Natural Computing
https://doi.org/10.1007/s11047-022-09918-x

Variants of string assembling systems

Martin Kutrib[1] · Matthias Wendlandt[1]

Accepted: 27 July 2022

Abstract
String assembling systems are biologically inspired mechanisms that generate strings from copies out of finite sets of assembly units. The underlying mechanism is based on piecewise assembly of a double-stranded sequence of symbols, where the upper and lower strand have to match. The generation is additionally controlled by the requirement that the first symbol of a unit has to be the same as the last symbol of the strand generated so far, as well as by the distinction of assembly units that may appear at the beginning, during, or at the end of the assembling process and by a length restriction on the units. We investigate the power of these model-inherent control mechanisms by considering variants where one or more of these mechanisms are relaxed. The generative capacities and the relative power of the variants are our main interest. In particular, we prove that the power gained in the different control mechanisms may yield strictly more powerful systems and incomparable capacities. Additionally, we generalize these systems to multi-stranded systems. We obtain a strong connection to one-way multi-head finite automata and show an infinite, dense, and strict strand hierarchy. Finally, we examine the closure properties of the different variants of string assembling systems.

Keywords String assembling · Double-stranded sequences · Stateless · Two-head finite automata · Decidability · Closure properties.

1 Introduction

In 1965 Gordon E. Moore predicted that the number of components per integrated circuit will double every 18 month (Moore 1965). Up to now this forecast that is also known as *Moore's Law* became more or less reality. Nevertheless, there are many real world problems requiring such a huge computational complexity that they cannot be solved for an appropriate size of the instance in this day and age, and also will not be solvable under the assumption of Moore's Law in a 1000 years unless new computing techniques are developed. This motivated 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. Among the realms of string generating mechanisms examples are Lindenmayer systems (Rozenberg and Salomaa 1980), splicing systems and sticker systems (Păun et al. 1998). The latter two types of devices model operations on DNA molecules and are therefore based upon double-stranded strings as raw material of the string generation process, where corresponding symbols are uniquely related. Sticker systems were introduced in Kari et al. (1998) in their basic one-way variant. Basically, they consist of dominoes that can be seen as double-stranded molecules. The dominoes are sticked together in a derivation process until a complete double strand is derived. Different variants especially of two-way systems have been investigated in (Freund et al. 1998; Păun and Rozenberg 1998; Păun et al. 1998). A main feature of sticker systems is that the upper and the lower fragment of the dominoes are glued together. So, when a domino is added to a double strand the length difference of the upper and the lower strand derived is the same as the length difference between the upper and the lower fragment of the dominoes. This implies that many variants of sticker systems are at most as powerful as linear context-free

Some preliminary parts of this work have been presented at CIAA 2018 Kutrib and Wendlandt (2018) and SOFSEM 2019 Kutrib and Wendlandt (2019)

✉ Martin Kutrib
kutrib@informatik.uni-giessen.de

Matthias Wendlandt
matthias.wendlandt@informatik.uni-giessen.de

[1] Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany

Published online: 20 August 2022

grammars Păun et al. (1998). String assembling systems, introduced in Kutrib and Wendlandt (2012), are another string generating mechanism based on double strands. The idea of string assembling systems (Bordihn et al. 2012; Kutrib and Wendlandt 2012, 2014) is basically motivated by the mechanisms of the Post Correspondence Problem. In particular, the basic assembly units are pairs of strings that have to be connected to the upper and lower string generated so far synchronously. In comparison to sticker systems, the strings are not connected to each other. This property enables the possibility to increase the length difference between the two strands arbitrarily and, moreover, to compare positions that are arbitrarily far away from each other in a given word. Thus, it is possible to generate non-context-free languages. Apart from the double strand, essentially, derivations in string assembling systems are controlled by the requirement that the first symbol of a string to be assembled has to match the last symbol of the strand generated so far, as well as by the distinction of assembly units that may appear at the beginning, during, or at the end of the assembling process. Moreover, the lengths of the strings in the assembly units are finite but are not limited a priori.

In Sect. 3 we investigate the power of these model-inherent control mechanisms by considering variants where one or more of these mechanisms are relaxed. Additionally, we study the case where the length of the strings in the assembly units is bounded. It turns out that the generative capacities and the relative power gained in the different control mechanisms may yield strictly more powerful systems as well as incomparable capacities. The relative power of the systems is summarized in Fig. 3. The last subsection studies the impact of the lengths of the assembling fragments. It is shown that increasing the lengths gives strictly more capacity, that is, an infinite and tight hierarchy follows.

The computational power of one-way k-head finite automata increases with each head added to the system (Yao and Rivest 1978). This suggests naturally to examine multi-stranded string assembling systems and to investigate whether a strand hierarchy can be achieved. In Sect. 4 we investigate these generalizations. The generative capacities and the relative power are our main interest. We explore the relations with one-way multi-head finite automata and show an infinite, dense, and strict strand hierarchy. Moreover, we consider the relations with the language families of the Chomsky Hierarchy and consider the unary variants of k-stranded string assembling systems.

Sect. 5 is devoted to examine the closure properties of the restricted variants as well as the extended variants of string assembling systems with respect to the usual operations on sets of languages. Finally, we conclude the paper in Sect. 6.

2 Preliminaries and definitions

We write Σ^* for the set of all words (strings) 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 $|w|$. For the number of occurrences of a symbol a in w we use the notation $|w|_a$. Generally, for a singleton set $\{a\}$ we also simply write a. We use $\subseteq$ for inclusions and $\subset$ for strict inclusions. In order to avoid technical overloading in writing, two languages L and L' are considered to be equal, if they differ at most by the empty word, that is, $L \setminus \{\lambda\} = L' \setminus \{\lambda\}$. A language is said to be *cofinite* if it is the complement of a finite language.

We are especially interested in how string assembling systems can be used to generate languages. To this end, we consider arbitrary alphabets and do not restrict to the natural DNA alphabet $\{A, G, C, T\}$. Clearly, there are ways to encode an arbitrary alphabet in the natural alphabet.

A string assembling system (SAS) generates double-stranded strings, iteratively connecting basic elements called units. A *unit* $u = (w_1, w_2)$ with $w_1, w_2 \in \Sigma^+$ consists of two strings over a given alphabet Σ. The length of a unit is the maximum of the lengths of w_1 and w_2. The first string w_1 is connected to the upper strand of a growing double-stranded string, while the second one w_2 is connected to the lower one. They can only be connected, (i) if the first symbol of w_1 is equal to the last symbol of the upper strand generated so far and, similarly, (ii) if the first symbol of w_2 is equal to the last symbol of the lower strand generated so far. In this case, the first symbols of the unit and the last symbols of the double-stranded sequence generated so far are glued together on top of each other if the derived upper and the lower strands match at each position.

Each *generation* has to begin with a unit out of the set of *axioms*. Afterwards the derivation continues with units of a second set, the set of *assembly units*, and it has to be finished by an *ending unit*. The generation is said to be *valid* if and only if both strands are identical in each position when the derivation process stops.

Formally, a *string assembling system* $S = \langle \Sigma, A, T, E \rangle$ is a quadruple, where

1. Σ is a finite, nonempty set of *symbols*,
2. $A \subset \Sigma^+ \times \Sigma^+$ is the finite set of *axioms* of the forms (uv, u) or (u, uv), where $u \in \Sigma^+$ and $v \in \Sigma^*$,
3. $T \subset \Sigma^+ \times \Sigma^+$ is the finite set of *assembly units*, and
4. $E \subset \Sigma^+ \times \Sigma^+$ is the finite set of *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.

Let $S = \langle \Sigma, A, T, E \rangle$ be an SAS. The *derivation relation* $\Rightarrow$ is defined on $\Sigma^+ \times \Sigma^+$ by

1. $(uv, u) \Rightarrow (uvx, uy)$ if
   (i) $uv = ta$, $u = sb$, and $(ax, by) \in T \cup E$, for $a, b \in \Sigma$, $x, y, s, t \in \Sigma^*$, and
   (ii) $vx = yz$ or $vxz = y$, for $z \in \Sigma^*$,
2. $(u, uv) \Rightarrow (uy, uvx)$ if
   (i) $uv = ta$, $u = sb$, and $(by, ax) \in T \cup E$, for $a, b \in \Sigma$, $x, y, s, t \in \Sigma^*$, and
   (ii) $vx = yz$ or $vxz = y$, for $z \in \Sigma^*$.

An illustration can be found in Fig. 1.

A derivation is said to be *successful* if it initially starts with an axiom from A, continues with assembling units from T, and ends with assembling an ending unit from E. The process stops when an ending assembly unit is added. The sets A, T, and E are not necessarily disjoint.

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

$$L(S) = \{ w \in \Sigma^+ \mid (u, v) \Rightarrow^* (w, w) \text{ is a successful derivation} \},$$

with $(u, v) \in A$, where $\Rightarrow^*$ refers to the reflexive, transitive closure of the derivation relation $\Rightarrow$.

In the following we will refer to systems based on this definition as *common* SAS or simply SAS. The next example illustrates the basic mechanisms of string assembling systems.

Example 1 Let Σ be an alphabet not containing the symbols $[\$]_1, [\$]_2, [\$]_3$. The following SAS $S = \langle \Sigma \cup \{[\$]_1, [\$]_2, [\$]_3\}, A, T, E \rangle$ generates the non-context-free language $\{ [\$]_1 w [\$]_2 w [\$]_3 \mid w \in \Sigma^+ \}$. The units are:

$$A = \{([\$]_1, [\$]_1)\}, \quad T = T_1 \cup T_2 \cup T_3,$$
$$E = \{([\$]_3, [\$]_3)\}, \text{ where for } x, y \in \Sigma,$$
$$T_1 = \{([\$]_1 x, [\$]_1), (xy, [\$]_1), (x[\$]_2, [\$]_1)\},$$
$$T_2 = \{([\$]_2 x, [\$]_1 x), (xy, xy), (x[\$]_3, x[\$]_2)\},$$
$$T_3 = \{([\$]_3, [\$]_2 x), ([\$]_3, xy), ([\$]_3, x[\$]_3)\}.$$

The units from T_1 are used to generate the upper strand $[\$]_1 w [\$]_2$ and the lower strand $[\$]_1$. Then units from the set T_2 are assembled to generate the upper strand $[\$]_1 w [\$]_2 w [\$]_3$ and the lower strand $[\$]_1 w [\$]_2$. Finally, we obtain $[\$]_1 w [\$]_2 w [\$]_3$ as upper and lower strands by the units in T_3. ■

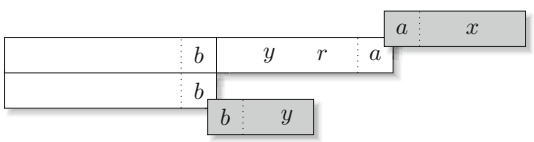

Fig. 1 Example of assembling a unit (ax, by) at the right

3 Restricted string assembling systems

There are different mechanisms to control the derivations of SAS. First, the units are arranged in three sets, such that initial units have to be taken from one set, ending units from another, and in between only units from the third set are allowed. Second, whenever the current strands are extended, the first symbols of the strings in the unit are glued on top of the last symbols of the strands generated so far. Third, the lengths of the units seem to be an important property. And the last control mechanism is the double strand. The power of double strands is already adumbrated by the Post Correspondence Problem, which goes along with the undecidability of emptiness. This chapter is devoted to study the generative capacity gained from these mechanisms. In particular, in each of the next subsections we remove control mechanisms and analyze the resulting language family.

3.1 Length-restricted string assembling systems

In this section we turn to a structural limitation of string assembling systems. The length difference of the strings in units can be seen as a very basic kind of memory. This length difference is bounded by the lengths of the units. In particular, if a unit has length n, then the maximal length difference can be $n - 1$. In the following we consider SAS where the lengths of the strings in units are restricted. This idea of storing information is illustrated in the next two examples.

Example 2 The language $\{ a^{2n} b \mid n \geq 1 \} \cup \{ a^{3n} c \mid n \geq 1 \}$ is generated by the SAS $S = \langle \{a, b, c\}, A, T, E \rangle$ with the following units.

1. $(a^2, a) \in A$
2. $(a^4, a^3) \in A$
3. $(a^6, a^5) \in A$
4. $(a^3, a) \in A$
5. $(a^6, a^4) \in A$
6. $(a^7, a^7) \in T$
7. $(ab, a^2 b) \in E$
8. $(ac, a^3 c) \in E$

The length difference that is given by one of the axioms can somehow be seen as a memory. If a word of the form $a^{2n} b$ is generated, one of the first three axioms (1),(2), or (3) has to be used. In this case, the length difference between the upper and the lower string is 1. Since the sole unit (a^7, a^7), unit (6), from the set T has matching upper

and lower strings, this length difference is kept until the end of the derivation. The only fitting ending unit in this case is unit (7), which finally gives a desired word.

For words of the form $a^{3n}c$ one of the axioms (4) or (5) has to be used, which initially establishes the length difference 2 between the upper and the lower strand. Again unit (6) can be used to extend both strands arbitrarily. Finally the only fitting ending unit in this case is unit (8). ■

The technique of Example 2 can be extended to generate non-regular languages, where the length difference between the upper and lower strand increases arbitrarily during the derivation, as it is the case in the next example.

Example 3 The language $\{ a^{2n}b^{2n} \mid n \geq 1 \} \cup \{ a^{3n}b^{3n} \mid n \geq 1 \}$ is generated by the SAS $S = \langle \{a, b\}, A, T, E \rangle$ with the following units.

1. $(a^2, a) \in A$
2. $(a^4, a) \in A$
3. $(a^6, a) \in A$
4. $(a^3, a) \in A$
5. $(a^7, a) \in T$
6. $(ab, a) \in T$
7. $(bb, aa) \in T$
8. $(b, ab) \in T$
9. $(b, bb) \in T$
10. $(b, b) \in E$

Again we use the technique of different axioms with different length differences to distinguish between possible multiplicities of 2 and 3. The basic idea of the construction is first to generate (a^n, a) where n can be any multiple of 2 or 3, then to extend the upper strand by b^n whereby the lower strand is extended coincidently to a^n. Finally, the missing b's are assembled in the lower strand.

At the beginning one of the axioms (1)–(4) is used, then unit (5) that extends the current upper strand by a^6 can repeatedly be assembled. Together with the chosen axiom in this way any a^n with n multiple of 2 and 3 can be generated in the upper strand and, moreover, these are the only strings that can be generated. This process is terminated by assembling the second possible unit (6). Now $(a^n b, a)$ has been generated. The only possibility to continue is with unit (7) until all a's of the upper strand are matched. Now $(a^n b^n, a^n)$ has been generated. The only possibility for a successful derivation is now to continue with unit (8) and subsequently with unit (9) until all b's of the upper strand are matched. So, $(a^n b^n, a^n b^n)$ has been generated, where n is a multiple of 2 or 3. ■

Now we turn to SAS where the lengths of the strings in units are limited by a constant, and explore the generative capacity gained in this way.

Let $k \geq 1$ be an integer. An SAS $S = \langle \Sigma, A, T, E \rangle$ is said to be *k-length-restricted* (k-SAS), if $|u| \leq k$ and $|v| \leq k$ for each unit $(u, v) \in A \cup T \cup E$.

First we consider unary languages and assume that their words are ordered according to their lengths. The question is, how large can the gaps between two consecutive words be if the language is generated by some k-SAS. The following example gives a quadratic lower bound.

Example 4 Let $k \geq 1$ be an integer. The language $L_k = a(a^{(k-1)^2})^*$ is generated by the k-SAS $S = \langle \{a\}, A, T, E \rangle$ with the following units.

1. $(a, a) \in A$
2. $(a^k, a) \in T$
3. $(a^2, a^k) \in T$
4. $(a, a) \in E$

Assembling the sole axiom and the sole ending unit results in the word a. So, the assembling of units from T must result in equally long upper and lower strands. The length difference between upper and lower string is $k-1$ in unit (2) and $k-2$ in unit (3). Since for $k \geq 3$ the numbers $k-1$ and $k-2$ are relatively prime, each $k-1$ units (3) and $k-2$ units (2) have to be assembled in order to obtain equally long upper and lower strands. Assembling $k-1$ units (3) extends the upper strand by $k-1$ and the lower strand by $(k-1)(k-1)$ symbols. Assembling $k-2$ units (2) extends the upper strand by $(k-2)(k-1)$ and the lower strand by 0 symbols. So, the current strands can be extended by strings of length $(k-1)+(k-2)(k-1)$ or equivalently $(k-1)(k-1)$. Adding the initial symbol a shows that S generates L_k, for $k \geq 3$. For $k = 1$, unit (2) does not extend the current strands and, thus, unit (3) cannot contribute to a successful derivation. Therefore, the initial word a is the only one that can be derived. Since $L_1 = \{a\}$, the SAS S generates the language L_k for $k = 1$ as well. Finally, if $k = 2$ then unit (3) extends the current strands by 1, which shows that $a^+ = L_2$ is generated. ■

Trivially for the special case of 1-SAS it holds that the language generated is either empty or consists of the singleton $\{a\}$ in the unary case. We turn to investigate the arbitrary unary case.

The next lemma shows that the lower bound for the gaps presented in Example 4 is tight.

Lemma 5 *Let $k \geq 2$ and L be a unary language generated by some k-SAS. Assume the words of L are ordered according to their lengths. Then the length difference between two consecutive words is at most $(k-1)^2$.*

Proof Assume in contrast to the assertion that there are two consecutive words a^m and a^n in L such that $m < n$ and $n - m > (k-1)^2$.

If there is a unit (a^x, a^x) in T, with $x > 1$, then $a^m \in L$ implies $a^{m+x-1} \in L$. Since $x \leq k$ we have that at least one word between a^m and a^n can be generated, because

$$m < m + x - 1 \leq m + k - 1 \leq m + (k-1)^2 < n,$$

a contradiction.

If there are units (a^{x_1}, a^{y_1}) and (a^{x_2}, a^{y_2}) in T, where $1 \leq y_1 < x_1 \leq k$ and $1 \leq x_2 < y_2 \leq k$, then assembling $y_2 - x_2$ times the unit (a^{x_1}, a^{y_1}) and $x_1 - y_1$ times the unit (a^{x_2}, a^{y_2}) extends the upper strand by

$$(y_2 - x_2)(x_1 - 1) + (x_1 - y_1)(x_2 - 1) = x_1 y_2 - x_2 y_1 - x_1 + x_2 + y_1 - y_2$$

and the lower strand by $(y_2 - x_2)(y_1 - 1) + (x_1 - y_1)(y_2 - 1) = x_1 y_2 - x_2 y_1 - x_1 + x_2 + y_1 - y_2$ symbols. So, both strands are extended by the same number of symbols. This number $x_1 y_2 - x_2 y_1 - x_1 + x_2 + y_1 - y_2$ is maximal for $x_1 = y_2 = k$ and $x_2 = y_1 = 1$, since the first product is maximal and the second minimal. Then it is equal to $(k-1)^2$. Similarly as before we conclude that $a^{m+(k-1)^2} \in L$ and obtain a contradiction since $m < m + (k-1)^2 < n$.

Finally, assume that there are units $(a^x, a^y) \in T$, $1 \leq y < x \leq k$, but no unit, where the upper string is shorter than the lower. In this case, the length difference between upper and lower string has to be equalized by axioms and ending units, thus, L is finite. Let a^m be generated by assembling some axiom (a^{x_1}, a^{y_1}) and some ending unit (a^{x_2}, a^{y_2}). In order to increase the length m we can use another axiom (a^{x_3}, a^{y_3}) where $|x_3 - y_3| > |x_1 - y_1|$. So, the difference between upper and lower string that can be equalized increases by at most $k-1$, and by assembling $k-1$ units with length difference 1 the length m is increased by at most $(k-1)^2$. Similarly, if there is no such axiom, we can alternatively use another ending unit which results in the same upper bound for the length increasing. If neither such axiom nor such ending unit exists, a contradiction is derived in the same way. The same argumentation can be applied for assembling units where the upper strand is shorter than the lower one. □

The previous Lemma and Example 4 yield an infinite and tight hierarchy dependent on the length restrictions (see Fig. 2).

Theorem 6 *Let $k \geq 1$ be an integer. The family of languages generated by k-SAS is strictly included in the family of languages generated by $(k+1)$-SAS.*

Proof By definition a k-SAS is a $(k+1)$-SAS. Thus it remains to be shown that the inclusion is strict.

For 1-SAS it is solely possible to generate finite languages, but 2-SAS can also generate infinite languages. Moreover, for $k \geq 1$, Example 4 shows that the regular language $L_{k+1} = a(a^{(k)^2})^*$ is generated by a $(k+1)$-SAS. But by Lemma 5 it cannot be generated by any k-SAS. □

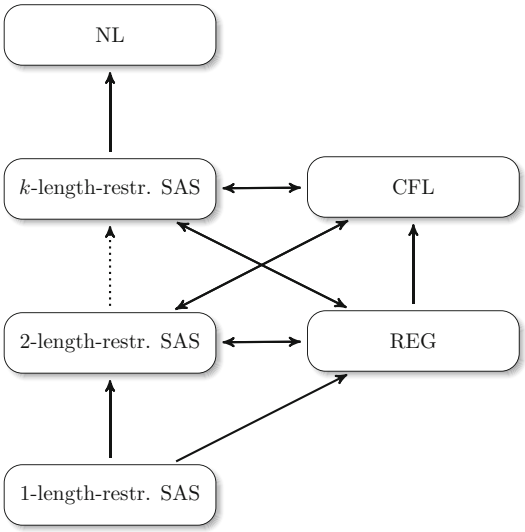

Fig. 2 Hierarchy on the length of the units. A single arrow means strict inclusion, a double arrow means incomparability, and the dotted arrow is for an infinite hierarchy depending on the maximal length of the units. Since 1-length-restricted SAS can only generate finite languages, they are a subset of the regular languages

3.2 Single-stranded string assembling systems

The next restriction examines the power of double strands. It has been pointed out before, that the power of double-stranded systems goes along with the undecidability of emptiness. Thus, in the following we remove the power of the double strands and turn to investigate single-stranded SAS (1-stranded SAS). The basic idea of overlappings by one symbol and one strand is closely related to the work on chop operators (Enaganti et al. 2017; Holzer et al. 2017).

A single-stranded string assembling system is an SAS $S = \langle \Sigma, A, T, E \rangle$, where all the units $u \in A \cup T \cup E$ are of the form $u \in \Sigma^+$.

The *derivation relation* $\Rightarrow$ is defined on specific subsets of Σ^+ by $ua \Rightarrow uav$ if $ua \in \Sigma^+$ and $av \in T \cup E$, for $a \in \Sigma, u, v \in \Sigma^*$. As for common SAS, a derivation is said to be *successful* if it initially starts with an axiom from A, continues with assembling units from T, and ends with assembling an ending unit from E. The sets A, T, and E are not necessarily disjoint.

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

$$L(S) = \{ w \in \Sigma^+ \mid u \Rightarrow^* w \text{ is a successful derivation} \}.$$

We turn to the comparison with the regular languages. The next theorem shows that the restriction to one strand reduces the power significantly.

Theorem 7 *The family of languages generated by* 1-*stranded* SAS *is strictly included in the family of regular languages.*

Proof Given a single-stranded SAS $S = \langle \Sigma, A, T, E \rangle$ we construct a nondeterministic finite automaton $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ such that $L(S) = L(M)$.

Since S is single-stranded, the units are words from Σ^+. Since units in T of length one do not have any effect for single-stranded SAS, they can safely be omitted. So, we assume that each unit in T is at least of length two.

The following NFA is successively developed. There are no elements beside the ones mentioned. For each axiom $u = x_1 x_2 \cdots x_\ell \in A$ we define the states $\{u_0, u_1, \ldots, u_\ell\}$ and the transitions $\delta(u_{j-1}, x_j) = u_j$, for $1 \leq j \leq \ell$. Similarly, for each unit $u = x_1 x_2 \cdots x_\ell \in T \cup E$ we define the states $\{u_1, u_2, \ldots, u_\ell\}$ and the transitions $\delta(u_{j-1}, x_j) = u_j$, for $2 \leq j \leq \ell$.

So far, we have a different chain of connected states for each unit of S. The chains are interconnected as follows. Let q_0 be a new state that is set to be the initial state. By the λ-transitions $\delta(q_0, \lambda) = u_0$, for all $u \in A$, the NFA guesses an axiom of S. The last state of each chain associated with units $u = x_1 x_2 \cdots x_\ell$ from A or T is connected to a next unit $v = y_1 y_2 \cdots y_m$ from T or E by λ-transitions $\delta(u_\ell, \lambda) = v_1$ if the last symbol of the first chain matches the first symbol of the second chain, that is, if $x_\ell = y_1$. In this way the assembling of units is simulated. Finally, the last state of each chain associated with an ending unit is made accepting, that is, $F = \{ q \mid q = v_m \text{ for some } v = y_1 y_2 \cdots y_m \in E \}$. So, there are no transitions from states at the end of chains associated with ending units.

The NFA constructed simulates derivations of S and vice versa, that is, for each derivation of a word $w \in L(S)$ that applies the units $u_0, u_1, \ldots, u_n$, there is an accepting path in M that starts from q_0 and runs through the chains associated with $u_0, u_1, \ldots, u_n$, and *vice versa*. This shows that every language generated by some single-stranded SAS is regular.

The strictness of the inclusion claimed follows since the regular language

$$L_{\text{un}} = \{a\} \cup \{ a^{2n} \mid n \geq 2 \}$$

is not even generated by any double-stranded SAS (Kutrib and Wendlandt 2012). □

Next we turn to two observations that are in the spirit of pumping lemmas. They are useful to show that languages are not generated by single-stranded SAS.

Lemma 8

1. Let $L \subseteq \Sigma^*$ be a language whose words have infinitely many unary factors, that is, for some $a \in \Sigma$, the set

   $$\{ n \geq 1 \mid \text{there are } u, v \in \Sigma^* \text{ such that } ua^n v \in L \}$$

   is infinite. If language L is generated by a single-stranded SAS then there is a unit $(a^i) \in T$, for some $i \geq 2$.
2. Let $L \subseteq \Sigma^*$ be a language such that for some $a, b \in \Sigma$, the set

   $$\{ n \geq 1 \mid \text{there are } u, v \in \Sigma^* \text{ such that } \ell, m \geq n \text{ and } ua^\ell b^m v \in L \}$$

   is infinite. If language L is generated by a single-stranded SAS then there is a unit $(a^i b^j) \in T$, for some $i, j \geq 1$.

Example 9 The language $L = \{ a^\ell b^m a^n \mid \ell, m, n \geq 1 \}$ is not generated by any single-stranded SAS. Contrarily let us assume that L is generated by a single-stranded SAS $S = \langle \Sigma, A, T, E \rangle$. Then Lemma 8 says that there are assembling units (a^{i_1}), $i_1 \geq 2$, (b^{i_2}), $i_2 \geq 2$, $(a^{i_3} b^{j_3})$, $i_3, j_3 \geq 1$, and $(b^{i_4} a^{j_4})$, $i_4, j_4 \geq 1$, in T. Moreover, there must be an axiom (a^r) with $r \geq 1$ in A and an ending unit (a^s) with $s \geq 1$ in E. Therefore, the word $a^{r+i_3-1} b^{j_3} b^{i_4-1} a^{j_4} a^{i_3-1} b^{j_3} b^{i_4-1} a^{j_4-1+s}$ can be derived as well, a contradiction. ■

It has been shown that the copy language $\{ [\$]_1 w [\$]_2 w [\$]_3 \mid w \in \{a, b\}^+ \}$, which is not even context free, is generated by some double-stranded SAS (Example 1). This result together with Theorem 7 implies that the generative capacity of double-stranded SAS is strictly larger than that of single-stranded SAS.

Theorem 10 *The family of languages generated by* 1-*stranded* SAS *is strictly included in the family of languages generated by* 2-*stranded* SAS.

3.3 Free string assembling systems

The next restricted variant are so-called *free* string assembling systems, where the control mechanism derived from the fact that the assembled strings have to overlap the last symbol of the current strand is relaxed. For this reason we allow the units to be subsets of $\Sigma^* \times \Sigma^*$.

A *free string assembling system* $S = \langle \Sigma, A, T, E \rangle$ is a string assembling system, where the units are elements of $\Sigma^* \times \Sigma^*$. That is, $A, T, E \subset \Sigma^* \times \Sigma^*$ These units are assembled according to the *derivation relation* $\overset{f}{\Rightarrow}$ by

1. $(uv, u) \overset{f}{\Rightarrow} (uvx, uy)$ if

(i) $(x, y) \in T \cup E$, for $x, y \in \Sigma^*$, and
(ii) $vx = yz$ or $vxz = y$, for $z \in \Sigma^*$, and

2. $(u, uv) \overset{f}{\Rightarrow} (uy, uvx)$ if

(i) $(y, x) \in T \cup E$, for $x, y \in \Sigma^*$, and
(ii) $vx = yz$ or $vxz = y$, for $z \in \Sigma^*$.

As for common SAS a derivation is said to be *successful* if it initially starts with an axiom from A, continues with assembling units from T, and ends with assembling an ending unit from E. The sets A, T, and E are not necessarily disjoint. Consequently, the free SAS can be seen as a special type of SAS, since they are neither a real restricted type nor an extension.

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

$$L(S) = \{\, w \in \Sigma^+ \mid (u, v) \overset{f}{\Rightarrow}{}^*(w, w) \text{ is a successful derivation} \,\},$$

with $(u, v) \in A$.

Example 11 Let $h : \{a_1, b_1\} \to \{a_2, b_2\}$ be a homomorphism with $h(a_1) = a_2$ and $h(b_1) = b_2$. The language $\{\, [\$]_0 w_1 [\$]_1 h(w_1) [\$]_2 \mid w_1 \in \{a_1, b_1\}^+ \,\}$ is not context free but generated by the free SAS $S = \langle \{a_1, a_2, b_1, b_2, [\$]_0, [\$]_1, [\$]_2\}, A, T, E \rangle$, where all following units are defined for all $x_1, y_1, z_1 \in \{a_1, b_1\}$.

1. $([\$]_0 x_1, \lambda) \in A$
2. $(x_1 y_1, \lambda) \in T$
3. $([\$]_1, [\$]_0) \in T$
4. $(x_1 [\$]_1, [\$]_0) \in T$
5. $(h(x_1)h(y_1), x_1 y_1) \in T$
6. $(h(x_1)[\$]_2, x_1 [\$]_1 h(z_1)) \in T$
7. $(h(x_1)h(y_1)[\$]_2, x_1 y_1 [\$]_1 h(z_1)) \in T$
8. $(\lambda, h(x_1)h(y_1)) \in T$
9. $(\lambda, [\$]_2) \in E$
10. $(\lambda, h(x_1)[\$]_2) \in E$

There are three basic ideas of the construction of S. First, the axiom (1) has an empty lower string. This ensures that subsequently only units (2) can be assembled until the derivation of the upper strand $[\$]_0 w_1 [\$]_1$ is completed by one of the units (3) or (4). Second, since the concluding $[\$]_2$ in the lower strand is only defined in the ending units (9) and (10), it is impossible to generate anything after $[\$]_2$. Third, the units assembling the upper w_1 extend the current strand at even positions and have length two, while units assembling w_1 in the lower strand extend the current strand at odd positions. Thus, they overlap, which ensures that the format of the generated words is correct, and that there are no symbols of $\{a_1, b_1\}$ and $\{a_2, b_2\}$ mixed up. These ideas are similarly applied to the construction of the units generating the subword $h(w_1)$. Here the units in the upper strand start at an odd position of $h(w_1)$ and at an even position in $h(w_1)$ in the lower strand.

Unit (5) copies the word w_1 with different indices after the $[\$]_1$ in the upper strand. The units (6) and (7) complete the derivation of the upper strand. Both place $[\$]_1$ in the lower strand, which ensures the equality of w_1 and $h(w_1)$. The generation of the lower strand and the whole derivation is completed by the units (8) and (9) or (10).

Since the order of the applications of the units is fixed, it is not possible to derive words of another form. ■

In order to show that the loss of the control mechanism based on overlapping weakens the generative capacity of SAS, we first show that certain witness languages cannot be generated by free string assembling systems.

Lemma 12 *Neither the language* $L = \{\, a^n b^n [\$] a^m \mid m, n \geq 1 \,\}$ *nor its iterated version* $L_+ = \{a\} L_0^*$ with $L_0 = \{\, a^{n-1} b^n [\$] a^m \mid m, n \geq 1 \,\}$ *can be generated by any free* SAS.

Proof In contrast to the assertion let us assume that L or L_+ is generated by some free SAS $S = \langle \{a, b, [\$]\}, A, T, E \rangle$. First, it is shown that there must exist units of both forms (a^{i_1}, a^{j_1}) with $i_1 > j_1 \geq 0$ and (a^{i_2}, a^{j_2}) with $j_2 > i_2 \geq 0$, or a unit of the form (a^i, a^i) with $i \geq 1$.

Consider the derivation of a word $w = ab[\$]a^m$ with m large enough. At some point in the derivation the generated strands must have the form (u, v), where one of the two strands, say u, is of the form $ab[\$]a^p$. If v is of the form $ab[\$]a^q$ as well, we may assume that $p \geq q$. In order to extend p and q to the large m, both strands can be extended by applying units of the form (a^i, a^i) with $i \geq 1$, or units of both forms (a^{i_1}, a^{j_1}), $i_1 > j_1 \geq 0$ and (a^{i_2}, a^{j_2}), $j_2 > i_2 \geq 0$, can be used. Now let $|v| < 3$. Then a unit of the form (a^k, λ), $k \geq 1$ can be applied which yields generated strands of the same form (u, v), or units (a^k, x), $k \geq 0$, $|x| \geq 1$, can be applied at most three times. Afterwards the generated strands are again of the same form (u, v). It follows that there must exist in S units of both forms (a^{i_1}, a^{j_1}), $i_1 > j_1 \geq 0$ and (a^{i_2}, a^{j_2}), $j_2 > i_2 \geq 0$, or a unit of the form (a^i, a^i), $i \geq 1$.

Next, consider the derivation of a word $w = a^n b^n [\$] a^m$ with m, n large enough. After an appropriate axiom has been chosen, the generated strands are of the form (u, v) with $u, v \in \{a\}^*$. Applying the unit (a^i, a^i), $i \geq 1$, shows that $a^{n+i} b^n [\$] a^m$ is generated by S as well, a contradiction. The same contradiction is obtained when units (a^i, a^i) do not exist but units of both forms (a^{i_1}, a^{j_1}), $i_1 > j_1 \geq 0$ and (a^{i_2}, a^{j_2}), $j_2 > i_2 \geq 0$. In this case applying $j_2 - i_2$ times the unit (a^{i_1}, a^{j_1}) and $i_1 - j_1$ times the unit (a^{i_2}, a^{j_2}) shows that $a^{n+i_1 j_2 - i_2 j_1} b^n [\$] a^m$ is also generated. Note that $i_1 j_2 > i_2 j_1$.

So, we conclude that neither L nor L_+ can be generated by any free SAS. □

Theorem 13 *There is a language generated by some* SAS *that cannot be generated by any free* SAS.

Proof The assertion is shown by the witness language

$$L_+ = \{a\}L_0^* \text{ with } L_0 = \{\, a^{n-1}b^n[\$]a^m \mid m,n \geq 1 \,\}$$

that cannot be generated by any free SAS by Lemma 12. However, L_+ is generated by the SAS $\langle \{a,b,[\$]\}, A, T, E\rangle$, where the units are defined as follows.

1. $(a,a) \in A$
2. $(aa,a) \in T$
3. $(ab,a) \in T$
4. $(bb,aa) \in T$
5. $(b[\$],ab) \in T$
6. $([\$],bb) \in T$
7. $([\$],b[\$]a) \in T$
8. $([\$],aa) \in T$
9. $([\$]a,a) \in T$
10. $(a,a) \in E$

This completes the description of the units of S. Starting with the axiom the derivation can immediately be terminated by assembling the ending unit, which results in the derivation of a word of $\{a\}(L_0)^0$. Alternatively, the unit (2) can be assembled arbitrarily often followed by assembling the unit (3). There are no other possibilities to start the derivation. In this way prefixes (a^nb,a) are generated for $n \geq 1$. Now the only possibility to continue is to assemble the unit (4) as long as the a's in the upper strand are matched, followed by assembling the unit (5). This generates $(a^nb^n[\$],a^nb)$. Next, unit (6) has to be used as long as the b's in the upper strand are matched, followed by unit (7). This generates $(a^nb^n[\$],a^nb^n[\$]a)$. Subsequently, the only applicable units are (8) or (9). Repeatedly assembling unit (8) generates $(a^nb^n[\$],a^nb^n[\$]a^m)$ for $m \geq 1$. This process ends when unit (9) is assembled. From now on unit (2) has to be used until $(a^nb^n[\$]a^m,a^nb^n[\$]a^m)$ is generated, which results in the derivation of $\{a\}(L_0)^1$. Finally, the whole process can be repeated where in each iteration the leftmost a overlaps. So, the language $\{a\}L_0^*$ is generated. □

Though there is a language that separates SAS from free SAS, the differences in the generative capacities disappear for unary languages. In order to show that SAS and free SAS are equally powerful with respect to unary languages the next lemma is useful.

Lemma 14 *For each free* SAS *accepting a unary language an equivalent free* SAS *can be constructed whose sole ending unit is* (λ,λ).

Proof Let $S = \langle \{a\}, A, T, E\rangle$ be a unary free SAS. Since S is unary, in any successful derivation the ordering of the applied units from T is arbitrary. For any ordering the same word is generated. Moreover, any successful derivation starts with some axiom and ends with some ending unit. Now the axioms and ending units are merged into new axioms. More precisely, the free SAS $S' = \langle \{a\}, A', T, E'\rangle$ is constructed by setting $E' = \{(\lambda,\lambda)\}$ and

$$A' = \{\, (a^{i_1+i_2}, a^{j_1+j_2}) \mid (a^{i_1}, a^{j_1}) \in A \text{ and } (a^{i_2}, a^{j_2}) \in E \,\}.$$

Since in any successful derivation in S an axiom, say $(a^{i_1}, a^{j_1}) \in A$, and an ending unit, say (a^{i_2}, a^{j_2}), is assembled, there is a successful derivation generating the same word in S' by assembling the axiom $(a^{i_1+i_2}, a^{j_1+j_2}) \in A'$ and the ending unit $(\lambda,\lambda) \in E'$, and vice versa. □

Proposition 15 *A unary language can be generated by an* SAS *if and only if it can be generated by a free* SAS.

Proof First, we show how a free string assembling system can simulate a given unary SAS $S = \langle \{a\}, A, T, E\rangle$. To this end, a free SAS $S' = \langle \{a\}, A, T', E'\rangle$ is constructed as follows. For every unit $(a^i, a^j) \in T$, $i,j \geq 1$ the unit $(a^{i-1}, a^{j-1}) \in T'$ is defined, and similarly for E and E'. In this way, clearly, each derivation step $(a^u, a^v) \Rightarrow (a^{u+i-1}, a^{v+j-1})$ in S using the unit (a^i, a^j) from T or E is simulated by a derivation step $(a^u, a^v) \overset{f}{\Rightarrow} (a^{u+i-1}, a^{v+j-1})$ in S' using the unit (a^{i-1}, a^{j-1}) from T' or E', and vice versa.

Second, it has to be shown how an SAS can simulate a given unary free SAS. Let $S = \langle \{a\}, A, T, E\rangle$ be a free SAS generating a unary language. By Lemma 14 we may assume that $E = \{(\lambda,\lambda)\}$. Now, an SAS $S' = \langle \{a\}, A', T', E'\rangle$ is constructed as follows. In order to overcome the problem that the units in A may have empty strings but the units in A' may not, some units have to be merged. This can be done, since unary units always fit to the current double strand. The set of axioms is defined as $A' = A'_0 \cup A'_1 \cup A'_2 \cup A'_3 \cup A'_4$, where

$$\begin{aligned}
A'_0 &= \{\, (a^i, a^j) \mid (a^i, a^j) \in A, \text{ for } i,j \geq 1 \,\},\\
A'_1 &= \{\, (a^{i_1+i_2}, a^j) \mid (a^{i_1}, \lambda) \in A \text{ and } (a^{i_2}, a^j) \in T,\\
&\qquad \text{for } i_1, j \geq 1 \,\},\\
A'_2 &= \{\, (a^i, a^{j_1+j_2}) \mid (\lambda, a^{j_1}) \in A \text{ and } (a^i, a^{j_2}) \in T,\\
&\qquad \text{for } i, j_1 \geq 1 \,\},\\
A'_3 &= \{\, (a^i, a^j) \mid (\lambda,\lambda) \in A \text{ and } (a^i, a^j) \in T, \text{ for } i,j \geq 1 \,\},\\
A'_4 &= \{\, (a^i, a^j) \mid (\lambda,\lambda) \in A \text{ and } (a^i, \lambda) \in T,\\
&\qquad (\lambda, a^j) \in T, \text{ for } i,j \geq 1 \,\}.
\end{aligned}$$

So, any axiom in A that has at least one empty string is replaced by an axiom without empty strings (unless the axiom does not appear in any successful derivation and is

useless). The set of ending units is defined as $E' = \{(a, a)\}$, and the set T' as

$$T' = \{\, (a^{i+1}, a^{j+1}) \mid (a^i, a^j) \in T \,\}.$$

Now, for a successful derivation generating a non-empty word in S that starts with axiom u_0, assembles the units $u_1, u_2, \ldots, u_k$, and ends with the sole ending unit (λ, λ) there is a successful derivation in S' as follows. If u_0 does not contain an empty string then the derivation starting with axiom $u'_0 = u_0 \in A'_0$, assembling the units $u'_1, u'_2, \ldots, u'_k$, where $u'_\ell = (a^{i+1}, a^{j+1})$ if $u_i = (a^i, a^j)$, $1 \le \ell \le k$, and ending with $(a, a) \in E'$ generates the same word in S', and vice versa.

If u_0 contains one empty string then there must exist a unit in the sequence $u_1, u_2, \ldots, u_k$ whose corresponding string is non-empty. Similarly, if u_0 contains two empty strings then there must exist either one unit in the sequence $u_1, u_2, \ldots, u_k$ whose both strings are non-empty, or there are two units in the sequence, where one unit has a non-empty upper string and the other one a non-empty lower string. Say there is one unit u_j in the sequence. Now the derivation starting with the axiom that merges u_0 and u_j, assembling the units $u'_1, u'_2, \ldots, u'_{j-1}, u'_{j+1}, \ldots, u'_k$ and ending with $(a, a) \in E'$ generates the same word in S', and vice versa. Similarly, if there are two units that are merged with u_0. So, we conclude that S and S' generate the same unary language. □

The general relation of SAS and free SAS is an open question. In Kutrib and Wendlandt (2012) a certain pumping lemma for SAS has been proven. It turns out to be a useful tool to show that certain languages cannot be generated by SAS. This lemma can be shown for free SAS in the same way as for common ones.

Lemma 16 *Let $L \subseteq \Sigma^*$ be a language generated by a free* SAS. *If $|a^+ \cap L| = \infty$, for some symbol $a \in \Sigma$, then there exist constants $p, q \ge 1$ such that $a^p v \in L$, $v \in \Sigma^*$, implies $a^{p+q} v \in L$.*

In Kutrib and Wendlandt (2012) it has been shown that the family of languages generated by SAS is a subset of the family of languages accepted by nondeterministic one-way two-head finite automata. Clearly this result can easily be adapted to free SAS.

Lemma 17 *Let S be a free* SAS. *There exists a nondeterministic one-way two-head finite automaton M that accepts $L(S)$.*

3.4 One-set string assembling systems

The next restricted variant are so-called *one-set* string assembling systems, where the control mechanism that the units are arranged in the three sets of axioms, assembling units, and ending units is relaxed. So, in particular, a derivation can end at any point of the derivation where both strands have the same length.

A string assembling system is said to be *one set*, if the sets A, T, and E are equal. Accordingly we shortly write $S = \langle \Sigma, T \rangle$. The units are assembled according to the *derivation relation* $\Rightarrow$ of common SAS. Now every derivation that begins with a single unit from T and ends with both strands identical is *successful*, and the *language $L(S)$ generated* by S is as before defined to be the set

$$L(S) = \{\, w \in \Sigma^+ \mid (p, q) \Rightarrow^* (w, w) \text{ is a successful derivation} \,\}$$

with $(p, q) \in T$.

Example 18 The language $L = \{\, a^m b^n \mid m + n \ge 2 \,\}$ is generated by the one-set SAS $S = \langle \{a, b\}, T \rangle$ with the following units.

1. $(aa, aa) \in T$
2. $(ab, ab) \in T$
3. $(bb, bb) \in T$ ■

The next lemma is helpful for showing that certain languages cannot be generated by one-set SAS.

Lemma 19 *Let $S = \langle \Sigma, T \rangle$ be a one-set* SAS, *$w_1, w_2 \in \Sigma^*$, and $x \in \Sigma$. If $w_1 x$ and $x w_2$ are generated by S then $w_1 x w_2$ is generated by S as well.*

Proof Let $S = \langle \Sigma, T \rangle$ be a one-set SAS that generates $w_1 x$ and $x w_2$. Then there are successful derivations $(u, v) \Rightarrow^* (w_1 x, w_1 x)$ and $(u', v') \Rightarrow^* (x w_2, x w_2)$ with assembling units $(u, v), (u', v') \in T$, where the first letter in u' and v' must be x.

This implies that $(u, v) \Rightarrow^* (w_1 x, w_1 x) \Rightarrow (w_1 u', w_1 v') \Rightarrow^* (w_1 x w_2, w_1 x w_2)$ is also a successful derivation. □

Theorem 20 *The family of languages generated by one-set* SAS *is strictly included in the family of languages generated by* SAS.

Proof A one-set SAS $\langle \Sigma, T \rangle$ can directly be simulated by an SAS $\langle \Sigma, A, T, E \rangle$, where $A = T$ and $E = T$. Therefore, the family of languages generated by one-set SAS is included in the family of languages generated by SAS.

The language $L = \{\, a^m b^n a^\ell \mid m, n, \ell \ge 1 \,\}$ is generated by some SAS. Assume that it is also generated by some one-set SAS. The words *aba* and *aaba* do belong to L. So, Lemma 19 implies that *abaaba* belongs to L as well, a contradiction. □

In order to study the unary case we first utilize the fact that in one-set SAS there are no explicit ending units. This allows us to characterize the unary one-set SAS languages.

A unary language L is said to be *expanded* from language L_0 by $z \geq 1$ if

$$L = \{\, a^n \mid n = z \cdot m + 1, \text{ for } a^m \in L_0 \,\}.$$

We recall a well-known useful fact which is related to number theory and Frobenius numbers (see, for example, Shallit (2008) for a survey).

Lemma 21 *Let $x_1, x_2, \ldots, x_k$ be positive integers. Then every sufficiently large integer can be written as a non-negative integer linear combination of the x_i's if and only if the x_i's are relatively prime, that is, if $\gcd(x_1, x_2, \ldots, x_k) = 1$. The largest positive integer which cannot be represented as a non-negative integer linear combination of the x_i is their Frobenius number $g(x_1, x_2, \ldots, x_k)$. In particular, for $k = 2$, the largest integer that cannot be represented is $x_1x_2 - x_1 - x_2$.*

We use the previous lemma to characterize the unary languages generated by one-set SAS.

Proposition 22 *Every unary language generated by a one-set* SAS *is either empty, the singleton $\{a\}$, or expanded from a cofinite language.*

Proof The one-set SAS $\langle \{a\}, \{(a, aa)\} \rangle$ generates the empty language, and the one-set SAS $\langle \{a\}, \{(a, a)\} \rangle$ generates the language $\{a\}$.

Since in one-set SAS there are no explicit ending units, any successful derivation can be extended by continuing with another successful derivation. The result is the concatenation of both words generated where one overlapping symbol is chopped off. In general, if more than one word is concatenated, for all but the first word one overlapping symbol is chopped off. So, any finite language generated by a unary one-set SAS is either empty or $\{a\}$.

Next, assume that the unary one-set SAS language L is infinite. If it includes two different words whose lengths are $x > 1$ and $y > 1$ such that $x - 1$ and $y - 1$ are relatively prime then, by Lemma 21, any word a^n with length $n > (x-1)(y-1) - (x-1) - (y-1) + 1$ is generated by S. So, L is expanded from a cofinite language by $z = 1$.

For the last case we assume that for each two different words a^x and a^y with $x, y > 1$, in L the numbers $x - 1$ and $y - 1$ are *not* relatively prime. Then we consider *all* words in L whose length is at least two and denote the greatest common divisor of their lengths decreased by one by d, where it is not excluded that $d = 1$. Now we consider the set $N = \{\, n \mid a^{d \cdot n + 1} \in L \,\}$. We choose an arbitrary $n_0 \in N$ and denote the prime factors of n_0 by $d_1, d_2, \ldots, d_k$.

Next, we choose some number $n_1 \in N$ that does not contain the prime factor d_1. Such a number must exist since otherwise all numbers in N would be divisible by d_1 which is a contradiction to the definition of N. Similarly, we continue to choose some (not necessarily distinct) numbers $n_i \in N$, $2 \leq i \leq k$, which are not divisible by d_i, respectively. We conclude that the greatest common divisor of $n_0, n_1, \ldots, n_k$ is 1. So, by Lemma 21 it follows that there exists a positive integer ℓ, that is the Frobenius number $g(n_0, n_1, \ldots, n_k)$, such that all numbers $n > \ell$ can be represented as non-negative integer linear combination of $n_0, n_1, \ldots, n_k$. In particular, the set of representable numbers is cofinite. Since all words in L have a length of the form $d \cdot n + 1$, L is expanded from a cofinite language by $z = d$.

Finally, if the word a belongs to L then S includes necessarily the unit (a, a). Except for generating the word a, the unit is useless since it does not extend the current strands. However, the word a is expanded from λ by any $z \geq 1$. □

Proposition 22 reveals that, for example, the finite singleton language $\{aa\}$ cannot be generated by any one-set SAS. On the other hand, they are able to generate infinite languages.

Corollary 23 *The family of finite languages and the family of languages generated by one-set* SAS *are incomparable.*

It turns out that the two possibilities to relax one of the two control mechanisms studied and to keep the other yields incomparable generative capacities.

Theorem 24 *The families of languages generated by free* SAS *and by one-set* SAS *are incomparable.*

Proof By Proposition 22 the singleton language $\{aa\}$ cannot be generated by any one-set SAS. On the other hand, it is generated by the free SAS

$$\langle \{a\}, \{(aa, aa)\}, \{(\lambda, \lambda)\}, \{(\lambda, \lambda)\} \rangle.$$

Conversely, Lemma 12 shows that the language

$$L_+ = \{a\}L_0^* \text{ with } L_0 = \{\, a^{n-1}b^n[\$]a^m \mid m, n \geq 1 \,\}$$

cannot be generated by any free SAS. On the other hand, the SAS generating L_+ constructed in the proof of Theorem 13 can easily be translated into a one-set SAS. The axiom (a, a) can safely be removed since any successful derivation has to continue with unit (2) or unit (3) starting with symbol a. Moreover the units (2) and (3) are the only units with which it can be started, since all the other units are not valid double strands. The ending unit (a, a) just terminates the derivation without extending the strands. □

3.5 Pure string assembling systems

The last restriction considered in this section is a combination of both restrictions studied above. For so-called *pure* string assembling systems neither the control mechanism

derived from the fact that the assembled strings have to overlap the last symbol of the current strand nor the mechanism derived from the fact that the units are arranged in the three sets of axioms, assembling units, and ending units are available. Since the control mechanisms are equal to those of the Post Correspondence Problem (PCP), this family of languages can be seen as the family of languages generated by PCP instances.

A one-set SAS $S = \langle \Sigma, T \rangle$ is said to be *pure* if $T \subset \Sigma^* \times \Sigma^*$ and its units are assembled according to the *derivation relation* $\overset{f}{\Rightarrow}$.

Example 25 Let $S = \langle \{a, b, \bar{a}, \bar{b}\}, T \rangle$ be the pure SAS, where the following units are defined for the homomorphism $h(a) = \bar{a}$ and $h(b) = \bar{b}$ and all $x \in \{a, b\}$.

1. $(x, \lambda) \in T$
2. $(h(x), x) \in T$
3. $(\lambda, h(x)) \in T$.

Since the context-free languages are closed under intersection with regular sets, and the intersection of the languages $L(S) \cap \{a, b\}^+ \{h(a), h(b)\}^+$ results in the non-context-free language $\{ wh(w) \mid w \in \{a, b\}^+ \}$, the language $L(S)$ is not context free.

In order to generate words from $L(S) \cap \{a, b\}^+ \{h(a), h(b)\}^+$ the derivation has to start with units (1). After assembling repeatedly units (1), the only possibility to obtain a matching lower string is to continue with a unit (2). Since the word generated has to belong to the regular set $\{a, b\}^+ \{h(a), h(b)\}^+$, units (2) have to be assembled until strands of the form ($wh(w)$, w) are derived. Again, the only possibility to obtain a matching lower string is to continue with units (3) until a word of the desired form is derived. ■

In general, we obtain the following property of pure SAS languages.

Proposition 26 *Every language L generated by a pure* SAS *is Kleene plus closed, that is, $L = L^+$.*

Proof Let S be a pure string assembling system and consider successful derivations $(u_1, v_1) \overset{f}{\Rightarrow}^* (w_1, w_1)$ and $(u_2, v_2) \overset{f}{\Rightarrow}^* (w_2, w_2)$ in S. Since there is no explicit ending unit, also the derivation

$$(u_1, v_1) \overset{f}{\Rightarrow}^* (w_1, w_1) \overset{f}{\Rightarrow} (w_1 u_2, w_1 v_2) \overset{f}{\Rightarrow}^* (w_1 w_2, w_1 w_2)$$

is successful. If there is no successful derivation in S at all then $L = L(S)$ is empty. Since it holds that $\emptyset^+ = \emptyset$, we conclude in any case $L^+ \subseteq L$ and, thus, $L = L^+$. □

For unary languages, the difference between one-set SAS and pure SAS is subtle. Following the argumentation in Kutrib and Wendlandt (2018) a unary language L is said to be *stretched* from language L_0 by $z \geq 1$ if $L = \{ a^n \mid n = z \cdot m, \text{ for } a^m \in L_0 \}$. So, the difference between the properties of being expanded and being stretched is the addition of one to the word lengths for expanded languages.

Proposition 27 *Every unary language generated by a pure* SAS *is either empty or a stretched cofinite language.*

Proof Almost literally, the proof of Proposition 22 applies here as well. The only difference is that for one-set SAS first the concatenated word lengths are each decreased by one (due to the overlapping) and then the result is extended by one symbol again (due to the initial unit). For pure SAS neither decreasing of the word lengths nor extending the result is necessary. This means that the unary languages generated by pure SAS are stretched instead of expanded, and that the singleton language $\{a\}$ is not generated. □

Corollary 28 *The family of finite languages and the family of languages generated by pure* SAS *are incomparable.*

In particular, the subtle difference between being stretched or being expanded from a cofinite language makes both language families incomparable.

Theorem 29 *The families of languages generated by (unary) one-set* SAS *and (unary) pure* SAS *are incomparable.*

Proof The unary language $L_e = \{ a^n \mid n \geq 2 \text{ and } n \text{ even} \}$ is generated by the pure SAS $\langle \{a\}, \{(aa, aa)\} \rangle$. Assume L_e is generated by some one-set SAS S. Since a^{2x} and a^{2y} belong to L_e for some $x, y \geq 1$, the fact that S is one-set yields that $a^{2x+2y-1}$ is generated by S as well. But $a^{2x+2y-1}$ does not belong to L_e.

Conversely, the unary language $L_o = \{ a^n \mid n \geq 3 \text{ and } n \text{ odd} \}$ is generated by the one-set SAS $\langle \{a\}, \{(aaa, aaa)\} \rangle$. Assume L_o is generated by some pure SAS S'. Since a^{2x+1} and a^{2y+1} belong to L_o for some $x, y \geq 1$, the fact that S' is pure yields that $a^{2x+2y+2}$ is generated by S' as well. But $a^{2x+2y+2}$ does not belong to L_o. □

On the other hand, the family of languages generated by pure SAS is strictly included in the family of languages generated by free SAS.

Theorem 30 *The family of languages generated by pure* SAS *is strictly included in the family of languages generated by free* SAS.

Proof The inclusion of the language families follows immediately from the definitions, since both types of SAS

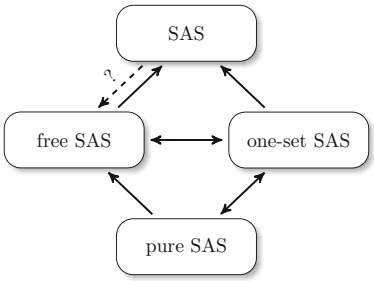

Fig. 3 Inclusion structure of language families. A single arrow means strict inclusion, a double arrow means incomparability, and the dashed line together with the solid line indicates the open problem whether there is a strict inclusion or incomparability

are based on the derivation relation $\overset{f}{\Rightarrow}$. So, a pure SAS $\langle \Sigma, T \rangle$ can directly be simulated by a free SAS $\langle \Sigma, A, T, E \rangle$, where $A = T = E$.

The strictness of the inclusion follows, for example, by Proposition 15 that says that each unary SAS language is also generated by some free SAS, and Theorem 29. Witnesses for the strictness are, for example, the languages $\{a\}$ or $\{ a^n \mid n \geq 3 \text{ and } n \text{ odd} \}$. □

The relations between the variants of string assembling systems studied so far are summarized in Fig. 3. We conclude the section by the relations with the language families of the Chomsky Hierarchy. While string assembling systems where the units are arranged in three sets, that is SAS and free SAS, can generate all finite languages by using corresponding axioms and a trivial ending unit, the absence of this control mechanism yields incomparability with the family of finite languages (see Corollary 23 and Corollary 28).

Proposition 31 *The families of languages generated by free* SAS*, one-set* SAS*, and pure* SAS *are incomparable with the families of regular and (deterministic) (linear) context-free languages.*

Proof The unary regular language $L_{\text{un}} = \{a\} \cup \{ a^{2n} \mid n \geq 2 \}$, is not generated by any SASKutrib and Wendlandt (2012). So, from the results above we conclude that it is not generated by any of the three remaining variants of SAS.

On the other hand, non-context-free languages generated by SAS, free SAS, and pure SAS are given in the Examples 1, 11, and 25. A slight modification of the construction of Example 25 shows that one-set SAS can generate non-context-free languages as well. □

The strict inclusion of the family of languages generated by SAS in the complexity class $\mathsf{NL} = \mathsf{NSPACE}(\log n)$ is shown in Kutrib and Wendlandt (2012). Since, in turn, NL is strictly included in $\mathsf{NSPACE}(n)$ (see, for example, Papadimitriou 1994), which is equal to the family of context-sensitive languages, we obtain the following corollary.

Corollary 32 *The families of languages generated by free* SAS*, one-set* SAS*, and pure* SAS *are strictly included in* NL *and, thus, in the family of context-sensitive languages.*

4 String assembling systems with multiple strands

The most powerful mechanism of string assembling systems is the ability to have two strands that can grow independently from each other. This seems to be the reason for the strong connection to one-way two-head finite automata. It has been shown that one-way k-head finite automata are less powerful than one-way $(k+1)$-head finite automata in Yao and Rivest (1978). Thus a proper head hierarchy can be concluded. Apart from biological reasons it seems to be natural to examine multi-stranded string assembling systems and to ask whether a similar hierarchy can be achieved. The special case of string assembling systems with one strand has been investigated in Subsection 3.2.

Before we start with the definition of such systems, we define what a *valid* k-stranded sequence is. A k-stranded sequence $(w_1, w_2, \ldots, w_k)$ is said to be *valid* if for every $1 \leq i < j \leq k$ it holds that either w_i is a prefix of w_j or *vice versa*.

A k-stranded *SAS* (SAS-k) is a quadruple $\langle \Sigma, A, T, E \rangle$, where

1. Σ is the finite, nonempty set of *symbols*,
2. $A \subset (\Sigma^+)^k$ is the finite set of *axioms* where each axiom is a valid sequence of words $(w_1, w_2, \ldots, w_k)$,
3. $T \subset (\Sigma^+)^k$ is the finite set of *assembly units*, and
4. $E \subset (\Sigma^+)^k$ is the finite set of *ending assembly units*.

The next definition formally says how the units are assembled.

Let $S = \langle \Sigma, A, T, E \rangle$ be a k-stranded SAS. The *derivation relation* $\Rightarrow$ is defined on specific subsets of $(\Sigma^+)^k$ by $(u_1 x_1, u_2 x_2, \ldots, u_k x_k) \Rightarrow (u_1 x_1 v_1, u_2 x_2 v_2, \ldots, u_k x_k v_k)$, $u_i, v_i \in \Sigma^*, x_i \in \Sigma$, if

i) $(x_1 v_1, x_2 v_2, \ldots, x_k v_k) \in T \cup E$ and
ii) $(u_1 x_1 v_1, u_2 x_2 v_2, \ldots, u_k x_k v_k)$ is a valid sequence of words.

As for common SAS a derivation is said to be *successful*, if it initially starts with an axiom from A, continues with assembling units from T, and ends with assembling an ending unit from E. The process necessarily stops when an ending assembly unit is added. The sets A, T, and E are not necessarily disjoint.

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

$$L(S) = \{\, w \in \Sigma^+ \mid (u_1, u_2, \ldots, u_k) \Rightarrow^* (w, w, \ldots, w) \text{ is a successful derivation} \,\},$$

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

First we derive an upper bound for the generative capacity. The upper bound is given by the recognition power of nondeterministic one-way k-head finite automata. Their variant of two-way k-head finite automata characterizes the complexity class $\mathsf{NL} = \mathsf{NSPACE}(\log n)$ Hartmanis (1972). So, together with the next theorem, we obtain that the family of languages generated by SAS-k, $k \geq 1$, is properly included in NL.

Theorem 33 *Let $k \geq 1$ and S be a k-stranded* SAS. *Then there exists a nondeterministic one-way k-head finite automaton M that accepts $L(S)$.*

Proof The main idea of the construction of M is that each of the k heads is used to check one strand. Initially, M guesses one of the axioms. The guess is verified by reading the k strands of the unit with the k input heads. In this way, it is checked whether the input given fits to the unit guessed. The last symbols are the new overlappings and the heads stay on them for the new guess. Next, M guesses depending on the currently scanned input symbols (the current overlappings), which assembly unit comes next. Then the guess is verified. After each verification, M determines whether the assembling process is completed and guesses another unit to be assembled otherwise. If an ending unit is guessed then after its verification M moves all heads synchronously one step to the right. If all heads see the right endmarker, M accepts. □

In Kutrib and Wendlandt (2012) it has been shown that double-stranded SAS can be simulated by one-way two-head finite automata. Thus, Theorem 33 allows an immediate generalization of results to k-stranded SAS.

It is known from Kutrib et al. (2016) that nondeterministic one-way k-head finite automata cannot accept the linear context-free language $\{\, wcw^R \mid w \in \{a,b\}^* \,\}$. Moreover, as mentioned above, the non-context-free language $\{\, [\$]_1 w [\$]_2 w [\$]_3 \mid w \in \{a,b\}^+ \,\}$ is generated by some common SAS. Since NL is strictly included in $\mathsf{NSPACE}(n)$ (see, for example, Papadimitriou 1994), which in turn is equal to the family of context-sensitive languages, we have the following relations.

Corollary 34 *For $k \geq 1$, the family of languages generated by SAS-k is strictly included in* NL *and, thus, in the family of context-sensitive languages.*

For $k \geq 2$, the family of languages generated by SAS-k is incomparable with the family of (deterministic) (linear) context-free languages.

Next, we turn to an infinite, dense, and strict strand hierarchy. The question whether there is a proper head hierarchy for one-way multi-head finite automata has been raised in Rosenberg (1966). It has been answered in the affirmative in Yao and Rivest (1978). The basic witness languages are

$$L'_n = \{\, w_1 * w_2 * \cdots * w_{2n} \mid w_i \in \{a,b\}^*, \; w_i = w_{2n+1-i}, 1 \leq i \leq 2n \,\}$$

which can be accepted by a (nondeterministic) one-way k-head finite automaton if and only if $n \leq \binom{k}{2}$. Here we use a slight modification of these languages.

Let $h : \{\, a_i, b_i \mid 1 \leq i \leq 2n \,\}^* \to \{a,b\}^*$ be a homomorphism with $h(a_i) = a$ and $h(b_i) = b$, for $1 \leq i \leq 2n$. Then we define

$$L_n = \{\, [\$]_0 w_1 [\$]_1 w_2 [\$]_2 \cdots [\$]_{2n-1} w_{2n} [\$]_{2n} \mid w_i \in \{a_i, b_i\}^*, h(w_i) = h(w_{2n+1-i}), 1 \leq i \leq 2n \,\}.$$

Clearly the head hierarchy for one-way multi-head finite automata is witnessed by the languages L_n as well, since the transduction from L'_n to L_n can be done by a sequential transducer, that can be simulated by a multi-head finite automaton in parallel. So, a strand hierarchy for SAS would follow if there is a k-stranded SAS generating the language L_n, $n = \binom{k}{2}$. The construction is shown in the proof below. Afterwards the construction is applied to an example to illustrate the details.

Theorem 35 *Let $k \geq 1$ be an integer. The family of languages generated by SAS-k is strictly included in the family of languages generated by* SAS-$(k+1)$.

Proof It can be concluded from Theorem 7 that the family of languages generated by 1-stranded SAS is a proper subfamily of the family of languages generated by 2-stranded SAS. As mentioned before, for $k \geq 2$ it is sufficient to show that language L_n with $n = \binom{k}{2}$ is generated by some SAS-k. To this end, we now construct the k-stranded string assembling system $S = \langle \Sigma, A, T, E \rangle$, where $A = \{([\$]_0, [\$]_0, \ldots, [\$]_0)\}$ and $E = \{([\$]_{2n}, [\$]_{2n}, \ldots, [\$]_{2n})\}$. The set T of assembly units is defined below. Example 36 illustrates the construction.

The examples above show how words between delimiters can be copied to another position of another strand. This is the underlying technique of the construction. However, whenever this technique is used to copy a

subword of L_n from one strand to another or to compare two subwords on different strands, clearly, due to the positions of matching subwords these two strands cannot be used for any further pair of matching subwords. The idea of the construction is as follows. Each pair of different strands is used to copy one subword to its matching position. In this way, $\binom{k}{2}$, that is, the required n matching subwords can be generated. The initial phase of the generation is slightly different.

Initially, the first strand is constructed up to subword $2n-(k-1)$. In this process the content of the subwords is nondeterministically guessed. For the rest of the proof let $x, y \in \{a, b\}$. For $1 \leq i \leq 2n-(k-1)$ we define the units

1. $([\$]_{i-1}x_i, [\$]_0, \ldots, [\$]_0)$,
2. $(x_i y_i, [\$]_0, \ldots, [\$]_0)$,
3. $(x_i[\$]_i, [\$]_0, \ldots, [\$]_0)$.

In particular, these units ensure that the format of the word generated is correct up to subword $2n-(k-1)$.

Next, the remaining $k-1$ subwords are copied from their matching positions. More precisely, subword $i-1$ of strand i is copied to position $2n+2-i$ of strand 1. Here we define the corresponding units. However, in order to apply these units, first the leftmost subwords of strand i have to be completed by taking them from strand 1 by units 10, 11, and 12 below. In order to know which subwords of strand i have not to be taken from strand 1 by these units, for any strand i, a set ω_i is maintained that contains all indices of subwords that are copied from or to the strand i. For $2 \leq i \leq k$ and $z_j \in \Sigma$, $j \in \{1, 2, \ldots, k\} \setminus \{1, i\}$, we define the units

4. $(z_1, z_2, \ldots, z_k)$ with $z_1 = [\$]_{2n+1-i}x_{2n+2-i}$ and $z_i = [\$]_{i-2}x_{i-1}$,
5. $(z_1, z_2, \ldots, z_k)$ with $z_1 = x_{2n+2-i}y_{2n+2-i}$ and $z_i = x_{i-1}y_{i-1}$,
6. $(z_1, z_2, \ldots, z_k)$ with $z_1 = x_{2n+2-i}[\$]_{2n+2-i}$ and $z_i = x_{i-1}[\$]_{i-1}$,

and add $i-1$ to ω_i and $2n+2-i$ to ω_1.

Now the first strand is complete, its format is correct, and the first $k-1$ subwords match their mates at the end of the strand. This completes the initial phase.

In the second phase, the remaining pairs of subwords have to be compared. The overall generation may only end successfully if the matching is verified. The comparisons are done by trying to copy one subword to the position of its matching subword. Clearly, the copying can only be successful if the subwords (which are already fixed in the first strand) are equal. To this end, pairs of strands are used. The first strand has already been used together with each other strand to ensure that $k-1$ subwords match their mates. Next, strand 2 is used together with strands 3 to k to compare the subwords k up to $k-1+k-2$ with their matching mates. Subsequently, strand 3 is used together with strands 4 to k to compare the subwords $k-1+k-2+1$ up to $k-1+k-2+k-3$ with their matching mates, and so on. Formally, for $2 \leq i < j \leq k$, strand i is used together with strand j to compare the subword at position $\alpha = 2n+(i+1)-j-(k-1+k-2+\cdots+k-(i-1))$ with the subword at position $\beta = j-i+(k-1+k-2+\cdots+k-(i-1))$. For $2 \leq i < j \leq k$ and $z_\ell \in \Sigma$, $\ell \in \{1, 2, \ldots, k\} \setminus \{i, j\}$, we define the units

7. $(z_1, z_2, \ldots, z_k)$ with $z_i = [\$]_{\alpha-1}x_\alpha$ and $z_j = [\$]_{\beta-1}x_\beta$,
8. $(z_1, z_2, \ldots, z_k)$ with $z_i = x_\alpha y_\alpha$ and $z_j = x_\beta y_\beta$,
9. $(z_1, z_2, \ldots, z_k)$ with $z_i = x_\alpha[\$]_\alpha$ and $z_j = x_\beta[\$]_\beta$,

and add α to ω_i and β to ω_j.

As mentioned before, in order to apply these units, first the leftmost subwords of strands i and j have to be generated. To this end, the subwords are completed with arbitrary symbols unless they are used for comparisons in which case the subword number appears in the associated set ω. Actually, completing with arbitrary symbols means to copy the subword from the first strand since there is no other possibility to generate a word successfully.

Since the first strand is already complete, no further units have to be provided for its subwords. So, for all $2 \leq j \leq k$, $i \in (\{1, 2, \ldots, 2n\} \setminus \omega_j)$, and $z_\ell \in \Sigma$, $\ell \neq j$ we define the units

10. $(z_1, z_2, \ldots, z_k)$ with $z_j = [\$]_{i-1}x_i$,
11. $(z_1, z_2, \ldots, z_k)$ with $z_j = x_i y_i$,
12. $(z_1, z_2, \ldots, z_k)$ with $z_j = x_i[\$]_i$.

This completes the construction of the k-stranded SAS. □

We present an example for language L_3 which is generated by a 3-stranded string assembling system.

Example 36 The language

$$L_3 = \{\, [\$]_0 w_1 [\$]_1 w_2 [\$]_2 w_3 [\$]_3 w_4 [\$]_4 w_5 [\$]_5 w_6 [\$]_6 \mid w_i \in \{a_i, b_i\}^*, h(w_i) = h(w_{2\cdot 3+1-i}), 1 \leq i \leq 6 \,\}$$

is generated by the 3-stranded string assembling system $S = \langle \Sigma, A, T, E \rangle$ with the set of axioms $A = \{([\$]_0, [\$]_0, [\$]_0)\}$ and the set of ending units $E = \{([\$]_6, [\$]_6, [\$]_6)\}$. Let $x, y \in \{a, b\}$. The first assembling units generate the first strand up to $[\$]_4$:

1. $([\$]_0 x_1, [\$]_0, [\$]_0)$
2. $(x_1 y_1, [\$]_0, [\$]_0)$
3. $(x_1[\$]_1, [\$]_0, [\$]_0)$
4. $([\$]_1 x_2, [\$]_0, [\$]_0)$
5. $(x_2 y_2, [\$]_0, [\$]_0)$

6. $(x_2[\$]_2, [\$]_0, [\$]_0)$
7. $([\$]_2x_3, [\$]_0, [\$]_0)$
8. $(x_3y_3, [\$]_0, [\$]_0)$
9. $(x_3[\$]_3, [\$]_0, [\$]_0)$
10. $([\$]_3x_4, [\$]_0, [\$]_0)$
11. $(x_4y_4, [\$]_0, [\$]_0)$
12. $(x_4[\$]_4, [\$]_0, [\$]_0)$.

The units (1)–(12) are used to derive a multi-strand of the form

$([\$]_0w_1[\$]_1w_2[\$]_2w_3[\$]_3w_4[\$]_4, [\$]_0, [\$]_0)$.

The following units are used to generate the subwords w_5 and w_6 of the first strand in parallel to the subwords w_2 and w_1 (after completing the first subword of the third strand with the units below).

13. $([\$]_4x_5, [\$]_0, [\$]_1x_2)$
14. $(x_5y_5, [\$]_0, x_2y_2)$
15. $(x_5[\$]_5, [\$]_0, x_2[\$]_2)$
16. $([\$]_5x_6, [\$]_0x_1, [\$]_2)$
17. $(x_6y_6, x_1y_1, [\$]_2)$
18. $(x_6[\$]_6, x_1[\$]_1, [\$]_2)$

Units (13)–(15) are used to complete and copy w_2 on the third strand to w_5 on the first strand. Similarly, units (16)–(18) are used to complete and copy w_1 on the second strand to w_6 on the first strand. After completing the first subword of the third strand with the units below and applying units (13)–(18) a multi-strand of the form

$([\$]_0w_1[\$]_1w_2[\$]_2w_3[\$]_3w_4[\$]_4w_5[\$]_5w_6[\$]_6, [\$]_0w_1[\$]_1, [\$]_0w_1[\$]_1w_2[\$]_2)$

is obtained. After this phase, the sets $\omega_i, 1 \leq i \leq 3$, are $\omega_1 = \{5,6\}$, $\omega_2 = \{1\}$ and $\omega_3 = \{2\}$.

With the next units subword w_4 on the second strand is completed and copied on the third strand to w_3.

19. $([\$]_6, [\$]_3x_4, [\$]_2x_3)$
20. $([\$]_6, x_4y_4, x_3y_3)$
21. $([\$]_6, x_4[\$]_4, x_3[\$]_3)$.

Afterwards the sets ω are $\omega_1 = \{5,6\}$, $\omega_2 = \{1,4\}$ and $\omega_3 = \{2,3\}$.

The units that complete the strands remain to be defined. As described in the construction above no subword of the first strand has to be completed. We define

22. $(z, [\$]_{i-1}x_i, z')$
23. (z, x_iy_i, z')
24. $(z, x_i[\$]_i, z')$

for $i \in \{2,3,5,6\}$ and $z, z' \in \Sigma$, since $\omega_2 = \{1,4\}$. Furthermore, define

25. $(z, z', [\$]_{i-1}x_i)$
26. (z, z', x_iy_i)
27. $(z, z', x_i[\$]_i)$

for $i \in \{1,4,5,6\}$ and $z, z' \in \Sigma$, since $\omega_3 = \{2,3\}$. This completes the description of the units of S. ■

The last theorem established an infinite, dense, and tight strand hierarchy of SAS. The hierarchy is based partially on the simulations by one-way k-head finite automata. Further relations with language families have been given in Corollary 34. Still open are the relations of the families of languages generated by SAS-k with the families of languages accepted by one-way k-head finite automata (is the inclusion proper?) and with the regular languages. The next results show that there is in fact a very simple, that is, unary regular language that cannot be generated by any SAS-k. For the proof we utilize the observation made in the proof of Lemma 14 that the ordering of applied units in the generation of unary words does not matter. Any permutations of the units in T yield the same word. This observation is now formalized in the next lemma.

Lemma 37 *Let $k \geq 1$ be an integer and $S = \langle \Sigma, A, T, E \rangle$ be a SAS-k. If some unary word x^n, for $x \in \Sigma$, is generated by the system S by using an axiom $u_a \in A$ and applying successively units $u_1, u_2, \ldots, u_m$ and an ending unit $u_e \in E$, then the same word x^n is generated by S by using u_a and applying successively units $u_{i_1}, u_{i_2}, \ldots, u_{i_m}$ and ending unit u_e, where $i_1, i_2, \ldots, i_m$ is an arbitrary permutation of $1, 2, \ldots, m$.*

Theorem 38 *For any $k \geq 1$, the regular language $L_{\text{un}} = \{a\} \cup \{ a^{2n} \mid n \geq 2 \}$ is not generated by any SAS-k.*

Proof In contrast to the assertion assume that language L_{un} is generated by some SAS-k $S = \langle \Sigma, A, T, E \rangle$. Let $T = \{u_1, u_2, \ldots, u_n\}$.

Lemma 37 reveals that any successful generation of S can be represented by its axiom u_a, its ending unit u_e and the multiplicities of applications of units from T, that is, by the n numbers $i_1, i_2, \ldots, i_n$, where $i_j \geq 0$ gives the number of applications of unit u_j, $1 \leq j \leq n$.

Since L is infinite, there is at least one pair of axiom u_a and ending unit u_e such that there are infinitely many different words generated by S by using u_a and u_e. Let $L_{u_a,u_e} \subseteq L(S)$ be the language of these words.

Since L_{u_a,u_e} is infinite, there exist two different multiplicities of applications of units $i_1, i_2, \ldots, i_n$ and $j_1, j_2, \ldots, j_n$ that represent successful generations of two different words $v, w \in L_{u_a,u_e}$, where $4 \leq |v| < |w|$, $i_\ell \leq j_\ell$, for all $1 \leq \ell \leq n$. We conclude that the multiplicities $j_1 - i_1, j_2 - i_2, \ldots, j_n - i_n$ represent a successful generation as well. So applying the units from T as often as given by $j_1 - i_1, j_2 - i_2, \ldots, j_n - i_n$ extends any of the k strands by the same number of symbols. Since $|v|$ and $|w|$ have to be even, $|w| - |v|$ is even. For the extension by an even number of

symbols, an odd number of symbols is needed since one symbol is lost by gluing the strands together.

Since $a \in L$, there is an axiom as well as an ending unit of the form $(a, a, \ldots, a)$ in S. Now we consider the word w' whose generation is described by this axiom and ending unit and the multiplicities $j_1 - i_1, j_2 - i_2, \ldots, j_n - i_n$. The generation is successful and $|w'| > 2$ is odd, since the sequence appended is of odd length as pointed out above, a contradiction. □

It has been shown that k-stranded SAS are able to generate non-context-free languages. On the other hand, the unary regular language L_{un} cannot be generated by any k-stranded SAS.

Thus, it can be concluded that the family of languages generated by k-stranded SAS is incomparable with the context-free languages as well as the regular languages.

Corollary 39 *Let $k \geq 1$ be an integer. (i) The family of languages generated by SAS-k is incomparable with the families of languages accepted by one-way k'-head finite automata if $k' < k$. (ii) The family of languages generated by SAS-k is strictly included in the families of languages accepted by one-way k'-head finite automata if $k' \geq k$. (iii) The family of languages generated by SAS-k is incomparable with the (unary) regular languages if $k \geq 2$.*

The results of this section concerning the generative power are summarized in Figure 4.

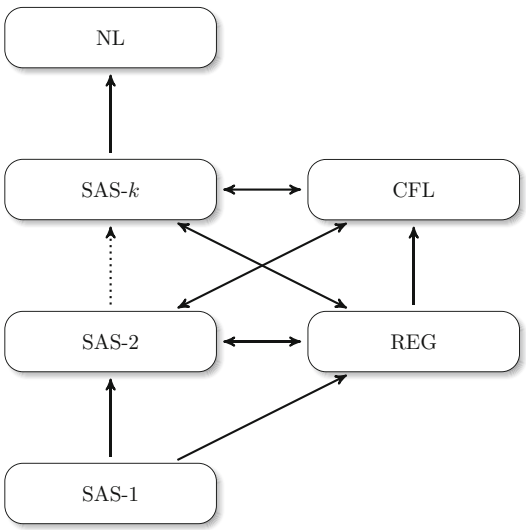

Fig. 4 Hierarchy of languages families dependent on the number of strands. A single arrow means strict inclusion, a double arrow means incomparability, and the dotted arrow is for an infinite hierarchy depending on the number of strands

5 Closure properties

In this section we examine the closure properties of the families of languages generated by different restricted variants of string assembling systems and the extension to k-stranded string assembling systems. It turns out that the language families generated by the restricted variants are not closed under the operations studied except for reversal. In many cases the witness languages are similar for each type of string assembling system. There are some adaptions that need to be done, especially for the restricted types. The reason is that either the restricted systems are not able to generate the basic language and thus some auxiliary symbols need to be added. Or the systems generate the languages in different and more complicated ways. The extended models sometimes require extensions of these basic languages.

In particular, we consider the witness languages

$$
\begin{aligned}
L_{\text{cup}} &= \{\, a^n b^n \mid n \geq 1 \,\} \cup \{\, a^n \mid n \geq 1 \,\},\\
L_{\text{un}} &= \{a\} \cup \{\, a^{2n} \mid n \geq 2 \,\},\\
L_{\text{com}} &= \overline{\{\, a^n b^n \mid n \geq 1 \,\}},\\
L_{\text{int}} &= \{\, [\$]_0 w_1 [\$]_1 w_2 [\$]_2 w_2 [\$]_3 w_1 [\$]_4 \mid w_1, w_2 \in \{a, b\}^+ \,\},\\
L_{\ell,\text{cat}} &= \{\, a^{\ell \cdot n} b^{\ell \cdot n} a^m \mid m, n \geq 1 \,\},\\
L_{\text{square}} &= \{\, a^{n^2+n+1} \mid n \geq 2 \,\}, \text{ and}\\
L_{\text{exp}} &= \{\, a a^2 a^4 \cdots a^{2^i} \mid i \geq 1 \,\}.
\end{aligned}
$$

The following subsections basically present the main ideas for showing the closure properties of the restricted string assembling systems. The last subsection examines the closure properties of the extended variant. The results are summarized in Table 1 on page 16.

Table 1 Summary of closure properties of families of languages generated by string assembling systems. The abbreviation lp. h stands for length-preserving homomorphisms. The properties of k-length-restricted SAS are for all $k \geq 2$ except for the operation inverse λ-free homomorphism for which $k \geq 3$ is assumed

	$\cup$	$\cap$	$\cap_{REG}$	$\bullet$	lp. h	h_λ^{-1}	*REV*
free SAS	−	−	−	−	−	−	−	+
one-set SAS	−	−	−	−	−	−	−	+
k-length restricted SAS	−	−	−	−	−	−	−	+
1-stranded SAS	−	−	−	−	−	−	−	+
k-stranded SAS	−	−	−	−	−	−	−	+
SAS	−	−	−	−	−	−	−	+

5.1 Free string assembling systems

In Theorem 38 it has been shown that the family of languages generated by SAS is not closed under union. The witness language is $L_{\text{un}} = \{a\} \cup \{ a^{2n} \mid n \geq 2 \}$. The following example shows that the two languages joined are generated by free SAS, although the construction is different to common SAS.

Example 40 The language $L = \{ a^{2n} \mid n \geq 2 \}$ is generated by a free string assembling system with the units

1. $(aaa, aaa) \in A$
2. $(aa, aa) \in T$
3. $(a, a) \in E$.

Clearly, the language $L' = \{a\}$ is generated by the free string assembling system with the units

1. $(a, a) \in A$
2. $(\lambda, \lambda) \in T$
3. $(\lambda, \lambda) \in E$. ■

In the unary case the family of languages generated by free SAS and SAS are similar. Thus, the following theorem can be concluded.

Theorem 41 *The family of languages generated by free* SAS *is not closed under union.*

Proof Example 40 shows that the languages $L = \{ a^{2n} \mid n \geq 2 \}$ and $L' = \{a\}$ are generated by free SAS. In Theorem 14 it has been shown that the family of languages generated by SAS and the family of languages generated by free SAS coincide in the unary case. The union $L \cup L'$ is equal to L_{un}. Since L_{un} can not be generated by any SAS, it can be concluded that the union $L \cup L'$ cannot be generated by any free SAS. □

The construction of the next example is a generalization of the construction given in Example 11. To this end, let $h : \{ a_i, b_i \mid 1 \leq i \leq 4 \}^* \to \{a, b\}^*$ be the homomorphism with $h(a_i) = a$ and $h(b_i) = b$, for $1 \leq i \leq 4$.

Example 42 The non-context-free language

$$L = \{ [\$]_0 w_1 [\$]_1 w_2 [\$]_2 w_3 [\$]_3 w_4 [\$]_4 \mid w_i \in \{a_i, b_i\}^+, 1 \leq i \leq 4, |w_i| \geq 2, h(w_1) = h(w_4) \}$$

is generated by a free string assembling S system with the following assembling units. The main problem is that S is free and thus symbols with different index could be mixed up in one of the words w_i. To avoid this, we have to use units that overlap on the two strands during the assembling process. Let $x_i, y_i, z_i \in \{a_i, b_i\}$, for $1 \leq i \leq 4$.

1. $([\$]_0 x_1, \lambda) \in A$
2. $(x_1 y_1, \lambda) \in T$
3. $([\$]_1 x_2, \lambda) \in T$
4. $(x_1 [\$]_1 x_2, \lambda) \in T$
5. $(x_2 y_2, \lambda) \in T$
6. $([\$]_2 x_3, \lambda) \in T$
7. $(x_2 [\$]_2 x_3, \lambda) \in T$
8. $(x_3 y_3, \lambda) \in T$
9. $([\$]_3, [\$]_0) \in T$
10. $(x_3 [\$]_3, [\$]_0) \in T$

The idea of the construction is that first the upper strand up to the first symbol of the subword w_4 is generated. The axiom (1) and the units (2) to (4) generate the first subword up to the first symbol of the subword w_2 leaving the lower strand empty. Next, the units (5) to (7) generate the second subword up to the first symbol of the subword w_3 leaving the lower strand empty. Similarly, the units (8) to (10) generate the third subword up to the $[\$]_3$ where in the last step the lower strand is assembled with the leading $[\$]_0$.

So far it is clearly still possible that incorrectly formatted strands are derived. However, after each symbol $[\$]_i$, $0 \leq i \leq 2$, there appears a symbol with correct index and the upper strand starts correctly with $[\$]_0$.

Next the units are used that complete the first subword of the lower strand and copy it to the fourth subword of the upper strand. In order to avoid the mixing of indices we proceed as follows. Since after applying the axiom there is one symbol of w_1 available and extensions without $[\$]_i$ are only possible by units having two symbols on the upper strand (units (2), (5), or (8)), these extensions appear at even positions in the word w_1. On the other hand, the only units that generate the leading $[\$]_0$ on the lower strand are units (9) and (10). So, these extensions appear at odd positions in the word w_1. By this different arity and the form of the units it is ensured that only upper strands are completed where all symbols in the first subword have the correct index 1. Moreover, the first subword is correctly copied to the last subword. The corresponding units are as follows.

11. $(x_4 y_4, x_1 y_1) \in T$ with $h(x_4 y_4) = h(x_1 y_1)$
12. $(x_4 [\$]_4, x_1 [\$]_1) \in T$ with $h(x_4) = h(x_1)$
13. $(x_4 y_4 [\$]_4, x_1 y_1 [\$]_1) \in T$ with $h(x_4 y_4) = h(x_1 y_1)$

Finally, it is ensured that also w_2 and w_3 are only completed if all of their symbols have the correct index (units (14) to (17)). We know already that w_4 is a correct copy of w_1. By the units (18) and (19) it is furthermore ensured that after the symbol $[\$]_3$ only symbols with index 4 can appear. The ending units are the only ones that assemble $[\$]_4$ on the lower strand. So, altogether we conclude that S generates the language as claimed.

14. $(\lambda, x_2 y_2) \in T$

15. $(\lambda, x_2 y_2 [\$]_2) \in T$
16. $(\lambda, x_2 y_2 z_2 [\$]_2) \in T$
17. $(\lambda, x_3 y_3) \in T$
18. $(\lambda, x_3 y_3 [\$]_3 x_4) \in T$
19. $(\lambda, x_3 y_3 z_3 [\$]_3 x_4) \in T$
20. $(\lambda, x_4 y_4) \in T$
21. $(\lambda, x_4 [\$]_4) \in E$
22. $(\lambda, x_4 y_4 [\$]_4) \in E$

■

The idea of the construction of the previous example can similarly be applied to define a free SAS generating the language

$$L' = \{\, [\$]_0 w_1 [\$]_1 w_2 [\$]_2 w_3 [\$]_3 w_4 [\$]_3 \mid w_i \in \{a_i, b_i\}^+, 1 \le i \le 4, |w_i| \ge 2, h(w_2) = h(w_3) \,\}$$

The example above is the main prerequisite to show that the family of languages generated by free string assembling systems is not closed under intersection.

Theorem 43 *The family of languages generated by free* SAS *is not closed under intersection.*

Proof Up to the condition that the subwords have to have a length of at least two, the intersection $L \cap L'$ is the language L_n with $n = 2$ that has been used to show a tight strand hierarchy in Section 4. Since the condition on the lengths does not affect the proof of the hierarchy, we derive that L_n cannot be generated by any k-stranded SAS unless $n \le \binom{k}{2}$. So, in order to generate L_2 at least 3 strands are necessary. We conclude that $L \cap L'$ is not generated by any free SAS. □

Before we continue with the remaining Boolean operation we consider the concatenation.

Example 44 The language $L_{\text{quad}} = \{\, a^{4n} b^{4n} \mid n \ge 1 \,\}$ is generated by the free SAS $S = \langle \{a, b\}, A, T, E \rangle$, with the assembling units

1. $(aaaa, aaa) \in A$
2. $(aaaa, aa) \in T$
3. $(bbbb, aa) \in T$
4. $(bbbb, ab) \in T$
5. $(\lambda, bb) \in T$
6. $(\lambda, b) \in E$.

The axiom (1) establishes an upper strand with an even number of symbols and a lower strand with an odd number of symbols. Then unit (2) can be used to extend the strands with a's, where four a's are appended to the upper and two a's to the lower strand. So, the arity of the strands is not changed. Moreover, after having assembled a^{4i} as upper strand, the lower strand is a^{2i+1}. Afterwards the only applicable unit to complete the lower strand up to one a is unit (3), which must be followed by assembling unit (4) once. This yields $(a^{4n} b^{4n}, a^{4n} b)$. Finally, the double strand can be completed by applying units (5) and (6). ■

Theorem 45 *The family of languages generated by free* SAS *is not closed under concatenation.*

Proof In Kutrib and Wendlandt (2012) it has been shown that the language $\{\, a^n b^n a^m \mid m, n \ge 1 \,\}$ cannot be generated by any SAS. A straightforward modification of the proof shows that, for all $\ell \ge 1$, the language $L_{\ell,\text{cat}} = \{\, a^{\ell \cdot n} b^{\ell \cdot n} a^m \mid m, n \ge 1 \,\}$ cannot be generated by any free SAS either.

Example 44 shows that L_{quad} is generated by a free SAS. Clearly, the simple free SAS with the units

1. $A = \{(\lambda, \lambda)\}$
2. $T = \{(a, a)\}$
3. $E = \{(a, a)\}$

generates $L = \{\, a^m \mid m \ge 1 \,\}$. However the concatenation $L_{\text{quad}} L$ coincides with $L_{4,\text{cat}}$ which shows the theorem. □

Theorem 46 *The family of languages generated by free* SAS *is not closed under complementation.*

Proof Example 44 shows that L_{quad} is generated by a free SAS. In Kutrib and Wendlandt (2012) it has been shown that the complement of the language $\{\, a^n b^n \mid n \ge 1 \,\}$ cannot be generated by any SAS. The proof can easily be adapted to show that the complement of L_{quad} cannot be generated by any free SAS either. □

Example 47 We construct a free SAS $S = \langle \{a, b, [\$], [\$]_1, [\$]_2\}, A, T, E \rangle$ that generates the language

$$L = \{\, [\$] a^2 [\$] a^4 [\$] \cdots [\$] a^{2^{i+2}} [\$]_1 b^{2^{i+2}} [\$]_2 \mid i \ge 1 \,\}.$$

In particular the units of S are defined as

1. $([\$] aa [\$], [\$] a) \in A$
2. $(aaaa, aa) \in T$
3. $(aaaa [\$], a [\$] a) \in T$
4. $(aaaa [\$]_1, a [\$] a) \in T$
5. $(bb, aa) \in T$
6. $(bb [\$]_2, a [\$]_1 b) \in T$
7. $(\lambda, bb) \in T$
8. $(\lambda, b [\$]_2) \in E$.

After starting with the single axiom (1), the only applicable units are (3) and (4). While unit (4) necessarily terminates the generation of the prefix up to $[\$]_1$, unit (3) together with unit (2) are used to increase the i. So, after i applications of unit (3) (with the only possible applications of unit (2) in between) we have the double strand

$$([\$] a^2 [\$] a^4 [\$] \cdots [\$] a^{2^{i+1}} [\$], [\$] a^2 [\$] a^4 [\$] \cdots [\$] a^{2^i} [\$] a)$$

. The remaining derivation is obvious by the construction of the remaining units. □

Theorem 48 *The family of languages generated by free* SAS *is not closed under length-preserving homomorphisms.*

Proof Consider the witness language

$$L = \{\, [\$]a^2[\$]a^4[\$]\cdots[\$]a^{2^{i+2}}[\$]_1 b^{2^{i+2}}[\$]_2 \mid i \geq 1 \,\}$$

of Example 47. Let the homomorphism $h : \{a, b, [\$], [\$]_1, [\$]_2\}^* \to \{a\}^*$ be defined by $h(a) = h(b) = h([\$]) = h([\$]_1) = h([\$]_2) = a$. It leads to the non-semilinear unary language $h(L) = \{\, a^{3\cdot 2^{i+2}+i+2} \mid i \geq 0 \,\}$.

By Lemma 17, the family of languages generated by free string assembling systems is a proper subset of the family of languages generated by nondeterministic one-way multi-head finite automata. Since one-way multi-head finite automata are not able to accept non-semilinear unary languages, it can be concluded that the family of languages generated by free SAS is not closed under length-preserving homomorphisms. □

Example 49 We construct a free SAS $S = \langle \{a, b, c\}, A, T, E \rangle$ that generates the language $L = \{\, (ac)^{2n+1}b^{2n+1} \mid n \geq 0 \,\} \cup \{\, (ac)^n \mid n \geq 1 \,\}$ with the units

1. $(ac, a) \in A$
2. $(ac, \lambda) \in A$
3. $(acac, ca) \in T$
4. $(ac, ac) \in T$
5. $(bb, ca) \in T$
6. $(b, cb) \in T$
7. $(\lambda, bb) \in T$
8. $(\lambda, bb) \in E$
9. $(b, cb) \in E$
10. $(\lambda, ac) \in E$.

Again, the main point of the construction is that the lower strand starts with an odd length while the upper strand starts with an even length (unit (1)), if a word of the form $(ac)^{2n+1}b^{2n+1}$ is generated. Unit (3) extends the number of ac's in the upper strand by two and extends the lower strand by two symbols. Thus, the number of ac's in the upper strand is always odd and we derive strands of the form $((ac)^{2i+1}, (ac)^i a)$. Then unit (5) has to be applied that generates for any remaining ca in the upper strand a bb in the upper strand. That is, we obtain strands of the form $((ac)^{2i+1}(bb)^i, (ac)^{2i}a)$. In order to obtain a valid derivation now unit (6) has to be applied yielding a strand $((ac)^{2i+1}b^{2i+1}, (ac)^{2i+1}b)$. The remaining derivation is obvious be the construction of the remaining units. Similarly, when words of the form $(ac)^i$ are generated. ■

Theorem 50 *The family of languages generated by free* SAS *is not closed under inverse λ-free homomorphisms.*

Proof We consider the witness language

$$L = \{\, (ac)^{2n+1}b^{2n+1} \mid n \geq 0 \,\} \cup \{\, (ac)^n \mid n \geq 1 \,\}$$

of Example 49 together with the λ-free homomorphism $h : \{a, b\}^* \to \{a, b, c\}^*$, where $h(a) = ac$ and $h(b) = b$. Then, language $h^{-1}(L)$ is

$$\{\, a^{2n+1}b^{2n+1} \mid n \geq 0 \,\} \cup \{\, a^n \mid n \geq 1 \,\}$$

which is not generated by any free SAS by applying Lemma 16. □

As for common string assembling systems the only positive closure is under reversal by mirroring all units and interchanging the axioms and the ending units.

Theorem 51 *The family of languages generated by free* SAS *is closed under reversal.*

5.2 One-set string assembling systems

Here we turn to examine the closure properties of the family of languages generated by one-set string assembling systems. Recall, that the basic restriction of this type is that the sets of axioms, assembling units, and ending units coincide.

The following regular languages $L = \{a\}$, $L' = \{\, a^n \mid n \geq 1 \,\}$ and the context-free languages $L'' = \{\, a^{2n}b^{2n} \mid n \geq 1 \,\}$ and $L''' = \{\, a^n b^n \mid n \geq 1 \,\}$ are generated by one-set string assembling systems in the same way as by common SAS, where the corresponding constructions use axioms and ending units consisting of single symbols. Thus in a derivation they can be neglected. These languages are applied as witness languages to show that the family of languages generated by common SAS is not closed under union, complementation, and concatenation (Kutrib and Wendlandt 2012). From these proofs the following result can straightforwardly be concluded.

Corollary 52 *The family of languages generated by one-set* SAS *is not closed under union, complementation, and concatenation.*

Concerning the intersection, we may come back to the language of Example 42. So, again, let $h : \{\, a_i, b_i \mid 1 \leq i \leq 4 \,\}^* \to \{a, b\}^*$ be the homomorphism $h(a_i) = a$ and $h(b_i) = b$, for $1 \leq i \leq 4$.

Example 53 The following language L is the language from Example 42 with the modification that the lengths of the words w_i also may be one. So, language

$$L = \{\, [\$]_0 w_1 [\$]_1 w_2 [\$]_2 w_3 [\$]_3 w_4 [\$]_4 \mid w_i \in \{a_i, b_i\}^+, 1 \le i \le 4 \text{ with } h(w_1) = h(w_4) \,\}$$

is generated by the one-set SAS S with the following units. Let $x_i, y_i \in \{a_i, b_i\}$, for $1 \le i \le 4$.

1. $([\$]_0 x_1, [\$]_0) \in T$
2. $(x_1 y_1, [\$]_0) \in T$
3. $(x_1 [\$]_1 y_2, [\$]_0) \in T$
4. $(x_2 y_2, [\$]_0) \in T$
5. $(x_2 [\$]_2 y_3, [\$]_0) \in T$
6. $(x_3 y_3, [\$]_1) \in T$
7. $(x_3 [\$]_3 y_4, [\$]_0 x_1) \in T$ with $h(y_4) = h(x_1)$
8. $(x_4 y_4, x_1 y_1) \in T$ with $h(x_4 y_4) = h(x_1 y_1)$
9. $(x_4 [\$]_4, x_1 [\$]_1 y_2) \in T$ with $h(x_4) = h(x_1)$
10. $([\$]_4, x_2 y_2) \in T$
11. $([\$]_4, x_2 [\$]_2 x_3) \in T$
12. $([\$]_4, x_3 y_3) \in T$
13. $([\$]_4, x_3 [\$]_3 x_4) \in T$
14. $([\$]_4, x_4 y_4) \in T$
15. $([\$]_4, x_4 [\$]_4) \in T$.

The construction of S ensures that unit (1) is the only unit with which the derivation can start. Moreover, unit (15) is the only unit that can be applied in the last derivation step, since it is the only unit with a $[\$]_4$ in the lower strand. Nothing can be appended in the lower strand afterwards, since there is no unit with a $[\$]_4$ as the first symbol in the lower strand. The copying by unit (8) ensures the relation between w_1 and w_4. The correct format is achieved by the units (2)–(7) and (9). The completion of the lower strand is done by the units (10)–(14). ■

Example 54 From Example 53 we can immediately derive a construction of a one-set SAS that generates the language

$$L' = \{\, [\$]_0 w_1 [\$]_1 w_2 [\$]_3 w_3 [\$]_3 w_4 [\$]_4 \mid w_i \in \{a_i, b_i\}^+, 1 \le i \le 4 \text{ with } h(w_2) = h(w_3) \,\}.$$

■

The intersection of similar languages as in Example 53 and Example 54 have been used to disprove the closure of the family of languages generated by free SAS. In the same way, we obtain the next theorem.

Theorem 55 *The family of languages generated by one-set* SAS *is not closed under intersection.*

Example 56 The language

$$L = \{\, [\$] a [\$] b^3 [\$] a^5 [\$] b^7 [\$] \cdots [\$] x^{2i+1} \# \mid i \ge 1 \,\}$$

with $x = a$ if i is even, and $x = b$ if i is odd, is generated by the one-set SAS $S = \langle \{a, b, [\$], \#\}, T \rangle$ with the assembling units

1. $([\$] a [\$] b, [\$] a) \in T$
2. $(bb, aa) \in T$
3. $(aa, bb) \in T$
4. $(bbb[\$] a, a[\$] b) \in T$
5. $(aaa[\$] b, b[\$] a) \in T$
6. $(bbb\#, a[\$] b) \in T$
7. $(aaa\#, b[\$] a) \in T$
8. $(\#, bb) \in T$
9. $(\#, aa) \in T$
10. $(\#, b\#) \in T$
11. $(\#, a\#) \in T$.

The derivation has to start with unit (1), since it is the only unit where the upper and the lower strand fit together. Afterwards, unit (1) cannot be applied anymore, since it starts with [$] in the upper as well as in the lower strand. Moreover, either unit (10) or (11) is used to the finish the derivation, since they are the only units with a # in the lower strand. Nothing can be appended afterwards, since there are no units beginning with a # in the lower strand. The alternation of a and b and the extension is controlled by the units (4) and (5), and the copying by the units (2) and (3). When the units (6) or (7) are applied, the upper strand cannot be extended anymore and the lower strand is completed by the units (9), (10) and (11). As mentioned before, units (10) and (11) are used to finish the derivation. ■

Theorem 57 *The family of languages generated by one-set* SAS *is not closed under length-preserving homomorphisms.*

Proof Consider the witness language

$$L = \{\, [\$] a [\$] b^3 [\$] a^5 [\$] b^7 [\$] \cdots [\$] x^{2i+1} \# \mid i \ge 1 \,\}$$

of Example 56. Let the homomorphism $h : \{a, b, [\$], \#\}^* \to \{a\}^*$ be defined by $h(a) = h(b) = h([\$]) = h(\#) = a$. It leads to the non-semilinear unary language

$$h(L) = L_{\text{square}} = \{\, a^{n^2+n+1} \mid n \ge 2 \,\}.$$

Since it has been shown in Kutrib and Wendlandt (2012) that string assembling systems are not able to generate non-semilinear unary languages in general, we conclude that the family of languages generated by one-set SAS is not closed under length-preserving homomorphisms. □

Example 58 We construct a one-set SAS $S = \langle \{a, b, c\}, T \rangle$ that generates the language $L = \{\, (ac)^n b^n \mid n \ge 1 \,\} \cup \{\, (ac)^n \mid n \ge 1 \,\}$ with the units

1. $(a, ac) \in T$
2. $(aca, cac) \in T$
3. $(ac, c) \in T$
4. $(ac, a) \in T$

5. $(cac, a) \in T$
6. $(cb, ac) \in T$
7. $(bb, cac) \in T$
8. $(b, cb) \in T$
9. $((b, bb) \in T$.

The units (1)–(3) are used to generate words of the form $(ac)^+$. For this purpose, unit (1) is the only one which can be used in the first derivation step. Afterwards unit (2) is applied to extend the strands arbitrarily. When unit (3) is applied, the derivation stops, since there are no units starting in the lower as well as in the upper strand with a c. It is not possible to apply units of (4)–(9) to a derivation by units of (1)–(3), since the last symbols of the current strands do not match with the units.

Units (4)–(9) are used to generate the words from $\{ (ac)^n b^n \mid n \geq 1 \}$. The idea of the derivation is the same as in the last sections. Here, unit (4) is the only unit that can be used as an axiom. Then the derivation is done as usual. The last symbols of the current strands do not allow that something is derived after the b block. ■

Theorem 59 *The family of languages generated by one-set* SAS *is not closed under inverse λ-free homomorphisms.*

Proof We consider the witness language $L = \{ (ac)^n b^n \mid n \geq 1 \} \cup \{ (ac)^n \mid n \geq 1 \}$ of Example 58 together with the λ-free homomorphism $h : \{a, b\}^* \to \{a, b, c\}^*$, where $h(a) = ac$ and $h(b) = b$. Then, language $h^{-1}(L)$ is

$$L_{\text{cup}} = \{ a^n b^n \mid n \geq 1 \} \cup \{ a^n \mid n \geq 1 \}$$

which is not generated by any SAS. □

The closure under reversal can be shown similar to the previous proofs by mirroring all units. The interchange of the axioms and the ending units is omitted, since they do not exist in one-set SAS.

Theorem 60 *The family of languages generated by one-set* SAS *is closed under reversal.*

5.3 Length-restricted string assembling systems

Since SAS whose units contain single symbols can only generate finite languages with words of length one, it is reasonable to consider a constant greater than one for limiting the lengths of the strands in the units when considering length-restricted SAS.

Since the constructions of the counterexamples for the non-closure of the families of languages generated by k-SAS under union, intersection, complementation, and concatenation in Kutrib and Wendlandt (2012) use only units with strands that have at most length two, the next theorem follows.

Theorem 61 *Given an integer $k \geq 2$, the family of languages generated by k-*SAS *is not closed under union, intersection, complementation, and concatenation.*

We turn to consider the closure under homomorphisms.

Example 62 The language

$$L = \{ [\$]abc[\$]a^3bc[\$]a^5bc[\$] \cdots [\$]a^{2i+1}bc\# \mid i \geq 1 \}$$

is generated by the 2-SAS $S = \langle \{a, b, c, [\$], \#\}, A, T, E \rangle$ with the units:

1. $([\$]a, [\$]) \in A$
2. $(ab, [\$]) \in T$
3. $(bc, [\$]) \in T$
4. $(c[\$], [\$]) \in T$
5. $([\$]a, [\$]a) \in T$
6. $(aa, aa) \in T$
7. $(aa, ab) \in T$
8. $(aa, bc) \in T$
9. $(a, c[\$]) \in T$
10. $(c\#, [\$]) \in T$
11. $(\#, aa) \in T$
12. $(\#, ab) \in T$
13. $(\#, bc) \in T$
14. $(\#, c\#) \in E$.

The only axiom is unit (1). Afterwards the units (2) to (5) have to be applied one after the other, since there is no other option. Then units (6)–(8) first copy the number of a's of the previous block and then generate a new a for the b and a new a for the c in the upper strand. Afterwards by the units (2) and (3) the suffix bc is added and by the units (4) and (5) a cycle that extends the word is completed. In each cycle the number of a's in the newly added factor increases by two in the upper strand. At some point of the derivation unit (10) is applied instead of unit (4), which initializes the completion of the lower strand by the units (11)–(14). ■

Theorem 63 *Given an integer $k \geq 2$, the family of languages generated by k-*SAS *is not closed under length-preserving homomorphisms.*

Proof Proceeding in a similar way as before, we apply the length-preserving homomorphism $h : \{a, b, c, [\$], \#\}^* \to \{a\}^*$ with $h(a) = h(b) = h(c) = h([\$]) = h(\#) = a$ to the language L of Example 62. This leads to the non-semilinear unary language $h(L) = \{ a^{n^2+3n+1} \mid n \geq 2 \}$. Since it has been shown that string assembling systems are not able to generate non-semilinear unary languages, the theorem follows. □

Concerning the closure under inverse homomorphisms we can apply the argumentation for one-set SAS in Theorem 59, since every language generated by a one-set SAS

is generated by an SAS and the witness SAS constructed in Example 58 is three-length-restricted. However, this gives the non-closure under inverse homomorphisms only for $k \geq 3$. The problem is open for $k = 2$.

Theorem 64 *Given an integer $k \geq 3$, the family of languages generated by k-SAS is not closed under inverse λ-free homomorphisms.*

The closure under reversal can be shown in a similar way to the previous proofs by mirroring all units and interchanging the axioms and the ending units.

Theorem 65 *Given an integer $k \geq 2$, the family of languages generated by k-SAS is closed under reversal.*

5.4 Single-stranded string assembling systems

String assembling systems with a single strand generate a subregular family of languages. Many of the previous theorems about closure properties are based on relations with non-regular languages. Thus, in these cases we have to consider new approaches here.

The regular languages $L = \{a\}$, $L' = \{ a^{2n} \mid n \geq 1 \}$, and $L'' = \{ a^n \mid n \geq 1 \}$ are generated by single-stranded string assembling systems. These languages are applied as witness languages to show that the family of languages generated by common SAS is not closed under union and intersection with regular languages Kutrib and Wendlandt (2012). So, we obtain the next corollary.

Corollary 66 *The family of languages generated by* 1*-stranded* SAS *is neither closed under union nor under intersection with regular languages.*

The same construction as in the previous sections can be applied to show that the family of languages generated by 1-stranded SAS is closed under reversal. Simply all units are reversed and the axioms and the ending units are interchanged.

Theorem 67 *The family of languages generated by* 1*-stranded* SAS *is closed under reversal.*

We turn to the closures that do not follow immediately.

Theorem 68 *The family of languages generated by* 1*-stranded* SAS *is not closed under concatenation.*

Proof Consider the language $L = \{ a^n b^m \mid m, n \geq 1 \}$ that is generated by the 1-stranded SAS $S = \langle \Sigma, A, T, E \rangle$ with the single axiom $A = \{(a)\}$, the single ending unit $E = \{(b)\}$, and the set of assembling units $T = \{(aa), (ab), (bb)\}$. Further consider the language $L' = \{ b^n a^m \mid m, n \geq 1 \}$ that is generated in a similar way. Assume that a 1-stranded SAS $S' = \langle \Sigma, A', T', E' \rangle$ generates the concatenation LL'. By Lemma 8, S' must have assembling units of the form $(a^i b^j) \in T'$ with integers $i, j \geq 1$ as well as $(b^{i'} a^{j'}) \in T'$ with integers $i', j' \geq 1$. But since there must be an axiom a^c and an ending unit $a^{c'}$, $c, c' \geq 1$ in S', also the word $a^{i+c-1} b^j b^{i'-1} a^{j'} a^{i-1} b^j b^{i'-1} a^{j'+c'-1}$ can be generated, a contradiction. □

Theorem 69 *The family of languages generated by* 1*-stranded* SAS *is not closed under complementation.*

Proof As shown above the language $L = \{ a^n b^m \mid m, n \geq 1 \}$ is generated by a 1-stranded SAS. Assume there is a 1-stranded SAS $S = \langle \Sigma, A, T, E \rangle$ generating the complement $\overline{L}$ of L. By Lemma 8, S, must have at least one assembling unit of the form $(a^i b^j) \in T$, $i, j \geq 1$, since there are infinitely many words of the form $aba^n b^m, m, n \geq 1$ in $\overline{L}$. The subsets $\{ a^n \mid n \geq 1 \} \subset \overline{L}$ and $\{ b^n \mid n \geq 1 \} \subset \overline{L}$ induce the existence of an axiom $(a^n) \in A$, $n \geq 1$, and an ending unit $(b^m) \in E$, $m \geq 1$.

However, then it is also possible to generate the word $a^{i+n-1} b^{j+m-1}$ by S which belongs to L, a contradiction. □

The section ends with the examination of the closure under homomorphisms and inverse homomorphisms.

Theorem 70 *The family of languages generated by* 1*-stranded* SAS *is not closed under length-preserving homomorphisms.*

Proof Consider the language $L = \{ a^{2n} \mid n \geq 1 \} \cup \{b\}$ that is generated by a 1-stranded SAS $S = \langle \Sigma, A, T, E \rangle$ with the axioms $A = \{(aa), (b)\}$, the ending units $E = \{(a), (b)\}$, and the single assembling unit $T = \{(aaa)\}$. The homomorphism $h(a) = h(b) = a$ leads to the language $h(L) = \{ a^{2n} \mid n \geq 1 \} \cup \{a\}$ for which it has been shown that it cannot be generated even by a common SAS. □

Theorem 71 *The family of languages generated by* 1*-stranded* SAS *is not closed under inverse λ-free homomorphisms.*

Proof Consider the language $L = L' \cup L''$ where

$$L' = \{ (cd)^n (ef)^m \mid m, n \geq 1 \} \text{ and } L'' = \{ (ef)^n (cd)^m \mid m, n \geq 1 \}.$$

It is generated by the 1-stranded SAS $S = \langle \Sigma, A, T, E \rangle$ with axioms $A = \{(c), (e)\}$, ending units $E = \{(efcd), (dcd), (cdef), (fef)\}$, and assembling units $T = T' \cup T''$ where $T' = \{(cdc), (cdef), (fef)\}$ and $T'' = \{(efe), (efcd), (dcd)\}$.

The derivation starts either with a c or with an e, depending on whether a word of L' or a word of L'' should be derived. The units of T' are used in the derivation of a word of L' and the units of T'' for a word of L''.

Consider the λ-free homomorphism h defined by $h(a) = cd$ and $h(b) = ef$. The language $h^{-1}(L)$ is

$h^{-1}(L) = \{ a^n b^m \mid m,n \geq 1 \} \cup \{ b^n a^m \mid m,n \geq 1 \}$.

By a similar argumentation as in the proof of Theorem 68 it is shown that $h^{-1}(L)$ cannot be generated by any 1-stranded SAS. □

The results for the closure properties of the variants of SAS are summarized in Table 1.

5.5 String assembling systems with multiple strands

Finally, we consider the closure properties of the families of languages generated by multi-stranded SAS. It will turn out that even the generalization to multiple strands does not lead to different properties with respect to the operations studied.

We are going to apply a certain pumping lemma that has been shown in Kutrib and Wendlandt (2012) and has already be used for free SAS (Lemma 16). In fact, almost literally its proof applies to multi-stranded SAS.

Lemma 72 *Given an integer $k \geq 2$, let $L \subseteq \Sigma^*$ be a language generated by a k-stranded* SAS. *If $|a^+ \cap L| = \infty$, for some symbol $a \in \Sigma$, then there exist constants $p, q \geq 1$ such that $a^q v \in L$, $v \in \Sigma^*$, implies $a^{q+p} v \in L$.*

Now, by Lemma 72 we directly can adapt some proofs of non-closure properties from the two-stranded to the k-stranded case.

Theorem 73 *Let $k \geq 2$ be an integer. The family of languages generated by k-stranded string assembling systems is not closed under union, intersection with regular languages, complementation, length-preserving homomorphisms and inverse λ-free homomorphisms.*

Proof Since in Theorem 38 it has been shown that $L_{\text{un}} = \{a\} \cup \{ a^{2n} \mid n \geq 1 \}$ cannot be generated by any k-stranded SAS, but clearly the two joined languages can, we obtain that the family generated by k-stranded SAS is not closed under union.

The unary language $L = \{ a^n \mid n \geq 1 \}$ can be generated by some k-stranded SAS. Since $L \cap L_{\text{un}} = L_{\text{un}}$, the non-closure under intersection with regular languages follows.

Assume that the language $L_{\text{com}} = \overline{\{ a^n b^n \mid n \geq 1 \}}$ is generated by some k-stranded SAS. Since all words a^i, for $i \geq 1$, belong to L_{com}, Lemma 72 can be applied with constants $p, q \geq 1$. So, $a^p b^{p+q} \in L_{\text{com}}$ implies $a^{p+q} b^{p+q} \in L_{\text{com}}$, a contradiction. Since $\{ a^n b^n \mid n \geq 1 \}$ is generated by some k-stranded SAS, the non-closure under complementation follows.

Similarly, the non-closure under inverse λ-free homomorphisms can be shown by the witness language $\{ (ac)^n b^n \mid n \geq 1 \} \cup \{ (ac)^m \mid m \geq 1 \}$, the homomorphism $h : \{a,b\}^* \to \{a,b,c\}^*$ with $h(a) = ac$, $h(b) = b$, and application of Lemma 72.

The non-semilinear language $L_s = \{ [\$]a^1[\$]a^3[\$] \cdots [\$]a^{2i+1}\# \mid i \geq 1 \}$ is generated by some common SAS Kutrib and Wendlandt (2012), thus also by some k-stranded SAS, for all $k \geq 2$. Let h be the length-preserving homomorphism $h : \{a, [\$], \#\}^* \to \{a\}^*$ with $h(a) = h([\$]) = h(\#) = a$. Then $h(L_s)$ is a non-semilinear unary language. Since a k-stranded SAS can be simulated by a k-head finite automaton which is not able to accept non-semilinear unary languages, non-closure under length-preserving homomorphisms follows.

□

The closure under reversal can be achieved by the usual construction.

Theorem 74 *Let $k \geq 2$ be an integer. The family of languages generated by k-stranded* SAS *is closed under reversal.*

The remaining closure properties are intersection and concatenation.

Theorem 75 *Let $k \geq 2$ be an integer. The family of languages generated by k-stranded* SAS *is not closed under intersection.*

Proof For $k = 2$, the non-closure has been shown in Kutrib and Wendlandt (2012).

Let $h : \{ a_i, b_i \mid 1 \leq i \leq 4 \}^* \to \{a,b\}^*$ be the homomorphism with $h(a_i) = a$ and $h(b_i) = b$, for $1 \leq i \leq 4$. In Theorem 35 it has been shown that the language

$$L_n = \{ [\$]_0 w_1 [\$]_1 w_2 [\$]_2 \cdots [\$]_{2n-1} w_{2n} [\$]_{2n} \mid w_i \in \{a_i, b_i\}^*, h(w_i) = h(w_{2n+1-i}), 1 \leq i \leq 2n \}.$$

with $n = \binom{k}{2}$ can be generated by some k-stranded SAS, but not by any $(k-1)$-stranded SAS. The construction of the proof of Theorem 35 can be adapted so that the following two variants can be generated as well.

$$L'_n = \{ z[\$]_n w_{n+1} [\$]_{n+1} w_{n+2} [\$]_{n+2} \cdots [\$]_{3n-1} w_{3n} [\$]_{3n} z' \mid w_{n+i} \in \{a_{n+i}, b_{n+i}\}^*, h(w_{n+i}) = h(w_{3n+1-i}), 1 \leq i \leq 2n, z \in \{[\$]_{j-1}, a_j, b_j\}^*, 1 \leq j \leq n, z' \in \{[\$]_j, a_j, b_j\}^*, 3n+1 \leq j \leq 4n \},$$

and

$$L''_n = \{ [\$]_0 w_1 [\$]_1 w_2 [\$]_2 \cdots w_n [\$]_n z [\$]_{3n} w_{3n+1} [\$]_{3n+1} \cdots w_{4n} [\$]_{4n} \mid |quad w_i \in \{a_i, b_i\}^*, w_{3n+i} \in \{a_{3n+i}, b_{3n+i}\}^*, h(w_i) = h(w_{4n+1-i}), 1 \leq i \leq n, z \in \{[\$]_j, a_j, b_j\}^*, n+1 \leq j \leq 3n \}.$$

For $k \geq 3$, we have $2\binom{k}{2} \geq \binom{k+1}{2}$, and derive that the intersection $L'_n \cap L''_n$, for $n = \binom{k}{2}$, cannot be generated by any k-stranded SAS. □

In Kutrib and Wendlandt (2012) it has been shown that the concatenation of the languages $\{ a^n b^n \mid n \geq 1 \}$ and $\{ a^m \mid m \geq 1 \}$ cannot be generated by any 2-stranded SAS. Next, we show that it can be generated by a 3-stranded SAS.

Example 76 The language $L = \{ a^n b^n a^m \mid m, n \geq 1 \}$ is generated by the SAS-3 $S = \langle \Sigma, A, T, E \rangle$ with the single axiom $A = \{(a, a, a)\}$, the ending units

$$E = \{(aa, aa, a), (aaa, aaa, aa), (a, ba, ba), (a, baa, baa), (ba, aba, ba)\},$$

and the assembling units in T being

1. (aa, a, aa)
2. (ab, a, ab)
3. (bb, aa, b)
4. (b, ab, b)
5. (b, bb, b)
6. (ba, b, b)
7. (a, b, bb)
8. (a, ba, b)
9. (aaa, aaa, b)
10. (a, a, baa)
11. (a, a, aaa).

The derivation starts with the sole axiom. Then the first a-block is generated in the first and the third strand by applying repeatedly unit (1), and unit (2) once. In a valid derivation it is not possible to apply unit (11) in this phase, since otherwise the number of a's in the first and the third strand will never match again. After applying unit (2), the current 3-stranded word is of the form $(a^n b, a, a^n b)$. Now units (3) and (4) become applicable. The repeated application of unit (3) followed by one application of unit (4) yields a 3-stranded word of the form $(a^n b^n, a^n b, a^n b)$. So, there are as many a's in the first block as b's in the second block of the first strand. After applying unit (4), units (5), and (6) from T become applicable. A repeated application of unit (5) completes the b-block of the second strand. After subsequently applying unit (6) once, a 3-stranded word of the form $(a^n b^n a, a^n b^n, a^n b)$ is derived. Note, if unit (5) is applied too often, application of unit (6) and, thus, a successful derivation becomes impossible. Now units (7) and (8) are applicable. The repeated application of unit (7) followed by one application of unit (8) yields a 3-stranded word of the form $(a^n b^n a, a^n b^n a, a^n b^n)$. Next, the third a-block is generated. In order to avoid that it is followed by another b-block, its parity is utilized. So, applying unit (9) repeatedly and unit (10) once gives a 3-stranded word of the form $(a^n b^n a^i, a^n b^n a^i, a^n b^n aa)$, where i is an odd number. Subsequent applications of unit (11) fill the a-block of the third strand but let its length be even. In this way further applications of units (1) or (2) fail. Finally, the derivation is made successful by selecting ending unit (aa, aa, a) or (aaa, aaa, aa). The further ending units are used to derive short words. ■

Theorem 77 *Let $k \geq 2$ be an integer. The family of languages generated by k-stranded* SAS *is not closed under concatenation.*

Proof The language $L = \{ a^n b^n \mid n \geq 1 \} \cup \{a\}$ is generated by the 2-stranded SAS $S = \langle \Sigma, A, T, E \rangle$ with $A = \{(a, a)\}$, $E = \{(a, a), (b, b)\}$, and the assembling units

1. $(aa, a) \in T$
2. $(ab, a) \in T$
3. $(bb, aa) \in T$
4. $(b, ab) \in T$
5. $(b, bb) \in T$.

Clearly, the language $L' = \{ a^m \mid m \geq 1 \}$ is generated by some 2-stranded SAS as well. Since $LL' = \{ a^n b^n a^m \mid m \geq 2, n \geq 1 \} \cup \{ a^m \mid m \geq 2 \}$, Lemma 72 can be applied to derive that LL' cannot be generated by any k-stranded SAS. Thus, the family of languages generated by k-stranded SAS is not closed under concatenation. □

6 Conclusion

The restrictions in the definition of SAS differentiate between axioms, assembling units, and ending units, using double strands, and the length of the units, do all have an impact on the generative capacities of string assembling systems. In particular, every restricted variant but the free SAS results in a subfamily of the family of languages generated by common SAS. For the free SAS and its subfamily of pure SAS, the relation to SAS is an open problem. In fact, it has been shown that there is a language generated by SAS which cannot be generated by any free SAS, but on the other side, currently no simulation of free SAS by SAS is known. The usage of empty axioms, which is not allowed for SAS, seems to be the problem. A first idea is that a free SAS can be simulated by an SAS with one more symbol concatenated at the beginning of each word. This symbol can be somehow seen as an auxiliary symbol to overcome the definition of an empty axiom.

Interestingly, the families of k-length-restricted SAS form a proper hierarchy depending on the maximal length k of the units. The hierarchy is established on unary languages. Clearly, the lowest level, the family of languages generated by 1-length-restricted SAS is a proper subset of the finite languages and thus of the regular languages. Increasing k by one, one gets the 2-length-restricted SAS that lead to a family of languages that includes non-context-free languages as $\{ a^n b^n c^n \mid n \geq 1 \}$.

The last restricted variant is the family of languages generated by single-stranded SAS. It has been shown that they form a subfamily of the regular languages. Even very weak regular languages as $\{ a^m b^n a^k \mid m, n, k \geq 1 \}$ are not included in this family.

Moreover, we investigated an extension of SAS where the system is allowed to have multiple strands. Clearly, the motivation for this extension cannot be seen in the double helix structure of DNA, but nevertheless the relation to one-way multi-head finite automata somehow advises to investigate this generalization. The head hierarchy on one-way multi-head finite automata can be adapted to k-stranded string assembling systems. In particular the family of languages generated by k-stranded SAS is a proper subset of the family of languages generated by $(k+1)$-stranded SAS. A desirable improvement of the results would be to find a separating language where the size of the alphabet is not related to the number of strands.

However, even with multiple strands, string assembling systems cannot generate the unary regular language $L_{\text{un}} = \{a\} \cup \{ a^{2n} \mid n \geq 2 \}$, no matter about the number of strands.

It turns out that all variants of string assembling systems are closed under reversal. On the other hand, no variant of SAS is closed under intersection. A main reason for this seems to be the strong relation to multi-head finite automata and the appropriate head hierarchy which is based on a suitable language for intersection.

The closure or non-closure of all types of SAS studied under iteration remains open. At first glance, an approach could be to combine the ending units and the axioms of a given system to new assembling units. The problem is that it can not be ensured that the strands slip apart. In particular, for an SAS S with an ending unit $u_e = (aaa, aaa)$ and an axiom $u_a = (abb, abb)$ we would construct the unit $u = (aaabb, aaabb)$. Maybe there is some derivation that leads to the double strand $w = (aaabba, a)$. Now, if we apply u on w we get $w' = (aaabbaaabb, aaabb)$. But for w it is not possible to apply u_e, since an ending unit needs to lead to a complete double strand. Thus the application would be unsuccessful.

Acknowledgements We would like to thank the anonymous referees for their careful reading of the manuscript and for giving many valuable comments that helped to improve the presentation of the paper.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Bordihn H, Kutrib M, Malcher A (2012) On the computational capacity of parallel communicating finite automata. Int J Found Comput Sci 23:713–732. https://doi.org/10.1142/S0129054112500062

Enaganti SK, Ibarra OH, Kari L, Kopecki S (2017) On the overlap assembly of strings and languages. Nat Comput 16:175–185. https://doi.org/10.1007/s11047-015-9538-x

Freund R, Păun G, Rozenberg G, Salomaa A (1998) Bidirectional sticker systems. In: Pacific symposium on biocomputing (PSB 1998), World Scientific, Singapore, pp 535–546.

Hartmanis J (1972) On non-determinancy in simple computing devices. Acta Inform 1:336–344

Holzer M, Jakobi S, Kutrib M (2017) The chop of languages. Theor Comput Sci 682:122–137. https://doi.org/10.1016/j.tcs.2017.02.002

Kari L, Păun G, Rozenberg G, Salomaa A, Yu S (1998) DNA computing, sticker systems, and universality. Acta Inform 35:401–420

Kutrib M, Malcher A, Wendlandt M (2016) Set automata. Int J Found Comput Sci 27:187–214. https://doi.org/10.1142/S0129054116400062

Kutrib M, Wendlandt M (2012) String assembling systems. RAIRO Inform Théor 46:593–613

Kutrib M, Wendlandt M (2014) Bidirectional string assembling systems. RAIRO Inform Théor 48:39–59. https://doi.org/10.1051/ita/2013048

Kutrib M, Wendlandt M (2018) Expressive capacity of subregular expressions. RAIRO Inform Théor 52:201–218. https://doi.org/10.1051/ita/2018014

Kutrib M, Wendlandt, M (2018) Parametrizing string assembling systems. In: Câmpeanu C (ed) Implementation and application of automata (CIAA 2018), LNCS vol. 10977. Springer, pp 236–247

Kutrib M, Wendlandt M (2019) Multi-stranded string assembling systems. In: Catania B, Královic R, Nawrocki JR, Pighizzini G (eds) Theory and practice of computer science (SOFSEM 2019), LNCS, vol 11376. Springer, pp 285–297. https://doi.org/10.1007/978-3-030-10801-4_23

Moore GE (1965) Cramming more components onto integrated circuits. Electronics 38:114–117

Papadimitriou CH. (1994) Computational complexity. Addison-Wesley

Păun G, Rozenberg G (1998) Sticker systems. Theor Comput Sci 204:183–203

Păun G, Rozenberg G, Salomaa A (1998) DNA computing: new computing paradigms. Texts in theoretical computer science. Springer

Rosenberg AL (1966) On multi-head finite automata. IBM J Res Dev 10:388–394

Rozenberg G, Salomaa A (1980) The mathematical theory of L systems. Academic Press

Shallit J (2008) The Frobenius problem and its generalizations. In: Ito M, Toyama M (eds) Developments in language theory (DLT 2008), LNCS, vol 5257. Springer, pp 72–83

Yao AC, Rivest RL (1978) $k+1$ heads are better than k. J ACM 25:337–340