RAIRO-Theor. Inf. Appl. 48 (2014) 39–59
DOI: 10.1051/ita/2013048

Available online at:
www.rairo-ita.org

BIDIRECTIONAL STRING ASSEMBLING SYSTEMS

Martin Kutrib[1] and Matthias Wendlandt[1]

Abstract. We introduce and investigate several variants of a bidirectional string assembling system, which is a computational model that generates strings from copies of assembly units. The underlying mechanism is based on two-sided piecewise assembly of a double-stranded sequence of symbols, where the upper and lower strand have to match. The generative capacities and the relative power of the variants are our main interest. In particular, we prove that bidirectional string assembling system generate languages not represented as any finite concatenation of one-sided string assembling systems. The latter build an infinite, strict and tight concatenation hierarchy. Moreover, it is shown that even the strongest system in question can only generate NL languages, while there are unary regular languages that cannot be derived. Furthermore, a finite strict hierarchy with respect to the different variants considered is shown and closure properties of the languages generated are presented.

Mathematics Subject Classification. 68Q05, 68Q42.

1. Introduction

String assembling systems [5] are language generating devices, where the raw material processed is double stranded in such a way that corresponding symbols have to be identical. In this way the correctness of the complementation of a strand is naturally given. With the advent of investigations of devices and operations that are inspired by the study of biological processes, and the growing interest in nature-based problems modeled in formal systems this old control mechanism of the generation process has been rekindled. The famous Post's Correspondence

Keywords and phrases. String assembling, double-stranded sequences, concatenation hierarchy, stateless, multi-head finite automata, closure properties.

[1] Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany.
{Martin.Kutrib,matthias.wendlandt}@informatik.uni-giessen.de

Article published by EDP Sciences

© EDP Sciences 2014

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

Problem can be seen as a first study showing the power of double-stranded string generation [8]. Basically, the idea of string assembling systems is to have basic assembly units that are pairs of substrings which have to be synchronously connected to the matching upper and lower strand in order to extend the string assembled to the right. In general, we provide two further control mechanisms. First, we require that the first symbol of a substring has to be the same as the last symbol of the strand to which it is connected. One can imagine that both symbols are glued together one at the top of the other and, thus, just one appears in the final string. Second, as for the notion of strictly locally testable languages [6,10], we distinguish between assembly units that may appear at the beginning, during, and at the end of the assembling process.

Here we generalize the devices to two-sided ones and introduce several types of bidirectional string assembling systems. The basic variant has a set of assembly units for each side. The assembling of units to the left and right is not synchronized. The control of the generation process is weakened for centralized variants, where units are still assembled at both ends, but all units are centralized into one set, that is, it is not distinguished whether a unit may be assembled at the left or right. A stronger control of the generation process is obtained by synchronizing the assembling of units at both ends. Here we consider systems where units are assembled simultaneously at the right and at the left. The last variant in question are synchronized bidirectional string assembling systems, where the units simultaneously assembled must be related.

A related recent approach of double-stranded string generation are sticker systems [1,3,7], where basically the assembly units may be more complex dominoes. So, the generation process of sticker systems is subject to control mechanisms and restrictions given, for example, by the shape of the pieces. The assembling units of string assembling systems do not form dominoes. Although the basic mechanisms of both types of systems are closely related, their generative capacities differ essentially. In [5] it is shown that the copy language $\{\$_1 w \$_2 w \$_3 \mid w \in \Sigma^+\}$ is generated by a one-sided string assembling system, while it is not generated by any sticker system. On the other hand, sticker systems can generate all regular languages, whereas it turns out that there is a regular language not generated even by the strongest bidirectional string assembling system under consideration.

The paper is organized as follows: The next section contains preliminaries and the definition of string assembling systems as well as some meaningful examples that are a starting point to explore the generative capacity of the systems. In particular, by a known result from [5] and an example we obtain that bidirectional string assembling systems are strictly stronger than unidirectional ones. Section 3 is devoted to the question whether or not bidirectional string assembling systems generate merely the concatenation of two languages generated by one-sided systems. It is shown that bidirectional systems are strictly more powerful, but also that they can generate languages which have no representation as any finite concatenation of one-sided system languages. In addition, an infinite, strict, and tight concatenation hierarchy of one-sided systems is obtained. Then Section 4 deals

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

with synchronized systems. In this way, the number of units assembled is identical for the right and left part of the strand. It is shown that synchronization yields strictly more powerful systems regardless of whether they are centralized or not. In turn the generative power is lowered when systems are centralized. As an upper bound on the generative capacity of bidirectional string assembling system a construction is given that allows to simulate such a system by some nondeterministic one-way five-head finite automaton. In Section 5 we investigate synchronized bidirectional string assembling systems, where the units simultaneously assembled must be related. It turns out that for such devices centralized and non-centralized versions are equally powerful. Moreover, they form the strongest system in question. However, nondeterministic two-way four-head finite automata can simulate these systems, which implies that none of the devices in question can generate a language not belonging to the complexity class NL. In particular, all languages generated are context-sensitive. On the other hand, it is shown that there is a unary regular language not belonging to any family of languages generated by string assembling systems. Interestingly, any unary regular language can be generated even by unidirectional string assembling systems if finitely many words are omitted, that is, finitely many errors are suffered. In Section 6 we consider closure properties of the families of languages generated by the systems in question. In particular, the non-closure under five of the six AFL operations is shown. Furthermore we obtain non-closure under complementation. The only positive closure property is for reversal.

2. Preliminaries and definitions

We write Σ^* for the set of all words over the finite alphabet Σ. The empty word is denoted by λ, and $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. The reversal of a word w is denoted by w^R, and for the length of w we write $|w|$. Generally, for a singleton set $\{a\}$ we simply write a. We use $\subseteq$ for inclusions and $\subset$ for strict inclusions.

Basically, a bidirectional string assembling system generates a double-stranded string by assembling substrings to the upper and lower strand at both ends, so that both strands match. Moreover, a substring can only be assembled to the right when its first symbol matches the last symbol of the strand, and to the left when its last symbol matches the first symbol of the strand. In these cases the matching symbols are glued together one at the top of the other. The substrings to be assembled are given by so-called units. The generation has to begin with a unit from the set of initial units. Then it may continue, and when a unit from the sets of ending units is applied, the assembling process in the corresponding direction stops. The generation is said to be valid if and only if both strands are identical when the process stops at the right and at the left.

In the sequel, we denote (incomplete) matching double strands by quintuples (v_1, v_2, u, w_1, w_2), where u is non-empty, either v_1 or v_2, or both are empty and, similarly, either w_1 or w_2, or both are empty. It is understood that u is the part where upper and lower strand match, v_1 (v_2) denotes the part of the upper (lower)

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

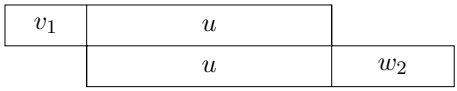


FIGURE 1. Example of an incomplete matching double strand ($v_2 = w_1 = \lambda$).

strand sticking out to the left, and w_1 (w_2) denotes the part of the upper (lower) strand sticking out to the right (see Fig. 1). The set of all (incomplete) matching double strands over an alphabet Σ is denoted by $\Sigma^{\rightleftharpoons}$.

Definition 2.1. A *bidirectional string assembling system* (2SAS) is a sextuple $\langle \Sigma, A, T_l, T_r, E_l, E_r \rangle$, where

1. Σ is the finite, nonempty set of *symbols* or *letters*,
2. $A \subset \Sigma^{\rightleftharpoons}$ is the finite set of *axioms*,
3. $T_l, T_r \subset \Sigma^+ \times \Sigma^+$ are the finite sets of *left and right assembly units*,
4. $E_l \subset \Sigma^+ \times \Sigma^+$ is the finite set of *left ending assembly units* of the forms (uv, u) or (u, uv), where $u \in \Sigma^+$ and $v \in \Sigma^*$, and
5. $E_r \subset \Sigma^+ \times \Sigma^+$ is the finite set of *right ending assembly units* of the forms (vu, u) or (u, vu), where $u \in \Sigma^+$ and $v \in \Sigma^*$.

The next definition formally says how the units are assembled (see Fig. 2).

Definition 2.2. Let $S = \langle \Sigma, A, T_l, T_r, E_l, E_r \rangle$ be a 2SAS. The *derivation relation* $\Rightarrow$ is defined on $\Sigma^{\rightleftharpoons}$ by

1. $(v_1, v_2, u, w_1, w_2) \Rightarrow (v_1, v_2, u', w_1', w_2')$ if
   (a) $uw_1 = ta$, $uw_2 = sb$, and $(ax, by) \in T_r \cup E_r$, for $a, b \in \Sigma$, $x, y, s, t \in \Sigma^*$, and
   (b) $uw_1x = uw_2yz$, $u' = uw_2y$, $w_1' = z$, $w_2' = \lambda$, or $uw_2y = uw_1xz$, $u' = uw_1x$, $w_1' = \lambda$, $w_2' = z$, for $z \in \Sigma^*$.
2. $(v_1, v_2, u, w_1, w_2) \Rightarrow (v_1', v_2', u', w_1, w_2)$ if
   (a) $v_1u = at$, $v_2u = bs$, and $(xa, yb) \in T_l \cup E_l$, for $a, b \in \Sigma$, $x, y, s, t \in \Sigma^*$, and
   (b) $xv_1u = zyv_2u$, $u' = yv_2u$, $v_1' = z$, $v_2' = \lambda$, or $yv_2u = zxv_1u$, $u' = xv_1u$, $v_1' = \lambda$, $v_2' = z$, for $z \in \Sigma^*$.

A derivation is said to be *successful* if it initially starts with an axiom from A, continues with assembling units from T_l and T_r, and ends with having assembled an ending unit from E_l to the left and an ending unit from E_r to the right. The process necessarily stops when both ending units have been added. The sets T_l, T_r, E_l, and E_r are not necessarily disjoint.

The *language* $L(S)$ *generated* by S is defined to be the set

$$L(S) = \{ w \in \Sigma^+ \mid (v_1, v_2, u, w_1, w_2) \Rightarrow^* (\lambda, \lambda, w, \lambda, \lambda) \text{ is a } \textit{successful} \text{ derivation} \}$$

where $\Rightarrow^*$ refers to the reflexive, transitive closure of the derivation relation $\Rightarrow$.

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

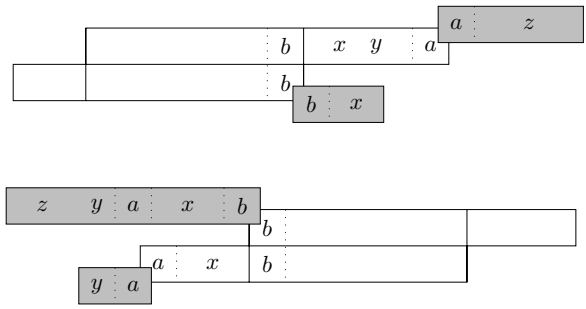


FIGURE 2. Examples of assembling a unit (az, bx) at the right (*top*) and a unit $(zyaxb, ya)$ at the left (*bottom*).

A 2SAS $\langle \Sigma, A, T_l, T_r, E_l, E_r \rangle$ is said to be a *unidirectional string assembling system* (1SAS) if $T_l = \emptyset$, $E_l = \{(a, a) \mid a \in \Sigma\}$, and all axioms are of the form $(\lambda, \lambda, u, w_1, w_2)$. That is, except for ending units that actually add no symbols, units are only assembled at the right.

In order to clarify the notation we give two examples.

Example 2.3. The 2SAS $S = \langle \{a, b\}, A, T_l, T_r, E_l, E_r \rangle$ generates the deterministic context-free language $\{a^n b^n \mid n \geq 1\} \cup a^+$, where

$$A = \{(\lambda, \lambda, a, \lambda, \lambda), (\lambda, \lambda, b, \lambda, \lambda)\},$$
$$T_l = \{(bb, b), (ab, b), (aa, bb), (a, ab), (a, aa)\}, \quad T_r = \{(aa, aa)\},$$
$$E_l = \{(a, a)\}, \quad E_r = \{(b, b), (a, a)\}.$$

The units in T_l are used to generate the words from $\{a^n b^n \mid n \geq 1\}$ to the left of the axiom $(\lambda, \lambda, b, \lambda, \lambda)$. The units in T_r are used to generate the words of the form a^+ to the right of the axiom $(\lambda, \lambda, a, \lambda, \lambda)$. Starting with this axiom it is only possible to add units (aa, aa) to the right, and end up at both ends with the units $(a, a) \in E_l$ and $(a, a) \in E_r$. If the unit (a, aa) would be used at the left side, the lower strand gets longer, but cannot be completed in the upper strand. If the axiom containing a b is used, first the suffix ab^n is generated by repeatedly assembling unit (bb, b) followed by one unit (ab, b). After that the unit (aa, bb) has to be assembled exactly as many times as the unit (bb, b) before. Now the only possibility is to add (a, ab) and complete the lower strand with (a, aa). Because the units of T_r cannot be used at the left side, (aa, aa) cannot be added.

Example 2.4. The 2SAS $S = \langle \{a, b\}, A, T_l, T_r, E_l, E_r \rangle$ generates the deterministic context-free language $\{a^n b^n a^m \mid n, m \geq 1\}$, where

$$A = \{(\lambda, \lambda, ba, \lambda, \lambda)\},$$
$$T_l = \{(bb, b), (ab, b), (aa, bb), (a, ab), (a, aa)\}, \quad T_r = \{(aa, aa)\},$$
$$E_l = \{(a, a)\}, \quad E_r = \{(a, a)\}.$$

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

Here, basically, the same mechanism as in the previous example is used. The main difference is that a derivation starts with an axiom consisting of a b at the left side and an a at the right, which leads to the generation of the concatenation of the languages $\{a^n b^n \mid n \geq 1\}$ and a^+.

Now we turn to weaken the control of the generation process. So-called *centralized* bidirectional string assembling systems (C-2SAS) cannot distinguish between the units that may be assembled at the left and right. To this end, formally it suffices to require $T_l = T_r$ and $E_l = E_r$.

Example 2.5. The C-2SAS $S = \langle \{a, b\}, A, T_l, T_r, E_l, E_r \rangle$ generates the context-free language $\{a^n b^n a^m b^m \mid n, m \geq 1\}$, where

$$A = \{(\lambda, \lambda, ba, \lambda, \lambda)\}, \quad T_l = T_r = \{(bb, b), (ab, b), (aa, bb), (a, ab), (a, aa)\},$$
$$E_l = \{(a, a)\}, \quad E_r = \{(b, b)\}.$$

As in Example 2.3, the words from $\{a^n b^n \mid n \geq 1\}$ are generated to the left of the axiom $(\lambda, \lambda, ba, \lambda, \lambda)$. In addition, the same units are used to generate a second word from the same set to the right. When a word from $\{a^k b^k a^l b^l \mid k, l \geq 1\}$ has been assembled, the derivation cannot be extended successfully. Though the unit (bb, b) can repeatedly be added at the right, the lower strand cannot be completed afterwards. Similarly, the unit (a, aa) can be added at the left, but the upper strand cannot be completed afterwards.

3. Concatenation hierarchy of 1SAS

In [5] it has been shown that the language $\{a^n b^n a^m \mid n, m \geq 1\}$ is not generated by any 1SAS. Therefore, by Example 2.4 we obtain the following inclusion.

Theorem 3.1. *The family of languages generated by 1SAS is properly included in the family of languages generated by 2SAS.*

While the previous result is evident, we next turn to the question whether or not 2SAS generate merely the concatenation of two 1SAS languages. The next theorem shows that 2SAS are in fact at least as powerful as two "concatenated" 1SAS. However, the rest of the section reveals not only that 2SAS are strictly more powerful, but also that they can generate languages which have no representation as any finite concatenation of 1SAS languages. In addition, an infinite, strict, and tight concatenation hierarchy of 1SAS is obtained.

Theorem 3.2. *Let S_1, S_2 be two 1SAS. There exists a 2SAS that generates the concatenation $L(S_1)L(S_2)$.*

Proof. Let

$$S_1 = \langle \Sigma', A', \emptyset, T'_r, \{(a, a) \mid a \in \Sigma'\}, E'_r \rangle \text{ and}$$
$$S_2 = \langle \Sigma'', A'', \emptyset, T''_r, \{(a, a) \mid a \in \Sigma''\}, E''_r \rangle.$$

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

Then the 2SAS $\langle \Sigma' \cup \Sigma'', A, T_l, T_r, E_l, E_r \rangle$ generates the concatenation $L(S_1)L(S_2)$, where

$$\begin{aligned} A &= \{(v_1, \lambda, v_2u, w_1, w_2) \mid (v_1v_2, v_2) \in E'_r, (\lambda, \lambda, u, w_1, w_2) \in A''\}, \\ &\quad \cup \{(\lambda, v_2, v_1u, w_1, w_2) \mid (v_1, v_2v_1) \in E'_r, (\lambda, \lambda, u, w_1, w_2) \in A''\}, \\ T_l &= T'_r, \quad T_r = T''_r, \\ E_l &= \{(uw_1, uw_2) \mid (\lambda, \lambda, u, w_1, w_2) \in A'\}, \text{ and} \\ E_r &= E''_r. \end{aligned}$$

□

In [5] it has been shown that the family of languages generated by 1SAS is not closed under concatenation. In order to derive the infinite, strict, and tight concatenation hierarchy of 1SAS we consider the four languages

$$\begin{aligned} L_0 &= \{a^nb^n \mid n \geq 1\} \cup d^+, & L_1 &= \{c^nd^n \mid n \geq 1\} \cup a^+, \\ L_2 &= \{b^na^n \mid n \geq 1\} \cup c^+, & L_3 &= \{d^nc^n \mid n \geq 1\} \cup b^+. \end{aligned}$$

and, for $i \geq 1$, their cyclic concatenations $L^{(1)} = L_0$ and $L^{(i+1)} = L^{(i)}L_{i \bmod 4}$.

The next lemma prepares to show that $L^{(i+1)}$ cannot be written as concatenation of at most i 1SAS languages.

Lemma 3.3. *Let $r \geq 0$ be a constant, and $L \subseteq \Sigma^*$ be a language so that*

$$\begin{aligned} &\{a^nb^n \mid n \geq 1\}a^+(ba)^r \subseteq L \cap \{a, b\}^+ \subseteq \{a^nb^n \mid n \geq 1\}a^+(b^+a^+)^r \text{ or} \\ &\{a^nb^n \mid n \geq 1\}a^+(ba)^rb \subseteq L \cap \{a, b\}^+ \subseteq \{a^nb^n \mid n \geq 1\}a^+(b^+a^+)^rb^+. \end{aligned}$$

Then L is not generated by any 1SAS.

Proof. In contrast to the assertion we assume that L is generated by some 1SAS $S = \langle \Sigma, A, T_l, T_r, E_l, E_r \rangle$. Recall that by definition $T_l = \emptyset$, $E_l = \{(x, x) \mid x \in \Sigma\}$, and all axioms are of the form $(\lambda, \lambda, u, w_1, w_2)$. Let $s + 1$ be the length of the longest string appearing in a unit from A, T_r, or E_r.

Next, we define subsets of T_r as follows. $T_e(a) = \{(a^i, a^i) \in T_r \mid i \geq 2\}$, $T_u(a) = \{(a^i, a^j) \in T_r \mid i > j \geq 1\}$, and $T_d(a) = \{(a^i, a^j) \in T_r \mid 1 \leq i < j\}$.

If $T_u(a)$ contains a unit (a^{i_1}, a^{j_1}) and $T_d(a)$ a unit (a^{i_2}, a^{j_2}), then the assembling of $j_2 - i_2$ units (a^{i_1}, a^{j_1}) and $i_1 - j_1$ units (a^{i_2}, a^{j_2}) extends the upper as well as the lower string by $\ell = i_1j_2 - i_2j_1 - i_1 - j_2 + i_2 + j_1$ symbols a. If $T_e(a)$ contains a unit (a^i, a^i), its assembling extends the upper as well as the lower string by $\ell = i - 1$ symbols a. Let $(\lambda, \lambda, u, w_1, w_2)$ be the unit from A which starts the generation of a string with prefix a^nb^na, where n is large enough, that is, $u, w_1, w_2 \in a^*$. Then the upper and lower string can be extended by ℓ symbols yielding $(\lambda, \lambda, a^{|u|+\ell}, w_1, w_2)$, while the remaining generation is unchanged. Therefore, a string with prefix $a^{n+\ell}b^na$ is generated as well, which is a contradiction. We conclude that $T_e(a)$ and one of $T_u(a)$ and $T_d(a)$ must be empty. Without loss of generality, we assume $T_d(a)$ is empty.

Now we claim that there are no units of the form (b, a^{j+1}) in T_r, for $j \geq 1$.

In order to prove the claim we assume contrarily that such a unit exists in T_r. Let $(\lambda, \lambda, u, w_1, w_2) \Rightarrow^* (\lambda, \lambda, a^{n_1}, a^{n-n_1}b^i, \lambda)$, $i \leq s$ and n large enough, be the

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

initial part of a successful generation which continues $(\lambda, \lambda, a^{n_1}, a^{n-n_1}b^i, \lambda) \Rightarrow^* (\lambda, \lambda, a^n b^n av, \lambda, \lambda)$, for $v \in \{a, b\}^*$. Since $T_e(a)$ and $T_d(a)$ are empty, such an initial part exists. Since n is large enough, some unit (a^{p+1}, a^{q+1}), $p > q \geq 0$ is assembled at least once. Assembling the unit (a^{p+1}, a^{q+1}) j times more at the beginning of the generation yields $(\lambda, \lambda, u, w_1, w_2) \Rightarrow^* (\lambda, \lambda, a^{n_1+j\cdot q}, a^{n+j\cdot p-(n_1+j\cdot q)}b^i, \lambda)$. Continuing the generation by assembling $(p-q)$ times the unit (b, a^{j+1}) yields $(\lambda, \lambda, a^{n_1+j\cdot q+(p-q)\cdot j}, a^{n+j\cdot p-(n_1+j\cdot q)-(p-q)\cdot j}b^i, \lambda) = (\lambda, \lambda, a^{n_1+j\cdot p}, a^{n-n_1}, \lambda)$. So, the generation

$$(\lambda, \lambda, u, w_1, w_2) \Rightarrow^* \left(\lambda, \lambda, a^{n_1+j\cdot p}, a^{n-n_1}b^i, \lambda\right) \Rightarrow^* \left(\lambda, \lambda, a^{n+j\cdot p}b^n av, \lambda, \lambda\right)$$

is successful as well. The contradiction concludes the proof of the claim.

Now we are prepared to prove the lemma. We denote the word $a^n b^n a^m (ba)^r$ or $a^n b^n a^m (ba)^r b$ by $w_{n,m}$, and consider these words for n large enough and m arbitrarily large compared with n. We distinguish three cases.

Case 1. For all constants d there is a word $w_{n,m}$ so that the initial part of its generation yields an (incomplete) matching double strand $(\lambda, \lambda, u, w_1, \lambda)$, where $uw_1 = a^n b^n a^{x_1}$ and $\min\{x_1, |w_1|\} \geq d$.
Since $T_e(a)$ and $T_d(a)$ are empty, we obtain the derivation $(\lambda, \lambda, u, w_1, \lambda) \Rightarrow^* (\lambda, \lambda, a^n b^n a^{x_2}, a^{m-x_2}b^i, \lambda)$, where $i \leq s$ and $m - x_2 \geq d$. In order to complete the lower strand for $d > (2r+1)s$, there must exists a unit of the form (b, a^{j+1}) or (a, a^{j+1}), $j \geq 1$. The latter is impossible since $T_d(a)$ is empty, the former is impossible by the claim above.

Case 2. For all constants d there is a word $w_{n,m}$ so that the initial part of its generation yields an (incomplete) matching double strand $(\lambda, \lambda, u, \lambda, w_2)$, where $uw_2 = a^n b^n a^{x_1}$ and $\min\{x_1, |w_2|\} \geq d$.
Since $T_e(a)$ and $T_d(a)$ are empty, we obtain $(\lambda, \lambda, a^{n_1}, a^{n-n_1}b^i, \lambda)$ as initial part of the generation, where $i \leq s$. If the first a of the third block is generated in the upper string first, the generation continues $(\lambda, \lambda, a^{n_1}, a^{n-n_1}b^i, \lambda) \Rightarrow^* (\lambda, \lambda, a^{n_2}, a^{n-n_2}b^n a^{i'}, \lambda)$, or $(\lambda, \lambda, a^{n_1}, a^{n-n_1}b^i, \lambda) \Rightarrow^* (\lambda, \lambda, a^n b^{n_3}, b^{n-n_3}a^{i'}, \lambda)$, where $i' \leq s$. Then there must be a unit in $T_d(a)$ since otherwise the situation $\min\{x_1, |w_2|\} \geq d$ is unreachable. But $T_d(a)$ is empty.
If the first a of the third block is generated in the lower string first, the generation continues $(\lambda, \lambda, a^{n_1}, a^{n-n_1}b^i, \lambda) \Rightarrow^* (\lambda, \lambda, a^n b^{n_2}, \lambda, b^{n-n_2}a^{i'})$, where $i' \leq s$. Then, for $d > n \cdot s$ there must be a unit (b, a^{j+1}), $j \geq 1$. Otherwise the situation $\min\{x_1, |w_2|\} \geq d$ is unreachable. But such a unit does not exist by the claim above.

Case 3. There is a constant d so that for all words $w_{n,m}$ the initial parts of their generation yields (incomplete) matching double strands $(\lambda, \lambda, a^n b^n a^x, w_1, w_2)$, with $x \geq 1$, $w_1, w_2 \in a^*$ with $||w_1| - |w_2|| < d$. In this case, either $T_e(a)$ or $T_d(a)$ must be nonempty, a contradiction.

So, in any case we obtain a contradiction to the assumption that L is generated by some 1SAS, which concludes the proof. □

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

Theorem 3.4. *Let $k \geq 1$ be a constant. Then $L^{(k)}$ can be represented as concatenation of k 1SAS languages, but there is no representation as concatenation of less than k 1SAS languages.*

Proof. The four languages L_0, L_1, L_2, L_3 are all 1SAS languages. Since $L^{(k)}$ is defined as concatenation of k of these languages, the first part of the assertion follows.

For the second part, we know by Lemma 3.3 that the concatenation L_0L_1 is not generated by any 1SAS. So, the assertion follows for $k = 2$. Concluding inductively, let $k \geq 3$ and assume that $L^{(k-1)}$ cannot be represented as concatenation of less than $k-1$ 1SAS languages.

It remains to be shown that $L^{(k)}$ cannot be represented as concatenation of less than k 1SAS languages. Lemma 3.3 shows that $L^{(k)}$ is not a 1SAS language and, thus, we consider its representation as concatenation of a 1SAS language L_l and a (not necessarily 1SAS) language L_r. That is, $L^{(k)} = L_lL_r$, where L_l is generated by some 1SAS.

First we notice that for any fixed k, the possible numbers and orders of a-, b-, c-, and d-blocks appearing in words from $L^{(k)}$ is uniquely determined from left to right. For example, any word may begin with a d-block, or an a-block followed by a b-block. These are followed by an a-block, or a c-block followed by a d-block, and so on. The possible sequences of blocks are called *signatures*.

Since all words in $L^{(k)}$ begin either with letter a or with letter d, all words in L_l begin with a or d as well.

Next we consider all words in $L^{(k)} \cap \{a, b\}^+$. If there is some word a^i, $i \geq 1$, in L_l, there is a matching word $a^{n-i}b^na^mv \in L_r$, where $v \in \{a, b\}^+$. Now the concatenation of any word in L_l beginning with a d and $a^{n-i}b^na^mv$ cannot belong to $L^{(k)}$ since it has an invalid signature. Similarly, if there is some word a^nb^j, $n > j \geq 1$, in L_l, there is a matching word $b^{n-j}a^mv \in L_r$. Again, the concatenation of any word in L_l beginning with a d and $b^{n-j}a^mv$ cannot belong to $L^{(k)}$ since it has an invalid signature. By Lemma 3.3, the words in $L_l \cap \{a, b\}^+$ cannot cover all prefixes $a^nb^na^m$, $n, m \geq 1$. We conclude that there are words of the form $a^nb^na^\ell$, $n \geq 1$ and $\ell \geq 0$, in L_l, and there are matching words $a^{\ell'}v$, where $\ell' \geq 1$ and $v \in \{a, b\}^+$. Moreover, if there is a word of the form $a^nb^na^\ell u$ in L_l, where $u \in \{a, b, c, d\}^*$, then the concatenation $a^nb^na^\ell u \cdot a^{\ell'}v$ has a valid structure if and only if $u \in a^*$. This implies that $L_l \cap a\{a, b, c, d\}^*$ includes only words $a^nb^na^\ell$, $n \geq 1$, $\ell \geq 0$. In addition, such a word belongs to L_l for any number n.

Now assume that L_r includes words beginning with the letter d, say du, where $u \in \{a, b, c, d\}^*$. Then the concatenation $a^nb^na^\ell \cdot du$ has an invalid signature. Therefore, there is no word in L_r beginning with d, and all words d^j, $j \geq 1$ appearing as prefixes in L_l. Moreover, if there is a word of the form du in L_l, where $u \in \{a, b, c, d\}^*$, then the concatenation $du \cdot a^{\ell'}v$, where $a^{\ell'}v \in L_r$ matches $a^nb^na^\ell$ in L_l, has a valid structure if and only if $u \in a^*$. Therefore, $L_l \cap d\{a, b, c, d\}^*$ includes only words d^ma^ℓ, $m \geq 1$, $\ell \geq 0$, and such a word belongs to L_l for any number m. Above it has already been derived that $L_l \cap a\{a, b, c, d\}^*$ includes only words $a^nb^na^\ell$, and such a word belongs to L_l for any number n.

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

Next consider words in $L^{(k)}$ that are of the forms $a^nb^nc^md^mv$ and $d^nc^md^mv$, where $m, n \geq 1$ and $v \in \{a, b, c, d\}^*$. From above it follows that there are matching words $c^md^mv \in L_r$. Since $a^nb^na^\ell \cdot c^md^mv$ as well as $d^na^\ell \cdot c^md^mv$ belong to $L^{(k)}$ if and only if $\ell = 0$, and language L_l includes only words beginning with a or d, it altogether follows $L_l = \{a^nb^n \mid n \geq 1\} \cup \{d^m \mid m \geq 1\}$, that is, $L_l = L_0$. Furthermore, L_r is up to renaming of letters equal to $L^{(k-1)}$. By the induction hypothesis this shows that $L^{(k)}$ cannot be represented as concatenation of less than k 1SAS languages. □

The concept of bidirectionality of string assembling systems is different from concatenations of 1SAS languages. In fact, there are languages on top of the infinite concatenation hierarchy that are generated by simple 2SAS.

Theorem 3.5. *There is a language generated by a 2SAS which cannot be represented as concatenation of any finite number of 1SAS languages.*

Proof. By Example 2.3, the language $L = \{a^nb^n \mid n \geq 1\} \cup a^+$ is generated by a 2SAS. Now assume in contrast to the assertion that it has a representation as concatenation of some $k \geq 1$ languages generated by the 1SAS $S_1, S_2, \ldots, S_k$: $L = L(S_1)L(S_2)\ldots L(S_k)$. Since L includes infinitely many words of the form a^+, at least one of the systems, say S_i, generates infinitely many of them. None of the systems S_j, with $1 \leq j < i$, generates a word containing the letter b, since otherwise by concatenation a word is obtained where a b is followed by an a. Moreover, none of the systems S_j, with $i < j \leq k$, generates a word containing the letter b. Otherwise two concatenations which differ only by two different words of the form a^+ from $L(S_i)$ and which include the word from $L(S_j)$ containing the letter b, must yield a word not belonging to L.

We conclude that system S_i generates all factors containing the letter b. Since, in addition, $L(S_i)$ contains infinitely many words of the form a^+, Lemma 4.3 is applicable, which shows that there are words generated that do not belong to L, a contradiction. □

4. Synchronized 2SAS

This section is devoted to investigate variants of 2SAS allowing a stronger control of the generation process. Basically, the idea is to synchronize the assembling of units at the right and left. First we consider S-2SAS (SC-2SAS), which are 2SAS (C-2SAS) where units are always assembled simultaneously at the right and at the left. In this way, the number of units assembled is the same for the right and left part of the strand.

Example 4.1. The SC-2SAS $S = \langle\{a, b\}, A, T_l, T_r, E_l, E_r\rangle$ generates the context-free language $\{a^nba^n \mid n \geq 1\}$, where $A = \{(\lambda, \lambda, aba, \lambda, \lambda)\}$, $T_l = T_r = \{(aa, aa)\}$, and $E_l = E_r = \{(a, a)\}$.

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

The only possibility to start with is the axiom $(\lambda, \lambda, aba, \lambda, \lambda)$. Then $n-1$ times the unit (aa, aa) is used, which is synchronously assembled at the left and at the right. Finally, the ending unit (a, a) completes the derivation.

Though for non-synchronized 2SAS the numbers of units assembled at the right and left may differ, they are special cases of synchronized systems. This can be seen as follows. Let $S = \langle \Sigma, A, T_l, T_r, E_l, E_r \rangle$ be a 2SAS. Then we construct an equivalent S-2SAS $S' = \langle \Sigma, A, T'_l, T'_r, E_l, E_r \rangle$ generating $L(S)$ by adding units that actually do not add anything to the strands. That is, we set $T'_l = T_l \cup \{(a, a) \mid a \in \Sigma\}$ and $T'_r = T_r \cup \{(a, a) \mid a \in \Sigma\}$. Clearly, S' generates the language $L(S)$. Since centralized variants meet the condition $T_l = T_r$ and the construction extends T_l as well as T_r by the same set of units, the construction works fine also for C-2SAS and SC-2SAS.

So, in order to separate synchronized from non-synchronized string assembling systems, it suffices to show that the language $\{a^n b a^n \mid n \geq 1\}$ of Example 4.1 is not generated by any 2SAS.

Theorem 4.2. (i) *The family of languages generated by 2SAS is properly included in the family of languages generated by S-2SAS.* (ii) *The family of languages generated by C-2SAS is properly included in the family of languages generated by SC-2SAS.*

Proof. By Example 4.1, the language $L = \{a^n b a^n \mid n \geq 1\}$ is generated by an SC-2SAS and, thus, by an S-2SAS. Assume that L is generated by a 2SAS $S = \langle \Sigma, A, T_l, T_r, E_l, E_r \rangle$. We distinguish two cases. First, let there be an axiom $(v_1, v_2, x_1 b x_2, w_1, w_2) \in A$ with $v_1, v_2, x_1, x_2, w_1, w_2 \in a^*$, so that infinitely many words of L are generated by starting the derivation with this axiom. Since the derivations are not synchronized, the single assembling steps at the right are independent of the steps at the left and *vice versa*. So, we can rearrange the single steps such that all units at the right are assembled before the units at the left, without changing the strings derived. Next, consider two successful derivations $(v_1, v_2, x_1 b x_2, w_1, w_2) \Rightarrow^* (v_1, v_2, x_1 b a^{n_1}, \lambda, \lambda) \Rightarrow^* (\lambda, \lambda, a^{n_1} b a^{n_1}, \lambda, \lambda)$ and $(v_1, v_2, x_1 b x_2, w_1, w_2) \Rightarrow^* (v_1, v_2, x_1 b a^{n_2}, \lambda, \lambda) \Rightarrow^* (\lambda, \lambda, a^{n_2} b a^{n_2}, \lambda, \lambda)$, where $n_1 \neq n_2$. Then there is a successful derivation

$$(v_1, v_2, x_1 b x_2, w_1, w_2) \Rightarrow^* (v_1, v_2, x_1 b a^{n_2}, \lambda, \lambda) \Rightarrow^* (\lambda, \lambda, a^{n_1} b a^{n_2}, \lambda, \lambda),$$

which contradicts our assumption.

Second, consider the case where all axioms containing letter b generate only finitely many words. Then there exists at least one axiom $(v_1, v_2, a^j, w_1, w_2)$ not containing b such that infinitely many words are generated from it. Moreover, for infinitely many of these words the letter b is added at the same side, say at the right. For this case, we consider two *different* successful derivations

$$\left(v_1, v_2, a^j, w_1, w_2\right) \Rightarrow^* \left(\lambda, \lambda, a^{i+j}, w_1, w_2\right) \Rightarrow^* \left(\lambda, \lambda, a^{i+j+k} b a^{i+j+k}, \lambda, \lambda\right)$$

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

and $(v_1, v_2, a^j, w_1, w_2) \Rightarrow^* (\lambda, \lambda, a^{i'+j}, w_1, w_2) \Rightarrow^* (\lambda, \lambda, a^{i'+j+k'}ba^{i'+j+k'}, \lambda, \lambda)$. We conclude that

$$\left(v_1, v_2, a^j, w_1, w_2\right) \Rightarrow^* \left(\lambda, \lambda, a^{i+j}, w_1, w_2\right) \Rightarrow^* \left(\lambda, \lambda, a^{i+j+k'}ba^{i'+j+k'}, \lambda, \lambda\right)$$

is successful as well. If $i \neq i'$ the word generated does not belong to L. Therefore, i is equal to i' for any derivation starting with the axiom in question.

Next, subsets of T_r are defined as follows: $T_{r,e}(a) = \{(a^i, a^i) \in T_r \mid i \geq 2\}$, $T_{r,u}(a) = \{(a^i, a^j) \in T_r \mid i > j \geq 1\}$, and $T_{r,l}(a) = \{(a^i, a^j) \in T_r \mid 1 \leq i < j\}$.

If $T_{r,u}(a)$ contains a unit (a^{i_1}, a^{j_1}) and $T_{r,l}(a)$ a unit (a^{i_2}, a^{j_2}), then the assembling of $j_2 - i_2$ units (a^{i_1}, a^{j_1}) and $i_1 - j_1$ units (a^{i_2}, a^{j_2}) extends the upper as well as the lower string by $\ell = i_1 j_2 - i_2 j_1 - i_1 - j_2 + i_2 + j_1$ symbols a. If $T_{r,e}(a)$ contains a unit (a^i, a^i), its assembling extends the upper as well as the lower string by $\ell = i - 1$ symbols a. So, the derivation above can successfully be extended as

$$\begin{aligned}\left(v_1, v_2, a^j, w_1, w_2\right) \Rightarrow^* \left(\lambda, \lambda, a^{i+j}, w_1, w_2\right) \Rightarrow^* \\ \left(\lambda, \lambda, a^{i+j+\ell}, w_1, w_2\right) \Rightarrow^* \left(\lambda, \lambda, a^{i+j+\ell+k'}ba^{i+j+k'}, \lambda, \lambda\right)\end{aligned}$$

which yields a contradiction. So, $T_{r,e}(a)$ and one of $T_{r,u}(a)$ and $T_{r,l}(a)$ must be empty. Without loss of generality, we assume $T_{r,l}(a)$ is empty.

Recall that there are infinitely many derivations of the form above where i must always be equal to i'. Since $T_{r,e}(a)$ and $T_{r,l}(a)$ are empty, any unit containing letters a only increases the length difference between the upper and lower strand. So, the only way to get long matching suffixes following the sole b is to assemble units of the form (b, a^+) that add letters a to the suffix in advance. However, to get long suffixes these units must be assembled before the first a of the suffix appears in the upper strand. Therefore, there are a fixed $i_0 \geq 0$ and infinitely many k' and $m(k')$ so that

$$\begin{aligned}(v_1, v_2, a^j, w_1, w_2) \Rightarrow^* \left(\lambda, \lambda, a^{i+j}, w_1, w_2\right) \Rightarrow^* \left(\lambda, \lambda, a^{i+j+k'}b, \lambda, a^{i_0}\right) \Rightarrow^* \\ \left(\lambda, \lambda, a^{i+j+k'}b, \lambda, a^{m(k')}\right) \Rightarrow^* \left(\lambda, \lambda, a^{i+j+k'}ba^{i+j+k'}, \lambda, \lambda\right)\end{aligned}$$

are successful derivations. Hence, for two different k'_1 and k'_2 we have

$$\begin{aligned}\left(v_1, v_2, a^j, w_1, w_2\right) \Rightarrow^* \left(\lambda, \lambda, a^{i+j}, w_1, w_2\right) \Rightarrow^* \left(\lambda, \lambda, a^{i+j+k'_1}b, \lambda, a^{i_0}\right) \Rightarrow^* \\ \left(\lambda, \lambda, a^{i+j+k'_1}b, \lambda, a^{m(k'_2)}\right) \Rightarrow^* \left(\lambda, \lambda, a^{i+j+k'_1}ba^{i+j+k'_2}, \lambda, \lambda\right).\end{aligned}$$

The contradiction shows that L is not generated by any 2SAS and concludes the proof. □

The next result is, to some extend, a pumping lemma for certain languages. It has been shown in [5] for 1SAS. Interestingly, it can be adapted even to synchronized bidirectional string assembling systems provided they are centralized.

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

Lemma 4.3. *Let $L \subseteq \Sigma^*$ be a language generated by an SC-2SAS. If $|a^+ \cap L| = \infty$, for some symbol $a \in \Sigma$, then there exist constants $p, q \geq 1$ such that $a^p v \in L$, $v \in \Sigma^*$, implies $a^{p+lq} v \in L$, for all $l \geq 1$.*

Proof. Let $S = \langle \{a, b\}, A, T_l, T_r, E_l, E_r \rangle$ be an SC-2SAS generating L, that is $T_l = T_r$ and $E_l = E_r$, and $a \in \Sigma$, $|a^+ \cap L| = \infty$, and three sets defined by $T_e = \{(a^i, a^i) \in T_l \mid i \geq 2\}$, $T_u = \{(a^i, a^j) \in T_l \mid i > j \geq 1\}$, and $T_d = \{(a^i, a^j) \in T_l \mid 1 \leq i < j\}$.

Since L includes infinitely many words from a^+, set T_e, or T_u and T_d both are non-empty. Otherwise only finitely many words from a^+ are generated. If T_e is not empty, choose a unit $(a^i, a^i) \in T_e$ and set $q = i - 1$. If, otherwise, T_u and T_d both are non-empty, choose a unit (a^{i_1}, a^{j_1}) from T_u and a unit (a^{i_2}, a^{j_2}) from T_d. Assembling $j_2 - i_2$ units (a^{i_1}, a^{j_1}) and $i_1 - j_1$ units (a^{i_2}, a^{j_2}) extends the upper as well as the lower string by $i_1 j_2 - i_2 j_1 - i_1 - j_2 + i_2 + j_1$ symbols. In this case set $q = i_1 j_2 - i_2 j_1 - i_1 - j_2 + i_2 + j_1$.

Consider the strings $B = \{v_i u w_i \mid i \in \{1, 2\}$ and $(v_1, v_2, u, w_1, w_2) \in A\}$ and set p to be the length of the longest string appearing in B or a unit from $T_l \cup E_l$. Then in any successful derivation of a word $a^p v \in L$ there appears an (incomplete) matching double strand of the form $(a^*, \lambda, au', w_1, w_2)$ or $(\lambda, a^*, au', w_1, w_2)$, that is extended to the left only by units of the form $(a^+, a^+) \in (T_l \cup E_l)$. So, assembling l times the unit(s) that extend(s) the upper as well as the lower string by q symbols a at the left of this double strand, and having the remaining derivation unchanged generates the string $a^{p+lq} v$. □

The next theorem applies the lemma to separate centralized from non-centralized string assembling systems.

Theorem 4.4. *(i) The family of languages generated by C-2SAS is properly included in the family of languages generated by 2SAS. (ii) The family of languages generated by SC-2SAS is properly included in the family of languages generated by S-2SAS.*

Proof. Let L be the language $\{a^n b^n \mid n \geq 1\} \cup a^+$, and assume L is generated by an SC-2SAS. Then Lemma 4.3 can be applied with constants $p, q \geq 1$. Since $a^p b^p \in L$, we derive $a^{p+q} b^p \in L$, a contradiction. So, L does not belong to the family of languages generated by SC-2SAS and, thus, does not belong to the family of languages generated by C-2SAS, either. On the other hand, by Example 2.3 language L is generated by a 2SAS and, thus by an S-2SAS. □

Next, we compare the generative capacity of synchronized 2SAS with the computational capacity of nondeterministic one-way multi-head finite automata in order to derive languages that are not generated by any S-2SAS. The following relation has been shown in [4].

Theorem 4.5. *Let S be an S-2SAS. There exists a nondeterministic one-way 5-head finite automaton M that accepts $L(S)$.*

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

5. Synchronized 2SAS with related units

The last variant in question are synchronized bidirectional string assembling systems, where the units which are simultaneously assembled must be related. Formally, we introduce a relation $R \subseteq (T_l \times T_r) \cup (E_l \times E_r)$, so that it is understood that when a unit p is assembled at the left simultaneously with a unit q at the right, then $(p, q) \in R$. We denote such systems by R-2SAS and their centralized versions by RC-2SAS. Though we denote the reversal of a word w by w^R, there will never be a conflict of notation with the relation R.

Example 5.1. The following RC-2SAS $S = \langle \{a, b\}, A, T_l, T_r, E_l, E_r, R\rangle$ generates the context-free language $\{ww^R \mid w \in \{a, b\}^+\}$.

$$\begin{aligned} A &= \{(\lambda, \lambda, aa, \lambda, \lambda), (\lambda, \lambda, bb, \lambda, \lambda)\} \\ T_l &= T_r = \{(bb, bb), (aa, aa), (ab, ab), (ba, ba)\} \\ E_l &= E_r = \{(a, a), (b, b)\} \\ R &= \{((bb, bb), (bb, bb)), ((aa, aa), (aa, aa)), ((ab, ab), (ba, ba)), \\ &\qquad ((ba, ba), (ab, ab)), ((a, a), (a, a)), ((b, b), (b, b))\} \end{aligned}$$

The derivation of ww^R starts in the center of the word with one of the axioms $(\lambda, \lambda, aa, \lambda, \lambda)$ or $(\lambda, \lambda, bb, \lambda, \lambda)$. The relation R associates a unit $(u, u) \in T_l$ with its reversed version $(u^R, u^R) \in T_r$. So, each time a unit of T_l is assembled at the left, the reversed counterpart is assembled at the right. Moreover, the units allow to add an a or a b regardless of the current symbol at the end of the strings. The derivation is simply completed by assembling an ending unit (a, a) or (b, b) at both ends.

While for all variants of bidirectional string assembling systems considered so far the centralized versions are strictly weaker than the non-centralized ones, this difference disappears when the units of synchronized systems are related.

Theorem 5.2. *A language is generated by an R-2SAS if and only if it is generated by an RC-2SAS.*

Proof. Let $S = \langle \Sigma, A, T_l, T_r, E_l, E_r, R\rangle$ be an R-2SAS. To turn it into an equivalent RC-2SAS $S' = \langle \Sigma, A', T'_l, T'_r, E'_l, E'_r, R'\rangle$ we set $A' = A$, $T'_l = T'_r = T_l \cup T_r$ and $E'_l = E'_r = E_l \cup E_r$. In order to make sure that the units from T_r (T_l) are only assembled at the right (left) and similarly for E_r and E_l, the relation R' is used. To this end, it is simply defined to be R. In this way, clearly, in a derivation of S' only pairs of units are assembled that also can be assembled in S. Since the set of axioms is unchanged we obtain $L(S) = L(S')$. □

It is worth mentioning that any S-2SAS is an R-2SAS, where the relation between units is $R = (T_l \times T_r) \cup (E_l \times E_r)$. Theorem 4.5 shows that every language generated by an S-2SAS is accepted by a nondeterministic one-way 5-head finite automaton. It is well known that the latter cannot accept the mirror language $\{ww^R \mid w \in \{a, b\}^+\}$. By Example 5.1 we obtain the following proper inclusions.

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

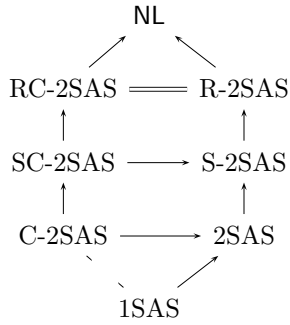


FIGURE 3. Inclusion structure of language families generated by string assembling systems. The arrows indicate strict inclusions. The relation between 1SAS and C-2SAS is an open problem.

Theorem 5.3. *The families of languages generated by S-2SAS and SC-2SAS are properly included in the family of languages generated by R-2SAS (or RC-2SAS).*

At the top of the hierarchy are the synchronized 2SAS with related units. An upper bound on their generative capacity is given by the power of nondeterministic two-way 4-head finite automata. From the complexity point of view, these devices are well explored. In [2] it has been shown that the computational complexity class $\mathsf{NL} = \mathsf{NSPACE}(\log n)$ is characterized by the class of nondeterministic two-way multi-head finite automata. So, together with the next theorem, we obtain that the family of languages generated by R-2SAS is properly included in NL. The theorem is shown in [4].

Theorem 5.4. *Let S be an R-2SAS. There exists a nondeterministic two-way 4-head finite automaton that accepts $L(S)$.*

Finally, we compare all language families generated by some variant of a 2SAS with (sub-) families of the Chomsky hierarchy. Theorem 5.4 revealed that all the former families are properly included in NL and, thus, in the family of context-sensitive languages. In [5] it has been shown that even 1SAS can generate non-context-free languages. The next two lemmas allow to conclude the incomparability of all 2SAS families with the regular and, thus, with the (deterministic) (linear) context-free languages.

Lemma 5.5. *A unary language is generated by an R-2SAS if and only if it is generated by a 1SAS.*

Proof. By Theorem 5.2 it suffices to show that any unary language generated by an RC-2SAS is also generated by a 1SAS. So, let $S = \langle \{a\}, A, T_l, T_r, E_l, E_r, R\rangle$ be a unary RC-2SAS. An equivalent 1SAS $S' = \langle \{a\}, A', T'_l, T'_r, E'_l, E'_r\rangle$ is constructed as follows. By definition we have $T'_l = \emptyset$ and $E'_l = \{(a, a)\}$. Since all units are unary

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

they always fit to the current double strand. Since the assembling is synchronized in S, in S' pairs of units assembled at the same time, that is, which are related, are merged into one unit:

$$T'_r = \Big\{ \left(a^{|u_1u_2|-1}, a^{|v_1v_2|-1}\right) \mid (u_1,v_1) \in T_l, (u_2,v_2) \in T_r, \text{ and } ((u_1,v_1),(u_2,v_2)) \in R \Big\},$$

$$E'_r = \Big\{ \left(a^{|u_1u_2|-1}, a^{|v_1v_2|-1}\right) \mid (u_1,v_1) \in E_l, (u_2,v_2) \in E_r, \text{ and } ((u_1,v_1),(u_2,v_2)) \in R \Big\}.$$

The axioms of S are shifted relative to each other until their left ends match:

$$A' = \Big\{ \left(\lambda, \lambda, a^{i+|u|}, a^{|v_1w_1|-i}, a^{|v_2w_2|-i}\right) \mid (v_1,v_2,u,w_1,w_2) \in A \text{ and } i = \min\{|v_1w_1|, |v_2w_2|\} \Big\}.$$

In order to give evidence of the correctness of the construction we consider an arbitrary derivation $(v_1,v_2,u,w_1,w_2) \Rightarrow^* (v'_1,v'_2,u',w'_1,w'_2)$ of S, and claim that there is a derivation $(\lambda,\lambda,z,x_1,x_2) \Rightarrow^* (\lambda,\lambda,z',x'_1,x'_2)$ in S' so that $|v_1uw_1| = |zx_1|$, $|v_2uw_2| = |zx_2|$, $|v'_1u'w'_1| = |z'x'_1|$, and $|v'_2u'w'_2| = |z'x'_2|$.

For $(v_1,v_2,u,w_1,w_2) \in A$ there is an axiom $(\lambda,\lambda,a^{i+|u|},a^{|v_1w_1|-i},a^{|v_2w_2|-i})$ in A', where $i = \min\{|v_1w_1|,|v_2w_2|\}$. Since $|v_1uw_1| = i + |u| + |v_1w_1| - i$ and $|v_2uw_2| = i + |u| + |v_2w_2| - i$, the claim is true for the axiom.

Concluding inductively, assume the claim is true for a derivation as above. Assume that assembling a pair of units $((a^p,a^q),(a^r,a^s)) \in R$ in S yields

$$(v'_1,v'_2,u',w'_1,w'_2) \Rightarrow ((v''_1,v''_2,u'',w''_1,w''_2)),$$

where $|v''_1u''w''_1| = |v'_1u'w'_1| + p + r$ and $|v''_2u''w''_2| = |v'_2u'w'_2| + q + s$. Then there is a unit (a^{p+r},a^{q+s}) in T'_r. Therefore, in S' we have the derivation $(\lambda,\lambda,z',x'_1,x'_2) \Rightarrow (\lambda,\lambda,z'',x''_1,x''_2)$, where $|z''x''_1| = |z'x'_1| + p + r$ and $|z''x''_2| = |z'x'_2| + q + s$. From $|v'_1u'w'_1| = |z'x'_1|$ and $|v'_2u'w'_2| = |z'x'_2|$ the claim follows.

The claim shows in particular that for any successful derivation in S there must be a successful derivation in S'. The converse is seen similarly and, thus, $L(S) = L(S')$. □

Lemma 5.6. *There is a (unary) regular language not generated by any R-2SAS.*

Proof. In [5] it has been shown that the regular language $\{a\} \cup \{a^{2n} \mid n \geq 2\}$ is not generated by any 1SAS and, thus, by Lemma 5.5 it is not generated by any R-2SAS, either. □

The proof of the previous lemma utilizes the language $\{a\} \cup \{a^{2n} \mid n \geq 2\}$ which is the union of a finite language and an infinite language. The idea of the proof is that the axioms and the ending units of the finite language have to be of a certain

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

form, and may be extended by units of the infinite language. For this witness language the problem disappears when the finite language is omitted from the union. So, the question arises whether this observation can be generalized. That is, given a unary regular language, is it always possible to omit a finite number of words from it in order to obtain a language generated by a 1SAS. The following lemma answers the question in the affirmative.

Lemma 5.7. *Let L be a unary regular language. Then there exists a finite set $W \subseteq L$ so that $L \setminus W$ is generated by a 1SAS.*

Proof. Let a regular language $L \subseteq \{a\}^*$ be given by a complete minimal deterministic finite automaton M with state set $Q = \{q_1, q_2, \ldots, q_n\}$, initial state q_1, set of accepting states F, and transition function δ. Since M is unary and deterministic, it consists of an initial tail of states followed by a cycle. That is, $\delta(q_i, a) = q_{i+1}$, for $1 \leq i \leq n-1$, and $\delta(q_n, a) = q_j$, for some $1 \leq j \leq n$. Let $F_1 = F \cap \{q_1, q_2, \ldots, q_{j-1}\}$ be the set of accepting states belonging to the initial tail and $F_2 = F \cap \{q_j, q_{j+1}, \ldots, q_n\}$ be the set of accepting states belonging to the cycle. Clearly, only finitely many words are accepted by M with states from F_1. The set $W = \{w \in a^* \mid w \in L(M) \text{ and } \delta(q_1, w) \in F_1\}$ includes exactly these words.

Next we construct a 1SAS $S = \langle \{a\}, A, T_l, T_r, E_l, E_r \rangle$ such that $L(S) = L \setminus W$, where by definition $T_l = \emptyset$ and $E_l = \{(a, a)\}$. To this end, consider a function $d : Q \times Q \to \{0, 1, \ldots, n\}$ that maps a pair of states q, q' to the length of the shortest non-empty path driving M from q to q'. If there does not exist such a path, d returns 0. Now, for each accepting state $q_f \in F_2$ an axiom is added to A, more precisely, $A = \{(\lambda, \lambda, a^{d(q_1, q_f)}, \lambda, \lambda) \mid q_f \in F_2\}$. Recall that $j-1$ is the length of the cycle. The set T_r is defined to be the singleton $\{(a^j, a^j)\}$, and the set of ending units is $E_r = \{(a, a)\}$.

Let w be a word accepted by M. If $\delta(q_1, w) \in F_1$, then the word w belongs to the finite set W and is excluded. If $\delta(q_1, w) = q_f \in F_2$, then w is generated by S starting with the axiom $(\lambda, \lambda, a^{d(q_1, q_f)}, \lambda, \lambda)$ and assembling the unit $(a^j, a^j) \in T_r$ as many times as M runs through the cycle. So, $(L \setminus W) \subseteq L(S)$.

Next, let w be generated by S where the derivation starts with an axiom $(\lambda, \lambda, a^{d(q_1, q_f)}, \lambda, \lambda) \in A$, continues with assembling the sole unit $(a^j, a^j) \in T_r$ several times, and ends by adding the sole ending unit (a, a). So, the length of w is $d(q_1, q_f) + i(j-1)$, for some $i \geq 0$. Since $\delta(q_1, a^{d(q_1, q_f)+i(j-1)}) = \delta(q_f, a^{i(j-1)}) = \delta(q_f, a^{(i-1)(j-1)}) = q_f$ and $q_f \in F_2$ is an accepting state, automaton M accepts w. So, we have $L(S) \subseteq (L \setminus W)$ and, thus, $L(S) = (L \setminus W)$. □

6. Closure properties

Here we consider closure properties of the families of languages generated by different versions of 2SAS. Some of the ideas of the proofs are inherited from 1SAS. First we examine the Boolean operations. In [5] it has been shown that 1SAS can not generate the language $\{a\} \cup \{a^{2n} \mid n \geq 2\}$. A small extension of the proof

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

shows that the language $\{a\} \cup \{a^{2^n} \mid n \geq 1\}$ is not generated by any 1SAS either. Together with Lemma 5.5, where it is shown that a unary language is generated by a R-2SAS if and only if it is generated by a 1SAS, we obtain the non-closure under union.

Theorem 6.1. *The families of languages generated by any type of 2SAS are not closed under union.*

The non-closure under complementation follows by the same witness language.

Theorem 6.2. *The families of languages generated by any type of 2SAS are not closed under complementation.*

Proof. From above we know that the language $\{a\} \cup \{a^{2^n} \mid n \geq 1\}$ is not generated by any type of 2SAS. However, its complement $\{a^{2^n+1} \mid n \geq 1\}$ is generated by the 1SAS $\langle \{a, \}, \{(\lambda, \lambda, a^3, \lambda, \lambda)\}, \emptyset, \{(a^3, a^3)\}, \{(a, a)\}, \{(a, a\}\rangle$. □

Since the families of languages generated by any type of 2SAS is incomparable with the family of regular languages, but includes the language Σ^*, for any alphabet Σ, their non-closure under intersection with regular languages follows immediately.

Lemma 6.3. *The families of languages generated by any type of 2SAS are not closed under intersection with regular languages.*

The general non-closure under intersection is now shown for all types of 2SAS except for R-2SAS, for which it is an open problem.

Lemma 6.4. *Let $k \geq 1$ be a constant. Then there are k languages generated by 1SAS whose intersection is accepted by a one-way ℓ-head finite automaton if and only if $k \leq \binom{\ell}{2}$.*

Proof. In [5] it is shown that the two languages

$$L_1 = \{\$_1 w \$_2 u_1 \$_3 u_2 \$_4 w \$_5 \mid u_1, u_2, w \in \{a, b\}^+\} \text{ and}$$
$$L_2 = \{\$_1 u_1 \$_2 w \$_3 w \$_4 u_2 \$_5 \mid u_1, u_2, w \in \{a, b\}^+\}$$

are generated by 1SAS. More general, we consider the languages

$$L_{k,i} = \{\$_1 u_1 \$_2 u_2 \dots \$_{2k} u_{2k} \$_{2k+1} \mid u_j \in \{a, b\}^+, 1 \leq j \leq 2k, u_i = u_{2k-i+1}\}.$$

which are generated by 1SAS as well. However, in [9] it is shown that the intersection

$$L_k = \bigcap_{i=1}^{k} L_{k,i} = \{\$_1 w_1 \$_2 w_2 \dots \$_k w_k \$_{k+1} w_k \$_{k+2} w_{k-1} \dots \$_{2k} w_1 \$_{2k+1} \mid w_j \in \{a, b\}^+, 1 \leq j \leq k\}$$

is accepted by a one-way ℓ-head finite automaton if and only if $k \leq \binom{\ell}{2}$. □

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

Theorem 6.5. *The families of languages generated by any type of 2SAS except R-2SAS are not closed under intersection.*

Proof. Theorem 4.5 shows that all language families in question are accepted by some one-way 5-head finite automaton. So, the theorem follows by Lemma 6.4. □

The next (non-)closure properties are for homomorphisms. It turns out that we can adopt the proof of [5] to show that all language families generated by 2SAS are not closed under non-erasing letter-to-letter homomorphisms.

Theorem 6.6. *The families of languages generated by any 2SAS except R-2SAS are not closed under λ-free letter-to-letter homomorphisms.*

Proof. The homomorphism $h : \{a, \$, \#\}^* \to \{a\}^*$ is applied to the language

$$\left\{ \$a^1\$a^3\$ \dots \$a^{2i+1}\# \mid i \geq 1 \right\}.$$

The result is a non-semilinear language which is not accepted by any one-way k-head finite automaton. □

Theorem 6.7. *The families of languages generated by 2SAS, S-2SAS and R-2SAS are not closed under concatenation.*

Proof. A straightforward modification of Example 2.3 yields 2SAS that generate the languages $L_1 = \{b^n a^n \mid n \geq 1\} \cup b^+$ and $L_2 = \{a^n b^n \mid n \geq 1\} \cup \{b\}$. We consider the concatenation $L' = L_1 L_2$ and assume that it is generated by some R-2SAS.

Strings from b^+b do belong to L'. We consider their assembling up to the last step. Recall that the ending units have to be assembled synchronously. During the assembling process at least one but maybe both ends of the current string are extended to get longer and longer. Moreover, since there is only a finite number of ending units, parts of the extension are in such a way that – after assembling a certain number of units – both strands at the left end and both strands at the right end are extended by the same number of symbols, say l symbols at the left and r symbols at the right end, where $l, r \geq 0$ and $l + r \geq 1$ (*cf.* proof of Lem. 3.3).

Next, we consider the generation of words $b^n a^n a^m b^m$, with m, n large enough. For such words, the ending units consist of symbols b only. Just before the ending units are added, we can assemble the units that extend the lower and upper strand at the left by l symbols b, and the lower and upper strand at the right by r symbols b. So, the string $b^{n+l} a^n a^m b^{m+r} \notin L'$ is generated, a contradiction. It follows that the families of languages generated by 2SAS, S-2SAS, and R-2SAS are not closed under concatenation. □

For the centralized variants the witness languages have to be modified.

Theorem 6.8. *The families of languages generated by C-2SAS and SC-2SAS are not closed under concatenation.*

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

TABLE 1. Summary of closure properties of bidirectional string assembling systems.

	$\sim$	$\cup$	$\cap$	$\cap_{REG}$	$\cdot$	h_λ	R
1SAS	no	no	no	no	no	no	yes
C-2SAS	no	no	no	no	no	no	yes
2SAS	no	no	no	no	no	no	yes
S-2SAS	no	no	no	no	no	no	yes
SC-2SAS	no	no	no	no	no	no	yes
R-2SAS	no	no	?	no	no	?	yes

Proof. The languages $L_1 = \{b^n a^n a^m b^m \mid n, m \geq 1\} \cup \{b\}$ and $L_2 = b^+$ are generated by some C-2SAS. As in the proof of Theorem 6.7, we assume that the concatenation $L' = L_1 L_2$ is generated by some SC-2SAS, and consider the assembling of a string $w = b^n a^n a^m b^m$ where m, n are large enough. For the centralized variants, the units can be used at both sides. So, before the last step at which the ending units are added, we extend the string at the left end and at the right end, and thus generate $b^{n+l+r} a^n a^m b^{m+l+r}$, where $l, r \geq 0$ and $l + r \geq 1$. But this word does not belong to L', a contradiction. □

Finally, we turn to a sole positive closure property, the reversal.

Theorem 6.9. *The families of languages generated by any type of 2SAS* are *closed under reversal.*

Proof. Let $S = \langle \Sigma, A, T_l, T_r, E_l, E_r \rangle$ be an 2SAS. Then we construct a 2SAS $S_R = \langle \Sigma, A', T'_l, T'_r, E'_l, E'_r \rangle$ with $L(S) = (L(S_R))^R$ as

$$A' = \left\{ (v_1^R, v_2^R, u^R, w_1^R, w_2^R) \mid (v_1, v_2, u, w_1, w_2) \in A \right\},$$
$$E'_l = \left\{ (u^R, v^R) \mid (u, v) \in E_r \right\}, \qquad E'_r = \left\{ (u^R, v^R) \mid (u, v) \in E_l \right\},$$
$$T'_l = \left\{ (u^R, v^R) \mid (u, v) \in T_r \right\}, \qquad T'_r = \left\{ (u^R, v^R) \mid (u, v) \in T_l \right\}.$$

So, the strings in the units are reversed, and the units from the left side are interchanged with the units from the right side. It is evident that for any derivation of a word w by S there is a derivation of w' by S_R. □

REFERENCES

[1] R. Freund, G. Păun, G. Rozenberg and A. Salomaa, Bidirectional sticker systems, in *Pacific Symposium on Biocomputing (PSB 1998)*. World Scientific, Singapore (1998) 535–546.
[2] J. Hartmanis, On non-determinancy in simple computing devices. *Acta Inform.* **1** (1972) 336–344.
[3] L. Kari, G. Păun and G. Rozenberg, A. Salomaa and S. Yu, DNA computing, sticker systems, and universality. *Acta Inform.* **35** (1998) 401–420.
[4] M. Kutrib and M. Wendlandt, Bidirectional string assembling systems, in vol. 290 of *Non-Classical Models of Automata and Applications* (NCMA 2011). books@ocg.at. Austrian Computer Society, Vienna (2012) 107–121.

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press

[5] M. Kutrib and M. Wendlandt, String assembling systems. *RAIRO: ITA* **46** (2012) 593–613.
[6] R. McNaughton, Algebraic decision procedures for local testability. *Math. Systems Theory* **8** (1974) 60–76.
[7] G. Păun and G. Rozenberg, Sticker systems. *Theoret. Comput. Sci.* **204** (1998) 183–203.
[8] E.L. Post, A variant of a recursively unsolvable problem. *Bull. AMS* **52** (1946) 264–268.
[9] A.C. Yao and R.L. Rivest, $k+1$ heads are better than k. *J. ACM* **25** (1978) 337–340.
[10] Y. Zalcstein, Locally testable languages. *J. Comput. System Sci.* **6** (1972) 151–167.

Communicated by M. Holzer.
Received February 1, 2013. Accepted December 11, 2013.

https://doi.org/10.1051/ita/2013048 Published online by Cambridge University Press