More Structural Characterizations of Some Subregular Language Families by Biautomata

Markus Holzer and Sebastian Jakobi
Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
{holzer,sebastian.jakobi}@informatik.uni-giessen.de

We study structural restrictions on biautomata such as, e.g., acyclicity, permutation-freeness, strongly permutation-freeness, and orderability, to mention a few. We compare the obtained language families with those induced by deterministic finite automata with the same property. In some cases, it is shown that there is no difference in characterization between deterministic finite automata and biautomata as for the permutation-freeness, but there are also other cases, where it makes a big difference whether one considers deterministic finite automata or biautomata. This is, for instance, the case when comparing strongly permutation-freeness, which results in the family of definite language for deterministic finite automata, while biautomata induce the family of finite and co-finite languages. The obtained results nicely fall into the known landscape on classical language families.

1 Introduction

The finite automaton is one of the first and most intensely investigated computational model in theoretical computer science, see, e.g., [15]. Its systematic study led to a rich and unified theory of regular subfamilies such as, for example, finite languages (are accepted by acyclic finite automata—here, except for non-accepting sink states, self-loops on states count as cycles), ordered languages (where the transitions of the accepting automata preserve an order on the state set), and star-free languages or non-counting languages (which can be described by regular like expressions using only union, concatenation, and complement or equivalently by permutation-free finite automata), to mention a few. Relations between several subregular language families, such as those mentioned above, are summarized in [6]. In particular, an extensive study of star-free regular languages can be found in [14]. Even nowadays the study of subregular language families from different perspectives such as, for instance, algebra, logic, descriptional, or computational complexity, is a vivid area of research.

Recently, an alternative automaton model to the deterministic finite automaton (DFA), the so called *biautomaton* (DBiA) [12] was introduced. Roughly speaking, a biautomaton consists of a *deterministic* finite control, a read-only input tape, and two reading heads, one reading the input from left to right (forward transitions), and the other head reading the input from right to left (backward transitions). Similar two-head finite automata models were introduced, e.g., in [5, 13, 17]. An input word is accepted by a biautomaton, if there is an accepting computation starting the heads on the two ends of the word meeting somewhere in an accepting state. Although the choice of reading a symbol by either head is nondeterministic, a deterministic outcome of the computation of the biautomaton is enforced by two properties: (i) The heads read input symbols independently, i.e., if one head reads a symbol and the other reads another, the resulting state does not depend on the order in which the heads read these single letters. (ii) If in a state of the finite control one head accepts a symbol, then this letter is accepted in this state by the other head as well. Later we call the former property the $\diamond$-property and the latter one the F-property. In [12] and a series of forthcoming papers [7, 8, 10, 11] it was shown that biautomata share

Z. Ésik and Z. Fülöp (Eds.): Automata and Formal Languages 2014 (AFL 2014)
EPTCS 151, 2014, pp. 271–285, doi:10.4204/EPTCS.151.19

a lot of properties with ordinary finite automata. For instance, as minimal DFAs, also minimal DBiAs are unique up to isomorphism [1, 12].

Now the question arises, which structural characterizations of subregular language families of DFAs carry over to biautomata. Let us give an example which involves partially ordered automata. A DFA with state set Q and input alphabet Σ is *partially ordered*, if there is a (partial) order $\leq$ on Q such that $q \leq \delta(q,a)$, for every $q \in Q$ and $a \in \Sigma$. In [4] it was shown that partially ordered DFAs characterize the family of $\mathscr{R}$-trivial regular languages, that is, a regular language L is $\mathscr{R}$-trivial if for its syntactic monoid M_L, the assumption $sM_L = tM_L$ implies $s = t$, for all $s,t \in M_L$. For the definition of the syntactic monoid of a regular language we refer to [1]. Adapting the definition of being partially ordered literally to DBiAs results in a characterization of the family of $\mathscr{J}$-trivial regular languages [11, 12]—originally the authors of [12] speak of acyclic biautomata instead, since loops are not considered as cycles there; we think that the term *partially ordered* is more suitable in this context. Here a regular language L is $\mathscr{J}$-trivial if for its syntactic monoid M_L, the assumption $M_L s M_L = M_L t M_L$ implies $s = t$, for all $s,t \in M_L$. Note that a language is $\mathscr{J}$-trivial regular if and only if it is piecewise testable [19]. A language $L \subseteq \Sigma^*$ is *piecewise testable* if it is a finite Boolean combination of languages of the form $\Sigma^* a_1 \Sigma^* a_2 \Sigma^* \dots \Sigma^* a_n \Sigma^*$, where $a_i \in \Sigma$ for $1 \leq i \leq n$. We can also ask whether a transfer of conditions can be done the other way around from DBiAs to DFAs. This is not that obvious, since structural properties on DBiAs may involve conditions on the forward and backward transitions. For instance, in [7] it was shown that biautomata, where for every state and every input letter the forward and the backward transition go to the same state, characterize the family of commutative regular languages. A regular language $L \subseteq \Sigma^*$ is *commutative* if for all words $u,v \in \Sigma^*$ and letters $a,b \in \Sigma$ we have $uabv \in L$ if and only if $ubav \in L$. Obviously, this condition can be used also to give a structural characterization of commutative regular languages on DFAs, namely that for every state q and letters $a,b \in \Sigma$ the finite state device satisfies $\delta(\delta(q,a),b) = \delta(\delta(q,b),a)$. This is the starting point of our investigations.

We study structural properties of DFAs appropriately adapted to DBiAs, since up to our knowledge most classical properties from the literature on finite automata were not studied for DBiAs yet. Our investigation is started in Section 3 with automata which transition functions induce permutations on the state set. Originally permutation DFAs were introduced in [20]. We show that both types of finite state machines, permutation DFAs and permutation DBiAs are equally powerful. Thus, an alternative characterization of the family of p-regular languages in terms of DBiAs is obtained. Next we take a closer look on quite the opposite of permutation automata, namely on permutation-free devices—see Section 4. A special case of a permutation-free automaton is an acyclic (expect for sink states) one. It is easy to see that acyclic DFAs as well as DBiAs characterize the family of finite languages. An important subregular language family, which can be obtained from finite languages by finitely many applications of concatenation, union, and complementation with respect to the underlying alphabet, is the class of star-free languages. It obeys a variety of different characterizations [14], one of them are permutation-free DFAs. We show that permutation-free DBiAs characterize the star-free languages, too. For strongly permutation-free automata, which are automata that are permutation-free and where also the identity permutation is forbidden, we find the first significant difference of DFAs and DBiAs. While for DFAs this property characterizes the family of definite languages, DBiAs describe only finite or co-finite languages. A language $L \subseteq \Sigma^*$ is *definite* [16] if and only if $L = L_1 \cup \Sigma^* L_2$, for some finite languages L_1 and L_2. Moreover, we find a relation between strongly permutation-free automata, and automata where all states are almost-equivalent—the notion of almost-equivalence was introduced in [2]. Then in Section 5 we continue our investigation with another important subfamily of star-free languages, namely ordered languages [18]. A DFA with state set Q and input alphabet Σ is *ordered* if there is a total order $\leq$ on the state set Q such that $p \leq q$ implies $\delta(p,a) \leq \delta(q,a)$, for every $p,q \in Q$ and $a \in \Sigma$.

Property	Automata type	
	DFAs	DBiAs
permutation	p-regular	p-regular
permutation-free	star-free	star-free
ordered	ordered	FIN $\cup$ co-FIN $\subset \cdot \subset$ ORD
partially ordered	$\mathscr{R}$-trivial	$\mathscr{J}$-trivial
strongly permutation-free	definite	finite and co-finite
acyclic; self-loops are cycles	finite	finite
non-exiting	prefix-free	circumfix-free
non-returning	strict superset of suffix-free	strict subset of non-returning DFAs

Table 1: Comparison of the results on structural properties on DFAs and DBiAs and their induced language families (shading represents results obtained in this paper); here FIN refers to the family of finite languages, co-FIN to the family of co-finite languages, and ORD to the family of ordered languages.

The family of ordered languages lies strictly in-between the family of finite and the family of star-free languages. Appropriately adapting this definition to biautomata results in a language class, which we call the family of bi-ordered languages, that is a proper superset of the family of finite and co-finite languages and a strict subset of the family of ordered languages. Moreover, it is shown that there is a subtle difference whether the order condition is applied to automata in general or to minimal devices only. In the next to last section we take a closer look on non-exiting and non-returning machines. It is well known that non-exiting DFAs characterize the family of prefix-free languages, while non-returning automata are related to suffix-free languages. We show that every biautomaton which is non-exiting must also be non-returning (unless it accepts the empty language), and that non-exiting minimal DBiAs characterize the family of circumfix-free languages. For non-returning minimal DBiAs we prove that the induced language family is a strict subset of the family of non-returning minimal DFAs languages. The obtained results are summarized in Table 1. In the last section we briefly discuss our findings and give some hints on future research directions on the subject under consideration.

2 Preliminaries

A *deterministic finite automaton* (DFA) is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is the finite set of *states*, Σ is the finite set of *input symbols*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *accepting states*, and $\delta\colon Q \times \Sigma \to Q$ is the *transition function*. As usual, the transition function δ can be recursively extended to $\delta\colon Q \times \Sigma^* \to Q$. The *language accepted* by A is defined as $L(A) = \{ w \in \Sigma^* \mid \delta(q_0, w) \in F \}$.

A *deterministic biautomaton* (DBiA) is a sixtuple $A = (Q, \Sigma, \cdot, \circ, q_0, F)$, where Q, Σ, q_0, and F are defined as for DFAs, and where $\cdot$ and $\circ$ are mappings from $Q \times \Sigma$ to Q, called the *forward* and *backward transition function*, respectively. It is common in the literature on biautomata to use an infix notation for these functions, i.e., writing $q \cdot a$ and $q \circ a$ instead of $\cdot(q,a)$ and $\circ(q,a)$. Similar as for the transition function of a DFA, the forward transition function $\cdot$ can be extended to $\cdot\colon Q \times \Sigma^* \to Q$ by $q \cdot \lambda = q$ and $q \cdot av = (q \cdot a) \cdot v$, for all states $q \in Q$, symbols $a \in \Sigma$, and words $v \in \Sigma^*$. Here λ refers to the *empty word*. The extension of the backward transition function $\circ$ to $\circ\colon Q \times \Sigma^* \to Q$ is defined as follows: $q \circ \lambda = q$ and $q \circ va = (q \circ a) \circ v$, for all states $q \in Q$, symbols $a \in \Sigma$, and words $v \in \Sigma^*$. Notice that $\circ$ consumes the input from right to left, hence the name backward transition function.

The DBiA A *accepts* a word $w \in \Sigma^*$ if there are words $u_i, v_i \in \Sigma^*$, for $1 \leq i \leq k$, such that w can be written as $w = u_1 u_2 \dots u_k v_k \dots v_2 v_1$, and

$$((\dots((((q_0 \cdot u_1) \circ v_1) \cdot u_2) \circ v_2) \dots) \cdot u_k) \circ v_k \in F.$$

The language accepted by A is $L(A) = \{ w \in \Sigma^* \mid A \text{ accepts } w \}$.

The DBiA A has the *$\diamond$-property*, if $(q \cdot a) \circ b = (q \circ b) \cdot a$, for all $a, b \in \Sigma$, and $q \in Q$, and it has the *F-property*, if for all $q \in Q$ and $a \in \Sigma$ it is $q \cdot a \in F$ if and only if $q \circ a \in F$. The biautomata as introduced in [12] always had to satisfy both these properties, while in [7, 8] also biautomata that lack one or both of these properties, as well as nondeterministic biautomata were studied. Throughout the current paper, when writing of biautomata, or DBiAs, we always mean deterministic biautomata that satisfy both the $\diamond$-property, and the F-property, i.e., the model as introduced in [12]. For such biautomata the following it is known from the literature [7, 12]:

- $(q \cdot u) \circ v = (q \circ v) \cdot u$, for all states $q \in Q$ and words $u, v \in \Sigma^*$,
- $(q \cdot u) \circ vw \in F$ if and only if $(q \cdot uv) \circ w \in F$, for all states $q \in Q$ and words $u, v, w \in \Sigma^*$.

From this one can conclude that for all words $u_i, v_i \in \Sigma^*$, with $1 \leq i \leq k$, it is

$$((\dots((((q_0 \cdot u_1) \circ v_1) \cdot u_2) \circ v_2) \dots) \cdot u_k) \circ v_k \in F$$

if and only if

$$q_0 \cdot u_1 u_2 \dots u_k v_k \dots v_2 v_1 \in F.$$

Therefore, the language accepted by a DBiA A can as well be defined as $L(A) = \{ w \in \Sigma^* \mid q_0 \cdot w \in F \}$.

Let A be DFA or a DBiA with state set Q. We say that a state $q \in Q$ is a *sink state* if and only if all outgoing transition (regardless whether they are forward or backward transitions) are self-loops only. Note, that in particular, one can distinguish between accepting and non-accepting sink states.

In the following we define the two DFAs contained in a DBiA, which accept the language, and the reversal of the language accepted by the biautomaton. Let $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ be a DBiA. We denote by Q_{fwd} the set of all states reachable from q_0 by only using forward transitions, and denote the set of states reachable by only using backward transitions by Q_{bwd}, i.e.,

$$Q_{\text{fwd}} = \{ q \in Q \mid \exists u \in \Sigma^* : q_0 \cdot u = q \} \quad \text{and} \quad Q_{\text{bwd}} = \{ q \in Q \mid \exists v \in \Sigma^* : q_0 \circ v = q \}.$$

Now we define the DFA $A_{\text{fwd}} = (Q_{\text{fwd}}, \Sigma, \delta_{\text{fwd}}, q_0, F_{\text{fwd}})$, with $F_{\text{fwd}} = Q_{\text{fwd}} \cap F$, and $\delta_{\text{fwd}}(q, a) = q \cdot a$, for all states $q \in Q_{\text{fwd}}$ and symbols $a \in \Sigma$. Similarly, we define the DFA $A_{\text{bwd}} = (Q_{\text{bwd}}, \Sigma, \delta_{\text{bwd}}, q_0, F_{\text{bwd}})$, with $F_{\text{bwd}} = Q_{\text{bwd}} \cap F$, and $\delta_{\text{bwd}}(q, a) = q \circ a$, for all $q \in Q$ and $a \in \Sigma$. One readily sees that $L(A_{\text{fwd}}) = L(A)$. Moreover, since $q \circ uv = (q \circ v) \circ u$, one can also see $L(A_{\text{bwd}}) = L(A)^R$. It is shown in [9] that if A is a minimal biautomaton, then the two DFAs A_{fwd} and A_{bwd} are minimal, too.

3 Permutation Automata

First we study automata where every input induces a permutation on the state set. Such finite automata were defined in [20]. A DFA $A = (Q, \Sigma, \delta, q_0, F)$ is a *permutation DFA* if $\delta(p, a) = \delta(q, a)$ implies $p = q$, for all $p, q \in Q$ and $a \in \Sigma$. A regular language is *p-regular* if it is accepted by a permutation DFA. We give a similar definition for biautomata: a biautomaton $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ is a *permutation biautomaton* if for all $p, q \in Q$ and $a \in \Sigma$ we have that $p \cdot a = q \cdot a$ implies $p = q$, and also $p \circ a = q \circ a$ implies $p = q$.

We will see that a language is p-regular if and only if it is accepted by a permutation biautomaton. Before we can show this, we describe a useful technique to construct a biautomaton from finite automata. In [12] a construction of a biautomaton from a given DFA A is described, that uses a cross-product construction of A with the power-set automaton of the reversal of A. In the following we describe how a biautomaton can be constructed from two arbitrary DFAs accepting a regular language and its reversal.

Let $L \subseteq \Sigma^*$ be a regular language, and for $i \in \{1,2\}$ let $A_i = (Q_i, \Sigma, \delta_i, q_0^{(i)}, F_i)$ be DFAs with $L(A_1) = L$, and $L(A_2) = L^R$. Further, for all states $p \in Q_1$ let u_p be some word with $\delta_1(q_0^{(1)}, u_p) = p$, and similarly for $q \in Q_2$ let v_q be a word with $\delta_2(q_0^{(2)}, v_q) = q$. Then define the automaton $B_{A_1 \times A_2} = (Q, \Sigma, \cdot, \circ, q_0, F)$ with state set $Q = Q_1 \times Q_2$, initial state $q_0 = (q_0^{(1)}, q_0^{(2)})$, accepting states $F = \{ (p,q) \in Q \mid u_p v_q^R \in L \}$, and where for all $(p,q) \in Q$ and $a \in \Sigma$ we have $(p,q) \cdot a = (\delta_1(p,a), q)$, and $(p,q) \circ a = (p, \delta_2(q,a))$. The following lemma proves the correctness of this construction.

Lemma 1 *For $i \in \{1,2\}$ let $A_i = (Q_i, \Sigma, \delta_i, q_0^{(i)}, F_i)$ be DFAs with $L(A_1) = L$, and $L(A_2) = L^R$. Then $B_{A_1 \times A_2}$ is a deterministic biautomaton, such that $L(B_{A_1 \times A_2}) = L$.*

Besides its usefulness for our result on permutation biautomata, this construction is also of relevance from a descriptional complexity point of view. Using the construction from [12] on an n-state DFA yields a biautomaton with $n \cdot 2^n$ states. In fact, a precise analysis in [10] that uses a similar construction as in [12] proves a tight bound of $n \cdot 2^n - 2(n-1)$ states for converting an n-state DFA into an equivalent biautomaton. However, this bound only takes into account the state complexity of the original language L, but not the state complexity of L^R. If the state complexity of L^R much smaller than 2^n then the bound from [10] is far off the number of states of the minimal biautomaton for L. Using our construction, we can deduce an upper bound of $n \cdot m$ for the number of states of a biautomaton for the language L, if n is the state complexity of L, and m is the state complexity of L^R.

Now we show our result on permutation automata.

Theorem 2 *A language is p-regular if and only if it is accepted by some permutation biautomaton.*

Proof: If A is a permutation biautomaton, then A_{fwd} is a permutation DFA, hence $L(A)$ is p-regular. For the reverse implication let L be some p-regular language over the alphabet Σ. Then L^R is p-regular, too [20], so there are permutation DFAs $A_i = (Q_i, \Sigma, \delta_i, q_0^{(i)}, F_i)$, for $i = 1,2$, that $L = L(A_1)$, and $L^R = L(A_2)$. Using the cross-product construction from Lemma 1, we obtain the biautomaton $B = B_{A_1 \times A_2}$. Recall that the states of B are of the form (p,q), with $p \in Q_1$ and $q \in Q_2$, and the transitions are defined such that $(p,q) \cdot a = (\delta_1(p,a), q)$, and $(p,q) \circ a = (p, \delta_2(q,a))$, for all states (p,q) and symbols $a \in \Sigma$. We will show in the following that B is a permutation automaton. Therefore let (p,q) and (p',q') be two states of B, and $a \in \Sigma$. If $(p,q) \cdot a = (p',q') \cdot a$ then $(\delta(p,a), q) = (\delta(p',a), q')$, which implies $\delta(p,a) = \delta(p',a)$ and $q = q'$. Since A_1 is a permutation DFA, we also obtain $p = p'$, hence $(p,q) = (p',q')$. With a similar reasoning, using the permutation property of A_2, we see that also $(p,q) \circ a = (p',q') \circ a$ implies $(p,q) = (p',q')$, therefore B is a permutation biautomaton. □

4 Permutation-Free Automata

An important subregular language family is the class of star-free languages. A language is star-free if it can be obtained from finite languages by finitely many applications of concatenation, union, and complementation with respect to the underlying alphabet. For the class of finite languages we have the following obvious theorem, which we state without proof.

Theorem 3 *A language is finite (co-finite, respectively) if and only if its minimal biautomaton is acyclic except for non-accepting (accepting, respectively) sink states; self-loops count as cycles.* □

The class of star-free languages obeys a variety of different characterizations [14], one of them being the following: a regular language is star-free if its minimal DFA is permutation-free. A DFA $A = (Q,\Sigma,\delta,q_0,F)$ is *permutation-free* if there is no word $w \in \Sigma^*$ such that the mapping $q \mapsto \delta(q,w)$, for all $q \in Q$, induces a *non-trivial* permutation, i.e., a permutation different from the identity permutation, on some set $P \subseteq Q$. Now the question arises whether a similar condition for biautomata also yields a characterization of the star-free languages. We feel that the following definition of permutation-freeness is a natural extension from the corresponding definition for DFAs. We say that a biautomaton $A = (Q,\Sigma,\cdot,\circ,q_0,F)$ is *permutation-free* if there are no words $u,v \in \Sigma^*$, such that the mapping $q \mapsto (q\cdot u)\circ v$ induces a non-trivial permutation on some set of states $P \subseteq Q$. It turns out that with this definition, permutation-free biautomata indeed characterize the star-free languages. We will later discuss some other possible definitions. Before we show our result on permutation-free biautomata, we prove the following lemma which helps us to relate permutations in biautomata to permutations in DFAs.

Lemma 4 *Let Q be a finite set and $\pi_1,\pi_2 \colon Q \to Q$ be two mappings satisfying $\pi_1(\pi_2(q)) = \pi_2(\pi_1(q))$ for all $q \in Q$. If there exists a subset $P \subseteq Q$ such that the mapping $\pi \colon P \to P$ defined by $\pi(p) = \pi_2(\pi_1(p))$ is a non-trivial permutation on P, then there exists a subset $P' \subseteq Q$ and an integer $d \geq 1$ such that π_1^d or π_2^d is a non-trivial permutation on P'.*

Proof: Consider the sequence of sets $\pi_1^0(P),\pi_1^1(P),\pi_1^2(P),\ldots \subseteq Q$. Since Q is a finite set, the number of different sets $\pi_1^j(P)$, for $j \geq 0$, is finite. Thus, there must be integers $m,d \geq 1$ such that the sets $\pi_1^0(P),\pi_1^1(P),\ldots\pi_1^{m+d-1}(P)$ are pairwise distinct, and $\pi_1^{m+d}(P) = \pi_1^m(P)$. Since $\pi = \pi_1\pi_2 = \pi_2\pi_1$ is a permutation on P, we obtain

$$P = \pi^{m+d}(P) = \pi_2^{m+d}(\pi_1^{m+d}(P)) = \pi_2^d(\pi_2^m(\pi_1^m(P))) = \pi_2^d(P),$$

which shows that π_2^d is a permutation on P. It follows that also π_1^d must be a permutation on P. If one of these is a non-trivial permutation we are done. Therefore assume that both π_1^d and π_2^d are the identity permutation on P. In this case it must be $d \geq 2$ because otherwise the permutation $\pi = \pi_1\pi_2$ would be trivial. Then the two sets P and $\pi_1(P)$ are different, so there is an element $q \in P$ with $\pi_1(q) \neq q$. On the other hand q must satisfy $\pi_1^d(q) = q$. Therefore the mapping π_1 is a permutation on the set $P' = \{\pi_1^0(p),\pi_1^1(p),\ldots,\pi_1^{d-1}(p)\}$, and it is non-trivial because $\pi_1^0(p) = p \neq \pi_1^1(p)$. □

Now we can show the following characterization of star-free languages in terms of permutation-free biautomata.

Theorem 5 *A language is star-free if and only if its* minimal *biautomaton is permutation-free.*

Proof: Clearly, if A is a minimal biautomaton that is permutation-free, then also the contained minimal DFA A_{fwd} is permutation-free, too. Therefore the language $L(A)$ is star-free.

For proving the reverse implication let $A = (Q,\Sigma,\cdot,\circ,q_0,F)$ be a minimal biautomaton that is *not* permutation-free. Then there are words $u,v \in \Sigma^*$, and a set of states $P \subseteq Q$, with $|P| \geq 2$, such that the mapping $\pi \colon Q \to Q$ defined as $\pi(q) = (q\cdot u)\circ v$ induces a non-trivial permutation on P. Notice that the $\diamond$-property of the biautomaton A implies $\pi(q) = \pi_1(\pi_2(q)) = \pi_2(\pi_1(q))$, for $\pi_1(q) = q\cdot u$, and $\pi_2(q) = q\circ v$. Therefore we can use Lemma 4, and obtain an integer $d \geq 1$ such that π_1^d or π_2^d induces a non-trivial permutation on some subset $P' \subseteq Q$. If π_1^d is non-trivial, then the word u^d induces a non-trivial permutation in the minimal DFA A_{fwd}, which in turn means that the language $L(A)$ is *not* star-free.

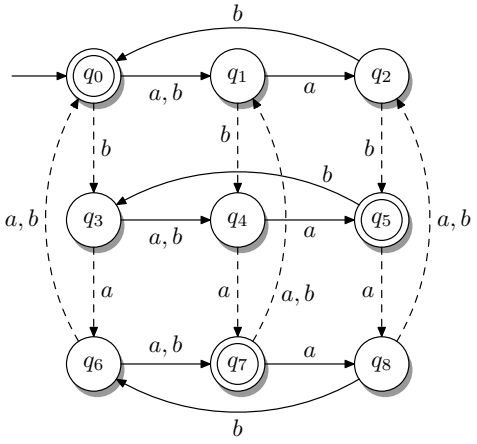

Figure 1: The permutation-free biautomaton A that has a word-cycle and a graph-cycle.

Otherwise, the mapping π_2^d is non-trivial, and the word v^d induces a non-trivial permutation on the minimal DFA A_{bwd}, which means that the language $L(A)^R$ is not star-free. Since the class of star-free languages is closed under reversal, we again conclude that the language $L(A)$ cannot be star-free in this case. □

In the above definition of permutation-free biautomata, from all states in the permutation induced by the word uv, the prefix u must be read with forward transitions and the suffix v with backward transitions. One could also think of other kinds of permutations in biautomata, and we shortly discuss two different such notions in the following. Since a permutation is composed of cycles, we describe the types of cycles. Let $P = \{p_0, p_1, \ldots, p_{k-1}\}$ be some set of states of a biautomaton A and w be some non-empty word over the input alphabet of A.

- We say that w induces a *word-cycle* on P if for $0 \leq i \leq k-1$ we have $p_{(i+1) \bmod k} = (p_i \cdot u_i) \circ v_i$, for some words u_i and v_i, with $u_i v_i = w$.
- We say that w induces a *graph-cycle* on P if $w = a_1 a_2 \ldots a_n$ and we have

$$p_{(i+1) \bmod k} = (\ldots((p_i \bullet_{i,1} a_1) \bullet_{i,2} a_2) \ldots) \bullet_{i,n} a_n,$$

  for $0 \leq i \leq k-1$, where $\bullet_{i,j} \in \{\cdot, \circ\}$, for $1 \leq j \leq n$. The intuition behind the definition of a graph-cycle is that the word w specifies the sequence of transitions (regardless whether they are forward or backward transitions) that are taken during the course of the computation.

Notice that if a biautomaton has a permutation as defined before Theorem 5, then it also has a word-cycle, and also a graph-cycle. Hence, if a language is accepted by a biautomaton that has no word-cycle or by a biautomaton that has no graph-cycle, then it is star-free. However, the converse is not true, as the following example shows.

Example 6 *Let $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ be the* minimal *DBiA for the language $L = \{aab, bab\}^*$. The biautomaton A is depicted in Figure 1—solid arrows denote forward transitions by $\cdot$, and dashed arrows denote backward transitions by $\circ$. By inspecting the DFA A_{fwd}, consisting of the states q_0, q_1, q_2, and the non-accepting sink state, which is not shown, one can see that A_{fwd} is permutation-free. Therefore the language L is star-free, and also the biautomaton A must be permutation-free. However, the word ab induces a word-cycle on the states q_0, q_1, and q_6 because $q_0 \circ ab = q_6$, $(q_6 \cdot a) \circ b = q_1$, and $q_1 \cdot ab = q_0$. Further, the word ab also induces a graph-cycle on the states q_0, q_4, and q_3 because $(q_0 \cdot a) \circ b = q_4$, $(q_4 \cdot a) \cdot b = q_3$, and $(q_3 \circ a) \circ b = q_0$.*

4.1 Strongly Permutation-Free Automata

A permutation-free automaton does not contain any *non-trivial* permutation, but it may contain the identity permutation. We may also forbid the identity permutation, which leads to the following definitions. A DFA $A = (Q, \Sigma, \delta, q_0, F)$ is *strongly permutation-free* if there is no non-empty word $w \in \Sigma^+$, such that the mapping $q \mapsto \delta(q, w)$ induces a permutation on some set $P \subseteq Q$, with $|P| \geq 2$. Similarly, a biautomaton $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ is *strongly permutation-free* if there are no words $u, v \in \Sigma^*$, with $uv \in \Sigma^+$, such that the mapping $q \mapsto (q \cdot u) \circ v$ induces a permutation on some set $P \subseteq Q$, with $|P| \geq 2$. In these definitions we require the words w and uv to be non-empty because the empty-word always induces the identity permutation on all sets of states. Further we only consider subsets $P \subseteq Q$ with $|P| \geq 2$ because every DFA and every biautomaton must contain a (maybe identity) permutation on a set $P \subseteq Q$ with $|P| \geq 1$, since by the pigeon hole principle there is a state which is repeatedly visited by only reading the letter a long enough.

Before we study which languages are accepted by strongly permutation-free automata, we give some further definitions which turn out to be related to strongly permutation-freeness. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA, and $w \in \Sigma^+$ some non-empty word. A state $q \in Q$ is called a *w-attractor* in A, if for all states $p \in Q$ there is an integer $k > 0$ such that $\delta(p, w^k) = q$. For a biautomaton $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ and words $u, v \in \Sigma^*$, with $|uv| \geq 1$, we denote by $\pi_{u,v}$ the mapping $p \mapsto (p \cdot u) \circ v$. Now a state $q \in Q$ is a *(u,v)-attractor* in A if for all $p \in Q$ there is an integer $k > 0$ such that $\pi_{u,v}^k(p) = q$. Notice that due to the $\diamond$-property of A the condition $\pi_{u,v}^k(p) = q$ can also be written as $(p \cdot u^k) \circ v^k = q$. Next we recall the definition of almost-equivalence. Two languages L_1 and L_2 are *almost-equivalent* ($L_1 \sim L_2$) if their symmetric difference $L_1 \triangle L_2 = (L_1 \setminus L_2) \cup (L_2 \setminus L_1)$ is finite. This notion naturally transfers to states as follows. For a state q of some DFA or biautomaton A we denote by $L_A(q)$ the language accepted by the automaton ${}_qA$ which is obtained from A by making state q its initial state. The language $L_A(q)$ is also called the *right language* of q. Now two states p and q of a DFA or biautomaton A are *almost-equivalent* ($p \sim q$) if their *right languages* $L_A(p)$ and $L_A(q)$ are almost-equivalent. We write $p \equiv q$, if p and q are *equivalent*, i.e., if $L_A(p) = L_A(q)$. Our next theorem connects the notions of almost-equivalence, w-attractors, and strongly permutation-freeness for DFAs, and shows that these conditions can be used to characterize the class of definite languages—a language L over an alphabet Σ is *definite* if there are finite languages L_1 and L_2 over Σ such that $L = L_1 \cup \Sigma^* L_2$. Interestingly the relation between definite languages and almost-equivalence was already studied in [16], long before the notion of almost-equivalence became popular in [2]—in [16] the used form of equivalence was not called "almost-equivalence," but simply "equivalent." Moreover, the relation between definite languages and strongly permutation-free automata was independently shown in [3]—compare also with [14, Exercise 28 of Chapter 4 and Exercise 13 of Chapter 5].

Theorem 7 *Let $A = (Q, \Sigma, \delta, q_0, F)$ be some* minimal *DFA, then the following statements are equivalent:*

1. *All states in Q are pairwise almost-equivalent.*
2. *For all words $w \in \Sigma^+$, there is a w-attractor in A.*
3. *A is strongly permutation-free.*
4. *$L(A)$ is a definite language.*

Now we turn to biautomata, where we will see that the equivalences between the first three conditions of Theorem 7 also hold in the setting of biautomata. However, we will see that the language class related to these conditions is different. Before we come to this result we recall the following lemma from [9]:

Lemma 8 *Let $A = (Q,\Sigma,\cdot,\circ,q_0,F)$ and $A' = (Q',\Sigma,\cdot',\circ',q_0',F')$ be two biautomata, and let $p \in Q$ and $q \in Q'$. Then $p \sim q$ if and only if $(p\cdot u)\circ v \sim (q\cdot' u)\circ' v$, for all words $u,v \in \Sigma^*$. Moreover, $p \sim q$ implies $(p\cdot u)\circ v \equiv (q\cdot' u)\circ' v$, for all words $u,v \in \Sigma^*$ with $|uv| \geq k = |Q \times Q'|$.*

The two automata A and A' in Lemma 8 need not be different, so this lemma can also be used for states p and q in one biautomaton A. Further, if this biautomaton A is minimal, then it does not contain a pair of different, but equivalent states. In this case the states p and q are almost-equivalent if and only if for all long enough words $uv \in \Sigma^*$ the two states $(p\cdot u)\circ v$ and $(q\cdot u)\circ v$ are the same state. We obtain the following corollary.

Corollary 9 *Let $A = (Q,\Sigma,\cdot,\circ,q_0,F)$ be a* minimal *biautomaton, and $k = |Q \times Q|$. Two states $p,q \in Q$ are almost-equivalent if and only if for all words $u,v \in \Sigma^*$, with $|uv| \geq k$, it is $(p\cdot u)\circ v = (q\cdot u)\circ v$.* □

Now we are ready for our result on strongly permutation-free biautomata.

Theorem 10 *Let $A = (Q,\Sigma,\cdot,\circ,q_0,F)$ be some* minimal *biautomaton, then the following statements are equivalent:*

1. *All states in Q are pairwise almost-equivalent.*
2. *There is a state $s \in Q$ that is a (u,v)-attractor in A, for all $u,v \in \Sigma^*$ with $|uv| \geq 1$.*
3. *A is strongly permutation-free.*
4. *$L(A)$ is a finite or co-finite language.*

Proof: Let $A = (Q,\Sigma,\cdot,\circ,q_0,F)$ be a minimal biautomaton. First assume $L(A)$ is a finite language, and let ℓ be the length of a longest word in $L(A)$. Then the right language $L_A(q)$ of every state $q \in Q$ is finite, so all states in Q are pairwise almost-equivalent. Since $L(A)$ is finite, and A is minimal, the biautomaton has a non-accepting sink state s, with $s\cdot a = s\circ a = s$ for all $a \in \Sigma$. Moreover, from any state $q \in Q$ the automaton always reaches this sink state s after reading at most ℓ symbols. Therefore, the state s is a (u,v)-attractor in A, for all words $u,v \in \Sigma^*$ with $|uv| \geq 1$. It also follows that the only permutation that is possible in A is the identity permutation on the singleton set $\{s\}$, hence A is strongly permutation-free. The case where $L(A)$ is a co-finite language is similar. The only differences are that the sink state s is an accepting state, and the integer ℓ must be the length of the longest word that is *not* in $L(A)$. This shows that statement 4 implies all other statements, and it remains to prove the other directions.

Assume that all states in Q are pairwise almost-equivalent, and let k be the integer from Corollary 9. If the length of every word in $L(A)$ is less than k then $L(A)$ is a finite language, so assume that there is a word $w \in L(A)$ with $|w| \geq k$. We show that in this case language $L(A)$ contains every word of length at least k, and thus, is co-finite. Since $|w| \geq k$, we can write w as $w = w_1w_2$, with $|w_2| = k$. Because $w \in L(A)$ we have $(q_0\cdot w_1)\circ w_2 \in F$. Now let $u \in \Sigma^{\geq k}$, and consider the states $p = (q_0\cdot w_1)$ and $q = (q_0\cdot u)$. Since all states are almost-equivalent, and the word w_2 has length k, we can use Corollary 9 to obtain $(p\cdot\lambda)\circ w_2 = (q\cdot\lambda)\circ w_2$. Hence the state $q\circ w_2 = (q_0\cdot u)\circ w_2$ is accepting. By the $\diamond$-property of A we have $(q_0\cdot u)\circ w_2 = (q_0\circ w_2)\cdot u$, and another application of Corollary 9 on the almost-equivalent states q_0 and $q_0\circ w_2$ we obtain $q_0\cdot u = (q_0\circ w_2)\cdot u$, because $|u| \geq k$. This shows that the word u is accepted by A, hence $L(A)$ is co-finite. This shows that statements 1 and 4 are equivalent.

Next assume there is a state $s \in Q$ that is a (u,v)-attractor for all words $u,v \in \Sigma^*$ with $|uv| \geq 1$. Then A must be strongly permutation-free, which can be seen as follows. Assume that there are words $u,v \in \Sigma^*$ with $|uv| \geq 1$ such that the mapping $\pi\colon q \mapsto (q\cdot u)\circ v$ is a permutation on some set $P \subseteq Q$, with $|P| \geq 2$. Then it must be $\left|\pi^i(P)\right| = |P| \geq 2$, for all $i \geq 0$. But since s is a (u,v)-attractor, there is an integer $m \geq 0$

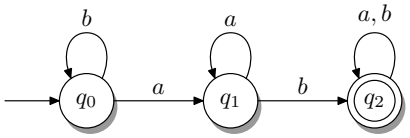

Figure 2: The minimal DFA A for the language $\Sigma^*ab\Sigma^*$ over the alphabet $\Sigma = \{a,b\}$. The states of A_1 can be ordered by $q_0 \leq q_1 \leq q_2$.

such that $(q \cdot u^m) \circ v^m = s$, for all states $q \in Q$. Then $|\pi^m(P)| = 1$, which is a contradiction, therefore A is strongly permutation-free.

Now it is sufficient to show that statement 3 implies statement 1. Therefore let A be strongly permutation-free, and assume for the sake of contradiction, that there are two states $p,q \in Q$ with $p \nsim q$. Corollary 9 now implies that there are words $u,v \in \Sigma^*$, with $|uv| \geq |Q \times Q|$, such that $(p \cdot u) \circ v \neq (q \cdot u) \circ v$. Since the number of steps in the computations $(p \cdot u) \circ v$ and $(q \cdot u) \circ v$ is at least $|Q \times Q|$, the words u and v can be written as $u = u_1u_2u_3$ and $v = v_3v_2v_1$ such that

$$\begin{aligned} &(p \cdot u_1) \circ v_1 = p_1, && (p_1 \cdot u_2) \circ v_2 = p_1, && (p_1 \cdot u_3) \circ v_3 = (p \cdot u) \circ v, \\ &(q \cdot u_1) \circ v_1 = q_1, && (q_1 \cdot u_2) \circ v_2 = q_1, && (q_1 \cdot u_3) \circ v_3 = (q \cdot u) \circ v. \end{aligned}$$

But then the mapping $\pi\colon r \mapsto (r \cdot u_2) \circ v_2$ is a permutation (the identity permutation) on the states $\{p_1,q_1\}$. Since A is strongly permutation-free, it follows $p_1 = q_1$, and in turn $(p \cdot u) \circ v = (q \cdot u) \circ v$. This contradicts the assumption $p \nsim q$, and concludes our proof. □

5 Ordered Automata

We now study automata where one can find an order on the state set that is compatible with the transitions of the automaton. Ordered DFAs and their accepted languages were studied in [18]. A DFA $A = (Q,\Sigma,\delta,q_0,F)$ is *ordered* if there exists some total order $\leq$ on the state set Q such that $p \leq q$ implies $\delta(p,a) \leq \delta(q,a)$, for all states $p,q \in Q$ and symbols $a \in \Sigma$. Similarly, a biautomaton $A = (Q,\Sigma,\cdot,\circ,q_0,F)$ is *ordered* if there is a total order $\leq$ on Q such that $p \leq q$ implies $p \cdot a \leq q \cdot a$ as well as $p \circ a \leq q \circ a$, for all states $p,q \in Q$ and symbols $a \in \Sigma$. A regular language is *ordered* if it is accepted by some ordered DFA, and it is *bi-ordered* if it is accepted by an ordered biautomaton. Moreover, a language is *strictly ordered* if its minimal DFA is ordered, and it is *strictly bi-ordered* if its minimal biautomaton is ordered.

The next two results show that the class of bi-ordered languages is located between the class of ordered languages and the class of finite and co-finite languages.

Theorem 11 *The class of bi-ordered languages is strictly contained in the class of ordered languages.*

Proof: If L is a bi-ordered language, then it is accepted by some ordered DBiA A. Then of course the automaton A_{fwd} is an ordered DFA. Therefore any bi-ordered language is an ordered language. The strictness of this inclusion is witnessed by the language $\Sigma^*ab\Sigma^*$ over the alphabet $\Sigma = \{a,b\}$, which is accepted by the ordered DFA A from Figure 2.

Next let us argue, why no biautomaton for the language $\Sigma^*ab\Sigma^*$ can be ordered. Therefore consider some biautomaton $B = (Q,\Sigma,\cdot,\circ,q_0,F)$ with $L(B) = \Sigma^*ab\Sigma^*$. In the following we use the notation $[u.v]$, for $u,v \in \Sigma^*$, to describe the state $(q_0 \cdot u) \circ v$ of B. Of course, different word pairs $[u.v]$ and $[u'.v']$ may describe the same state. First note that the three states $[\lambda.\lambda]$, $[a.\lambda]$, and $[\lambda.b]$ must be pairwise distinct,

because every one of these states leads to an accepting state on a different input string. Assume there is some order $\leq$ on the state set Q that is compatible with the transition functions of B. There are six different possibilities to order the three above mentioned states:

$$[\lambda.\lambda] \leq [a.\lambda] \leq [\lambda.b], \qquad [a.\lambda] \leq [\lambda.\lambda] \leq [\lambda.b], \qquad [\lambda.b] \leq [a.\lambda] \leq [\lambda.\lambda],$$
$$[\lambda.\lambda] \leq [\lambda.b] \leq [a.\lambda], \qquad [a.\lambda] \leq [\lambda.b] \leq [\lambda.\lambda], \qquad [\lambda.b] \leq [\lambda.\lambda] \leq [a.\lambda].$$

If $[\lambda.\lambda] \leq [a.\lambda] \leq [\lambda.b]$ then it must be $[\lambda.b^i] \leq [a.b^i] \leq [\lambda.b^{i+1}]$, for all $i \geq 0$. Since the number of states in Q is finite, there must be integers $j > k \geq 0$ for which $[\lambda.b^k] = [\lambda.b^j]$. It then follows that $[\lambda.b^k] = [a.b^k] = [\lambda.b^{k+1}]$. This is a contradiction because $[a.b^k]$ describes an accepting state, while $[\lambda.b^k]$ and $[\lambda.b^{k+1}]$ describe non-accepting states.

If $[\lambda.\lambda] \leq [\lambda.b] \leq [a.\lambda]$ then we obtain $[a^i.\lambda] \leq [a^i.b] \leq [a^{i+1}.\lambda]$ for all $i \geq 0$. Similar to the case above we get a contradiction: because Q is finite there is an integer $k \geq 0$ with $[a^k.\lambda] = [a^k.b] = [a^{k+1}.\lambda]$, but the state $[a^k.b]$ is accepting while the other two states are non-accepting.

Next consider the case $[a.\lambda] \leq [\lambda.\lambda] \leq [\lambda.b]$. By reading symbol a with a forward transition we obtain $[a.\lambda] \leq [a.b]$ from the second inequality, and by reading b with a backward transition, the first inequality implies $[a.b] \leq [\lambda.b]$. Note that both times $[a.b]$ describes the same state because B has the $\diamond$-property. Further, this state must be different from the three states $[a.\lambda]$, $[\lambda.\lambda]$, and $[\lambda.b]$ because it is an accepting state, and the others are not. Now there are two possibilities for the placement of state $[a.b]$ in the order:

$$[a.\lambda] \leq [a.b] \leq [\lambda.\lambda] \leq [\lambda.b] \quad \text{or} \quad [a.\lambda] \leq [\lambda.\lambda] \leq [a.b] \leq [\lambda.b].$$

The first case implies $[a^{i+1}.\lambda] \leq [a^{i+1}.b] \leq [a^i.\lambda]$, for all $i \geq 0$, which leads to the contradictory equation $[a^{k+1}.\lambda] = [a^{k+1}.b] = [a^k.\lambda]$, for some $k \geq 0$. The second case implies $[\lambda.b^i] \leq [a.b^{i+1}] \leq [\lambda.b^{i+1}]$, for all $i \geq 0$, and to the contradiction $[\lambda.b^k] = [a.b^{k+1}] = [\lambda.b^{k+1}]$, for some $k \geq 0$.

With similar argumentation, the remaining three cases $[a.\lambda] \leq [\lambda.b] \leq [\lambda.\lambda]$, $[\lambda.b] \leq [a.\lambda] \leq [\lambda.\lambda]$, and $[\lambda.b] \leq [\lambda.\lambda] \leq [a.\lambda]$ lead to contradictions—we omit the details. This shows that there is no order of the state set Q that is compatible with the transition functions of B. Therefore, the ordered language $\Sigma^* ab \Sigma^*$ is not a bi-ordered language. □

In the proof of the following result we use the lexicographic order $<_{\text{lex}}$ of words, which is defined as follows. Let Σ be an alphabet of size k and fix some order $a_1, a_2, \ldots, a_k$ of the symbols from Σ. For two words $w_1, w_2 \in \Sigma^*$ let $w_1 <_{\text{lex}} w_2$ if and only if either w_1 is a prefix of w_2, or $w_1 = u a_i w_1'$ and $w_2 = u a_j w_2'$, for some words $u, w_1', w_2' \in \Sigma^*$ and symbols $a_i, a_j \in \Sigma$, with $i < j$.

Theorem 12 *The class of finite and co-finite languages is strictly contained in the class of bi-ordered languages.*

Proof: Let L be some finite language over the alphabet Σ and let ℓ be the length of the longest word in L. We construct an ordered biautomaton for L as follows. Let $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ be the biautomaton with state set $Q = \{ (u,v) \mid u, v \in \Sigma^*, |uv| \leq \ell \} \cup \{s\}$, initial state $q_0 = (\lambda, \lambda)$, set of accepting states $F = \{ (u,v) \in Q \mid uv \in L \}$, and where the transition functions $\cdot$ and $\circ$ are defined as follows: for all symbols $a \in \Sigma$ let $s \cdot a = s \circ a = s$, and for all states $(u,v) \in Q$ let

$$(u,v) \cdot a = \begin{cases} (ua, v) & \text{if } |uav| \leq \ell, \\ s & \text{otherwise,} \end{cases} \qquad (u,v) \circ a = \begin{cases} (u, av) & \text{if } |uav| \leq \ell, \\ s & \text{otherwise.} \end{cases}$$

One readily sees that A has both the $\diamond$-property, and the F-property. Now let us define the order $\leq$ on Q as follows. First of all let $(u,v) \leq s$, for all $(u,v) \in Q$, so the non-accepting sink state is the largest element of Q. Next, for two different states (u_1, v_1) and (u_2, v_2) let $(u_1, v_1) \leq (u_2, v_2)$ if and only if

- $|u_1v_1| < |u_2v_2|$, or
- $|u_1v_1| = |u_2v_2|$, and $|u_1| < |u_2|$, or
- $|u_1v_1| = |u_2v_2|$, $|u_1| = |u_2|$, and $u_1 <_{\text{lex}} u_2$, or
- $|u_1v_1| = |u_2v_2|$, $u_1 = u_2$, and $v_1 <_{\text{lex}} v_2$.

Notice that if none of the four cases above holds, then $(u_1,v_1) = (u_2,v_2)$. It remains to show that the transitions of A respect the order $\leq$. Since state s goes to itself on every symbol, because it is the largest element, we have $(u,v)\cdot a \leq s\cdot a$, and $(u,v)\circ a \leq s\circ a$. Next let (u_1,v_1) and (u_2,v_2) be two different states of A with $(u_1,v_1) \leq (u_2,v_2)$. Then it must be $|u_1v_1| \leq |u_2v_2| \leq \ell$. If $|u_2v_2| = \ell$, then (u_2,v_2) goes to the sink state s on both the forward, and the backward a-transition. Since s is the largest element we obtain $(u_1,v_1)\cdot a \leq (u_2,v_2)\cdot a$, and $(u_1,v_1)\circ a \leq (u_2,v_2)\circ a$. Therefore, in the following argumentation we assume that $|u_2,v_2| < \ell$, so that we have $(u_1,v_1)\cdot a = (u_1a,v_1)$ and $(u_2,v_2)\cdot a = (u_2a,v_2)$, as well as $(u_1,v_1)\circ a = (u_1,av_1)$ and $(u_2,v_2)\circ a = (u_2,av_2)$. Now we have to show $(u_1a,v_1) \leq (u_2a,v_2)$ and $(u_1,av_1) \leq (u_2,av_2)$, for which we distinguish four cases.

- If $|u_1v_1| < |u_2v_2|$ then clearly $|u_1av_1| < |u_2av_2|$, from which we conclude $(u_1a,v_1) \leq (u_2a,v_2)$, and $(u_1,av_1) \leq (u_2,av_2)$.
- If $|u_1v_1| = |u_2v_2|$, and $|u_1| < |u_2|$, then also $|u_1a| < |u_2a|$. Again, we can conclude $(u_1a,v_1) \leq (u_2a,v_2)$, and $(u_1,av_1) \leq (u_2,av_2)$.
- Next assume $|u_1v_1| = |u_2v_2|$, $|u_1| = |u_2|$, and $u_1 <_{\text{lex}} u_2$. Since u_1 and u_2 are of same length, the fact $u_1 <_{\text{lex}} u_2$ implies $u_1a <_{\text{lex}} u_2a$. Thus, we get $(u_1a,v_1) \leq (u_2a,v_2)$, and $(u_1,av_1) \leq (u_2,av_2)$.
- Finally let $|u_1v_1| = |u_2v_2|$, $|u_1| = |u_2|$, $u_1 = u_2$, and $v_1 <_{\text{lex}} v_2$. From $v_1 <_{\text{lex}} v_2$ follows $av_1 <_{\text{lex}} av_2$, so we obtain $(u_1a,v_1) \leq (u_2a,v_2)$, and $(u_1,av_1) \leq (u_2,av_2)$.

This shows that the biautomaton A is an ordered biautomaton.

In case of a co-finite language $L \subseteq \Sigma^*$ we first take its complement $\Sigma^* \setminus L$, which is finite, and apply the above given construction. Then we obtain an ordered biautomata A. Finally, exchanging accepting and non-accepting states—this is the ordinary complementation construction known for DFAs applied to DBiAs—results in an ordered biautomata for the language L.

Finally, strictness of the inclusion is witnessed by the infinite and not co-finite language $a^* + b$, which is accepted by the bi-ordered biautomaton from Figure 3. □

Notice that the language $\Sigma^*ab\Sigma^*$ from the proof of Theorem 11 is even a strictly ordered language, since its minimal DFA A from Figure 2 is ordered. As we have seen, this language is not a bi-ordered language, therefore the class of bi-ordered languages does not even contain all strictly ordered languages. On the other hand, if a language is strictly bi-ordered, i.e., if its minimal biautomaton B is ordered, then also the minimal DFA B_{fwd} is ordered. Therefore, the class of strictly bi-ordered languages is contained in the classes of strictly ordered languages. We summarize our findings in the following corollary.

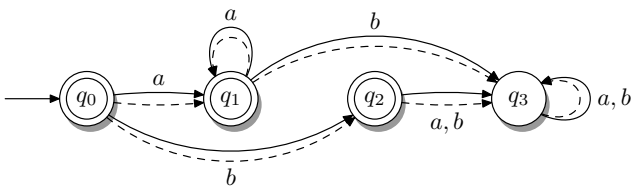

Figure 3: A bi-ordered biautomaton with order $q_0 \leq q_1 \leq q_2 \leq q_3$ for the language $a^* + b$.

Corollary 13 *The class of strictly bi-ordered languages is proper subset of the class of strictly ordered languages.* □

Concerning the relation between bi-ordered languages and strictly ordered languages, we can see that these are incomparable to each other. We have seen that the strictly ordered language $\Sigma^* ab \Sigma^*$ is not bi-ordered. On the other hand, we know that every finite language is bi-ordered. But one can see that the minimal DFA for the finite language $\{ab\}$ is not ordered—the reader is invited convince himself of this fact. A proof of a more general result, saying that a single word language is strictly ordered if and only if the word is of the form a^i for some alphabet symbol a and integer $i \geq 0$, can be found in [18].

Corollary 14 *The classes of bi-ordered languages and of strictly ordered languages are incomparable to each other.* □

Moreover, with a similar argumentation as above we obtain:

Corollary 15 *The class strictly bi-ordered languages is a proper subset of the class of bi-ordered languages.* □

6 Non-Exiting and Non-Returning Automata

In this last section we study so called non-exiting automata and non-returning automata. A biautomaton or finite automaton A is *non-exiting* if all outgoing transitions from accepting states go to a non-accepting sink state, and it is *non-returning* if the initial state does not have any ingoing transitions. We say that A is *exiting* if it is *not* non-exiting, and it is *returning* if it is *not* non-returning.

In the DFA case non-exiting automata are known to characterize the class of prefix-free languages, while non-returning automata are related to suffix-free languages. However this latter relation is not a characterization: it is true that every DFA that accepts a (non-empty) suffix-free language must be non-returning, but the reverse implication does not hold.

Concerning the situation for biautomata, we first show that every biautomaton which is non-exiting must also be non-returning (unless it accepts the empty language).

Lemma 16 *Let A be a biautomaton with $L(A) \neq \emptyset$. If A is non-exiting, then A is non-returning.*

Proof: We prove the contraposition of the lemma. Assume $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ is a returning biautomaton. Then there must be words $u, v \in \Sigma^*$, with $|uv| \geq 1$, such that $(q_0 \cdot u) \circ v = q_0$. It follows that $(q_0 \cdot u^n) \circ v^n = (q_0 \circ v^n) \cdot u^n = q_0$, for all $n \geq 0$. Since the number of states in A is finite, there are integers $i, j \geq 0$ and $x, y \geq 1$ such that $q_0 \cdot u^i = q_0 \cdot u^{i+x}$ and $q_0 \circ v^j = q_0 \circ v^{j+y}$. We obtain $q_0 = (q_0 \cdot u^{i+x}) \circ v^i$ and $q_0 = (q_0 \circ v^{j+y}) \cdot u^j$. Now let $w \in L(A)$, i.e., $q_0 \cdot w \in F$. From our considerations above we get $((q_0 \cdot u^i) \circ v^i) \cdot w \in F$ and

$$((q_0 \cdot u^i) \circ v^i) \cdot w = ((q_0 \cdot u^{i+x}) \circ v^i) \cdot w = (((q_0 \cdot u^i) \circ v^i) \cdot w) \cdot u^x \in F.$$

Recall that $|uv| \geq 1$. If $|u| \geq 1$, then we see that A is exiting because the accepting state $((q_0 \cdot u^{i+x}) \circ v^i) \cdot w$ cannot go to a non-accepting sink state on every input symbol. If $|u| = 0$ then it must be $|v| \geq 1$. Now a similar argumentation gives $((q_0 \circ v^j) \cdot u^j) \cdot w \in F$ and

$$((q_0 \circ v^j) \cdot u^j) \cdot w = ((q_0 \circ v^{j+y}) \cdot u^j) \cdot w = (((q_0 \circ v^j) \cdot u^j) \cdot w) \circ v^y \in F.$$

Here the accepting state $((q_0 \circ v^j) \cdot u^j) \cdot w$ cannot go to a non-accepting sink state on every alphabet symbol, which shows that A is exiting. □

The converse of Lemma 16 is not true which can easily be seen by the minimal biautomaton for the language $\{a, aa\}$. Since the language is finite, there cannot be a cycle $q_0 = (q_0 \cdot u) \circ v$, hence the biautomaton is non-returning. However, since both states $q_0 \cdot a$ and $(q_0 \cdot a) \cdot a$ are accepting, the automaton cannot be non-exiting.

Now we study the classes of languages accepted by biautomaton that are non-exiting or non-returning. While minimal non-exiting DFAs characterize the class of prefix-free languages, we show in the following that minimal non-exiting biautomata characterize a different language class, namely the class of circumfix-free languages. A word $v \in \Sigma^*$ is a *circumfix* of a word $w \in \Sigma^*$, if $w = w_1 w_2 w_3$ and $v = w_1 w_3$, for some words $w_1, w_2, w_3 \in \Sigma^*$. A language L is called *circumfix-free* if there are no two *different* words $w, v \in L$, such that v is a circumfix of w. Notice that prefixes and suffixes of a word are also circumfixes (where one "side" is λ). Therefore the class of circumfix-free languages is contained in both the classes of prefix-free languages and suffix-free languages.

Theorem 17 *A regular language is circumfix-free if and only if its* minimal *biautomaton is non-exiting.*

Proof: Let $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ be a minimal biautomaton and $L = L(A)$. First assume that A is exiting, i.e., there is an accepting state $q \in F$ and a non-empty word $w \in \Sigma^+$ such that $q \cdot w \in F$. Since q must be reachable from the initial state of A, there are words $u, v \in \Sigma^*$ with $(q_0 \cdot u) \circ v = q$. Then both words uv and uwv belong to $L(A)$, but uv is a circumfix of w. Therefore, if $L(A)$ is circumfix-free then A must be non-exiting.

For the reverse implication notice that whenever there are two different words w and w' in $L(A)$ such that w' is a circumfix of w, then $w = w_1 w_2 w_3$ and $w' = w_1 w_3$, with $w_1, w_3 \in \Sigma^*$ and $w_2 \in \Sigma^+$ (because $w \neq w'$). It follows $(q_0 \cdot w_1) \circ w_3 \in F$ and $((q_0 \cdot w_1) \circ w_3) \cdot w_2 \in F$, and since $w_2 \neq \lambda$ the automaton A must be exiting. Thus, if A is non-exiting then $L(A)$ must be circumfix-free. □

Now we consider languages accepted non-returning biautomata. If a minimal biautomaton A is non-returning then clearly the contained minimal DFA A_{fwd} is non-returning, too. Therefore the class of languages accepted by minimal non-returning biautomata is contained in the class of languages accepted by minimal non-returning DFAs. Moreover, this inclusion is strict because the minimal DFA for the language ab^* is non-returning, while the minimal biautomaton for that language is not (it has a backward transition loop for symbol b on its initial state). Therefore we have the following result.

Theorem 18 *The class of languages accepted by* minimal *non-returning biautomata is strictly contained in the class of languages accepted by* minimal *non-returning deterministic finite automata.* □

7 Conclusions

We continued the study of structural properties on biautomata started in [7, 11, 12]. Our focus was on the effect of classical properties of deterministic finite automata such as, e.g., permutation-freeness, strongly permutation-freeness, and orderability, on biautomata. It is shown that this approach on structurally restricting the recently introduced biautomata model was worth looking at. A comparison of the induced language families on structurally restricted deterministic automata and biautomata is given in Table 1. Future research on the subject under consideration may consist on some further properties such as, e.g., biautomata where all states are final or all are initial. In the case of ordinary deterministic finite automata the family of prefix-closed languages is obtained by the former property, while the latter gives the family of suffix-closed languages. Moreover, it would be also interesting to study, which structural properties can be successfully applied to nondeterministic biautomata, as introduced in [8].

References

[1] M. A. Arbib (1969): *Theories of Abstract Automata*. Automatic Computation, Prentice-Hall, London.

[2] A. Badr, V. Geffert & I. Shipman (2009): *Hyper-Minimizing Minimized Deterministic Finite State Automata*. *RAIRO–Informatique théorique et Applications / Theoretical Informatics and Applications* 43(1), pp. 69–94. doi:10.1051/ita:2007061

[3] J. Brzozowski & B. Liu (2021): *Syntactic Complexity of Finite/Cofinite, Definite, and Reverse Definite Languages*. arXiv:1203.2873v1 [cs.FL].

[4] J. A. Brzozowski & F. E. Fitch (1980): *Languages of $\mathscr{R}$-Trivial Monoids*. *Journal of Computer and System Sciences* 20(1), pp. 32–49. doi:10.1016/0022-0000(80)90003-3

[5] J.-M. Champarnaud, J.-P. Dubernard, H. Jeanne & L. Mignot (2013): *Two-Sided Derivatives for Regular Expressions and for Hairpin Expressions*. In A. H. Dediu, C. Martín-Vide & B. Truthe, editors: *Proc. of the 7th International Conference on Language and Automata Theory and Applications*, *LNCS* 7810, Springer, Bilbao, Spain, pp. 202–213. doi:10.1007/978-3-642-37064-9_19

[6] I. M. Havel (1969): *The theory of regular events II*. *Kybernetika* 6, pp. 520–544.

[7] M. Holzer & S. Jakobi (2013): *Minimization and Characterizations for Biautomata*. In S. Bensch, F. Drewes, R. Freund & F. Otto, editors: *Proc. of the 5th International Workshop on Non-Classical Models of Automata and Applications*, *books@ocg.at* 294, Österreichische Computer Gesellschaft, Umeå, Sweden, pp. 179–193.

[8] M. Holzer & S. Jakobi (2013): *Nondeterministic Biautomata and Their Descriptional Complexity*. In H. Jürgensen & R. Reis, editors: *Proc. of the 15th International Workshop on Descriptional Complexity of Formal Systems*, *LNCS* 8031, Springer, London, Ontario, Canada, pp. 112–123. doi:10.1007/978-3-642-39310-5_12

[9] M. Holzer & S. Jakobi (2014): *Minimal and Hyper-Minimal Biautomata*. IFIG Research Report 1401, Institut für Informatik, Justus-Liebig-Universität Gießen, Arndtstr. 2, D-35392 Gießen, Germany.

[10] G. Jirásková & O. Klíma (2012): *Descriptional Complexity of Biautomata*. In M. Kutrib, N. Moreira & R. Reis, editors: *Proc. of the 14th International Workshop Descriptional Complexity of Formal Systems*, *LNCS* 7386, Springer, Braga, Portugal, pp. 196–208. doi:10.1007/978-3-642-31623-4_15

[11] O. Klíma & L. Polák (2012): *Biautomata for k-Piecewise Testable Languages*. In H.-C. Yen & O. H. Ibarra, editors: *Proc. of the 16th International Conference Developments in Language Theory*, *LNCS* 7410, Springer, Taipei, Taiwan, pp. 344–355. doi:10.1007/978-3-642-31653-1_31

[12] O. Klíma & L. Polák (2012): *On Biautomata*. *RAIRO–Informatique théorique et Applications / Theoretical Informatics and Applications* 46(4), pp. 573–592. doi:10.1051/ita/2012014

[13] R. Loukanova (2007): *Linear Context Free Languages*. In C. B. Jones, Z. Liu & J. Woodcock, editors: *Proc. of the 4th International Colloquium Theoretical Aspects of Computing*, *LNCS* 4711, Springer, Macau, China, pp. 351–365.

[14] R. McNaughton & S. Papert (1971): *Counter-free automata*. *Research monographs* 65, MIT Press.

[15] M. L. Minsky (1967): *Computation: Finite and Infinite Machines*. Automatic Computation, Prentice-Hall.

[16] M. Perles, M. O. Rabin & E. Shamir (1963): *The Theory of Definite Automata*. *IEEE Transactions on Electronic Computers* EC-12(3), pp. 233–243. doi:10.1109/PGEC.1963.263534

[17] A. L. Rosenberg (1967): *A Machine Realization of the Linear Context-Free Languages*. *Information and Control* 10, pp. 175–188. doi:10.1016/S0019-9958(67)80006-8

[18] H.-J. Shyr & G. Thierrin (1974): *Ordered Automata and Associated Languages*. *Tamkang Journal of Mathematics* 5(1).

[19] I. Simon (1975): *Piecewise Testable Events*. In H. Brakhage, editor: *Proc. of the 2nd GI Conference on Automata Theory and Formal Languages*, *LNCS* 33, Springer, Kaiserslautern, Germany, pp. 214–222.

[20] G. Thierrin (1968): *Permutation Automata*. *Mathematical Systems Theory* 2(1), pp. 83–90. doi:10.1007/BF01691347