International Journal of Foundations of Computer Science
Vol. 31, No. 8 (2020) 1113–1132

DOI: 10.1142/S012905412042006X

State Complexity of Overlap Assembly

Janusz A. Brzozowski*, Lila Kari† and Bai Li‡

David R. Cheriton School of Computer Science
University of Waterloo,
Waterloo, ON, Canada N2L 3G1
**brzozo@uwaterloo.ca*
†lila@uwaterloo.ca
‡bai.li.2005@gmail.com

Marek Szykuła§

Institute of Computer Science, University of Wrocław
Joliot-Curie 15, PL-50-383 Wrocław, Poland
msz@cs.uni.wroc.pl

Received 14 December 2018
Revised 18 September 2019
Accepted 20 October 2019
Published 15 December 2020
Communicated by Cezar Câmpeanu

The *state complexity* of a regular language L_m is the number m of states in a minimal deterministic finite automaton (DFA) accepting L_m. The state complexity of a regularity-preserving binary operation on regular languages is defined as the maximal state complexity of the result of the operation where the two operands range over all languages of state complexities $\leq m$ and $\leq n$, respectively. We determine, for $m \geq 2$, $n \geq 3$, the exact value of the state complexity of the binary operation *overlap assembly* on regular languages. This operation was introduced by Csuhaj-Varjú, Petre, and Vaszil to model the process of self-assembly of two linear DNA strands into a longer DNA strand, provided that their ends "overlap". We prove that the state complexity of the overlap assembly of languages L_m and L_n, where $m \geq 2$ and $n \geq 1$, is at most $2(m-1)3^{n-1}+2^n$. Moreover, for $m \geq 2$ and $n \geq 3$ there exist languages L_m and L_n over an alphabet of size n whose overlap assembly meets the upper bound and this bound cannot be met with smaller alphabets. Finally, we prove that $m+n$ is the state complexity of the overlap assembly in the case of unary languages and that there are binary languages whose overlap assembly has exponential state complexity at least $m(2^{n-1}-2)+2$.

Keywords: Overlap assembly; regular language; state complexity; tight upper bound.

§Corresponding author.

1. Introduction

The *state complexity* of a regular language is the number of states in a minimal deterministic finite automaton (DFA) accepting the language. The state complexity of a regularity-preserving binary operation on regular languages is defined as the maximal state complexity of the result of the operation when the operands range over all languages of state complexities $\leq m$ and $\leq n$; it is a function of m and n. State complexity was introduced by Maslov [31] in 1970, but his short paper was relatively unknown for many years. Maslov stated without proof that the state complexity of the (Kleene) star of a language L_n of state complexity n is $2^{n-1} + 2^{n-2}$, that of reversal is 2^n, that of concatenation of languages L_m and L_n of state complexities m and n, respectively, is $(m-1)2^n + 2^{n-1}$, and that of union is mn. A more complete study of state complexity including proofs was presented by Yu, Zhuang, and Salomaa [34] in 1994. They proved that the state complexity of intersection is also mn. The same bound also holds for other binary Boolean functions such as symmetric difference and difference [1]. Since the publication of the paper by Yu, Zhuang, and Salomaa, many authors have written on this subject; for an extensive bibliography see the recent surveys [2, 17]. In particular, the state complexities of the so-called basic operations, namely Boolean operations, concatenation, star and reversal in various subclasses of the class of regular languages have been studied [2].

In this paper, we consider the state complexity of a biologically inspired binary word and language operation called *overlap assembly*. Formally, overlap assembly is a binary operation which, when applied to two input words xy and yz (where y is their nonempty *overlap*), produces the output xyz. As a formal language operation, overlap assembly was introduced by Csuhaj-Varjú, Petre, and Vaszil [6] under the name “self-assembly”. It has been studied by Enaganti, Ibarra, Kari and Kopecki [9, 10] for closure properties of various language families, decision problems, and the possible use of iterated overlap assembly to generate combinatorial DNA libraries. A particular case of overlap assembly, called *chop operation*, where the overlap consists of a single letter, was studied in [20, 21], and generalized to an arbitrary length overlap in [19]. Other similar operations have been studied in the literature, such as the *short concatenation* [4], which uses only the maximum-length (possibly empty) overlap y between operands, the Latin product of words [18] where the overlap consists of only one letter, and the operation $\bigotimes$ which imposes the restriction that the non-overlapping part xz is not empty [23]. Overlap assembly can also be considered as a particular case of semantic shuffle on trajectories with trajectory $0^*\sigma^+1^*$ [8],[a] or as a generalization of the operation $\bigodot_N$ from [8] which imposes the length of the overlap to be at least N.

[a]Informally, during a shuffle between two words with a trajectory over $\{0, 1, \sigma\}^+$, the symbols of the trajectory are interpreted as follows: 0 (respectively 1) signifies that the corresponding letter from the first (respectively second) word is retained, and σ signifies that a letter from the first word is retained, provided it coincides with the corresponding letter in the second word.

The study of overlap assembly as a formal language operation was initiated in the context of research on DNA-based information and DNA-based computation, as a formalization of a biological lab procedure that combines short linear DNA strands into longer ones, provided that their ends "overlap". The process of overlap assembly is enabled by an active agent called the DNA polymerase enzyme, which has the property of being able to extend DNA strands, under certain conditions. Other DNA bio-operations enabled by the action of the DNA polymerase enzyme, which have been modeled and studied as formal language operations, include hairpin completion and its inverse operation, hairpin reduction [5, 26, 28, 29], overlapping concatenation [30], and directed extension [11]. Experimentally, (parallel) overlap assembly of DNA strands under the action of the DNA polymerase enzyme was used for gene shuffling in, e.g. [33]. In the context of experimental DNA computing, overlap assembly was used in, e.g. [7, 12, 24, 32] for the formation of combinatorial DNA or RNA libraries. Overlap assembly can also be viewed as modeling a special case of an experimental lab procedure called cross-pairing PCR, introduced in [15] and studied in, e.g. [13, 14, 16, 27].

In this paper, we investigate the state complexity of overlap assembly as a binary operation on regular languages. Except that regular languages were known to be closed under overlap assembly, the topic was not studied before. The paper is organized as follows. Section 2 describes the biological motivation of overlap assembly. Section 3 introduces our notation and describes the construction of an NFA that accepts the results of overlap assembly of two regular languages, given by their accepting DFAs. In Sec. 4, we prove that the state complexity of the overlap assembly of languages L_m and L_n, where $m \geq 2$ and $n \geq 1$, is at most $2(m-1)3^{n-1} + 2^n$ (Theorem 3). Moreover, for $m \geq 2$ and $n \geq 3$ there exist languages L_m and L_n over an alphabet of size n whose overlap assembly meets the upper bound (Theorem 5) and, in addition, this bound cannot be met with smaller alphabets (Theorem 4). Section 5 proves that $m + n$ is a tight upper bound on the state complexity of overlap assembly of two unary regular languages L_m and L_n (Theorem 6), and in Sec. 6 we show that in the case of a binary alphabet the state complexity can be at least $m(2^{n-1} - 2) + 2$, thus is already exponential in n.

A shorter version of this work not containing the results about unary and binary alphabets has appeared in [3].

2. Overlap Assembly

The bio-operation of overlap assembly was intended to model the procedure whereby short DNA single strands can be concatenated (assembled) together into longer strands under the action of the enzyme DNA polymerase, provided they have ends that "overlap". Recall that DNA single strands are oriented words from the DNA alphabet $\Delta = \{A, C, G, T\}$, where one end of a strand is labeled by $5'$ and the other by $3'$. Watson/Crick (W/C) complementarity of DNA strands couples A to T and C to G and acts as follows: Given two W/C single strands, of opposite orientation,

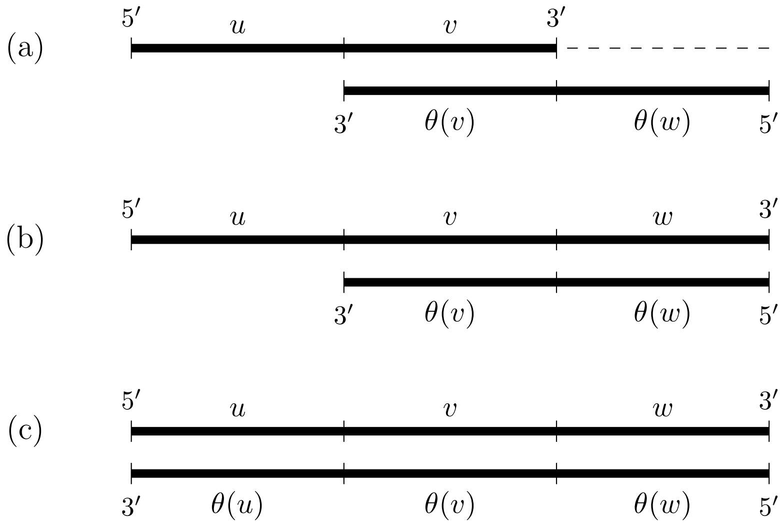

Fig. 1. (a) The input DNA single strands uv and $\theta(w)\theta(v)$ (by convention, all strands are written in the $5'$ to the $3'$ direction) bind together by the binding of their complementary segments v and $\theta(v)$, to form a partially double-stranded DNA molecule. (b) The DNA polymerase enzyme extends the $3'$ end of the strand uv, to form uvw. (c) The DNA polymerase enzyme extends the $3'$ end of the strand $\theta(w)\theta(v)$ to form $\theta(w)\theta(v)\theta(u)$. The resulting DNA double strand, whose "top" strand is uvw, is considered to be the output of the overlap assembly applied to the two input single strands. Adapted from [10].

and whose letters are complementary at each position, the W/C complementarity of DNA strands binds the two single strands together by covalent bonds, to form a DNA double strand. The W/C complementarity of DNA strands has been traditionally modeled [22, 25] as an antimorphic involution $\theta : \Delta^* \to \Delta^*$, that is, an involution on Δ (θ^2 is the identity on Δ) extended to an antimorphism on Δ^*, whereby $\theta(uv) = \theta(v)\theta(u)$ for all $u, v \in \Delta^*$. In this formalism, the W/C complement of a DNA strand $u \in \Delta^+$ is $\theta(u)$.

Using the convention that a word x over the DNA alphabet represents the DNA single strand x in the $5'$ to $3'$ direction (usually depicted as the top strand of a double DNA strand), the *overlap assembly* of a strand uv with a strand $\theta(w)\theta(v)$ first forms a partially double-stranded DNA molecule, where the substrand v in uv binds to the substrand $\theta(v)$ in $\theta(w)\theta(v)$; see Fig. 1(a). The DNA polymerase enzyme will then extend the $3'$ end of uv with the strand w; see Fig. 1(b). Similarly, the $3'$ end of $\theta(w)\theta(v)$ will be extended, resulting in a full double strand whose upper strand is $5' - uvw - 3'$, and bottom strand is $5' - \theta(w)\theta(v)\theta(u) - 3'$, see Fig. 1(c). Thus, in principle, the overlap assembly between uv and $\theta(w)\theta(v)$ results in the strands uvw and $\theta(uvw) = \theta(w)\theta(v)\theta(u)$.

Assuming that all involved DNA strands are initially double-stranded, that is, whenever the strand x is available, its W/C complement $\theta(x)$ is also available, this model was further simplified [6] as follows: Given words x, y over an alphabet Σ, the *overlap assembly of x with y* is defined as:

$$x \odot y = \{z \in \Sigma^+ \mid \exists u, w \in \Sigma^*, \exists v \in \Sigma^+ : x = uv, y = vw, z = uvw\}.$$

This can be naturally generalized to languages: Given languages L_m and L_n of state complexities m and n, respectively, the overlap assembly of L_m and L_n is defined as:

$$L_m \odot L_n = \{z \mid z = x \odot y, x \in L_m, y \in L_n\}.$$

3. An ε-NFA for Overlap Assembly

A *deterministic finite automaton (DFA)* is a quintuple $\mathcal{D} = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite non-empty set of *states*, Σ is a finite non-empty *alphabet*, $\delta : Q \times \Sigma \to Q$ is the *transition function*, $q_0 \in Q$ is the *initial* state, and $F \subseteq Q$ is the set of *final* states. We extend δ to functions $\delta : Q \times \Sigma^* \to Q$ and $\delta : 2^Q \times \Sigma^* \to 2^Q$ as usual. A DFA $\mathcal{D}$ *accepts* a word $w \in \Sigma^*$ if $\delta(q_0, w) \in F$. The language accepted by $\mathcal{D}$ is denoted by $L(\mathcal{D})$. If q is a state of $\mathcal{D}$, then the language $L_q(\mathcal{D})$ of q is the language accepted by the DFA $(Q, \Sigma, \delta, q, F)$. A state is *empty* (or *dead* or a *sink state*) if its language is empty. Two states p and q of $\mathcal{D}$ are *equivalent* if $L_p(\mathcal{D}) = L_q(\mathcal{D})$. A state q is *reachable* if there exists $w \in \Sigma^*$ such that $\delta(q_0, w) = q$. A DFA $\mathcal{D}$ is *minimal* if it has the smallest number of states and the smallest alphabet among all DFAs accepting $L(\mathcal{D})$. It is well known that a DFA is minimal if it uses the smallest alphabet, all of its states are reachable, and no two states are equivalent.

A *nondeterministic finite automaton (NFA)* is a quintuple $\mathcal{N} = (R, \Sigma, \eta, I, F)$, where R, Σ, and F are as Q, Σ, and F in a DFA respectively, $\eta : R \times \Sigma \to 2^R$, and $I \subseteq R$ is the *set of initial states*. Each triple (p, a, q) with $p, q \in R$, $a \in \Sigma$ is a *transition* if $q \in \eta(p, a)$. A sequence $((p_0, a_0, q_0), (p_1, a_1, q_1), \ldots, (p_{k-1}, a_{k-1}, q_{k-1}))$ of transitions, where $p_{i+1} = q_i$ for $i = 0, \ldots, k-2$ is a *path* in $\mathcal{N}$. The word $a_0 a_1 \cdots a_{k-1}$ is the word *spelled* by the path. A word w is *accepted* by $\mathcal{N}$ if there exists a path with $p_0 \in I$ and $q_{k-1} \in F$ that spells w. If $q \in \eta(p, a)$ we also use the notation $p \xrightarrow{a} q$. We extend this notation also to words, and write $p \xrightarrow{w} q$ for $w \in \Sigma^*$. An *ε-NFA* is an NFA in which transitions under the empty word ε are also permitted.

Given any two DFAs, we construct an ε-NFA that recognizes the overlap assembly of the languages accepted by the DFAs. This proves constructively that the family of regular languages is closed under overlap assembly.

Let $\mathcal{D}_m = (Q_m, \Sigma, \delta_m, 0, F)$ and $\mathcal{D}'_n = (Q'_n, \Sigma, \delta'_n, 0', F')$ be two DFAs with $\mathcal{D}_m$ recognizing L_m and $\mathcal{D}'_n$ recognizing L'_n, where $F = \{f_1, \ldots, f_h\}$ and $F' = \{f'_1, \ldots, f'_{h'}\}$. Let $Q_m = \{0, \ldots, m-1\}$, $Q'_n = \{0', \ldots, (n-1)'\}$, and let 0 and $0'$ be the initial states. We claim that the NFA $\mathcal{N}$, constructed as shown below, accepts the result of the overlap assembly of L_m and L'_n.

The NFA is defined as $\mathcal{N} = (R, \Sigma, \eta, \{r_0\}, F_{\mathcal{N}})$ where the set of states is $R = (Q_m \cup \{t\}) \times (Q'_n \cup \{s'\})$ with s', t being new symbols not occurring in $Q_m \cup Q'_n$, the initial state is $r_0 = (0, s')$, and the set of final states is $F_{\mathcal{N}} = \{(t, q') \mid q' \in F'\}$. Intuitively, the NFA simulates reading the word first by $\mathcal{D}_m$, then by both $\mathcal{D}_m$ and

$\mathcal{D}'_n$, and then by $\mathcal{D}'_n$. Hence the states in R contain a state of $\mathcal{D}_m$ and a state of $\mathcal{D}'_n$. The states with s' indicate that the second DFA has not yet read any letter, while the states with t indicate that the first DFA has finished its reading. The set of transitions η is defined below. The informal explanations at the right of transition definitions assume two operands $uv \in L_m$ and $vw \in L'_n$ respectively. The word $z = uvw$ belongs to their overlap assembly.

(i) $\{(q_i, s') \xrightarrow{a} (q_j, s') \mid q_i \xrightarrow{a} q_j \in \delta_m\}$; read u.
(ii) $\{(q_i, s') \xrightarrow{a} (q_j, q'_k) \mid q_i \xrightarrow{a} q_j \in \delta_m,\ 0' \xrightarrow{a} q'_k \in \delta'_n\}$; read the first letter of v.
(iii) $\{(q_i, q'_k) \xrightarrow{a} (q_j, q'_\ell) \mid q_i \xrightarrow{a} q_j \in \delta_m,\ q'_k \xrightarrow{a} q'_\ell \in \delta'_n\}$; read the remainder of v.
(iv) $\{(f_i, q'_k) \xrightarrow{\varepsilon} (t, q'_k) \mid f_i \in F,\ q'_k \in Q'_n\}$; v has been read.
(v) $\{(t, q'_k) \xrightarrow{a} (t, q'_\ell) \mid q'_k \xrightarrow{a} q'_\ell \in \delta'_n\}$; these rules read w.

Figure 2 illustrates the construction of such an NFA, denoted by $\mathcal{N}'$, for two particular two-state DFAs $\mathcal{D}_2$ and $\mathcal{D}'_2$ accepting the languages $L(D_2)$ (all words over $\{a, b\}^*$ that have an odd number of as) and $L(D'_2)$ (all words over $\{a, b\}^*$ that end in the letter a). Note that the overlap assembly of $L(D_2)$ and $L(D'_2)$ is $L(D'_2)$.

In the automaton $\mathcal{N}'$ of Fig. 2, states $(0, s')$ and $(1, s')$ behave as specified in Rule (i), using the transitions of $\mathcal{D}_2$. Rule (ii) moves the states from the first row to the second row of the figure. In the second row, the transitions are those of the direct product of $\mathcal{D}_2$ and $\mathcal{D}'_2$, as directed by Rule (iii). Note that neither Rule (i) nor Rule (ii) can be used again since s' does not appear as a component of any state

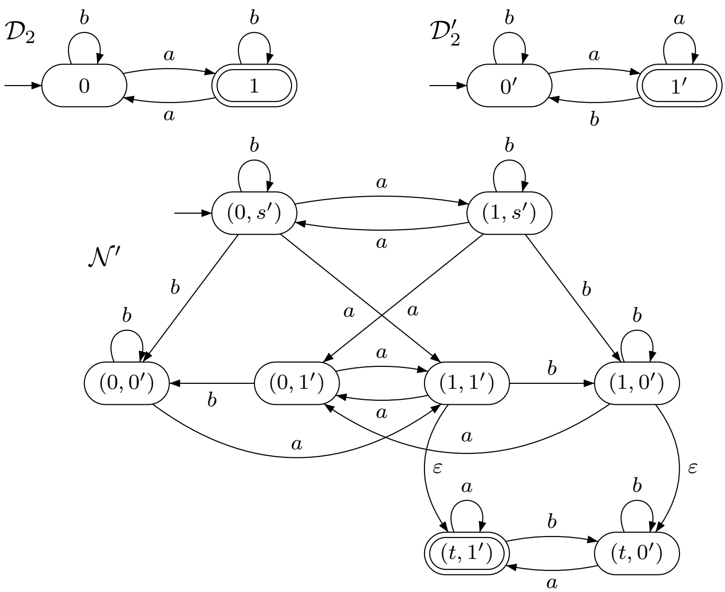

Fig. 2. An example of an NFA $\mathcal{N}'$ that accepts the overlap assembly of the languages accepted by the DFAs $\mathcal{D}_2$ (which accepts all words over $\{a, b\}^*$ that have an odd number of as) and $\mathcal{D}'_2$ (which accepts all words over $\{a, b\}^*$ that end in the letter a).

after Rule (iii) is used. When $\mathcal{N}'$ is in a state where the first component is 1, which is a final state of $\mathcal{D}_2$, $\mathcal{N}'$ can move to the next row following Rule (iv) and change the first component of the state to t. Note that Rule (iii) cannot be used again since t appears as the first component of every state after Rule (iv) is used. Finally, $\mathcal{N}'$ moves to the third row and follows the transitions of $\mathcal{D}'_2$. Note that Rule (iv) cannot be used again because of t. While the NFA $\mathcal{N}'$ has eight states, converting it to a DFA and minimizing this DFA results in D'_2. The NFA $\mathcal{N}'$ accepts the overlap assembly of $L(D_2)$ and $L(D'_2)$. In general, the following result holds:

Proposition 1. *Let L_m and L'_n be two regular languages accepted by the DFAs defined above, and let the NFA $\mathcal{N}$ be the automaton constructed as above. NFA $\mathcal{N}$ has the following properties*:

(1) *If $uv \in L_m$ and $vw \in L'_n$, then $r_0 \xrightarrow{uvw} r_f$ in $\mathcal{N}$ where $r_f \in F_{\mathcal{N}}$.*
(2) *If $r_0 \xrightarrow{z} r_f$ in $\mathcal{N}$, then there exist $u, w \in \Sigma^*$, $v \in \Sigma^+$ such that $z = uvw$, where $uv \in L_m$ and $vw \in L'_n$.*
(3) *$\mathcal{N}$ accepts $L_m \odot L'_n$.*

Proof. (1) For the first claim, let $v = ax$, where $a \in \Sigma$. If $uv \in L_m$ then $0 \xrightarrow{uax} f_i$, for some $f_i \in F$ in $\mathcal{D}_m$. So there exist q_i and q_j in Q_m such that $0 \xrightarrow{u} q_i \xrightarrow{a} q_j \xrightarrow{x} f_i$ in $\mathcal{D}_m$. Similarly, if $vw \in L_n$, then there exist q'_k and q'_ℓ in Q'_n such that $0' \xrightarrow{a} q'_k \xrightarrow{x} q'_\ell \xrightarrow{w} f'_j$, for some $f'_j \in F'$ in $\mathcal{D}'_n$.

By construction we have in $\mathcal{N}$:

$$(0, s') \xrightarrow[(i)]{u} (q_i, s') \xrightarrow[(ii)]{a} (q_j, q'_k) \xrightarrow[(iii)]{x} (f_i, q'_\ell) \xrightarrow[(iv)]{\varepsilon} (t, q'_\ell) \xrightarrow[(v)]{w} (t, f'_j),$$

which proves our first claim.

(2) Suppose that $r_0 \xrightarrow{z} r_f$ in $\mathcal{N}$, where $r_f \in F_{\mathcal{N}}$. By the construction of $\mathcal{N}$, such a path must proceed by i applications of rule (i), one application of rule (ii), j applications of rule (iii), one ε-transition via rule (iv), and k applications of rule (v), where $i, j, k \geq 0$. Thus there exist u, v, and w in Σ^* such that $z = uvw$, $|u| = i$, $|v| = j + 1$, and $|w| = k$. Owing to the construction of $\mathcal{N}$, there must exist paths $0 \xrightarrow{uv} f_i$ in $\mathcal{D}_m$ and $0' \xrightarrow{vw} f'_j$ in $\mathcal{D}'_n$, which means $uv \in L_m$ and $vw \in L'_n$.

(3) If $x \in L_m$ and $y \in L'_n$, then by (1), for every u, v, w where $x = uv$ and $y = vw$, uvw is recognized by $\mathcal{N}$; so $L_m \odot L_n \subseteq L(\mathcal{N})$. Conversely, if a word z is recognized by $\mathcal{N}$, then by (2), $z = uvw$ for some u, v, w where $uv \in L_m$ and $vw \in L_n$; so $L(\mathcal{N}) \subseteq L_m \odot L_n$. Hence $L(\mathcal{N}) = L_m \odot L_n$. □

Figure 3 shows the overall structure of the NFA $\mathcal{N}$, with examples of transitions of different types.

4. The State Complexity of Overlap Assembly in the General Case

To establish the state complexity of overlap assembly we need to determinize the ε-NFA $\mathcal{N} = (R, \Sigma, \eta, r_0, F_{\mathcal{N}})$ defined in Sec. 3, and then minimize the resulting

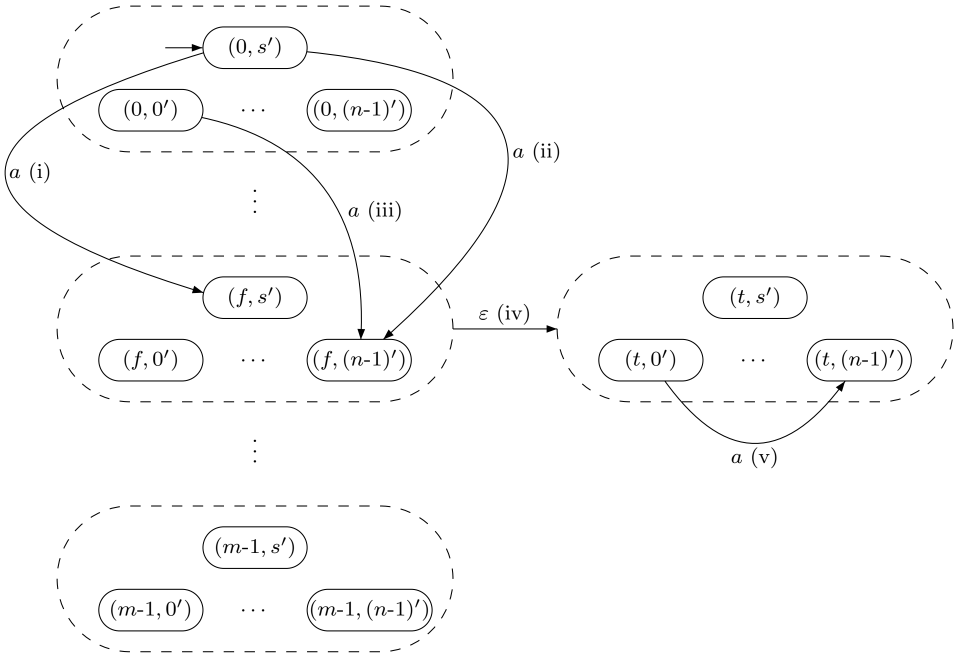

Fig. 3. The structure of the NFA that accepts the overlap assembly of two regular languages L_m and L'_n, with example transitions of every type. Assume that D_m has the transition $0 \xrightarrow{a} f$, that D'_n has the transition $0' \xrightarrow{a} (n-1)'$ and that f is one of the final states of D_m. The first of these two transitions gives rise to $(0, s') \xrightarrow{a} (f, s')$ (type (i)), while the first and second transition together give rise to $(0, s') \xrightarrow{a} (f, (n-1)')$ (type (ii)) and $(0, 0') \xrightarrow{a} (f, (n-1)')$ (type (iii)). Since f is final, a transition $(f, j') \xrightarrow{\varepsilon} (t, j')$ (type (iv)) exists for all $0 \le j \le (n-1)$. Lastly, the second transition gives rise to $(t, 0') \xrightarrow{a} (t, (n-1)')$ (type (v)).

DFA. The first step is to find an upper bound on the number of subsets S of the set R of states of $\mathcal{N}$. We begin by characterizing the reachable subsets of R. They all have the form

$$S = \{(q, s')\} \cup (\{q\} \times S') \cup (\{t\} \times T'), \tag{1}$$

where $q \in Q_m$, $T' \subseteq S' \subseteq Q'_n$ if $q \notin F$, $T' = S' \subseteq Q'_n$ if $q \in F$, and S' is non-empty unless $S = \{(0, s')\}$. We call q the *selector* of S, subset $S' \setminus \{0'\}$ is its *core*, and subset T' is its *subcore*.

We illustrate this using the NFA of Fig. 2. The initial subset is $\{(0, s')\}$; this has form (1) with $S' = T' = \emptyset$. From this initial subset we reach by b the subset $\{(0, s'), (0, 0')\} = \{0, s'\} \cup (\{0\} \times \{0'\})$; here $T' = \emptyset$ and $S' = \{0'\}$. By a we reach $\{(1, s')\} \cup \{(1, 1')\} \cup \{(t, 1')\} = \{(1, s')\} \cup (\{1\} \times \{1'\}) \cup (\{t\} \times \{1'\})$; here $S' = T' = \{1'\}$.

We now proceed to prove the claim about form (1).

Lemma 2. *Let $m \geq 2$, $n \geq 1$, and let $\mathcal{D}$ be the DFA obtained by determinization of the NFA for the overlap assembly $L_m \odot L_n$. Every reachable subset of $\mathcal{D}$ is of the form* (1)*. Moreover, if $q \notin F$, then S cannot be distinguished from $S \cup \{(q, 0')\}$.*

Proof. First we show that every reachable subset $S \subseteq R$ is of the desired form. We prove this claim by induction. The initial subset $\{(0, s')\}$ has this form. Suppose that S has this form, consider a letter $a \in \Sigma$, and the subset $U = \eta(S, a)$. Observe that $(\delta_m(q, a), s')$ is the only pair in U containing s', because of the transitions (i) and because $\mathcal{D}_m$ is deterministic. Also, every state (q, p'),where $p' \in Q'_n \cup \{s'\}$, is mapped to a state $(\delta_m(q, a), r') \in \{\delta_m(q, a)\} \times Q'_n$ by the transitions (ii) and (iii). Finally, the states in $\{t\} \times T'$ are mapped only to states from $\{t\} \times Q'_n$ by the transitions (iv) and (v).

Note that subsets S with $S' = \emptyset$ are not reachable, unless S is the initial subset $\{(0, s')\}$.

We show that if $S = \{(q, s')\} \cup (\{q\} \times S') \cup (\{t\} \times T')$ is reachable, then $T' \subseteq S'$. Let $r' \in T'$. Then there exists a word xy such that:

$$(0, s) \xrightarrow{x} (q_1, p') \xrightarrow{\varepsilon} (t, p') \xrightarrow{y} (t, r'),$$

where $q_1 \in F$. We also have:

$$(q_1, p') \xrightarrow{y} (q_2, r').$$

Thus $(q_2, r') \in S$, and so $r' \in S'$.

We observe that if $q \in F$, then by ε-transitions (transitions (iv)), every state $(q, r') \in S$ is mapped to (t, r'); thus $T' = S'$, which concludes the characterization of reachable subsets.

Finally, we show that if $q \notin F$, then S cannot be distinguished from $S \cup \{(q, 0')\}$. Indeed, let $a \in \Sigma$ be any letter. Then $\eta((q, 0'), a) = \eta((q, s'), a)$ because the transitions (iii) and (ii) coincide. Since $(q, s') \in S$, we have $\eta(S, a) = \eta(S \cup \{(q, 0')\}, a)$. $\square$

From Lemma 2 two reachable subsets with a different selector, or a different core, or a different subcore are potentially distinguishable. If two reachable subsets have the same selector, core, and subcore, then they can differ only by state $(q, 0')$ if the selector q is not in F; thus they cannot be distinguished. If two reachable subsets have the same selector q that is in F, then they cannot differ just by $(q, 0')$, as by ϵ-transitions from $(q, 0')$ we immediately obtain $(t, 0')$.

Theorem 3. *For $m \geq 2$ and $n \geq 1$, the state complexity of $L_m \odot L_n$ is at most*

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

Proof. Using Lemma 2, we count the number of potentially reachable and distinguishable subsets $S = \{(q, s')\} \cup (\{q\} \times S') \cup (\{t\} \times T')$.

Reachable subsets: For every state $q \in Q_m$, we count the number of potentially reachable subsets with selector q. There are 2 cases:

- If q is non-final, we can choose any non-empty set $S' \subseteq Q'_n$ of cardinality k and any subset T' of S'. The number of ways of doing this is $\sum_{k=1}^{n} \binom{n}{k} 2^k$.
- If q is final, again we choose any non-empty set S', but now $T' = S'$ is fixed. The number of ways of doing this is $2^n - 1$.

There is also the initial subset $\{(0, s')\}$ which contributes 1 to the sum. In total, this yields:

$$(m - |F|) \cdot \left(\sum_{k=1}^{n} \binom{n}{k} 2^k \right) + |F| \cdot (2^n - 1) + 1.$$

Distinguishable subsets: The above formula gives the number of potentially reachable subsets but overestimates the state complexity because not all subsets are distinguishable. Recall that by Lemma 2 if the selector q is not in F, then S cannot be distinguished from $S \cup \{(q, 0')\}$. Thus we do not need to count subsets S without $0'$, as $S \cup \{(q, 0')\}$ is potentially reachable and always equivalent to S. Hence, for a given $q \in Q_m \backslash F$ we choose S' to be any subset of Q'_n that contains $0'$, and again let T' be any subset of S'. This can be done in $\sum_{k=1}^{n} \binom{n-1}{k-1} 2^k$ ways. Thus the total number of potentially reachable and distinguishable subsets is at most

$$(m - |F|) \cdot \left(\sum_{k=1}^{n} \binom{n-1}{k-1} 2^k \right) + |F| \cdot (2^n - 1) + 1.$$

By algebra, we have $\sum_{k=1}^{n} \binom{n-1}{k-1} 2^k = 2 \cdot 3^{n-1}$, which is greater than $2^n - 1$; so this formula is maximized when $|F| = 1$, and we conclude that the maximum state complexity of overlap assembly is $2(m-1)3^{n-1} + 2^n$. $\square$

Theorem 4. *At least n letters are required to meet the bound from Theorem 3.*

Proof. Let $q \in F$ be a final state of $\mathcal{D}_m$. For each $p' \in Q'_n$ we consider the subset

$$T_{p'} = \{(q, s'), (q, p'), (t, p')\}.$$

If the upper bound is met, then, in particular, all subsets S with $q \in F$ must be reachable in view of Lemma 2. These subsets were counted in the upper bound, and there are no other subsets of reachable form that could be equivalent to them when the upper bound is met. Hence, in particular, all subsets $T_{p'}$ must be reachable.

Suppose that $T_{p'}$ is reachable by a word $w_{p'} a_{p'}$, for some letter $a_{p'}$. Note that (q, p') is the only one of the three states in $T_{p'}$ that can be reached by transitions (ii) of the NFA. Consider $\eta(r_0, w_{p'})$; it must contain (r, s') for some $r \in Q_m$, because by Lemma 2 every reachable subset has exactly one such pair. Thus, (r, s') must be mapped by transitions (ii) induced by $a_{p'}$ to (q, p'). Therefore, $\delta'_n(0', a_{p'}) = p'$, which proves that $a_{p'}$ are different for every p'. $\square$

We define the witness DFAs for $m, n \geq 2$. Let $\Sigma = \{a_0, \ldots, a_{n-1}\}$. Let $\mathcal{W}_m = (Q_m, \Sigma, \delta_m, 0, F)$ be defined as follows:

- $F = \{0\}$;
- $a_i : \mathbf{1}_m$ for $i \in \{0, 2, \ldots, n-1\}$, where $\mathbf{1}_m$ is the identity transformation on Q_m;
- $a_1 : (0, 1, \ldots, m-1)$ is a cyclic permutation of Q_m.

Let $\mathcal{W}'_n = (Q'_n, \Sigma, \delta'_n, 0', F')$ be defined as follows:

- $F = \{(n-1)'\}$;
- $a_0 : (Q'_n \to 0')$ maps all the states of Q'_n to $0'$;
- $a_i : (1', 2', 3', \ldots, (i-1)', 0', i', \ldots, (n-1)')$ for $i \in \{1, \ldots, n-1\}$. Here a_i permutes the states of Q'_n, mapping $1'$ to $2'$, $2'$ to $3'$, etc., then $(i-1)'$ to $0'$, $0'$ to i', and then i' to $(i+1)'$, etc., and $(n-1)'$ to $1'$.

The transitions of these DFAs with $m = 3$ and $n = 4$ states are illustrated in Fig. 4. Let L_m and L'_n be the languages of $\mathcal{W}_m$ and $\mathcal{W}'_n$, respectively.

By a *cyclic shift* of a core subset $S' \subseteq \{1', \ldots, (n-1)'\}$ we understand any subset obtained by shifting the states along the cycle $(1', \ldots, (n-1)')$, i positions clockwise, i.e. the subset $\{(((p-1+i) \bmod (n-1)) + 1)' \mid p' \in S'\}$ for any $i \geq 0$. The *next* and *previous* cyclic shifts correspond to $i = 1$ and $i = n-2$, respectively.

The transitions of letters $a_1, a_2, \ldots, a_{n-1}$ produce next cyclic shifts of the states in $\{1', \ldots, (n-1)'\}$, with the exception that state $0'$ replaces one of the states in

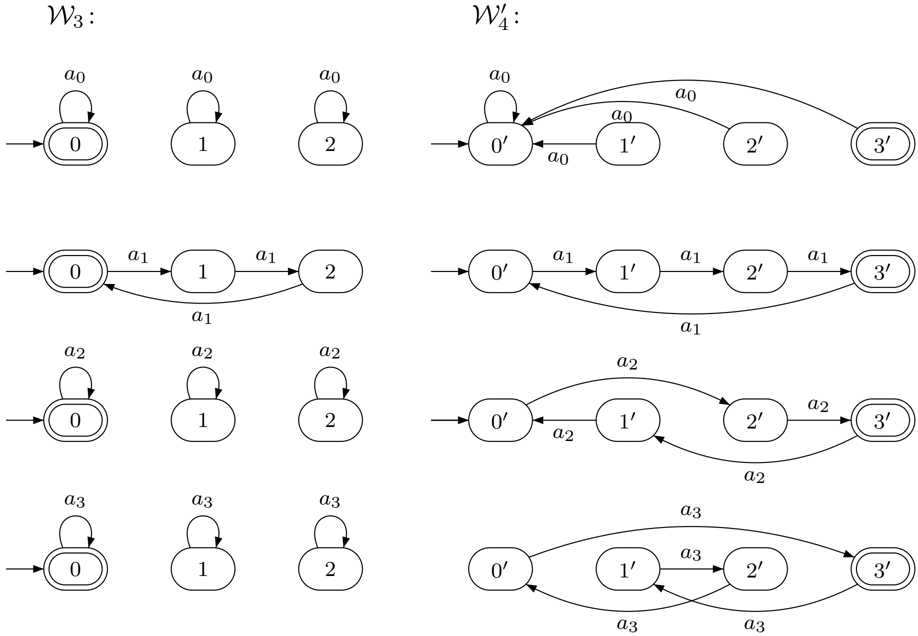

Fig. 4. The actions of the letters in $\mathcal{W}_3$ and $\mathcal{W}'_4$.

the cycle. The idea behind the witness is that we can add an arbitrary state to the core using these letters and produce arbitrary cyclic shifts as well, as will be shown later. Letter a_0 plays an important role of reset, which is necessary to reach small subsets. The main difficulty is that a_1 shares both roles of producing cyclic shifts and switching the selector.

Theorem 5. *For $m \geq 2$ and $n \geq 3$, $L_m \odot L'_n$ meets the upper bound.*

Proof. *Reachability*: It is enough to show that all subsets S from Lemma 2 are reachable, with the exception that if $q \notin F$ then it suffices to show reachability of either $S \backslash \{(q, 0')\}$ or $S \cup \{(q, 0')\}$.

• First we show that for all subsets

$$S = \{(q, s')\} \cup (\{q\} \times S'),$$

where $q \in Q_m \backslash \{0\}$ and $\emptyset \neq S' \subseteq Q'_n \backslash \{0'\}$, either $S \backslash \{(q, 0')\}$ or $S \cup \{(q, 0')\}$ is reachable. These subsets have core S' and an empty subcore.

We prove this by induction on the size $|S'|$ of the core. For $|S'| = 0$, apply $a_1^q a_0$ to $(0, s')$; this yields $\{(q, s'), (q, 0')\}$.

Consider $|S'| = 1$. If $q = 1$, then we just use a_1, which yields $\{(1, s'), (1, 1')\}$. To meet the other subsets $\{(1, s'), (1, p')\}$ for $p \geq 2$, from $\{(1, s'), (1, 1')\}$ we use $a_0 a_p$. For $q \geq 2$, we use $a_1^{q-1} a_0 a_1$, which yields $\{(q, s'), (q, 1')\}$. Then to meet the other subsets $\{(q, s'), (q, p')\}$ for $p \geq 2$, from $\{(q, s'), (q, 1')\}$ we also use $a_0 a_p$.

Consider $|S'| \geq 2$ and assume the induction hypothesis for subsets S with a smaller core. Since S' contains at least two states different from $0'$, there is a state $p' \in S' \backslash \{1'\}$. Let X' be the previous cyclic shift of $S' \backslash \{p'\}$. Since $p' \notin S' \backslash \{p'\}$, X' does not contain $(p-1)'$, but this is its only difference from the previous cyclic shift of S'. By the inductive assumption, $\{(q, s')\} \cup (\{q\} \times X')$ is reachable. We apply a_p to this subset, which maps X' to its next cyclic shift, and also (q, s') to (q, p'), which yields $\{(q, s')\} \cup (\{q\} \times S')$.

• Now we show reachability of subsets

$$S = \{(0, s')\} \cup (\{0\} \times S') \cup (\{t\} \times S'),$$

where $\emptyset \neq S' \subseteq Q'_n$. These are all potentially reachable subsets with selector 0.

First consider the case $0' \notin S'$. For $\{(m-1, s'), (m-1, 1')\}$ we apply $a_0 a_1$, which yields $\{(0, s'), (0, 1'), (t, 1')\}$. Then we continue the induction on $|S'|$ as before when $|S'| \geq 2$, with just $\{t\} \times S'$ added to the subsets.

Now consider the case $0' \in S'$. The case $S' = \{0'\}$ is easily covered by applying a_0 to $\{(0, s'), (0, 1'), (t, 1')\}$. If $S' = \{0', 1'\}$, then from $\{(m-1, s'), (m-1, (n-1)')\}$ we apply a_1 and get $\{(0, s'), (0, 0'), (0, 1'), (t, 0'), (t, 1')\}$ as desired. Let $S' \neq \{0', 1'\}$. We already know that $\{(0, s')\} \cup (\{0, t\} \times X')$ is reachable, where X' is the previous cyclic shift of $S' \backslash \{0'\}$. Since $|S'| \geq 2$ and $S' \neq \{0', 1'\}$, there is a $p' \in S' \backslash \{1'\}$. We apply a_p to $\{(0, s')\} \cup (\{0, t\} \times X')$. We have $X' \backslash \{(p-1)'\}$ mapped to $S' \backslash \{p'\}$ and $(p-1)'$ mapped to $0'$, which gives $(\{0\} \times (S' \cup \{0'\} \backslash \{p'\})$ by transitions (iii), and $(0,p')$ is added by transitions (ii). Thus, after completing by ε-transitions this yields $\{(0,s')\} \cup (\{0,t\} \times S')$.

• Finally, we show that for all subsets

$$S = \{(q,s')\} \cup (\{q\} \times S') \cup (\{t\} \times T'),$$

where $q \neq 0$ and $\emptyset \neq T' \subseteq S' \subseteq Q'_n$, either $S\backslash\{(q,0')\}$ or $S \cup \{(q,0')\}$ is reachable.

Consider the special case $S' = T' = \{0'\}$. We reach it from $\{(0,s'),(0,0'),(t,0')\}$ by applying $a_1^q a_0$. For the rest, assume that $S'\backslash\{0'\}$ is non-empty.

We need an auxiliary argument that from $\{(0,s')\}$ we can reach a subset with selector q, core S', and an empty subcore, using a word from $\{a_1,a_2,\dots,a_{n-1}\}^*$ (any word without a_0). We prove this by induction on the core size $|S'\backslash\{0'\}|$. For $|S'\backslash\{0'\}| = 1$, at the beginning we use a_1, which yields $\{(1,s'),(1,1')\}$. Now we can reach $\{(1,s'),(1,0'),(1,p')\}$ for any $p' \in \{2',\dots,(n-1)'\}$ by using $a_2a_3\dots a_p$. Then, from $\{(1,s'),(1,0'),(1,(n-1)')\}$ we reach $\{(2,s'),(2,0'),(2,1')\}$, and it remains to repeat the argument to reach every remaining subset of the form $\{(q,s'),(q,0'),(q,p')\}$ for $q \in Q_m\backslash\{0,1\}$ and $p' \in Q'_n\backslash\{0'\}$. For $|S'\backslash\{0'\}| \geq 2$ we follow the first part of the reachability argument as before, but we reach either $\{(q,s')\}\cup(\{q\}\times(S'\backslash\{0'\})$ or $\{(q,s')\}\cup(\{q\}\times(S'\cup\{0'\}))$, instead of just the former. Let $w \in \{a_1,a_2,\dots,a_{n-1}\}^*$ be a word that reaches either $\{(q,s')\}\cup(\{q\}\times(S'\backslash\{0'\})$ or $\{(q,s')\} \cup (\{q\} \times (S' \cup \{0'\}))$.

Suppose that we start from the subset

$$S_0 = \{(0,s')\} \cup (\{0,t\} \times T'_0),$$

where T'_0 is some subset such that $\emptyset \neq T'_0 \subseteq Q'_n$. We already know that for every T'_0, subset S_0 is reachable. After applying a_1w, we reach either

$$S_q = \{(q,s')\} \cup (\{q\} \times (S' \cup T'_q\backslash\{0'\})) \cup (\{t\} \times T'_q),$$

or $S_q \cup \{(q,0')\}$, where T'_q is obtained by applying some permutation π of Q'_n to T'_0. This is because $\{(0,s')\}$ is mapped by a_1w to $\{(q,s')\} \cup (\{q\} \times (S'\backslash\{0'\})$ or $\{(q,s')\} \cup (\{q\} \times (S' \cup \{0'\}))$, word a_1w acts as a permutation on $(\{t\} \times Q'_q)$, and $\{0\} \times T'_0$ is mapped to $(\{q\} \times T'_q)$. Note that a_1w does not depend on T'_0, so we can choose T'_0 arbitrarily. Let $T'_0 = \pi^{-1}(T')$, so $\pi(T'_0) = T'$. We obtain either

$$S_q = \{(q,s')\} \cup (\{q\} \times ((S'\backslash\{0'\}) \cup T') \cup (\{t\} \times T'),$$

or

$$S_q = \{(q,s')\} \cup (\{q\} \times ((S' \cup \{0'\}) \cup T') \cup (\{t\} \times T').$$

Recall that $T' \subseteq S'$ and if $0' \in T$, then also $0' \in S'$; hence $(S'\backslash\{0'\}) \cup T'$ is either S' or $S'\backslash\{0'\}$, and $(S' \cup \{0'\}) \cup T' = S' \cup \{0'\}$. Thus, S_q is either $S\backslash\{(q,0')\}$ or $S \cup \{(q,0')\}$.

Distinguishability: Consider two reachable subsets

$$S_1 = \{(q_1,s')\} \cup (\{q_1\} \times S'_1) \cup (\{t\} \times T'_1),$$

and

$$S_2 = \{(q_2, s')\} \cup (\{q_2\} \times S'_2) \cup (\{t\} \times T'_2),$$

with different selectors, different cores, or different subcores. Thus we have $q_1 \neq q_2$, or $T'_1 \neq T'_2$, or $(S'_1 \setminus \{(q_1, 0')\}) \neq (S'_2 \setminus \{(q_2, 0'\})$. These are precisely all the reachable and potentially distinguishable subsets in view of Lemma 2. Note that the initial subset also has this form, where $q_1 = 0$ and S'_1 and T'_1 are empty.

If $q_1 \neq q_2$, then without loss of generality let $q_1 < q_2$. We apply $a_1^{m-q_2} a_0 a_{n-1}^2$. For S_1, first $a_1^{m-q_2} a_0$ maps it to a subset $\{(q, s'), (0, s')\}$ or $\{(q, s'), (q, 0'), (t, 0')\}$ (if T'_1 is non-empty) for some $q \neq 0$. Then a_{n-1}^2 results in a subset that from the states from $(\{t\} \times Q'_n)$ contains at most $(t, 1')$, which is not final. On the other hand, S_2 by $a_1^{m-q_2} a_0$ is mapped to $\{(0, s'), (0, 0'), (t, 0')\}$. Then a_{n-1}^2 yields $\{(0, s'), (0, 0'), (t, 1'), (t, (n-1)')\}$, where $(t, (n-1)')$ is final.

So suppose that $q_1 = q_2$. If $q_1 \neq 0$ and $T'_1 \neq T'_2$, then we apply a_{n-1}^i for a suitable $i \geq 0$. Since a_{n-1} acts cyclically on all states $(\{t\} \times Q'_n)$ and no other states from the subsets are mapped to $(\{t\} \times Q'_n)$, we can repeat the cycle so that exactly one of $\eta(\{t\} \times T'_1, a_{n-1}^i)$ and $\eta(\{t\} \times T'_2, a_{n-1}^i)$ contains the final state $(t, (n-1)')$. If $q_1 = 0$ and $T'_1 \neq T'_2$, then also $S'_1 \neq S'_2$, so it remains to cover this case.

Suppose that $S'_1 \neq S'_2$. If $q_1 = q_2 = 0$, then also $T'_1 \neq T'_2$. We apply a_1, which maps S_1 to the subset

$$\{(1, s')\} \cup (\{1\} \times (\delta_m(S'_1, a_1) \cup \{2'\})) \cup (\{t\} \times \delta'_n(T'_1, a_1)),$$

and analogously S_2. Since $T'_1 \neq T'_2$ and a_1 acts cyclically on Q'_n, we have $\delta'_n(T'_1, a_1) \neq \delta'_n(T'_1, a_1)$. The case of these subsets has been already covered in the previous paragraph.

There remains the case where $T'_1 = T'_2$, $S'_1 \neq S'_2$, $q_1 = q_2 \neq 0$. We follow the induction on the selector q_1 starting with $q_1 = m - 1$ and decreasing it. We will show for $q_1 = m - 1$ that we can reach subsets with selector 0 that still have different cores. We have already shown in the previous paragraph that the subsets with selector 0 and different cores can be distinguished. For $q_1 < m - 1$ we will show that we can reach subsets with the same property but with selector $q_1 + 1$, which will follow by the inductive assumption. So let p be the largest index such that, without loss of generality, $p' \in S'_1$ and $p' \notin S'_2$. Note that $p \neq 0$, because then the subsets cannot be distinguished. If $p < n - 1$, then we apply a_1, which yields subsets with the desired property. If $p = n - 1$, then we first apply a_2, which yields the subset with $p' = 1'$, and then we can apply a_1 as before. $\square$

5. Unary Alphabet

In this section, we consider overlap assembly of languages over a one-letter alphabet. First note that if the longest word that is in a unary language L is of length n, then the state complexity of L is exactly $n + 2$. Similarly, if the longest word that is *not* in a unary language L is of length n, then the state complexity of L is exactly $n + 2$ [34].

Theorem 6. *Let $m, n \geq 1$, and let L_m and L_n be two unary languages of state complexities m and n, respectively. The state complexity of $L_m \odot L_n$ is at most $m+n$, and this bound is met by $L_m = \{a^{mk+n-1} \mid k \in \mathbb{Z}, mk+n-1 \geq 0\}$ and $L_n = \{a^{nk+m-1} \mid k \in \mathbb{Z}, nk+m-1 \geq 0\}$.*

Proof. We consider three cases:

Two infinite languages

Since languages L_m and L_n are regular and infinite, there are some $i, j \leq m$ and $i', j' \leq n$ such that $L_m \supseteq \{a^{ik+j} \mid k \geq 0\}$ and $L_n \supseteq \{a^{i'k'+j'} \mid k' \geq 0\}$.

Let $t \geq m+n-1$; we show that $a^t \in L_m \odot L_n$. Choose k and k' to be the maximum integers such that $ik+j \leq t$ and $i'k'+j' \leq t$. The longest word in $a^{ik+j} \odot a^{i'k'+j'}$ is $a^{(ik+j)+(i'k'+j')-1}$. By definition of k, we have $ik+j+i > t$; so $ik+j \geq t-i+1$. Similarly, $i'k'+j' \geq t-i'+1$. However,

$$\begin{aligned}(ik+j)+(i'k'+j')-1 &\geq (t-i+1)+(t-i'+1)-1 \\ &= 2t-i-i'+1 \geq 2t-m-n+1 \geq t.\end{aligned}$$

Therefore for any $t \geq m+n-1$, $a^t \in a^{ik+j} \odot a^{i'k'+j'}$. The longest word that might not be in $L_m \odot L_n$ is a^{m+n-2}, and so the state complexity of $L_m \odot L_n$ is at most $m+n$.

Next, we prove that the bound is met by the languages given in the theorem. Since we showed that $L_m \odot L_n$ contains all a^t with $t \geq m+n-1$, it is sufficient to show that a^{m+n-2} is not in $L_m \odot L_n$. Note that a^{m+n-1} is in both L_m and L_n, and we cannot obtain a^{m+n-2} if either word in L_m or L_n has length $\geq m+n-1$. Therefore we only need to consider the next longest words, which are $a^{n-1} \in L_m$ and $a^{m-1} \in L_n$. Since the longest word in $a^{n-1} \odot a^{m-1}$ is a^{m+n-3}, we have $a^{m+n-2} \notin L_m \odot L_n$. Therefore the state complexity is $m+n$.

Two finite languages

Now the longest word in L_m is a^{m-2} and the longest word in L_n is a^{n-2}. Therefore the longest word in $L_m \odot L_n$ is a^{m+n-5}. Hence the state complexity of $L_m \odot L_n$ is exactly $m+n-3$.

An infinite language and a finite one

We prove the following claim: Let $m, n \geq 1$, let L_m be an infinite unary language, and let L_n be a finite unary language. If $m \leq n-2$, then the state complexity of $L_m \odot L_n$ is at most $n-1$. Otherwise, it is at most $m+n-2$.

We consider the following two cases:

(1) $m \leq n-2$

We show that for $t \geq n-2$, $a^t \in L_m \odot L_n$. By definition of L_m, there exists $a^s \in L_m$ with $s \leq t$ and $t-s \leq m-1 \leq n-3$. Hence $a^t \in a^s \odot a^{n-2}$ and so $a^t \in L_m \odot L_n$. Therefore the state complexity of $L_m \odot L_n$ is at most $n-1$.

(2) $m > n-2$
We show that there is $i \geq 1$ such that for all $t \geq n+m-2$ we have $a^t \in L_m \odot L_n$ if and only if $a^{t-i} \in L_m \odot L_n$. This proves that the quotients of a^t and of a^{t-i} are equal, so there exists a unary DFA (not necessarily minimal) recognizing $L_m \odot L_n$ with a cycle of length i and $n+m-2$ states.

Let i be the length of the cycle in a minimal DFA of L_m. Then $i \leq m$ and $m-i$ is the number of states in the initial path in this DFA. Since L_n is finite, a^{n-2} is its longest word.

First assume that $a^t \in L_m \odot L_n$. Then there are $a^{ik+x} \in L_m$ and $a^y \in L_n$ such that $k \geq 0$, $x \leq m-1$, $y \leq n-2$, and $\max\{ik+x, y\} \leq t \leq ik+x+y-1$. Because $x+y-1 \leq m+n-4$ and $t \geq n+m-2$, it must be that $k \geq 1$. Then $a^{i(k-1)+x} \in L_m$. We have $t-i \geq (n+m-2)-m \geq n-2 \geq y$ and $i(k-1)+x \leq t-i$, thus $\max\{i(k-1)+x, y\} \leq t-i$. Also, from $t \leq ik+x+y-1$ we have $t-i \leq i(k-1)+x+y-1$. Therefore, $a^{i(k-1)+x} \in L_m$ and $a^y \in L_n$ form $a^{t-i} \in L_m \odot L_n$.

Now assume that $a^{t-i} \in L_m \odot L_n$. Since $a^{t-i} \in L_m \odot L_n$, there are $a^{ik+x} \in L_m$ and $a^y \in L_n$ such that $k \geq 0$, $x \leq m-1$, $y \leq n-2$, and $\max\{ik+x, y\} \leq t-i \leq ik+x+y-1$. If $x \leq m-i-1$, then $x+y-1 \leq (m-i-1)+(n-2)-1 = m+n-i-4$ but $t-i \geq n+m-2-i$, which yields a contradiction. If $x \geq m-i$, then a^{ik+x} is accepted in a state in the cycle of the DFA of L_m. Thus $a^{i(k+1)+x} \in L_m$ and, together with a^y, $a^{i(k+1)+x}$ forms $a^t \in L_m \odot L_n$. Hence the state complexity of $L_m \odot L_n$ is at most $m+n-2$.

In summary, the largest upper bound occurs if both languages are infinite, and the theorem holds. $\square$

6. Binary Alphabet

We define the following binary DFAs for $m, n \geq 2$. Let $\Sigma = \{a_0, a_1\}$. Let $\mathcal{B}_m(Q_m, \Sigma, \delta_m, 0, F)$ be defined as follows:

- $F = \{0\}$;
- $a_0 : \mathbf{1}_m$;
- $a_1 : (0, 1, \ldots, m-1)$.

Let $\mathcal{B}'_n(Q'_n, \Sigma, \delta'_n, 0', F')$ be defined as follows:

- $F = \{(n-1)'\}$;
- $a_0 : (1', \ldots, (n-1)')$;
- $a_1 : (0', 1', \ldots, (n-1)')$.

Theorem 7. *For $m \geq 2$ and $n \geq 3$, the state complexity of $L(\mathcal{B}_m) \odot L(\mathcal{B}'_n)$ is at least $m(2^{n-1}-2)+2$.*

Proof. The proof is based on ideas similar to those in the proof of Theorem 5.

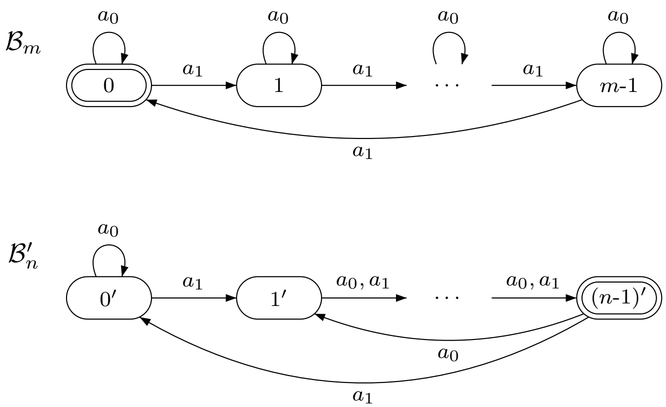

Fig. 5. Binary automata $\mathcal{B}_m$ and $\mathcal{B}'_n$ such that $L(\mathcal{B}_m) \odot L(\mathcal{B}'_n)$ has exponential state complexity.

Reachability: We show that for each selector $q \in Q_m$ and each core $\emptyset \neq S' \subseteq Q'_n \backslash \{0'\}$, there exists a reachable subset S with some subcore, that is:

$$S = \{(q, s')\} \cup (\{q\} \times (S' \cup \{0'\})) \cup (\{t\} \times T'),$$

for some subcore $T' \subseteq S' \cup \{0'\}$.

First, we show that we can reach a subset of that form but for some selector $p \in Q_m$ that is not necessarily q. We prove this by induction on $|S'|$. For $S' = \{r'\}$, we apply $a_1 a_0^{r-1}$, which yields $\{(1, s'), (1, 0'), (1, r')\}$. Let $|S'| \geq 2$ and assume that the claim holds for smaller subsets S'. Let $r' \in S'$ be a state and let $X' = S' \backslash \{r'\}$, By assumption we can reach

$$X = \{(p, s')\} \cup (\{p\} \times (X' \cup \{0'\})) \cup \{\{t\} \times Y'\},$$

for some $Y' \subseteq X'$. We apply $a_0^{m-1-r} a_1 a_0^{r-1}$ for X. This first maps X' to its cyclic shift without state $(m-1)'$, then state $1'$ is added by a_1 and the selector is changed, and we again cyclically shift to get X'. Finally, we apply a_0^{n-1} to ensure that $(q, 0')$ is present; this yields the desired subset S.

Now, to change the selector from p to q we use the same technique. It is enough to show that from a subset with selector p we can reach a subset with the selector $(p + 1) \bmod m$ and the same core S'. We choose a state $r' \in S'$, and then use $a_0^{m-1-r} a_1 a_0^{r-1}$. This first changes the core so that $(m-1)'$ is there, then the selector is changed by a_1, and the core is cyclically shifted back to S'.

Distinguishability: We will show that all the subsets above such that $S' \neq Q'_n \backslash \{0'\}$ together with the initial subset and one of the subsets with $S' = Q'_n \backslash \{0'\}$ are pairwise distinguishable. The number of non-empty and not full cores S' is $2^{n-1} - 2$, which together with the m choices for the selector q yields $m(2^{n-1} - 2)$. Adding the initial subset and the subset with full S' yields the desired formula.

Without loss of generality, let

$$S_1 = \{(q_1, s')\} \cup (\{q_1\} \times (S'_1 \cup \{0'\})) \cup \{\{t\} \times T'_1\},$$

$$S_2 = \{(q_2, s')\} \cup (\{q_2\} \times (S'_2 \cup \{0'\})) \cup \{\{t\} \times T'_2\},$$

be such that $\emptyset \neq S'_1 \subsetneq Q'_n \backslash \{0'\}$, $\emptyset \neq S'_2 \subseteq Q'_n \backslash \{0'\}$, $T'_1, T'_2 \subseteq Q'_n$, and $S'_1 \neq S'_2$ or $q_1 \neq q_2$. Moreover, we can assume that $|S'_1| \leq |S'_2|$.

First consider the case $q_1 \neq q_2$. Let r' be such that $r' \in S'_1$. As before, by applying $a_0^{n-1-r} a_1 a_0^{r-1}$, from S_1 we reach a subset with selector $(q_1 + 1) \bmod m$ and the same core S'_1. Similarly S_2 is mapped to a subset with selector $(q_2 + 1) \bmod m$. We repeat this procedure until S_2 is mapped to a subset with selector $(m-1, s')$, that is, for S_1 and S_2 we apply $(a_0^{n-1-r} a_1 a_0^{r-1})^{m-1-r}$. Since $q_1 \neq q_2$, the first subset obtained from S_1 has selector $q \neq m-1$. Now let $p' \in Q'_n \backslash (S'_1 \cup \{0'\})$. We apply a_0^{n-1-p}, which causes $(n-1)'$ to be absent from the core of the first subset. Since a subcore is always a subset of the core with $(0, t')$ added, $(n-1)'$ is also absent from the subcore of the first subset. We apply a_1 and obtain:

$$X_1 = \{(q+1, s')\} \cup (\{q+1\} \times Y'_1) \cup \{\{t\} \times Z'_1\},$$

$$X_2 = \{(0, s')\} \cup (\{0\} \times Y'_2) \cup \{\{t\} \times Z'_2\},$$

for some $Z'_1 \subseteq Y'_1 \subseteq Q'_n$ and $Z'_2 \subseteq Y'_2 \subseteq Q'_n$. Since $(n-1)'$ was not in the subcore of the first subset and $q + 1 \neq 0$, we have $0' \notin Z'_1$. We apply a_0^{n-1}. Since $0' \notin Z'_1$ and $q + 1 \neq 0$, from X_1 we obtain a subset that does not have final state $(t, (n-1)')$. On the other hand, from X_2 state $(0, s')$ is mapped by a_0 to $(0, 1')$ and then by an ε-transition to $(t, 1')$. This is then mapped to final state $(t, (n-1)')$ by a_0^{n-2}.

Now consider the case $q_1 = q_2$ and $S'_1 \neq S'_2$. Since S'_2 is not a subset of S'_1, there is a state p' such that $p' \notin S'_1$ and $p' \in S'_2$. Let $r' \in S'_1$. We apply $a_0^{m-1-r} a_1 a_0^{r-1}$ as before, which changes the selector to $(q_1 + 1) \bmod m$, but does not change the core S'_1 of the first subset. We repeat this until selector 0 is reached. Then we still have $p' \notin S'_1$ but $p' \in Y'_2$, where Y'_2 is the core of the second subset. We apply a_0^{n-1-p}. Then the first subset does not have final state $(t, (n-1)')$, but the second one does.

Finally, we need to distinguish the initial subset from the other subsets. For the initial subset, we observe that applying either $a_0 a_1 a_0^{n-1}$ or $a_1 a_0^{n-1}$ results in $\{(1, s'), (1, 0'), (1, 1')\}$. On the other hand, every other subset that we have to consider has a non-empty core S'_2. If $S'_2 = \{(n-1)'\}$ then we apply $a_0 a_1 a_0^{n-1}$, otherwise $a_1 a_0^{n-1}$. In both cases, this results in a subset that has a different core than $\{1'\}$, thus can be distinguished from $\{(1, s'), (1, 0'), (1, 1')\}$ as we showed before. □

7. Conclusions

We have determined the state complexity of overlap assembly of regular languages. The complexity is similar to that of the ordinary binary product (concatenation), yet, in contrast with that, requires a growing linear alphabet to reach the maximum. Nevertheless, a binary alphabet suffices for an exponential state complexity, quite close to the upper bound, whereas for a unary alphabet it does not exceed $m + n$.

In the general case, we left the border cases of $m = 1$ or $n \leq 2$, where our general witness family does not work. When $m = 1$ or $n = 1$, one of the languages is either universal or empty. An empty language causes the overlap assembly to be empty. A universal language yields the overlap assembly equal to $\Sigma^* L_n$ or $L_m \Sigma^*$, which are special cases of the product, and where the tight upper bounds on the state complexity are 2^{n-1} and m, respectively. The case of $n = 2$ is left open.

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada under grants No. OGP0000871 and R2824A01, and by the National Science Centre, Poland, under project number 2014/15/B/ST6/00615.

References

[1] J. A. Brzozowski, In search of the most complex regular languages, *Int. J. Found. Comput. Sci.* **24**(6) (2013) 691–708.

[2] J. A. Brzozowski, Towards a theory of complexity of regular languages, *J. Automata, Languages and Combinatorics* **23**(1–3) (2018) 67–101.

[3] J. A. Brzozowski, L. Kari, B. Li and M. Szykuła, State complexity of overlap assembly, *CIAA 2018*, ed. C. Câmpeanu, LNCS, Vol. 10977 (Springer, 2018), pp. 109–120.

[4] A. Carausu and G. Paun, String intersection and short concatenation, *Revue Roumaine des Mathematiques Pures et Appliquees* **26** (1981) 713–726.

[5] D. Cheptea, C. Martín-Vide and V. Mitrana, A new operation on words suggested by DNA biochemistry: hairpin completion, *Proc. Transgressive Computing*, *TC* (2006), pp. 216–228.

[6] E. Csuhaj-Varjú, I. Petre and G. Vaszil, Self-assembly of strings and languages, *Theoret. Comput. Sci.* **374**(1–3) (2007) 74–81.

[7] A. R. Cukras, D. Faulhammer, R. J. Lipton and L. F. Landweber, Chess games: A model for RNA based computation, *Biosystems* **52**(1–3) (1999) 35–45.

[8] M. Domaratzki, Minimality in template-guided recombination, *Inform. Comput.* **207**(11) (2009) 1209–1220.

[9] S. K. Enaganti, O. H. Ibarra, L. Kari and S. Kopecki, On the overlap assembly of strings and languages, *Nat. Comput.* **16**(1) (2016) 175–185.

[10] S. K. Enaganti, O. H. Ibarra, L. Kari and S. Kopecki, Further remarks on DNA overlap assembly, *Inform. Comput.* **253**(1) (2017) 143–154.

[11] S. K. Enaganti, L. Kari and S. Kopecki, A formal language model of DNA polymerase activity, *Fundamenta Informaticae* **138** (2015) 179–192.

[12] D. Faulhammer, A. R. Cukras, R. J. Lipton and L. F. Landweber, Molecular computation: RNA solutions to chess problems, *Proc. Natl. Acad. Sci.* **97**(4) (2000) 1385–1389.

[13] G. Franco, A polymerase based algorithm for SAT, *Theoretical Computer Science*, eds. M. Coppo, E. Lodi and G. Pinna, LNCS, Vol. 3701 (Springer, Berlin, Heidelberg, 2005), pp. 237–250.

[14] G. Franco, C. Giagulli, C. Laudanna and V. Manca, DNA extraction by XPCR, *Proc. DNA Computing*, (*DNA 11*), eds. C. Ferretti, G. Mauri and C. Zandron, LNCS, Vol. 3384 (2005), pp. 104–112.

[15] G. Franco and V. Manca, Algorithmic applications of XPCR, *Nat. Comput.* **10**(2) (2011) 805–819.

[16] G. Franco, V. Manca, C. Giagulli and C. Laudanna, DNA recombination by XPCR, *Proc. DNA Computing*, (*DNA 12*), eds. A. Carbone and N. A. Pierce, LNCS, Vol. 3892 (2006), pp. 55–66.

[17] Y. Gao, N. Moreira, R. Reis and S. Yu, A survey on operational state complexity, *J. Autom. Lang. Comb.* **21**(4) (2016) 251–310.

[18] J. S. Golan, *The Theory of Semirings with Applications in Mathematics and Theoretical Computer Science* (Addison-Wesley Longman Ltd., 1992).

[19] M. Holzer, S. Jakobi and M. Kutrib, The chop of languages, *Theoret. Comput. Sci.* **682** (2017) 122–137.

[20] M. Holzer and S. Jakobi, Chop operations and expressions: Descriptional complexity considerations, *DLT*, LNCS (Springer, 2011), pp. 264–275.

[21] M. Holzer and S. Jakobi, State complexity of chop operations on unary and finite languages, *DCFS*, LNCS, Vol. 7386 (Springer, 2012), pp. 169–182.

[22] S. Hussini, L. Kari and S. Konstantinidis, Coding properties of DNA languages, *Proc. DNA Computing*, (*DNA 7*), eds. N. Jonoska and N. C. Seeman, LNCS, Vol. 2340 (2002), pp. 57–69.

[23] M. Ito and G. Lischke, Generalized periodicity and primitivity for words, *Math. Log. Quart.* **53**(1) (2007) 91–106.

[24] P. D. Kaplan, Q. Ouyang, D. S. Thaler and A. Libchaber, Parallel overlap assembly for the construction of computational DNA libraries, *J. Theoret. Biol.* **188**(3) (1997) 333–341.

[25] L. Kari, R. Kitto and G. Thierrin, Codes, involutions, and DNA encodings, *Formal and Natural Computing*, eds. W. Brauer, H. Ehrig, J. Karhumäki and A. Salomaa, LNCS, Vol. 2300 (2002), pp. 376–393.

[26] S. Kopecki, On iterated hairpin completion, *Theoret. Comput. Sci.* **412**(29) (2011) 3629–3638.

[27] V. Manca and G. Franco, Computing by polymerase chain reaction, *Math. Biosci.* **211**(2) (2008) 282–298.

[28] F. Manea, C. Martín-Vide and V. Mitrana, On some algorithmic problems regarding the hairpin completion, *Discr. Appl. Math.* **157** (2009) 2143–2152.

[29] F. Manea and V. Mitrana, Hairpin completion versus hairpin reduction, *Proc. Computability in Europe*, *CiE*, eds. S. B. Cooper, B. Löwe and A. Sorbi, LNCS, Vol. 4497 (2007), pp. 532–541.

[30] C. Martín-Vide, G. Păun, J. Pazos and A. Rodríguez-Patón, Tissue P systems, *Theoret. Comput. Sci.* **296**(2) (2003) 295–326.

[31] A. N. Maslov, Estimates of the number of states of finite automata, *Dokl. Akad. Nauk SSSR* **194** (1970) 1266–1268 (Russian), English translation: Soviet Math. Dokl. **11** (1970) 1373–1375.

[32] Q. Ouyang, P. D. Kaplan, S. Liu and A. Libchaber, DNA solution of the maximal clique problem, *Science* **278**(5337) (1997) 446–449.

[33] W. P. Stemmer, DNA shuffling by random fragmentation and reassembly: in vitro recombination for molecular evolution, *Proc. Natl. Acad. Sci.* **91**(22) (1994) 10747–10751.

[34] S. Yu, Q. Zhuang and K. Salomaa, The state complexities of some basic operations on regular languages, *Theoret. Comput. Sci.* **125** (1994) 315–328.