word $w = a_1 \cdots a_n$, $a \in \Sigma$, $i = 1, \ldots, n$, is $w^R = a_n \cdots a_1$ and the reversal of a language L is $L^R = \{w^R \mid w \in L\}$.

The notions associated with orthogonality have close connections to formal power series and, for example, the proof of the main decidability result for orthogonal concatenation [Anselmo and Restivo 1996] relies on power series. Here we do not discuss power series in detail and the interested reader can find more information e.g. in [Kuich and Salomaa 1986].

A deterministic finite automaton (DFA) is a five-tuple

$$A = (Q, \Sigma, \delta, q_0, F), \tag{1}$$

where Σ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is the start state, $F \subseteq Q$ is the set of accepting states and $\delta : Q \times \Sigma \to Q$ defines the transitions of A. In the standard way the transition function δ is extended to a function $Q \times \Sigma^* \to Q$. Further, for a set $P \subseteq Q$ and $a \in \Sigma$, we use the shorthand $\delta(P, a) = \{\delta(q, a) \ : \ q \in P\}$.

We say that a state q is *reachable* from a state p if there exists $w \in \Sigma^*$ such that $\delta(p, w) = q$. A *dead state* is a state $q \in Q - F$ such that only q is reachable from q. States $q_1, q_2 \in Q$ are said to be *equivalent* if for any $w \in \Sigma^*$,

$$\delta(q_1, w) \in F \text{ iff } \delta(q_2, w) \in F.$$

A DFA $A = (Q, \Sigma, \delta, q_0, F)$ is a *permutation automaton* if, for each $b \in \Sigma$, the function $\delta(\cdot, b) : Q \to Q$ is a permutation of the set of states Q.

The language recognized by a DFA A as in (1) is $L(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\ \}$. Deterministic finite automata accept exactly the regular languages [Wood 1987, Yu 1997]. Any regular language has a unique DFA with a minimal number of states. In a minimal DFA all states are reachable from the start state and pairwise inequivalent.

For all unexplained notions concerning finite automata we refer the reader to [Wood 1987, Yu 1997]. More information on state complexity and references can be found in [Goldstine et al. 2002, Holzer and Kutrib 2009, Wood et al. 2004, Yu 2001].

To conclude this section we recall the notion of shuffle on trajectories operation originally introduced in [Mateescu et al. 1998].

A trajectory t is a word over the alphabet $\{0,1\}$. For $x, y \in \Sigma^*$, the shuffle of x and y along a trajectory t, $x \sqcup\!\sqcup_t y$, is defined inductively as follows.

If $x = ax'$, $y = by'$ (with $a, b \in \Sigma$) and $t = et'$ (with $e \in \{0,1\}$), then

$$x \sqcup\!\sqcup_{et'} y = \begin{cases} a(x' \sqcup\!\sqcup_{t'} by') \text{ if } e = 0; \\ b(ax' \sqcup\!\sqcup_{t'} y') \text{ if } e = 1. \end{cases}$$

If $x = ax'$ $(a \in \Sigma)$, $y = \epsilon$ and $t = et'$ $(e \in \{0,1\})$, then

$$x \sqcup\!\sqcup_{et'} \epsilon = \begin{cases} a(x' \sqcup\!\sqcup_{t'} \epsilon) \text{ if } e = 0; \\ \emptyset \qquad\qquad\quad \text{otherwise.} \end{cases}$$

If $x = \epsilon$, $y = by'$ $(b \in \Sigma)$ and $t = et'$ $(e \in \{0,1\})$, then

$$\epsilon \sqcup\!\sqcup_{et'} y = \begin{cases} b(\epsilon \sqcup\!\sqcup_{t'} y') \text{ if } e = 1; \\ \emptyset \qquad\qquad\quad \text{otherwise.} \end{cases}$$

We let $x \sqcup\!\sqcup_\epsilon y = \emptyset$ if $\{x, y\} \neq \{\epsilon\}$. Finally, if $x = y = \epsilon$, then $\epsilon \sqcup\!\sqcup_t \epsilon = \epsilon$ if $t = \epsilon$ and $\emptyset$ otherwise.

We extend shuffle on trajectories to *sets* $T \subseteq \{0,1\}^*$ *of trajectories* as follows:

$$x \sqcup\!\sqcup_T y = \bigcup_{t \in T} x \sqcup\!\sqcup_t y.$$

Further, for $L_1, L_2 \subseteq \Sigma^*$, we define

$$L_1 \sqcup\!\sqcup_T L_2 = \bigcup_{\substack{x \in L_1 \\ y \in L_2}} x \sqcup\!\sqcup_T y.$$

3 Definition and Basic Properties of Orthogonality

We begin by introducing the notion of operational orthogonality. By a binary[2] word operation on Σ^* we mean a function $\circ : (\Sigma^*)^2 \to 2^{\Sigma^*}$. The operation $\circ$ is extended to languages $L_1, L_2 \subseteq \Sigma^*$ as

$$L_1 \circ L_2 = \bigcup_{x \in L_1, y \in L_2} x \circ y.$$

Definition 1. Let L, L_1, L_2 be languages over Σ. We say that L is an *orthogonal* $\circ$-*composition* of L_1 and L_2, denoted

$$L = L_1 \circ_\perp L_2,$$

if the following two conditions hold

[2] All below notions can be extended in an obvious way for n-ary operations, $n \geq 2$. For simplicity below we discuss only the binary case.

(OR1) $L = L_1 \circ L_2$, and

(OR2) $(\forall u_i, v_i \in L_i, i = 1, 2)$ if $(u_1, u_2) \neq (v_1, v_2)$ then $u_1 \circ u_2 \cap v_1 \circ v_2 = \emptyset$.

Given languages L_1 and L_2, we define their orthogonal $\circ$-composition as

$$L_1 \circ_\perp L_2 = \begin{cases} L_1 \circ L_2 & \text{if condition (OR2) holds,} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

If $L_1 \circ_\perp L_2$ is defined, we say also that the languages L_1 and L_2 are $\circ$-orthogonal. Note that in the above statement the order of the languages is significant since the $\circ$-orthogonality relation is not symmetric.

Example 1. Let Σ be an alphabet containing symbols 0 and 1. For $x = x_1 \cdots x_m$, $x_i \in \Sigma$, $i = 1, \ldots, m$, $m \geq 0$ and $y \in \Sigma^*$, define the *marked concatenation* of words x and y as

$$x \bullet y = x_1 0 \cdots x_m 011 y.$$

Any languages $L_1, L_2 \subseteq \Sigma^*$ are $\bullet$-orthogonal.

Example 2. Let $\bullet$ be the marked concatenation from Example 1 and consider the operation $\diamond : (\Sigma^*)^2 \to 2^{\Sigma^*}$ defined by setting

$$x \diamond y = \{x \bullet y, y \bullet x\}, \quad x, y \in \Sigma^*.$$

Languages $L_1, L_2 \subseteq \Sigma^*$ are $\diamond$-orthogonal if and only if $L_1 \cap L_2$ has at most one word. Note that for any distinct words $z_1, z_2 \in L_1 \cap L_2$, $z_1 \diamond z_2 \cap z_2 \diamond z_1 \neq \emptyset$, resulting in a violation of the condition (OR2). However, if $L_1 \cap L_2 = \{z\}$, the pair (z, z) does not violate (OR2) although $z \diamond z$ contains a word that is a marked concatenation of a word of L_1 with a word of L_2, and vice versa.

For commutative word operations we can make the following observations:

Lemma 2. *Let $\circ$ be a commutative binary word operation. Then L_1 and L_2 are $\circ$-orthogonal if and only if L_2 and L_1 are $\circ$-orthogonal.*

Proof. Let $\circ$ be a commutative binary word operation, and assume that L_1 and L_2 are $\circ$-orthogonal. Assume, contrary to what we want to prove, that L_2 and L_1 are not $\circ$-orthogonal. Then there exists $x \in L_2 \circ L_1$ with two factorizations $x \in y_1 \circ z_1$ and $x \in y_2 \circ z_2$, where $y_1, y_2 \in L_2$ and $z_1, z_2 \in L_1$. But now since $y_i \circ z_i = z_i \circ y_i$, $i = 1, 2$, we have contradicted that L_1 and L_2 are $\circ$-orthogonal. $\square$

More generally, using the notation from [Kari 1994], let $\circ^R$ be the word operation defined by $x \circ^R y = y \circ x$. Then the following result is established:

Lemma 3. *Let $\circ$ be a binary word operation. Then L_1 and L_2 are $\circ$-orthogonal if and only if L_2 and L_1 are $\circ^R$-orthogonal.*

We note that the converse of Lemma 2 does not hold. In particular, if $\circ$ is the binary word operation defined by $a \circ b = \{ab\}$ and $x \circ y = \emptyset$ for $(x, y) \neq (a, b)$, then the operation $\circ$ is not commutative, but any pair of languages L_1 and L_2 are $\circ$-orthogonal.

4 Concatenation Orthogonality

In the following we concentrate on the concatenation operation. When we need to use notation for orthogonality of the operation, we denote concatenation by $\odot$. When we do not need notation for orthogonality we denote, as usual, concatenation of languages L_1 and L_2 as $L_1 \cdot L_2$, or simply as L_1L_2.

Let $L \subseteq \Sigma^*$ be a language. If L is a code then $L \odot_\perp L$ is defined, i.e., L is $\odot$-orthogonal with itself. However, the converse does not hold since

$$\{a, b, ab\} \odot_\perp \{a, b, ab\} = \{aa, ab, aab, ba, bb, bab, aba, abb, abab\},$$

and hence $\{a, b, ab\}$ is $\odot$-orthogonal with itself.

By definition, a sufficient condition for languages L and L' to be $\odot$-orthogonal is that

(i) L is a prefix code, or,

(ii) L' is a suffix code.

In particular, it follows that

$$L \text{ and } \Sigma^* \text{ are } \odot\text{-orthogonal iff } L \text{ is a prefix code,} \tag{2}$$

and,

$$\Sigma^* \text{ and } L \text{ are } \odot\text{-orthogonal iff } L \text{ is a suffix code.} \tag{3}$$

The below result showing that the operation $\odot_\perp$ is associative for non-empty languages follows easily from the corresponding well known property of formal power series.

Lemma 4. *Let L_i, $i = 1, 2, 3$, be non-empty languages. Then*

$$(L_1 \odot_\perp L_2) \odot_\perp L_3 = L_1 \odot_\perp (L_2 \odot_\perp L_3).$$

The above equality includes the condition that the left side is defined if and only if the right side is defined.

Lemma 4 needs the assumption on the non-emptiness of the languages. For example, if $L = \{a, aa\}$ then

$$(\emptyset \odot_\perp L) \odot_\perp L = \emptyset$$

but $\emptyset \odot_\perp (L \odot_\perp L)$ is undefined. In the statement of Lemma 4 it would be sufficient to require that only L_1 and L_3 are non-empty.

A natural question will be for which languages we can decide whether or not the languages are concatenation-orthogonal.

Proposition 5. *Given regular languages L_1 and L_2, it is decidable whether or not L_1 and L_2 are $\odot$-orthogonal.*

Proof. The result is well-known and follows also from Theorem 12 below since concatenation can be expressed as shuffle along the regular set of trajectories 0^*1^*. □

As can be expected, the result of Proposition 5 cannot be extended for context-free languages, not even in the case where only one of the languages L_1 or L_2 is context-free. Since it is undecidable whether or not a given linear context-free language is a prefix- (respectively, a suffix-) code, see e.g. [Jürgensen and Konstantinidis 1997], the observations (2) and (3) give the following undecidability result.

Theorem 6. *Given a linear language L and a regular language R it is undecidable whether or not*

(i) *L and R are $\odot$-orthogonal,*

(ii) *R and L are $\odot$-orthogonal.*

If L_1 and L_2 are languages over a unary alphabet $\{a\}$, L_1 and L_2 cannot be $\odot$-orthogonal whenever they contain any two words in common. This relates to the commutativity of unary concatenation. For commutative operations it may be appropriate to consider a somewhat different definition of orthogonality. We hope to return to this topic in later work.

4.1 Language equations

Here we consider language equations involving the orthogonal concatenation operation. The result of Proposition 5 means that given regular languages L, L_1 and L_2 we can effectively decide whether or not the equation

$$L = L_1 \odot_\perp L_2$$

having no variables holds. Next we consider one-variable equations

$$L = L_1 \odot_\perp X, \tag{4}$$

or its symmetric variant

$$L = X \odot_\perp L_1, \tag{5}$$

and two variable equations

$$L = X \odot_\perp Y. \tag{6}$$

Recall that the orthogonality property is not symmetric. However, we can make the observation that if L_1 and L_2 are $\odot$-orthogonal then L_2^R and L_1^R are $\odot$-orthogonal and

$$(L_1 \odot_\perp L_2)^R = L_2^R \odot_\perp L_1^R.$$

Thus an equation $L = X \odot_\perp L_1$ has a solution X_0 if and only if the equation $L^R = L_1^R \odot_\perp X$ has a solution X_0^R. This means, in particular, that when the language constants are regular (or restricted to any language family that is closed under reversal) an equation (5) can always be reduced to an equation (4). In the following we restrict one-variable equations to be of type (4).

A two-variable equation (6) has always the solution

$$L = L \odot_\perp \{\varepsilon\} = \{\varepsilon\} \odot_\perp L.$$

We say that these solutions are *trivial solutions.*

Without loss of generality we can assume that all solutions to equations (4) or (6) must be over an alphabet Σ where Σ contains all symbols occurring in words of L. The alphabet Σ is usually not mentioned separately.

The following strong result is established in [Anselmo and Restivo 1996].

Theorem 7. [Anselmo and Restivo 1996] *For regular languages L and L_1 it is decidable whether or not a one-variable equation (4) has a solution. A possible solution is regular and can be effectively constructed.*

The solutions for one-variable language equations $L = L_1X$ obviously do not need to be unique if we do not impose the condition of orthogonality. The proof of Theorem 7 in [Anselmo and Restivo 1996] implies that if (4) has a solution, it is unique. The proof given there uses formal power series and below we give an elementary proof of the fact that possible solutions for one-variable equations with orthogonal concatenation are unique, even without assuming regularity of the languages L and L_1.

Lemma 8. *Let L and L_1 be nonempty languages. If an equation $L = L_1 \odot_\perp X$ has a solution for the variable X, the solution is unique.*

Proof. For the sake of contradiction assume that there exist languages $L_2 \neq L_2'$ such that

$$L = L_1 \odot_\perp L_2 = L_1 \odot_\perp L_2'. \tag{7}$$

Let v_2 be a word of minimal length in the symmetric difference of L_2 and L_2'. Without loss of generality we can assume that $v_2 \in L_2 - L_2'$. Let u_1 be a word of minimal length in L_1. Now (7) implies that there exist $u_1' \in L_1$ and $v_2' \in L_2'$ such that

$$u_1 v_2 = u_1' v_2'. \tag{8}$$

If $v_2' \in L_2$, then the above equation would violate the $\odot$-orthogonality of L_1 and L_2, and we can conclude that $v_2' \in L_2' - L_2$.

In particular, $v_2' \neq v_2$ and since v_2 is of minimal length in the symmetric difference of L_2 and L_2' it follows that $|v_2'| > |v_2|$. Now by (8), $|u_1'| < |u_1|$ which contradicts the assumption that u_1 is of minimal length in L_1. □

It is known that any one-variable equation involving (ordinary) concatenation

$$L = L_1 X \tag{9}$$

has a minimal solution [Kari and Thierrin 1996]. As a consequence of Theorem 7 or Lemma 8 we observe that any solution to (4) has to be a minimal solution to the corresponding equation $L = L_1 X$ involving ordinary concatenation.

The analogy of Lemma 8 does not hold for two-variable equations (6). For example, by considering decompositions of individual words it is easy to see that solutions to (6) need not be unique.

For two-variable equations involving ordinary concatenation

$$L = XY, \tag{10}$$

where L is regular, the existence of non-trivial solutions is decidable [Conway 1971, Kari and Thierrin 1996, Mateescu et al. 2002].

For a given regular language L it is possible that the equation (10) has a non-trivial solution but the corresponding two-variable equation with orthogonal concatenation (6) has only trivial solutions. As an example we can consider the finite language $\{\varepsilon, a, a^2\} = \{1, a\} \cdot \{1, a\}$ where concatenation is clearly non-orthogonal.

While any solution to (4) has to be regular, assuming that L and L_1 are regular, it is noted in [Anselmo and Restivo 1996] that a regular language can be the orthogonal concatenation of two non-regular languages. For example,

$$a^* = \widehat{\Pi}_{i=0}^{\infty}(\varepsilon + a^{2^{2i}}) \odot_\perp \widehat{\Pi}_{i=0}^{\infty}(\varepsilon + a^{2^{2i+1}}). \tag{11}$$

Above we denote by $\widehat{\Pi}_{i=0}^{\infty} L_i$ the set consisting of infinite products of words $w_i \in L_i$, $i = 1, 2, \ldots$, where for all but finitely many indices $i \in \mathbb{N}$, $w_i = \varepsilon$. This means that the infinite product $\widehat{\Pi}_{i=0}^{\infty} L_i$ denotes a set of finite words.

The unary language a^* has different orthogonal decompositions into regular components, however, in this case one of the components has to be finite.

Lemma 9. *Let L be a regular unary language. If (6) has a solution (L_X, L_Y) where L_X and L_Y are regular, then one of the languages L_X or L_Y has to be finite.*

Proof. If L_X and L_Y are infinite and regular, there exist $m_1, m_2 \geq 0$ and $n_1, n_2 \geq 1$ such that $a^{m_1}(a^{n_1})^* \subseteq L_X$ and $a^{m_2}(a^{n_2})^* \subseteq L_Y$. This means that the word $a^{m_1+m_2+n_1n_2}$ has two different decompositions into words of L_X and L_Y, and the languages are not $\odot$-orthogonal. $\square$

¿From Theorem 7 it follows that in any solution for (6), where L is regular, if the language for one of the variables X or Y is regular, also the language for the other variable has to be regular. This property clearly does not hold for solutions of equation (10) involving ordinary concatenation.

Open problem 10 (i) *Is it possible, for a regular language L, that (6) has non-regular solutions for X and Y but no non-trivial regular solution?*

(ii) *Given a regular language L is it decidable whether or not (6) has a non-trivial solution (respectively, a non-trivial regular solution)?*

Note that the result of Theorem 7 cannot, at least not directly, be used to enumerate all possible regular solutions for two-variable equations (6) because there is no known state complexity upper bound for regular solutions for X and Y as a function of the state complexity of L.

To conclude this section we show that existence of solutions for two-variable equations is undecidable when the constant language is context-free.

Theorem 11. *Given a linear context-free language L it is undecidable whether or not the equation (6) has a non-trivial solution.*

Proof. Let $I_{PCP} = (u_1, \ldots, u_m; v_1, \ldots, v_m)$, $u_i, v_i \in \{a, b\}^+$, $i = 1, \ldots, m$, $m \geq 1$, be an arbitrary instance of the Post correspondence problem (PCP). Without loss of generality we assume that

$$u_1 \neq v_1. \tag{12}$$

We denote

$$I'_{PCP} = (u_1, \ldots, u_m, u_{m+1}; v_1, \ldots, v_m, v_{m+1}) \text{ where } u_{m+1} = c,\ v_{m+1} = cc.$$

Now I'_{PCP} is a PCP instance over the alphabet $\{a, b, c\}$. The new elements u_{m+1} and v_{m+1} have been added only for technical reasons and they clearly cannot be used in any solution for I'_{PCP} (because they have different length and no other words of I'_{PCP} contain occurrences of c). Thus

$$\text{the instance } I'_{PCP} \text{ has a solution iff the instance } I_{PCP} \text{ has a solution.} \quad (13)$$

We define

$$\Omega = \{1, \ldots, m, m+1, \#, a, b, c\} \text{ and } \Sigma = \Omega \cup \{\$\}.$$

For notational convenience, the alphabet Σ is allowed to depend on the given PCP instance. Everything below works if we code the symbols $1, \ldots, m+1$ over a fixed alphabet where the coding is chosen so that the first symbol of the encoding of $m+1$ is distinct from the first symbol of the encoding of 1.

We define a linear context-free language $L \subseteq \Sigma^*$:

$$\begin{aligned} L = &\{i_k \cdots i_1 \# u_{i_1} \cdots u_{i_k} \mid k \geq 1, 1 \leq i_j \leq m+1, j = 1, \ldots, k\} \cdot \{\varepsilon, \$\} \\ &\cup \{i_k \cdots i_1 \# v_{i_1} \cdots v_{i_k} \mid k \geq 1, 1 \leq i_j \leq m+1, j = 1, \ldots, k\} \cdot \{\$, \$\$\}. \end{aligned}$$

First we establish some properties of any decomposition of L as a non-trivial concatenation of two languages (without imposing any orthogonality condition). Then we show that (6) has a non-trivial solution if and only if the PCP instance I'_{PCP} does not have a solution.

Claim 1 *If we can write*

$$L = L_1 \cdot L_2, \quad (14)$$

where $L_1 \neq \{\varepsilon\}$, then $L_2 \subseteq \{\varepsilon, \$, \$\$\}$.

Proof. If the claim does not hold, then (since the symbols \$ occur only at the end of words of L) the language L_2 contains a word $y \in \Omega^+ \cdot \{\varepsilon, \$, \$\$\}$. If $y \in L$, then there cannot exist any nonempty word w such that $wy \in L$. In other words, if $\varepsilon \in L_1$ then $L_1 = \{\varepsilon\}$ which is a contradiction.

Hence we can conclude that $\varepsilon \notin L_1$ and (14) implies that there exist $z_1, z_2 \in \Sigma^*$ such that $1 \cdot z_1 \in L_1$ and $(m+1) \cdot z_2 \in L_1$. However, only one of the words $1 \cdot z_1 \cdot y$ and $(m+1) \cdot z_2 \cdot y$ can be in L because u_{m+1}, v_{m+1} end with c and u_1, v_1 do not end with c.

This concludes the proof of the claim. □

Claim 2 *If we can write $L = L_1 \cdot L_2$ where $L_1 \neq \{\varepsilon\}$, then $\$\$ \notin L_2$.*

Proof. By Claim 1, we know that $L_2 \subseteq \$^*$. Since $1\#u_1 \in L$, it has to be the case that $1\#u_1 \in L_1$. Now if $\$\$ \in L_2$, then $1\#u_1\$\$ \in L_1 \cdot L_2$ but $1\#u_1\$\$ \notin L$ by the definition of L and (12). □

Now we continue to show that I'_{PCP} has a solution if and only if the equation (6) does not have any non-trivial solution.

First assume that $i_1, \ldots, i_k$ is a solution for I'_{PCP}. Let L_1 and L_2 be an arbitrary non-trivial solution for X and Y in (6). By Claim 1 and Claim 2 we know that the only possibilities are that $L_2 = \{\$\}$ or $L_2 = \{\varepsilon, \$\}$. The first case is impossible, since there exist words in L which do not end with \$. Thus, we must have $L_2 = \{\varepsilon, \$\}$.

Since $w_1 = i_k \cdots i_1 \# u_{i_1} \cdots u_{i_k} \in L$, it follows that $w_1 \in L_1$. Since

$$i_k \cdots i_1 \# v_{i_1} \cdots v_{i_k} \$\$ \in L,$$

it must be the case that $w_2 = i_k \cdots i_1 \# v_{i_1} \cdots v_{i_k} \$ \in L_1$. (Note that the word $i_k \cdots i_1 \# v_{i_1} \cdots v_{i_k} \$\$$ cannot be in L_1 since $L_1\$ \subseteq L$.) Now the word $w_1 \cdot \$ = w_2 \cdot \varepsilon$ has two different decompositions and the languages L_1 and L_2 cannot be $\odot$-orthogonal.

Second, assume that I'_{PCP} does not have a solution. We define

$$\begin{aligned} L_1 = &\{i_k \cdots i_1 \# u_{i_1} \cdots u_{i_k} \mid k \geq 1, 1 \leq i_j \leq m+1, j = 1, \ldots, k\} \\ &\cup \{i_k \cdots i_1 \# v_{i_1} \cdots v_{i_k} \mid k \geq 1, 1 \leq i_j \leq m+1, j = 1, \ldots, k\} \cdot \{\$\}. \end{aligned}$$

We have $L = L_1 \cdot \{\varepsilon, \$\}$ and we verify that L_1 and $\{\varepsilon, \$\}$ are $\odot$-orthogonal. If this were not the case, there must exist $r, s \geq 1$ and $1 \leq i_1, \ldots, i_r, j_1, \ldots, j_s \leq m+1$, such that

$$i_r \cdots i_1 \# u_{i_1} \cdots u_{i_r} \cdot \$ = j_s \cdots j_1 \# v_{j_1} \cdots v_{j_s} \$ \cdot \varepsilon.$$

The above implies that $r = s$, $i_r \cdots i_1 = j_r \cdots j_1$ and $u_{i_1} \cdots u_{i_r} = v_{i_1} \cdots v_{i_r}$. This is impossible since I'_{PCP} was assumed not to have a solution.

By (13) this concludes the proof of the theorem. $\square$

4.2 Shuffle on Trajectories Orthogonality

In this subsection we consider orthogonality modulo the shuffle on trajectories operation. We consider the question of the decidability of the orthogonality property for this operation.

Theorem 12. *Given regular languages $L_1, L_2 \subseteq \Sigma^*$ and a regular set of trajectories T, it is decidable whether L_1 and L_2 are $\sqcup\!\sqcup_T$-orthogonal.*

Proof. [3] Let f_i be the rational transduction defined by

$$f_i = \{(u, u') : u' \in u \rightsquigarrow_T L_i\}$$

[3] We are grateful to an anonymous referee of an earlier version of this paper who suggested this construction.

for $i = 1, 2$. Here $\rightsquigarrow_T$ is the deletion along trajectories operation [Domaratzki 2004, Kari and Sosík 2005]; the closure properties of $\rightsquigarrow_T$ easily show that f_i is a rational transduction for $i = 1, 2$. Let $L = L_1 \sqcup\!\sqcup_T L_2$. Then L_1 and L_2 are $\sqcup\!\sqcup_T$-orthogonal if and only if $f_i|_L$ is a function for $i = 1, 2$. Whether or not each $f_i|_L$ is a function can be decided by a result of [Schützenberger 1976].

We note that [Iwama 1983] has proven that it is decidable whether L_1 and L_2 are $\sqcup\!\sqcup$-orthogonal for regular languages L_1, L_2 (i.e., the case of $T = \{0, 1\}^*$).

5 State Complexity

Recall that the concatenation of languages recognized by an m-state DFA A and an n-state DFA B needs at most $m \cdot 2^n - 2^{n-1}$ states and there exist worst-case examples where the bound can be reached [Jirásková 2005, Maslov 1970, Yu et al. 1994, Yu 1997]. The worst-case examples given in the above references that reach this bound are clearly not orthogonal. State complexity of concatenation of prefix-free and suffix-free languages was investigated in [Wood et al. 2009(a), Han and Salomaa 2007]. Pairs of prefix- and suffix-free languages are necessarily orthogonal. For prefix-free languages state complexity of concatenation is linear in m and n, but the situation is not symmetric for suffix-free languages.

Here we establish a tight bound for the state complexity of orthogonal concatenation. First we show that the state complexity of the concatenation orthogonal languages recognized, respectively, by an m state DFA A and an n state DFA B can reach $m2^{n-1} - 2^{n-2}$ in cases where B has a dead state. On the other hand, if B does not have a dead state, orthogonality places restrictions on the DFA A that give a corresponding upper bound for the state complexity.

In the following let

$$A = (Q, \Sigma, \delta_A, q_0, F_A), \quad B = (P, \Sigma, \delta_B, p_0, F_B) \tag{15}$$

be two DFAs. First without making any assumptions on orthogonality, we recall from [Yu 1997] the construction of a DFA

$$C = (R, \Sigma, \gamma, r_0, F_C) \tag{16}$$

that recognizes $L(A)L(B)$.

We choose $R = Q \times \mathcal{P}(P) - F_A \times \mathcal{P}(P - \{p_0\})$, $r_0 = (q_0, \emptyset)$, $F_C = \{(q, X) \in R \mid X \cap F_B \neq \emptyset\}$ and the transitions of γ are defined by setting for $q \in Q$, $X \subseteq P$, $a \in \Sigma$, $\gamma((q, X), a) = (\delta_A(q, a), Y)$, where

$$Y = \begin{cases} \delta_B(X, a) \cup \{p_0\} & \text{if } \delta_A(q, a) \in F_A, \\ \delta_B(X, a) & \text{if } \delta_A(q, a) \notin F_A. \end{cases} \tag{17}$$

This construction gives the upper bound for the state complexity of concatenation by choosing A to have one accepting state [Yu 1997]. Also, assuming that B has a dead state p_{dead}, we note that in the DFA C states (q, X) and $(q, X - \{p_{\text{dead}}\})$ are always equivalent. This gives the following upper bound.

Lemma 13. *Let A and B be (minimal) DFAs with m and n states, respectively, and we assume that B has a dead state. Then the state complexity of the language $L(A)L(B)$ is at most*

$$m2^{n-1} - 2^{n-2}. \tag{18}$$

The next lemma establishes that the upper bound of Lemma 13 can be reached by a pair of orthogonal languages.

Lemma 14. *Let $m, n \geq 3$. There exist a DFA A with m states and a DFA B with n states such that B has a dead state and the state complexity of $L(A) \odot_{\perp} L(B)$ is $m2^{n-1} - 2^{n-2}$.*

Proof. Let $\Sigma = \{a, b, c, d\}$ and for A and B we use notations as in (15).

We choose $Q = \{0, 1, \ldots, m-1\}$, $q_0 = 0$, $F_A = \{m-2\}$, and the transitions of δ_A are defined by setting

1. $\delta_A(0, a) = 0$, $\delta_A(m-2, c) = 0$,
2. $\delta_A(i, b) = i + 1$, $i = 0, 1, \ldots, m-3$,
3. $\delta_A(i, d) = i + 1$, $i = 0, 1, \ldots, m-4$, $\delta_A(m-2, d) = 0$,
4. all transitions not listed in the above cases go to the dead state $m-1$.

The DFA A is depicted in Figure 1.

For the DFA B we choose $P = \{0, 1, \ldots, n-1\}$, $p_0 = 0$, $F_B = \{1\}$, and δ_B is defined by setting

1. $\delta_B(i, a) = i + 1$, $i = 1, \ldots, n-3$, $\delta_B(n-2, a) = 1$,
2. $\delta_B(i, b) = i$, $i = 1, 2, \ldots, n-2$,
3. $\delta_B(i, c) = i$, $i = 2, 3, \ldots, n-2$, $\delta_B(0, c) = 1$
4. $\delta_B(i, d) = i$, $i = 0, 1, \ldots, n-2$,
5. all transitions not listed in the above cases go to the dead state $n-1$.

The DFA B is depicted in Figure 2.

We show that $L(A)$ and $L(B)$ are concatenation orthogonal. For the sake of contradiction, assume that there exist $u_i \in L(A)$, $v_i \in L(B)$, $i = 1, 2$, where

$$u_1 v_1 = u_2 v_2 \text{ and } u_1 <_{\text{p}} u_2. \tag{19}$$

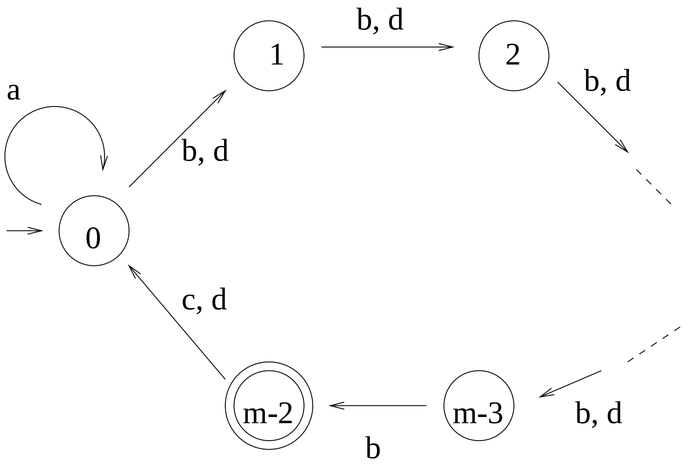

Figure 1: The DFA A. The figure does not show the dead state $m-1$ and the transitions into it.

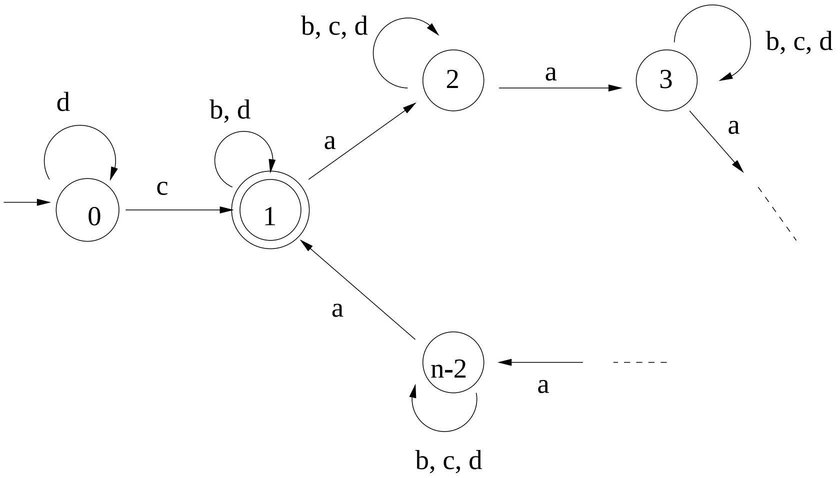

Figure 2: The DFA B. The figure does not show the dead state $n-1$.

Thus there exists $w \in \Sigma^+$ such that $v_1 = wv_2$.

Since $v_2 \in L(B)$, we can write $v_2 = d^i cz$, $i \geq 0$, $z \in \Sigma^*$ and the number of symbols a in z has to be of the form $j_z \cdot (n-2)$, $j_z \geq 0$. This follows from the observation that in the cycle of B the transitions on symbols other than a are self-loops and the only accepting state is 1.

Now $u_2 = u_1 w$ and w cannot end with a symbol d since A does not accept any words ending with d. This means that after reading w, the DFA B cannot be in the start state. Hence the computation of B on $v_1 = wd^i cz$ cannot be in state 1 after reading the prefix $wd^i c$ (since the only c-transition entering 1 is from the start state). After reading the following $j_z(n-2)$ a-transitions the computation cannot end in an accepting state of B.

We have seen that (19) produces a contradiction and, consequently, $L(A)$ and $L(B)$ have to be orthogonal.

Since B has a dead state, by Lemma 13, we know that the state complexity of $L(A) \odot_\perp L(B)$ is at most $m2^{n-1} - 2^{n-2}$ and hence it is sufficient to show that this value is also a lower bound.

Let $C = (R, \Sigma, \gamma, r_0, F_C)$ be the DFA constructed from A and B as in (16). We denote by R_1 the subset of R that consists of all elements $(q, X) \in R$ where $n-1 \notin X$. Since R_1 has $m2^{n-1} - 2^{n-2}$ states, it is sufficient to show that all states of R_1 are reachable and pairwise inequivalent in the DFA C.

Claim 3 *All states of R_1 are reachable.*

Proof. First we consider a state

$$(0, X), \text{ where } X \subseteq \{1, \ldots, n-2\}. \tag{20}$$

Using induction on $|X|$ we show that $(0, X)$ is reachable. As the base case, $(0, \emptyset)$ is the start state of C. Consider then

$$X = \{j_1, \ldots, j_r\}, \ \ 1 \leq j_1 < \ldots < j_r \leq n-2, \ \ r \geq 1. \tag{21}$$

By the silent inductive assumption the state $r_0 = (0, \{j_2 - j_1 + 1, \ldots, j_r - j_1 + 1\})$ is reachable. We note that in the DFA B, b-transitions on the states in $\{1, \ldots, n-2\}$ are self-loops. Hence

$$\gamma(r_0, b^{m-2}) = (m-2, \{0, j_2 - j_1 + 1, \ldots, j_r - j_1 + 1\}).$$

Then by applying one c-transition and shifting the second component by $(j_1 - 1)$ a-transitions we get the state $(0, X)$ where X is as in (21).

Next if $Y = \{0\} \cup X$ where $X \subseteq \{1, \ldots n-2\}$ we note that

$$\gamma((0, X), b^{m-2} d) = (0, Y).$$

Finally, from a state $(0, Z)$, $Z \subseteq \{0, 1, \ldots, n-2\}$ we get any state (i, Z), $1 \leq i \leq m-3$ or $i = m-1$ using only d-transitions. Note that $\delta_A(m-3, d)$ is the dead state $m-1$. Assuming that $0 \in Z$, we have $\gamma((0, Z), d^{m-3} b) = (m-2, Z)$. Recall that $(m-2, Z) \notin R_1$ if $0 \notin Z$.

This concludes the proof of Claim 3. □

Claim 4 *All states of R_1 are pairwise inequivalent.*

Proof. Let (i_1, X_1) and (i_2, X_2) be two distinct states in R_1.

First we consider the case where $X_1 \neq X_2$; without loss of generality let $x \in X_1 - X_2$. If $x \in \{1, \dots, n-2\}$ we note that $\gamma((i_1, X_1), a^{n-1-x}) \in F_C$ since the a-transitions in B take x to the accepting state 1. For the same reason $\gamma((i_2, X_2), a^{n-1-x}) \notin F_C$. Note that the a-transitions of A keep i_2 unchanged or take it to the dead state and hence the a-transitions of C do not create any new elements in the second component.

If $x = 0$, then applying the letter c takes (i_1, X_1) to a final state. However, since only the state 0 in B has a transition labelled c which enters the final state 1, (i_2, X_2) is not mapped to a final state by c.

Second we consider the case where $X_1 = X_2$ and $0 \leq i_1 < i_2 \leq m-1$.

(i) First consider the case where $i_1 < m-2$. Now $\gamma((i_1, X_1), b^{m-2-i_1}c) = (0, Y)$ where $1 \in Y$ and hence $(0, Y) \in F_C$. Note that $m-2-i_1 > 0$ and the transitions along b^{m-2-i_1} take (i_2, X_2) to a state $(m-1, Z)$ where $0 \notin Z$. We observe that the last c-transition cannot take $(m-1, Z)$ to an accepting state of C.

(ii) The only remaining case case is $i_1 = m-2$, $i_2 = m-1$. Now in A the word cb^{m-2} takes the state i_1 $(= m-2)$ to the accepting state $m-2$ and hence $\gamma((i_1, X_1), cb^{m-2}c) = (0, Y')$ where $1 \in Y'$. On the other hand, $\gamma((i_2, X_2), cb^{m-2}) = (m-1, Z')$ where $0 \notin Z'$ and hence $\gamma((i_2, X_2), cb^{m-2}c) \notin F_C$.

This concludes the proof of Claim 4 and the proof of Lemma 14. □

□

We note that the DFA B used in the proof of Lemma 14 has a dead state and, by Lemma 13, the result is tight for this type of automata.

Next we consider the situation where B does not have a dead state.

Lemma 15. *Let A and B as in (15) be minimal DFAs such that $L(A)$ and $L(B)$ are concatenation orthogonal. If B does not have a dead state, then no accepting state of A can be reachable from itself along a nonempty word.*

Proof. For the sake of contradiction assume that there exists $q \in F_A$ where $\delta_A(q, u) = q$, $\delta_A(q_0, w) = q$, for some $w \in \Sigma^*$ and $u \in \Sigma^+$.

Since B does not have a dead state, there exist $0 \leq i < j$ such that $\delta_B(p_0, u^i) = \delta_B(p_0, u^j) = p$ and $\delta_B(p, v) \in F_B$ for some $v \in \Sigma^*$. Thus, the words w and wu^{j-i} are in $L(A)$ and the words $u^j v$ and $u^i v$ are in $L(B)$. Since $w \cdot u^j v = wu^{j-i} \cdot u^i v$ this would imply that $L(A)$ and $L(B)$ are not orthogonal.

□

Now if B does not have a dead state, and assuming that $L(A)$ and $L(B)$ are orthogonal, we can define an anti-reflexive partial ordering of accepting states of A,

$$<_{\rm acc} \subseteq F_A \times F_A \text{ such that } p_1 <_{\rm acc} p_2 \text{ iff } p_2 \text{ is reachable from } p_1. \tag{22}$$

Furthermore, we know that if $p \in F_A$ is maximal with respect to $<_{\rm acc}$, the only state reachable from p is the dead state, using also the fact that A is minimal. Note that these conditions do not restrict $L(A)$ to be finite.

Under the above conditions, if we construct C as in (16), more than $m2^{n-1} - 2^{n-2}$ states of C can be reachable only if A has at least n accepting states. To see this, note that if (q, X) is a state of C, where $q \in Q$, $X \subseteq P$, and (q, X) is reachable along a word w, then the computation of A on w has to enter an accepting state at least $|X|$ times. Since the reachability relation between accepting states of A is an anti-reflexive partial ordering and A has a dead state, it is easy to verify that if $n \geq m$, at least half of the states of C constructed as in (16) must be unreachable.

Corollary 16. *If $n \geq m \geq 3$, the worst-case state complexity of the orthogonal concatenation of an m-state and an n-state DFA is $m2^{n-1} - 2^{n-2}$.*

However, assuming $m > n$, the above observations do not directly prevent the state complexity of orthogonal concatenation from exceeding the bound $m2^{n-1} - 2^{n-2}$ in cases where B does not have a dead state. In order to cover also these cases we need to look more carefully at the restrictions that orthogonality places on A in the situation where the second DFA B does not have a dead state.

In the following of this section, unless otherwise mentioned, A and B are always minimal DFAs with notations as in (15), where A has m states and B has n states. Furthermore, we assume that

(A1) $L(A)$ and $L(B)$ are orthogonal, and,

(A2) B does not have a dead state.

In particular, by Lemma 15, we know that the accepting states of A can be ordered by a relation $<_{\rm acc}$ as in (22). Also, C is the DFA constructed from A and B as in (16).

Lemma 17. *Let $p_1, p_2 \in P$, $p_1 \neq p_2$. If*

$$\textit{there exists } b \in \Sigma \textit{ such that } \delta_B(p_1, b) = \delta_B(p_2, b), \tag{23}$$

then for any set $\{p_1, p_2\} \subseteq X \subseteq P$ and any $q \in Q$, the state (q, X) cannot be reachable in C.

Proof. Recall that the start state of C is $(q_0, \emptyset)$ and new states can be added to the second component according to the rule (17). Thus, if X occurs as a second component of a reachable state there exist $u_1, u_3 \in \Sigma^*$ and $u_2 \in \Sigma^+$ such that $\delta_A(q_0, u_1) \in F_A$, $\delta_A(q_0, u_1u_2) \in F_A$, $\delta_B(p_0, u_2u_3) = p_1$ and $\delta_B(p_0, u_3) = p_2$. Here, due to symmetry, we can assume that the predecessor of p_1 is first generated by rule (17) after reading u_1, and the predecessor of p_2 is generated after reading u_1u_2.

Denote $p = \delta_B(p_1, b) = \delta_B(p_2, b)$ where b is as in (23). Since B does not have a dead state, there exists $w \in \Sigma^*$ such that $\delta_B(p, w) \in F_B$.

With the above assumptions we note that A accepts u_1 and u_1u_2. On the other hand, B accepts u_2u_3bw and u_3bw. This means that the word $u_1u_2u_3bw$ would have two different decompositions as a concatenation of words in $L(A)$ and $L(B)$, respectively. $\square$

For any distinct states p_1 and p_2 satisfying the assumptions of Lemma 17 we know that p_1 and p_2 cannot both occur in the second component of a reachable state of C. Also, we note that if $q_1 \in F_A$ is a minimal state with respect to $<_{\mathrm{acc}}$, the only reachable state of C with first component q_1 is $(q_1, \{p_0\})$, and these observations give the following corollary.

Corollary 18. *Let A and B be DFAs with m and n states, respectively, such that $L(A)$ and $L(B)$ are orthogonal, and, B does not have a dead state. Then the state complexity of $L(A) \odot_\perp L(B)$ can exceed $m2^{n-1} - 2^{n-2}$ only if the DFA B is a permutation automaton.*

Note that two states as given in the statement of Lemma 17 exist if and only if B is not a permutation automaton.

For a state $q \in Q$, by the *valid second components of* q we mean the sets $X \subseteq P$ such that (q, X) is a reachable state in C.

Lemma 19. *Assume that B is a permutation automaton. Then for every $q \in Q$ such that an accepting state is reachable from q along a nonempty word, there exists $p_q \in P$ such that p_q is not in any valid second component of q.*

Proof. Let $q \in Q$ be such that

$$\delta_A(q, w) \in F_A \tag{24}$$

for some $w \in \Sigma^+$. Since B is a permutation automaton we can choose as $p_q \in Q$ the state with the property

$$\delta_B(p_q, w) = p_0. \tag{25}$$

Now assume that p_q occurs in some valid second component of q. Recalling that the transitions of C are defined by (17), this means that there exist $u_1, u_2 \in \Sigma^*$ such that $\delta_A(q_0, u_1) \in F_A$, $\delta_A(q_0, u_1u_2) = q$, and $\delta_B(p_0, u_2) = p_q$.

Choose v to be any nonempty word in $L(B)$. Now using (24) and (25) we note that A accepts the words u_1 and u_1u_2w. On the other hand, B accepts the words u_2wv and v. Since $w \neq \varepsilon$, this produces a contradiction with the orthogonality of $L(A)$ and $L(B)$. □

Combining all of the above we can prove the following upper bound.

Lemma 20. *Let $m, n \geq 4$. Let A and B be DFAs with m and n states respectively such that $L(A)$ and $L(B)$ are orthogonal and B does not have a dead state. Then the state complexity of $L(A) \odot_\perp L(B)$ is at most $m2^{n-1} - 2^{n-2}$.*

Proof. We give an upper bound estimate for the number of reachable states in C. By Corollary 18 we can assume that B is a permutation automaton.

Let $q_1, q_2, q_3 \in F_A$ be such that q_1 is minimal with respect to $<_{\mathrm{acc}}$ and

$$q_1 <_{\mathrm{acc}} q_2 <_{\mathrm{acc}} q_3.$$

Note that if $<_{\mathrm{acc}}$ does not admit a chain of length three, the valid second components of any state $q \in Q$ would have cardinality at most two, and the claim holds trivially. Without loss of generality we can choose q_2 to have distance at most one from any minimal element of F_A in the ordering $<_{\mathrm{acc}}$, and similarly q_3 to have distance at most two from any minimal element.

Now the only valid second component of q_1 is $\{p_0\}$. Since any computation reaching q_2 can have passed at most one accepting state, the possible valid second components of q_2 are of the form $\{p_0\} \cup Y$ where Y is a singleton or the empty set. Also since $q_2 <_{\mathrm{acc}} q_3$, by Lemma 19, there exists some element of P that cannot occur in Y. This means that there exists at most $(n-1)$ possibilities for the valid second component of q_2.

By Lemma 15, we know that A has a dead state q_{dead} and an upper bound for the number of valid second components of q_{dead} is 2^n.

If $q \in Q - F_A$ is not the dead state, then some accepting state must be reachable from q along a nonempty word. Thus, by Lemma 19, an upper bound for the number of valid second components of q is 2^{n-1}.

If $q' \in F_A - \{q_1, q_2\}$, we know that any valid second components of q' must contain p_0 and an upper bound for the number of valid second components is again 2^{n-1}.

Thus, an upper bound for the number of reachable states of C is

$$1 + n - 1 + 2^n + (m-3) \cdot 2^{n-1}.$$

When $n \geq 4$ the above value is bounded above by the value of $m2^{n-1} - 2^{n-2}$. □

Now we can state the main result of this section.

Theorem 21. *For $m \geq 3$ and $n \geq 4$, the worst-case state complexity of the orthogonal concatenation of, respectively, an m-state and an n-state DFA language is $m2^{n-1} - 2^{n-2}$.*

Proof. This follows by Lemma 13, Lemma 14, Corollary 16 and Lemma 20. □

Note that the lower bound construction of Lemma 14 assumes $m, n \geq 3$ and the result of Theorem 21 does not cover some small values of m and n. Also, the construction given in Lemma 14 uses an alphabet with 4 letters and the precise state complexity remains open for alphabets of size 2 or 3.

5.1 Nondeterministic State and Transition Complexity

We briefly consider the nondeterministic state complexity of orthogonal concatenation. The state complexity of concatenation was studied by [Holzer and Kutrib 2003], who showed that if L_1 (resp., L_2) has nondeterministic state complexity n_1 (resp., n_2) then there is an NFA for their concatenation with $n_1 + n_2$ states, and this bound is tight. For prefix-free languages the tight bound is $n_1 + n_2 - 1$ [Wood et al. 2009(b)]. The languages used to show the tightness of the bound in [Holzer and Kutrib 2003] are $(a^{n_1})^*$ and $(b^{n_2})^*$, which are obviously orthogonal. Thus, we get immediately the following result:

Theorem 22. *The worst-case nondeterministic state complexity of the orthogonal concatenation of, respectively, an m-state and an n-state NFA language is precisely $m + n$.*

The same witness languages are also used for lower bounds on the transition complexity of regular languages [Domaratzki and Salomaa 2007], and thus the bounds for transition complexity are also unaffected by orthogonality.

5.2 State Complexity: Conclusions

We have investigated the state complexity of orthogonal concatenation. The worst-case (deterministic) state complexity of orthogonal concatenation is precisely half that of ordinary catenation. However, the nondeterministic state complexity and transition complexity remain unchanged.

We note that the upper bound known for state complexity of unique concatenation [Rampersad et al. 2009] is markedly different compared to the bound of Theorem 21 and, in fact, larger than the worst-case bound for ordinary concatenation. The tight bound for the state complexity of unique square of an n-state DFA is $n \cdot 3^n - 3^{n-1}$[Rampersad et al. 2009]. We can intuitively explain the difference by noting that in the case of unique concatenation, it is the automata that are responsible for excluding those words which have multiple factorizations.

However, for orthogonal concatenation, the languages are known beforehand to obey the appropriate uniqueness restriction. Thus, the bound is considerably lower in the latter case.

The situation for nondeterministic state complexity is similar: for orthogonal concatenation the nondeterministic state complexity is the sum of sizes of the NFAs for the the component languages, while in the unique concatenation case it is at least exponential [Rampersad et al. 2009].

The state complexity of powers of regular languages was recently investigated in [Domaratzki and Okhotin 2009]. The state complexity of the orthogonal square of a regular language is still open.

References

[Anselmo and Restivo 1996] Anselmo, M., Restivo, A.: "On languages factorizing the free monoid"; Internat. J. Algebra and Computation 6 (1996) 413–427.

[Castaing and De Lathauwer 2004] Castaing, J., De Lathauwer, L.: "An algebraic technique for the blind separation of DS-CDMA Signals"; Proc. 12th European Signal Processing Conference, EUSIPCO 2004, Vienna, (2004) 377–380.

[Czyzowicz et al. 2003] Czyzowicz, J., Fraczac, W., Pelc, A., Rytter, W.: "Linear time decomposition of regular prefix-codes"; Internat. J. Foundations of Comput. Sci. 14 (2003) 1019–1031.

[Conway 1971] Conway, J.H.: "Regular algebra and finite machines"; Chapman and Hall (1971)

[Daley et al. 2007] Daley, M., Domaratzki, M., Salomaa, K.: "Orthogonality of language operations"; Proc. Theory and Applications of Languages Equations, TALE 2007, Kunc, M., Okhotin, A. (eds.), TUCS General Publication No. 44, (June 2007) 43–53.

[Daley et al. 2008] Daley, M., Domaratzki, M., Salomaa, K.: "State complexity of orthogonal catenation"; Proc. 10th Workshop Descriptional Complexity of Formal Systems, DCFS 2008, Câmpeanu, C., Pighizzini, G., (eds.), (2008) 134–144.

[Domaratzki 2004] Domaratzki, M.: "Deletion along trajectories"; Theoret. Comput. Sci. 320, 2–3 (2004) 293–313.

[Domaratzki and Okhotin 2009] Domaratzki, M., Okhotin, A.: "State complexity of power"; Theoret. Comput. Sci. 410, 24–25 (2009) 2377–2392.

[Domaratzki and Salomaa 2007] Domaratzki, M., Salomaa, K.: "Transition complexity of language operations"; Theoret. Comput. Sci. 387 (2007) 147–154.

[Goldstine et al. 2002] Goldstine, J., Kappes, M., Kintala, C.M.R., Leung, H., Malcher, A., Wotschke, D.: "Descriptional complexity of machines with limited resources"; J. Universal Comput. Sci. 8 (2002) 193–234.

[Han and Salomaa 2007] Han, Y.-S., Salomaa, K.: "State complexity of basic operations on suffix-free regular languages"; Theoret. Comput. Sci. 410, 27–29 (2009) 2537–2548.

[Holzer and Kutrib 2003] Holzer, M., Kutrib, M.: "Nondeterministic descriptional complexity of regular languages"; Internat. J. Foundations of Comput. Sci. 14 (2003) 1087–1102.

[Holzer and Kutrib 2009] Holzer, M., Kutrib, M.: "Descriptional and computational complexity of finite automata"; Proc. LATA'09, Lect. Notes Comput. Sci. 5457, Springer (2009) 23–42.

[Iwama 1983] Iwama, K.: "Unique decomposability of shuffled strings"; Proc. 15th Annual ACM Symposium on Theory of Computing, Johnson, D. et al. (eds.), (1983) 374–381.

[Jirásková 2005] Jirásková, G.: "State complexity of some operations on binary regular languages"; Theoret. Comput. Sci. 330 (2005) 287–298.

[Jürgensen and Konstantinidis 1997] Jürgensen, H., Konstantinidis, S.: "Codes"; in Handbook of Formal Languages, Vol. I, Rozenberg, G., Salomaa, A. (eds.) Springer (1997) 511–607.

[Kari 1994] Kari, L.: "On language equations with invertible operations", Theoret. Comput. Sci. 132 (1994) 129–150.

[Kari and Sosík 2005] Kari, L., Sosík, P.: "Aspects of shuffle and deletion on trajectories"; Theoret. Comput. Sci. 332 (2005) 47–61.

[Kari and Thierrin 1996] Kari, L., Thierrin, G.: "Maximal and minimal solutions to language equations"; J. Comput. System Sci. 53 (1996) 487–496.

[Kuich and Salomaa 1986] Kuich, W., Salomaa, A.: "Semirings, Automata, Languages"; EATCS Monographs on Theoretical Computer Science, Springer-Verlag (1986)

[Maslov 1970] Maslov, A.N.: "Estimates of the number of states of finite automata"; Soviet Math. Dokl. 11 (1970) 1373–1375.

[Mateescu et al. 1998] Mateescu, A., Rozenberg, G., Salomaa, A.: "Shuffle on trajectories: Syntactic constraints"; Theoret. Comput. Sci. 197 (1998) 1–56.

[Mateescu et al. 2002] Mateescu, A., Salomaa, A., Yu, S.: "Factorizations of languages and commutativity conditions"; Acta Cybernetica 15 (2002) 339–351.

[Rampersad et al. 2009] Rampersad, N., Ravikumar, B., Santean, N., Shallit, J.: "State complexity of unique rational operations"; Theoret. Comput. Sci. 410, 24–25 (2009) 2431–2441.

[Rao and Dianat 2005] Rao, R., Dianat, S.: "Basics of Code Division Multiple Access (CDMA)"; SPIE, (2005)

[Salomaa 2008] Salomaa, K.: "Language decompositions, primality, and trajectory-based operations"; Proc. CIAA 2008, Lect. Notes Comput. Sci. 5148, Springer (2008) 17–22.

[Schützenberger 1976] Schützenberger, M.P.: "Sur les relation rationnelles entre monoïdes libres"; Theoret. Comput. Sci. 3 (1976) 243–259.

[Wood 1987] Wood, D.: "Theory of Computation"; Harper & Row, New York, NY (1987)

[Wood et al. 2004] Salomaa, A., Wood, D., Yu, S.: "On the state complexity of reversals of regular languages"; Theoret. Comput. Sci. 320(2–3) (2004) 315–329.

[Wood et al. 2007] Han, Y.-S., Salomaa, A., Salomaa, K., Wood, D., Yu, S.: "On the existence of prime decompositions"; Theoret. Comput. Sci. 376 (2007) 60–69.

[Wood et al. 2009(a)] Han, Y.-S., Salomaa, K., Wood, D.: "State complexity of prefix-free regular languages"; in: Automata, Formal Languages, and Related Topics, Ésik, Z., Fülöp, Z. (eds.), Institute of Informatics, University of Szeged (March 2009) 99–115.

[Wood et al. 2009(b)] Han, Y.-S., Salomaa, K., Wood, D.: "Nondeterministic state complexity of basic operations for prefix-free regular languages"; Fundamenta Inform. 90 (2009) 93–106.

[Yu et al. 1994] Yu, S., Zhuang, Q., Salomaa, K.: "The state complexities of some basic operations on regular languages"; Theoret. Comput. Sci. 125 (1994) 315–328.

[Yu 1997] Yu, S.: "Regular languages"; in Handbook of Formal Languages, Vol. I, Rozenberg, G., Salomaa, A. (eds.) Springer, (1997) 41–110.

[Yu 2001] Yu, S., "State complexity of regular languages"; J. Automata, Languages and Combinatorics 6 (2001) 221–234.